การศึกษาระบบควบคุมระยะไกล

A Study of Remote monitoring and Control

นางสาวโชษิตา แหลมคม

นางสาวฐิตินารถ ขำสอาด

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

การศึกษาระบบควบคุมระยะไกล

A Study of Remote monitoring and Control

นางสาวโชษิตา แหลมคม

นางสาวฐิตินารถ ขำสอาด

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมการวัดคุม
ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2556

A Study of Remote monitoring and Control

CHOSITA LEAMKHOM

TITINART KAMSA-ARD

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF RHE REQUIREMENT FOR THE DEGREE OF
BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING
DEPARTMENT OF INSTRUMENTATION ENGINEERING
FACULTY OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2013

ภาควิชาวิศวกรรมการวัดคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ใบรับรองปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	**การศึกษาการควบคุมระยะไกล**		
	A Study of Remote monitoring and Control		
นักศึกษาผู้จัดทำ	นางสาวโชษิตา แหลมคม	**รหัสประจำตัว**	53010391
	นางสาวฐิตินารถ ขำสอาด	**รหัสประจำตัว**	53010409
ปริญญา	วิศวกรรมศาสตรบัณฑิต		
สาขาวิชา	วิศวกรรมการวัดคุม		
ปีการศึกษา	2556		

อาจารย์ผู้ควบคุมปริญญานิพนธ์	ลายมือชื่อ
รศ.วิริยะ กองรัตน์	

วัน/เดือน/ปี ที่สอบ วันพุธที่ 26 กุมภาพันธ์ พ.ศ. 2557
สถานที่สอบ ณ ห้องสอบปริญญานิพนธ์ ภาควิชาวิศวกรรมการวัดคุม

ภาควิชารับรองแล้ว

(รศ.ดร.ฬุศักดิ์ ชีวสุวิทย์)
หัวหน้าภาควิชาวิศวกรรมการวัดคุม

หัวข้อปริญญานิพนธ์	การศึกษาการควบคุมระยะไกล
	A Study of Remote monitoring and Control
นักศึกษาผู้จัดทำ	นางสาวโชษิตา แหลมคม
	นางสาวฐิตินารถ ขำสอาด
อาจารย์ที่ปรึกษา	รศ.วิริยะ กองรัตน์
ปีการศึกษา	2556

บทคัดย่อ

เนื่องจากในปัจจุบันเป็นยุคของดิจิตอลและเทคโนโลยีได้เข้ามามีบทบาทในชีวิตประจำวันของมนุษย์มากขึ้น เพื่อสร้างความสะดวกสบายและรวดเร็วในการทำงาน ดังนั้นปริญญานิพนธ์ฉบับนี้จึงได้นำเสนอการควบคุมระยะไกลโดยใช้ไมโครคอนโทรลเลอร์ AT89C4051 เป็นตัวหลักในการควบคุมการทำงานของระบบ โดยสามารถทำการสังเกตการณ์ผลตอบสนองของระบบ ทั้งค่าอินพุทและเอาท์พุท ผ่านโปรแกรม visual basic 2010 ซึ่งเป็นส่วนที่ใช้เชื่อมต่อผู้ใช้งานกับระบบ และยังสามารถสังเกตการณ์ผลตอบสนองของระบบได้แม้ไม่ได้อยู่ในบริเวณนั้นโดยผ่านเครือข่ายแลน และเนื่องจากปริญญานิพนธ์ฉบับนี้เน้นไปทางการศึกษาระบบ ทั้งนี้ค่าอินพุทและเอาท์พุทที่รับและส่งในระบบจึงเป็นสัญญาณมาตรฐาน และมีความอเนกประสงค์ในการใช้งาน จึงสามารถนำไปประยุกต์ใช้กับระบบการควบคุมระยะไกลอื่นๆในชีวิตประจำวันได้จริง

Thesis Title	A Study of Remote monitoring and Control
Author	Ms. Chosita Leamkhom
	Ms. Titinart Kamsa-ard
Thesis Advisor	Assoc.Prof. Viriya Kongrattana
Year	2013

ABSTRACT

Nowadays is the era of digital and technology has role in the daily life to be convenient. Thus, this thesis presents a remote control using microcontroller AT89C4051 is the main of system by observed response both input and output via Visual Basic 2010 which is used communicate user with system. This thesis focuses on educational system that standard signal and multi-purpose to work and can apply to other remote control.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดีเพราะได้รับความเมตตาจากรองศาสตราจารย์ วิริยะ กองรัตน์ ที่ได้ให้คำปรึกษาและคำแนะนำแก่ผู้วิจัยมาโดยตลอด อีกทั้งยังเอื้อเฟื้ออุปกรณ์และเครื่องมือต่างๆ ในการทำปริญญานิพนธ์ฉบับนี้ คณาจารย์ภาควิชาวิศวกรรมการวัดคุมทุกท่านที่ได้ให้คำแนะนำอันเป็นประโยชน์ ผู้วิจัยรู้สึกซาบซึ้งและกราบขอพระคุณเป็นอย่างสูง

และที่ลืมเสียมิได้คือ ขอกราบขอบพระคุณ คุณพ่อและคุณแม่ อันเป็นที่รักยิ่งที่สนับสนุนและเป็นแรงบันดาลใจในการทำปริญญานิพนธ์ฉบับนี้ ตลอดจนเพื่อนๆทุกคนที่ให้ความช่วยเหลือและคำแนะนำในด้านต่าง

คุณค่าและประโยชน์อันพึงมีจากปริญญานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

คณะผู้จัดทำ

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและเหตุจูงใจของการวิจัย

ปัจจุบันเทคโนโลยีในด้านต่างๆได้ถูกพัฒนาไปอย่างก้าวไกลให้มีความทันสมัยมากขึ้น ทั้งนี้ก็เพื่อความสะดวกสบายในการดำเนินชีวิตประจำวันของมนุษย์ และด้วยวิถีชีวิตที่เร่งรีบประกอบกับความรวดเร็วในการทำสิ่งต่างๆนั้น การลดปัญหาเรื่องระยะทางความห่างไกลและช่วยประหยัดเวลา ดังนั้นระบบการควบคุมระยะไกลจึงเป็นอีกหนึ่งเรื่องที่น่าสนใจที่จะนำมาช่วยอำนวยความสะดวกสบายเพื่อใช้ในชีวิตประจาวันของมนุษย์ จึงทำให้เกิดแนวคิดที่จะทำการศึกษาเรื่องระบบการควบคุมระยะไกลขึ้น โดยให้สามารถนำมาประยุกต์ใช้กับระบบการควบคุมต่างๆได้ ประกอบกับความเจริญก้าวหน้าทางด้านโปรแกรมและคอมพิวเตอร์มีความสะดวกรวดเร็วในการประมวลผลและติดต่อสื่อสารข้อมูล ทำให้เราสามารถที่จะสังเกตการณ์ผลตอบสนองและสั่งการทำงานผ่านโปรแกรมบนคอมพิวเตอร์ได้ และโปรแกรมที่มีความสามารถในการติดต่อระหว่าง software และ hardware ที่นิยมใช้กันมากอยู่ในปัจจุบันนี้ก็คือโปรแกรม Visual Basic 2010 ซึ่งถูกนำมาใช้ในการเขียนโปรแกรมเพื่อเชื่อมต่อระหว่างส่วนของผู้ใช้งานกับส่วนของ Hardware ให้สามารถติดต่อสื่อสารข้อมูลกันได้โดยผ่านทางพอร์ทอนุกรม RS-232 ไปให้ที่ตัวควบคุมไมโครคอนโทรลเลอร์เพื่อทำการประมวลผลให้กับอุปกรณ์ต่อไป และเมื่อเราต้องการที่จะสังเกตการณ์ผลตอบสนองหรือสั่งการทำงานผ่านโปรแกรมบนคอมพิวเตอร์เมื่อเราไม่ได้อยู่ในบริเวณนั้นเราก็สามารถทำได้โดยการเชื่อมต่อคอมพิวเตอร์และอุปกรณ์เข้ากับเครือข่ายอินเทอร์เน็ต(แลน) ถือเป็นความสะดวกสบายและรวดเร็วในการติดต่อสื่อสารข้อมูลได้อย่างดีเลยทีเดียว

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1.เพื่อศึกษาการทำงานของระบบการควบคุมระยะไกลด้วยไมโครคอนโทรลเลอร์Atmel AT89C4051

2. เพื่อศึกษาการสื่อสารในการรับส่งข้อมูลระหว่างอุปกรณ์

3. เพื่อศึกษาดูการสังเกตผลตอบสนองของข้อมูลผ่านทางคอมพิวเตอร์โดยใช้โปรแกรม Visual Basic 2010

4. ศึกษาการเขียนโปรแกรมควบคุมบน Visual Basic 2010 เพื่อแสดงผลข้อมูลให้กับส่วนผู้ใช้งาน

5. ศึกษาการติดต่อสื่อสารข้อมูลโดยผ่านเครือข่ายแลน

1.3 ขอบเขตของปริญญานิพนธ์

1. สามารถรับส่งข้อมูลที่เป็นทั้งค่าอนาลอกและดิจิตอลจากอุปกรณ์ผ่านเข้าคอมพิวเตอร์ซึ่งเป็นส่วนผู้ใช้งานได้

2. เข้าใจการติดต่อสื่อสารข้อมูลผ่านพอร์ทอนุกรม(RS232)โดยมีโปรแกรม Visual Basic 2010 เป็นตัวแสดงผลที่คอมพิวเตอร์ได้

3. สามารถทำการติดต่อสื่อสารรับส่งข้อมูลในระยะไกลได้โดยผ่านเครือข่ายแลน

1.4 ขั้นตอนการศึกษา

1.ศึกษาโครงสร้างระบบการควบคุมระยะไกลโดยบอร์ดไมโครคอนโทรลเลอร์ Atmel AT89C4051

2. ศึกษาเรื่องการติดต่อสื่อสารข้อมูลภายในอุปกรณ์แต่ละตัว

3. ศึกษาการเชื่อมต่อกับเครือข่ายแลนเพื่อให้ผู้ใช้สามารถติดต่อสื่อสารข้อมูลได้จากระยะไกล

4. ศึกษาการเขียนโปรแกรม Visual Basic 2010 เพื่อแสดงผลและติดต่อกับผู้ใช้งานได้

5. ศึกษาคำสั่งในการทำงานของแต่ละโปรโตคอล เพื่อให้เข้าใจการทำงานของการรับส่งข้อมูล

6. ทำการเขียนโปรแกรม Visual Basic 2010 เพื่อเป็นการติดต่อสื่อสารระหว่างอุปกรณ์กับผู้ใช้งาน

7. ทำการรับส่งค่าข้อมูลทั้งอินพุทและเอาท์พุทที่เป็นทั้งดิจิตอลและอนาลอก

8. ทำการต่ออุปกรณ์เข้ากับเครือข่ายแลน เชื่อมต่อคอมพิวเตอร์เข้ากับเครือข่ายเดียวกันกับอุปกรณ์และทำการทดลองรับส่งค่าข้อมูล

1.5 ประโยชน์ที่คาดว่าจะได้รับ

ประโยชน์ที่คาดว่าจะได้รับจากการทำปริญญานิพนธ์เล่มนี้คือ ได้เข้าใจโครงสร้างและการทำงานของระบบการควบคุมระยะไกล ทราบถึงองค์ประกอบต่างๆที่สำคัญเกี่ยวกับการควบคุม ทั้งในเรื่องของการติดต่อสื่อสารระหว่างอุปกรณ์ ซึ่งเป็นสิ่งที่เราไม่สามารถมองเห็นแต่เมื่อเข้าใจการทำงานของอุปกรณ์หรือไอซีแต่ละตัวว่าทำหน้าที่อะไรแล้วก็จะทำให้เข้าระบบที่เราศึกษามากขึ้น

บทที่ 2
ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 ทฤษฎีและหลักการของ Visual Basic 2010

โปรแกรม Visual Basic 2010 จัดว่าเป็นโปรแกรมภาษาที่สมบูรณ์แบบมากที่สุดภาษาหนึ่ง เพราะประกอบด้วยเครื่องมือที่ช่วยให้การพัฒนาแอพพลิเคชั่นที่หลากหลาย และสะดวกสบายต่อการใช้งานมากกว่าเวอร์ชันก่อนๆมาก

2.1.1ความเป็นมาของVisualBasic2010

โปรแกรมภาษา Visual Basic นั้นพัฒนาขึ้นมาจากภาษาดั้งเดิมคือ ภาษา Basic ซึ่งภาษาที่ใช้ในการเขียนโปรแกรมในระยะเริ่มต้นจะใช้งานในแบบ Text Mode ต่อมาประมาณปี ค.ศ.1990 Microsoft ได้ประกาศเปิดตัวภาษา Visual Basic ซึ่งเป็นเหมือนกับชุดเครื่องมือ (Tool) ในการสร้างส่วนติดต่อกับผู้ใช้ในแบบกราฟิก (Graphic User Interface ; GUI) โดยใช้ภาษา Basic ควบคุมการทำงาน หลังจากนั้นมา Visual Basic ก็ได้รับความนิยมเพิ่มมากขึ้นเรื่อยๆจนกลายมาเป็นภาษาคอมพิวเตอร์ที่มีผู้ใช้งานมากที่สุดในปัจจุบัน เนื่องจากมีโครงสร้างภาษาที่ง่าย มีชุดเครื่องมือ (Tool) ในการสร้างส่วนติดต่อกับผู้ใช้ (User Interface) อย่างครบถ้วนและสะดวกต่อการใช้งาน ทำให้สามารถเรียนรู้การพัฒนาโปรแกรมได้ในระยะเวลาอันสั้นโดย Visual Basic ได้มีการพัฒนามาตั้งแต่ Version 1 จนถึง Version 6 (VB 6, ในชุด Visual Studio 98) ซึ่งเป็นแนวทางเดิมโดยการทำงานจะยึดติดกับระบบปฏิบัติการ Windowsเป็นหลัก จนกระทั่งปี พ.ศ.2002 ได้เปลี่ยนเป็น Visual Basic.NET (หรือVB7) ที่ทำงานบนแพลตฟอร์มแบบใหม่ของ Microsoft ที่เรียกว่า .NET Framework แล้วให้มีการพัฒนามาเป็น Visual Basic 2003, 2005, 2008 และในที่สุดก็มาเป็น Visual Basic 2010

2.1.2 ภาษาวิชวลเบสิก (Visual Basic language) คืออะไร

วิชวลเบสิก (Visual Basic) หรือ VB เป็นโปรแกรมภาษาคอมพิวเตอร์ตัวหนึ่ง ซึ่งใช้สำหรับสร้างหรือพัฒนาโปรแกรมใช้งานบนวินโดวส์มีความสามารถในการทำงานที่คล้ายกับภาษาคอมพิวเตอร์อื่นๆ เช่น C , Pascal , C++, C# สร้างโดยบริษัทไมโครซอฟท์ ภาษานี้เป็นหนึ่งในภาษาโปรแกรมยอดนิยมสำหรับโปรแกรมที่ใช้ในด้านธุรกิจ ซึ่งเป็นภาษาคอมพิวเตอร์ระดับสูง ใช้งานง่าย เหมาะสำหรับผู้เริ่มต้น เพราะใช้คำในภาษาอังกฤษที่เข้าใจง่าย และเมื่อเป็น Visual Basic ซึ่งใช้ลักษณะของ การมองเห็นได้ (Visual) ที่เป็นการติดต่อกับผู้ใช้ด้วยกราฟิก หรือ รูปภาพ (Graphical User Interface -GUI) จึงทำให้การพัฒนาโปรแกรมใช้งานได้สะดวกและรวดเร็วขึ้น ถึงแม้จะใช้งานง่าย แต่ก็มีความสามารถสูง เหมาะสำหรับการพัฒนาโปรแกรมใช้งานได้หลายด้าน เช่น งานคำนวณทั่วไป งานด้านฐานข้อมูล เกม ฯลฯ และยังได้พัฒนาต่อเป็นภาษา VB.NET อีกด้วย

ความหมายของ .NET (ดอทเน็ต) เป็นกลุ่มของเทคโนโลยีทางซอฟต์แวร์ที่เชื่องโยงข้อมูล, ข่าวสาร, คน, ระบบและอุปกรณ์ต่างๆ เข้าด้วยกัน ด้านการเขียนโปรแกรมดอทเน็ต หมายถึง .NET Framework คือ สภาพแวดล้อมที่สนับสนุนการพัฒนา และรันโปรแกรมในรูปแบบของ .NET Framework จะทำหน้าที่ควบคุมการรันโปรแกรม และให้บริการทรัพยากรต่างๆ แก่โปรแกรมที่รัน
เช่น การโหลดโปรแกรมขึ้นมาทำงาน การจัดการหน่วยความจำ การจัดเตรียมไลบรารีให้โปรแกรมเรียกใช้งาน

2.1.3 วิวัฒนาการของภาษาวิชวลเบสิก

ภาษาวิชวลเบสิกนั้นได้มีการพัฒนามาตั้งแต่รุ่นแรกๆ ที่ทำงานบนระบบปฏิบัติการ DOS มาจนถึงปัจจุบันที่ทำงานอยู่บนระบบปฏิบัติการวินโดวส์ ก่อนจะมาเป็น ภาษา Visual Basic.NET ให้เราใช้กันได้นั้น เราอาจจะคิดว่า ภาษา BASIC เป็นของ Microsoft คิดค้นขึ้น แท้จริงแล้ว ภาษา BASIC เป็นคำที่เกิดจากอักษรย่อของคำว่า Beginner's Allpurpose Symbolic Instruction Code ถูกพัฒนาขึ้นตั้งแต่ในช่วงต้นปี คศ. 1963 ที่วิทยาลัย Dartmount College ในสหรัฐอเมริกา โดย จอห์น เคมเมนี John G. Kemeny และ ธอมัส เคิรตส์ Thomas E.Kurtz ถูกออกแบบมาให้เป็นภาษาคอมพิวเตอร์ที่ใช้งานได้ง่ายในการเขียนโปรแกรม โดย สมัยก่อน มีการใช้งานบนเครื่องไมโครคอมพิวเตอร์ โดยมีการนำมาทำเป็น ชุดคำสั่งถาวร หรือ Firmware เพื่อเก็บไว้ใน ROM บนไมโครคอมพิวเตอร์รุ่นแรกๆ รวมถึงการเขียนโปรแกรมทั่วไปด้วย ด้วยความง่าย จึงเป็นที่แพร่หลายและได้รับความนิยม และต่อมา ก็ได้เกิด รุ่นต่างๆ ของ BASIC มาอีก เช่น * ในปี 1975 BASIC for Altair by Bill Gates * ในปี 1980 GWBasic by Microsoft * ในปี 1980 QuickBasic by Microsoft * ในปี 1991 Visual Basic by Microsoft และ ยังมี Turbo BASIC อีก จากนั้นได้ผ่านการพัฒนาต่อเนื่องมาอีกหลายรุ่นนับตั้งแต่ Visual Basic รุ่นแรก จนมาเป็น Visual Basic 6 ในปี 1998 และเมื่อมีการพัฒนา NET Framework ขึ้น การเปลี่ยนแปลงทางด้านโครงสร้างของภาษา BASIC ครั้งใหญ่ จนกลายมาเป็น Visual Basic.NET ที่มีใช้กันอยู่ในปัจจุบัน โดยปี 2002 ไมโครซอฟต์ได้ออกแบบภาษา Visual Basic ให้สนับสนุนการเขียนโปรแกรมเชิงวัตถุ (oop) อย่างสมบูรณ์ เป็น Visual Basic เวอร์ชั่น 7 พร้อมกับ .NET Framework 1.0

- Visual Basic 2005 พร้อมกับ .NET Framework 2.0
- Visual Basic 2008 พร้อมกับ .NET Framework 3.5
- Visual Basic 2010 พร้อมกับ .NET Framework 4.0
- Visual Basic 2012 พร้อมกับ .NET Framework 4.5

แนะนำโปรแกรม Visual Studio 2010 Express

รุ่น Express หมายถึง รุ่นที่เปิดให้ดาวน์โหลดไปใช้ได้ฟรี โดยข้อดีของรุ่น express นี้ก็คือ นอกจากจะใช้ในการศึกษาแล้วยังสามารถนำไปพัฒนาโปรแกรมในเชิงพาณิชย์ได้ครับ ประหยัดค่าลิขสิทธิ์ซอฟต์แวร์ ซึ่งรุ่นสูงกว่า (Ultimate, Premium และ Professional)เราอาจจะต้องเสียค่าลิขสิทธิ์พร้อมสิทธิประโยชน์พิเศษด้านอื่นๆ ในราคาระหว่าง $7xx ถึง $1x,xxx

Visual Studio คือ ซอฟต์แวร์ประเภท IDE (Integrated Development Environment) ช่วยให้ผู้พัฒนาโปรแกรมสามารถเขียนโปรแกรมด้วยความสะดวกสบายขึ้น สามารถแก้ไขข้อผิดพลาดในการเขียนโปรแกรมได้ง่าย รวดเร็ว

2.1.4 จุดเด่นของ Visual Basic

2.1.4.1 สร้างแอพพลิเคชั่นได้ง่ายและรวดเร็ว

โปรแกรม Visual Basic ได้รับการวางตัวให้เป็นเครื่องมือที่ช่วยให้ในการสร้างแอพพลิเคชั่นเป็นไปได้อย่างรวดเร็วและง่ายดาย เพื่อลดเวลาในการสร้างแอพพลิเคชั่นให้สั้นลง

ซึ่งเรียกรูปแบบอันนี้ว่า Rapid Application Development หรือ RAC ทั้งนี้เพราะได้ทำการขจัดงานที่โปรแกรมเมอร์ต้องทำซ้ำซากออกไปเหลือเฉพาะที่ต้องการโฟกัสเกี่ยวกับปัญหาที่เกิดขึ้นของงานจริงๆแล้วเขียนโปรแกรมจัดการกับปัญหานั้นๆส่วนเรื่องอื่นๆเหลือให้ Visual Basic จัดการ

2.1.4.2 การเขียนโปรแกรมที่ง่ายต่อการเรียนรู้

ถ้าได้มีโอกาสในการเขียนโปรแกรมด้วย Visual Basic แล้วจะเห็นว่าภาษา basic ในตัวของ Visual Basic นั้นอ่านง่าย คืออ่านแล้วใกล้เคียงกับภาษาที่เราใช้งานกันปกติคืออ่านแล้วสื่อความหมายเข้าใจได้ง่ายกว่าภาษาของโปรแกรมอื่นๆทำให้ผู้ที่เพิ่งเริ่มต้นในการเขียนโปรแกรมสามารถทำความเข้าใจกับการเขียนโปรแกรมได้อย่างรวดเร็ว

2.1.4.3 รวมเครื่องมืออำนวยความสะดวกในการเขียนโปรแกรม

นอกจากจะง่ายต่อการเรียนรู้แล้ว Visual Basic ยังมีเครื่องมือที่ช่วยให้การเขียนโปรแกรมเป็นเรื่องที่ไม่ยุ่งยากเพราะจะมีเครื่องมือที่ช่วยให้ไม่ต้องจดจำไวยากรณ์ภาษาที่ยุ่งยากและตรวจสอบโดยอัตโนมัติว่าโปรแกรมที่เขียนนั้นมีความถูกต้องตามหลักของภาษาหรือไม่ซึ่งจะมีการแยกแยะส่วนของโปรแกรมอย่างเป็นระเบียบ ทำให้งานของโปรแกรมเมอร์นั้นลดลงได้มาก

นอกจากจะมีเครื่องมือช่วยในการเขียนโปรแกรมแล้ว ยังมีเครื่องมือที่ใช้ทดสอบแก้ไขโปรแกรม(Debugger)ที่เขียนขึ้นมาว่าทำงานได้ถูกต้องหรือไม่ มีระบบขอความช่วยเหลือ(Online Help)ไว้อ้างอิง และขอความช่วยเหลือในจุดที่เราสงสัยข้องใจได้

เครื่องมือทั้งหมดที่กล่าวมานี้จะถูกจัดรวมไว้ในสภาพแวดล้อมการทำงานเดียวกัน (เรียกย่อๆว่า IDE)ทำให้เรียกใช้งานได้สะดวกตั้งแต่เขียนโปรแกรม, ทดสอบ, แก้ไข, สร้างชุดติดตั้ง รวมทั้งระบบการขอความช่วยเหลือ ซึ่งเราสามารถเพิ่มเติมเครื่องมือชนิดใหม่ๆเข้าไปได้เรื่อยๆหรือถอดเครื่องมือที่ไม่จำเป็นต้องใช้ออกเพื่อประหยัดพื้นที่ฮาร์ดดิสก์ก็ได้เช่นกัน

2.1.5 รูปแบบการพัฒนาแอพพลิเคชั่นด้วย Visual Basic

เมื่อเรามองเห็นว่าโปรแกรม Visual Basic สามารถช่วยให้เราทำการสร้างแอพพลิเคชั่นบน Windows ได้ง่ายและรวดเร็วแล้วยังมีรูปแบบหนึ่งที่โปรแกรม Visual Basic สามารถสร้างขึ้นมาได้อีก คือ

2.1.5.1 พัฒนาแอพพลิเคชั่นกับ ActiveX Control

เทคโนโลยีที่มีชื่อว่า ActiveX เป็นตัวอยู่เบื้องหลังความสำเร็จของ Visual Basic ซึ่งช่วยลดการทำงานที่ซ้ำซ้อนของการเขียนโปรแกรมลงไปมาก

ตัวอย่างเช่น การเขียนโปรแกรมเพื่อรับค่าของข้อมูลจากผู้ใช้แต่เดิมเราจะต้องมาเขียนโปรแกรมเพื่อวาดหน้าจอ, เขียนโปรแกรมวาดรูปของปุ่ม และช่องรับข้อความรวมทั้งเขียนโปรแกรมเพื่อจัดการกับข้อมูลที่ผู้ใช้ป้อนเข้ามาแต่ใน ActiveX จะทำให้เราสนใจเฉพาะการจัดการกับส่วนของข้อมูลที่ผู้ใช้ป้อนเข้ามาเท่านั้นที่เหลือจัดการให้เองโดยช่องรับข้อความปุ่มต่างๆนั้นเราจะทำการใช้ ActiveX Control จัดการ

นอกจากจะลดความซับซ้อนลงแล้วการที่เราใช้ ActiveX Control ช่วยในการเขียนโปรแกรมจะทำให้โปรแกรมที่เราเขียนกับโปรแกรมที่คนอื่นๆเขียนนั้นตั้งอยู่บนมาตรฐานเดียวกัน ทำให้การปรับปรุงโปรแกรมทำได้ง่าย เพราะใครๆก็เข้าใจมาตรฐานของ ActiveX Control นี้ทำให้ไม่ต้องกังวลว่าโปรแกรมที่เขียนขึ้นจะมีเฉพาะคนเขียนเท่านั้นที่เข้าใจ

2.1.5.2 สร้างแอพพลิเคชั่นที่ใช้งานกับฐานข้อมูล

เป็นแอพพลิเคชั่นที่มีการใช้งานมากที่สุด เพราะระบบร้านค้า, คลังสินค้า,ระบบบัญชี, ระบบบริหาร งานบุคคลหรือแม้แต่ E-Commerce ทั้งหมดต่างๆต้อมีส่วนที่ติดต่อกับฐานข้อมูลที่แน่นอน

โปรแกรม Visual Basic ได้ช่วยให้การสร้างแอพพลิเคชั่นกับฐานข้อมูลเป็นเรื่องที่ทำได้ง่ายดาย เพราะมีเครื่องมือต่างๆที่อำนวยความสะดวกในการเขียนโปรแกรมเพื่อใช้งานข้อมูลจากฐานข้อมูลซึ่งไม่จำกัดด้วยว่าจะเป็นฐานข้อมูลแบบใดทั้งฐานข้อมูลส่วนบุคคล, ฐานข้อมูลผ่านเครือข่าย หรือฐานข้อมูลผ่านอินเทอร์เน็ต จากความสามารถที่หลากหลายเหล่านี้ทำให้ Visual Basic จึงเป็นตัวเลือกอันดับต้นๆของการสร้างแอพพลิเคชั่นที่เกี่ยวกับฐานข้อมูลในโลกธุรกิจยุคปัจจุบัน

2.1.5.3 สร้างแอพพลิเคชั่นแบบใหม่กับอินเทอร์เน็ต

อินเทอร์เน็ตนับว่ามีความสำคัญกับชีวิตของคนที่ใช้ไอทีมากขึ้นทุกวัน ซึ่ง Visual Basic เปิดโอกาสให้เราสามารถสร้างแอพพลิเคชั่นเพื่อรองรับการทำงานร่วมกับอินเทอร์เน็ตได้ด้วยการใช้ความรู้เดิมที่เรามีอยู่แล้วจากการสร้างแอพพลิเคชั่นปกติ รวมทั้งมีเครื่องมือเสริมการทำงานต่างๆอย่างมากมาย

จุดเด่นของ Visual Basic อีกข้อหนึ่งคือ เปิดโอกาสให้เรานำแอพพลิเคชั่นปกติที่เดิมทำงานกับคอมพิวเตอร์ส่วนบุคคลสามารถดัดแปลงใช้งานกับอินเทอร์เน็ตได้อย่างไม่ยากเย็นนัก ทำให้ไม่ต้องทำการโละงานเดิมๆทิ้ง

2.2 ฐานข้อมูล (Data Base)

ฐานข้อมูล (Database) หมายถึง กลุ่มของข้อมูลที่มีความสัมพันธ์กัน นำมาเก็บรวบรวมเข้าไว้ด้วยกันอย่างมีระบบและข้อมูลที่ประกอบกันเป็นฐานข้อมูลนั้น ต้องตรงตามวัตถุประสงค์การใช้งานขององค์กรด้วยเช่นกัน เช่น ในสำนักงานก็รวบรวมข้อมูล ตั้งแต่หมายเลขโทรศัพท์ของผู้ที่มาติดต่อจนถึงการเก็บเอกสารทุกอย่างของสำนักงาน ซึ่งข้อมูลส่วนนี้จะมีส่วนที่สัมพันธ์กันและเป็นที่ต้องการนำออกมาใช้ประโยชน์ต่อไปภายหลัง ข้อมูลนั้นอาจจะเกี่ยวกับบุคคล สิ่งของสถานที่ หรือเหตุการณ์ใด ๆ ก็ได้ที่เราสนใจศึกษา หรืออาจได้มาจากการสังเกต การนับหรือการวัดก็เป็นได้ รวมทั้งข้อมูลที่เป็นตัวเลข ข้อความ และรูปภาพต่าง ๆ ก็สามารถนำมาจัดเก็บเป็นฐานข้อมูลได้ และที่สำคัญข้อมูลทุกอย่างต้องมีความสัมพันธ์กัน เพราะเราต้องการนำมาใช้ประโยชน์ต่อไปในอนาคต

2.2.1 ระบบฐานข้อมูล (Database System)

หมายถึง การรวมตัวกันของฐานข้อมูลตั้งแต่ 2ฐานข้อมูลเป็นต้นไปที่มีความสัมพันธ์กัน โดยมีวัตถุประสงค์เพื่อเป็นการลดความซ้ำซ้อนของข้อมูล และทำให้การบำรุงรักษาตัวโปรแกรมง่ายมากขึ้น โดยผ่านระบบการจัดการฐานข้อมูล หรือ เรียกย่อ ๆ ว่า DBMS

2.2.2 องค์ประกอบของระบบฐานข้อมูล

ระบบฐานข้อมูลเป็นเพียงวิธีคิดในการประมวลผลรูปแบบหนึ่งเท่านั้น แต่การใช้ฐานข้อมูลจะต้องประกอบไปด้วยองค์ประกอบหลักดังต่อไปนี้

2.2.2.1. แอพพลิเคชั่นฐานข้อมูล (Database Application)

เป็นแอพพลิเคชั่นที่สร้างไว้ให้ผู้ใช้งานสามารถติดต่อกับฐานข้อมูลได้อย่างสะดวกซึ่งมีรูปแบบการติดต่อกับฐานข้อมูลแบบเมนูหรือกราฟิก โยผู้ใช้ไม่จำเป็นต้องมีความรู้เกี่ยวกับฐานข้อมูลเลยก็สามารถเรียกใช้งานฐานข้อมูลได้เช่น บริการเงินสด ATM

2.2.2.2. ระบบจัดการฐานข้อมูล (Database Management System หรือ DBMS)

ระบบจัดการฐานข้อมูล หมายถึง กลุ่มโปรแกรมหรือซอฟต์แวร์ชนิดหนึ่ง ที่สร้างขึ้นมาเพื่อทำหน้าที่บริหารฐานข้อมูลโดยตรง ให้มีประสิทธิภาพมากที่สุด เป็นเครื่องมือที่ช่วยอำนวยความสะดวกให้ผู้ใช้สามารถเข้าถึงข้อมูลได้ โดยที่ผู้ใช้ไม่จำเป็นต้องรับรู้เกี่ยวกับรายละเอียดภายในโครงสร้างฐานข้อมูล พูดง่าย ๆ ก็คือ DBMS นี้เป็นตัวกลางในการเชื่อมโยงระหว่างผู้ใช้ และโปรแกรมต่างๆ ที่เกี่ยวข้องกับระบบฐานข้อมูล ตัวอย่างของ DBMS ที่นิยมใช้ในปัจจุบัน ได้แก่Microsoft Access, FoxPro, SQL Server, Oracle, Informix, DB2 เป็นต้น

หน้าที่ของระบบจัดการฐานข้อมูล มีดังนี้

1. กำหนดมาตรฐานข้อมูล
2. ควบคุมการเข้าถึงข้อมูลแบบต่าง ๆ
3. ดูแล-จัดเก็บข้อมูลให้มีความถูกต้องแม่นยำ
4. จัดเรื่องการสำรอง และฟื้นสภาพแฟ้มข้อมูล
5. จัดระเบียบแฟ้มทางกายภาพ (Physical Organization)
6. รักษาความปลอดภัยของข้อมูลภายในฐานข้อมูล และป้องกันไม่ใช้ข้อมูลสูญหาย
7. บำรุงรักษาฐานข้อมูลให้เป็นอิสระจากโปรแกรมแอพพลิเคชั่นอื่น ๆ
8. เชื่อมโยงข้อมูลที่มีความสัมพันธ์เข้าด้วยกัน เพื่อรองรับความต้องการใช้ข้อมูลในระดับต่าง ๆ

2.2.2.3. ดาต้าเบสเซิร์ฟเวอร์ (Database Server)

เป็นคอมพิวเตอร์ที่คอยให้บริการการจัดการฐานข้อมูล ซึ่งก็คือเครื่องคอมพิวเตอร์ที่ระบบจัดการฐานข้อมูลทำงานอยู่นั่นเองเพราะฉะนั้นควรเป็นคอมพิวเตอร์ที่มีความรวดเร็วในการทำงานสูงกว่าคอมพิวเตอร์ที่ใช้งานโดยทั่วไป

2.2.2.4. ข้อมูล (Data)

ข้อมูล คือ เนื้อหาของข้อมูลที่เราใช้งาน ซึ่งจะถูกเก็บในหน่วยความจำของดาต้าเบสเซิร์ฟเวอร์ โดยจะถูกเรียกมาใช้งานจากระบบจัดการฐานข้อมูล

2.2.2.5. ผู้บริหารฐานข้อมูล (Database Administrator หรือ DBA)

ผู้บริหารฐานข้อมูล คือ กลุ่มบุคคลที่ทำหน้าที่ดูแลข้อมูลผ่านระบบจัดการฐานข้อมูล ซึ่งจะควบคุมให้การทำงานเป็นไปอย่างราบรื่น นอกจากนี้ยังทำหน้าที่กำหนดสิทธิการใช้งานข้อมูล กำหนดในเรื่องความปลอดภัยของการใช้งาน พร้อมทั้งดูแลดาต้าเบสเซิร์ฟเวอร์ให้ทำงานอย่างปกติด้วย

2.2.3ประโยชน์ของการจัดทำระบบฐานข้อมูล

การรวบรวมข้อมูลต่างๆ เข้าเป็นระบบฐานข้อมูล โดยมีการออกแบบ การวิเคราะห์และสร้างความสัมพันธ์ของข้อมูล ซึ่งมีประโยชน์ต่อการใช้งานในสำนักงานทั่วไปอย่างมาก ดังนี้

2.2.3.1.ช่วยลดปัญหาของความซ้ำซ้อนของข้อมูลที่จัดเก็บเนื่องจากในขั้นตอนของการออกแบบฐานข้อมูล เมื่อพบข้อมูลบางส่วนที่ซ้ำซ้อนกันก็จะสามารถลดและปรับข้อมูลให้น้อยลง ขณะที่ยังคงความสามารถในการเรียกดูข้อมูลได้ ดังเดิมโดยใช้การกำหนดความสัมพันธ์ระหว่างข้อมูล

2.2.3.2.สามารถใช้ร่วมกันได้หลายคนและหลายหน่วยงานได้ไม่จำกัดเฉพาะโปรแกรมในปัจจุบันเท่านั้นแต่สามารถใช้กับโปรแกรมที่จะพัฒนาในอนาคตด้วย

2.2.3.3. สามารถหลีกเลี่ยงความขัดแย้งกันของข้อมูลได้ในระดับหนึ่ง เนื่องจากความซ้ำซ้อนของข้อมูล ดังเหตุผลในข้อแรกเมื่อลดความซ้ำซ้อนของข้อมูลแล้ว ระบบฐานข้อมูลก็จะมีข้อมูลเรื่องใดๆ อยู่น้อยชุดที่สุด ซึ่งสะดวกในการแก้ไขปรับปรุงต่างจากในกรณีที่มีข้อมูลอย่างเดียวกันหลายชุด ถ้ามีการแก้ไขแล้วไม่ได้แก้ไขข้อมูลครบทุกชุด เมื่อมีการเรียกใช้ข้อมูลจะ พบข้อมูลเรื่องเดียวกันแต่มีเนื้อหาต่างกัน

2.2.3.4. สามารถควบคุมความถูกต้องของข้อมูล ทั้งในเรื่องความถูกต้องของข้อมูลในแฟ้มข้อมูล (Relational Integrity) และความถูกต้องของความสัมพันธ์ระหว่างข้อมูล (Referential Integrity) สามารถควบคุมมาตรฐานของข้อมูลได้ ทั้งในลักษณะรูปแบบของข้อมูล (Format) การกำหนดรหัส (Coding)ในข้อมูลเรื่องเดียวกันให้เหมือนกัน

2.2.3.5. การจัดทำระบบฐานข้อมูล จะเป็นการวางแผนระบบข้อมูลขององค์กร หรือหน่วยงานอย่างมีประสิทธิภาพ ลดความสูญเสียและความขัดแย้งของข้อมูลที่อาจจะมีขึ้น ถ้าแต่ละแผนกแยกกันพัฒนาระบบข้อมูลของตนเอง

2.2.3.6. สามารถควบคุมและรักษาความปลอดภัยของข้อมูลได้ เนื่องจากข้อมูลต่างๆ ถูกนำเข้ามาจัดเก็บในระบบฐานข้อมูลซึ่งอยู่ที่ส่วนกลาง มีผู้ดูแลข้อมูลอย่างชัดเจน ผู้บริหารระบบฐานข้อมูล (Database Administration) ก็จะสามารถควบคุมการเข้าใช้ การแก้ไขข้อมูลของผู้เข้าใช้ทุกคน

2.2.3.7. ทำให้มีความเป็นอิสระในการจัดการฐานข้อมูล ถ้าต้องการเปลี่ยนแปลงวิธีการจัดเก็บหรือการเรียกใช้ข้อมูล การประยุกต์ใช้ทำได้ง่าย

2.2.4 ชุดคำสั่งที่ใช้จัดการกับฐานข้อมูล

แบ่งออกเป็น 2 ประเภทหลักๆคือ

2.2.4.1 จัดการกับโครงสร้างฐานข้อมูล เป็นการสร้าง ลบ หรือแก้ไขฐานข้อมูลและตาราง เช่น สร้างตารางฐานข้อมูลของพนักงานขึ้นมา ชุดคำสั่งต่างๆที่เกี่ยวกับการสร้างหรือเปลี่ยนแปลงโครงสร้างของฐานข้อมูลมีศัพท์เรียกว่า Data Definition Language หรือ DDL

2.2.4.2 จัดการกับส่วนของฐานข้อมูลที่มีอยู่ในตารางต่างๆในฐานข้อมูล เป็นการทำงานในเรื่องของการเพิ่ม การลบ หรือการแก้ไขข้อมูลในตารางต่างๆเช่น การเพิ่มข้อมูลพนักงานเข้าใหม่เข้าไปในตารางข้อมูลพนักงาน คำสั่งประเภทนี้เรียกว่า Data Management Language หรือ DML

2.2.5 วิธีติดต่อกับฐานข้อมูลของ Visual Basic

โปรแกรม Visual Basic มีความสามารถในการจัดการกับฐานข้อมูลได้โดยจะมีวิธีในการจัดการอยู่ 2 แบบ ดังนี้

2.2.5.1 การใช้ Data Control เป็นวิธีการที่ง่ายและมีความสะดวกที่สุดในการที่จะทำการติดต่อกับฐานข้อมูลเนื่องจาก Data Control จะทำการติดต่อกับฐานข้อมูลและจัดการกับฐานข้อมูลในตารางโดยอัตโนมัติ เช่นการเปิดฐานข้อมูล การแสดงและแก้ไขข้อมูลในตาราง อย่างไรก็ดีการใช้ Data Control ยังมีข้อจำกัดอยู่พอสมควร เช่น ไม่มีฟังก์ชันในการลบข้อมูล

2.2.5.2 การใช้ Data Object วิธีนี้จะต้องเขียนโปรแกรมเพื่อทำการติดต่อกับฐานข้อมูลโดยการใช้ Object ต่างๆที่ Visual Basic 2010 มีมาให้โดยการเพิ่ม ลบ หรือการแก้ไขข้อมูลจะต้องทำการเขียนโปรแกรมเอง แต่ข้อดีของวิธีนี้คือสามารถควบคุมความผิดพลาดต่างๆได้ดีกว่าใช้ Data Control รวมทั้งสามารถใช้ภาษา SQL เพื่อจัดการกับฐานข้อมูลได้อีกด้วย

2.3 ทฤษฎีการสื่อสารข้อมูล (Data Communication)

2.3.1 บทนำ

วิวัฒนาการของการสื่อสารข้อมูลและเครือข่ายคอมพิวเตอร์นั้น ทำให้ชีวิตของมนุษย์ในปัจจุบันได้เปลี่ยนแปลงไปเป็นอย่างมาก เช่น ในธุรกิจที่มีการแข่งขันกันสูง ดังนั้นข้อมูลที่รวดเร็วและทันการถือว่าเป็นหัวใจหลักอย่างหนึ่งในการตัดสินใจ จากในอดีตที่เราต้องส่งข้อมูลกันด้วยวิธีเก่าๆซึ่งอาจจะต้องใช้เวลานานกว่าข้อมูลจะถึงปลายทาง แต่ปัจจุบันนี้เราสามารถส่งข้อมูลกันได้อย่างรวดเร็ว ทำให้การตัดสินใจต่างๆสะดวกและมีความถูกต้องมากยิ่งขึ้น ดังนั้นเราจึงต้องมาศึกษากันในเรื่องของเทคโนโลยีที่จะช่วยในการสื่อสารข้อมูล เพื่อที่จะทำให้เราได้เข้าใจและสามารถนำไปใช้ให้เป็นประโยชน์ต่อไป

2.3.2 การสื่อสารข้อมูล

การสื่อสารข้อมูล เป็นการแลกเปลี่ยนข่าวสารหรือข้อมูลกันระหว่างอุปกรณ์ที่ใช้ในการสื่อสารโดยต้องมีสื่อในการโอนถ่ายข้อมูลกัน เช่น สายทองแดงหรือดาวเทียม เป็นต้น การที่เราจะสามารถสื่อสารข้อมูลกันได้นั้นจะต้องอาศัยทั้งฮาร์ดแวร์และซอฟท์แวร์โดยที่คุณสมบัติพื้นฐานของการสื่อสารข้อมูลนั้นจะประกอบไปด้วย

2.3.2.1 **ความถูกต้องของการส่งข้อมูล (Delivery)** หมายถึง ข้อมูลจะต้องสามารถไปถึงปลายทางได้อย่างถูกต้อง

2.3.2.2 **ความถูกต้องของข้อมูล (Accuracy)** หมายถึง ข้อมูลที่ไปถึงปลายทางนั้นจะต้องเหมือนกับที่ต้นทางที่ส่งไป

2.3.2.3 **เวลาที่เหมาะสม (Timeliness)** หมายถึง เวลาที่ใช้ในการเดินทางของข้อมูลนั้นต้องมีระยะเวลาที่เหมาะสมไม่นานจนเกินไป

2.3.3 องค์ประกอบของการสื่อสาร

- **ข้อมูลข่าวสาร (Message)** : สามารถเป็นได้ทั้งข้อความ ตัวเลข ภาพ เสียง หรือวิดีโอ เป็นต้น
- **ผู้ส่ง (Sender)** : เป็นอุปกรณ์ที่สามารถใช้ในการส่งข้อมูลได้ เช่น คอมพิวเตอร์ โทรศัพท์ กล้อง วิดีโอ เป็นต้น
- **ผู้รับ (Receiver)** : เป็นอุปกรณ์ที่ใช้สำหรับสามารถใช้ในการส่งข้อมูลได้ เช่น คอมพิวเตอร์ โทรศัพท์ กล้อง วิดีโอ เป็นต้น
- **สื่อที่ใช้ในการส่ง (Medium)** : เป็นสื่อกลางที่ใช้ในการนำส่งข้อมูลระหว่างผู้รับกับผู้ส่ง เช่น สายคู่ตีเกลียว สายโคแอ็กเชียล สายไฟเบอร์ออปติก ดาวเทียม เป็นต้น

- **โปรโตคอล (Protocol)** : เป็นกฎหรือข้อกำหนดของการสื่อสาร โดยก่อนที่ผู้ส่งและผู้รับจะสามารถติดต่อกันได้นั้น จะต้องสร้างข้อตกลงของการสื่อสารกันก่อน เพื่อที่จะให้สามารถที่จะเข้าใจกันได้ เช่น สมมติว่าคนไทยต้องการพูดคุยกับคนต่างชาติ จะต้องมีการตกลงกันก่อนว่าในการสื่อสารกันนั้นจะใช้ภาษาอะไร เพื่อที่จะให้พูดจาเป็นภาษาเดียวกันและเข้าใจตรงกัน

2.3.3.1 ทิศทางการติดต่อสื่อสารข้อมูล

- **แบบทิศทางเดียว (Simplex)** เป็นทิศทางการสื่อสารข้อมูลแบบที่ข้อมูลจะถูกส่งจากทิศทางหนึ่งไปยังอีกทิศทาง โดยไม่สามารถส่งข้อมูลย้อนกลับมาได้ เช่นระบบวิทยุ หรือโทรทัศน์

- **แบบกึ่งสองทิศทาง (Half Duplex)** เป็นทิศทางการสื่อสารข้อมูลแบบที่ข้อมูลสามารถส่งกลับกันได้ 2 ทิศทาง แต่จะไม่สามารถส่งพร้อมกันได้ โดยต้องผลัดกันส่งครั้งละทิศทางเท่านั้น เช่น วิทยุสื่อสารแบบผลัดกันพูด

- **แบบสองทิศทาง (Full Duplex)** เป็นทิศทางการสื่อสารข้อมูลแบบที่ข้อมูลสามารถส่งพร้อม ๆ กันได้ทั้ง 2 ทิศทาง ในเวลาเดียวกัน เช่น ระบบโทรศัพท์

2.3.4 โปรโตคอล

ในระบบเครือข่ายคอมพิวเตอร์นั้นจะมีการรับส่งข้อมูลกันอยู่ตลอดเวลา ดังนั้นสิ่งใดก็ตามที่สามารถรับส่งข้อมูลข่าวสารภายในเครือข่ายคอมพิวเตอร์กันได้ จะเรียกว่า "เอนติตี้" หรือว่าสามารถกล่าวอีกนัยหนึ่งได้ว่าการสื่อสารข้อมูลของเครือข่ายคอมพิวเตอร์ก็คือการรับส่งข้อมูลกันระหว่างเอนติตี้และการที่แต่ละเอนติตี้จะสามารถรับส่งข้อมูลกันได้จะต้องมีโพรโตคอลโดยที่โพรโตคอลจะหมายถึง ข้อกำหนดหรือกฎของการสื่อสารข้อมูล ดังนั้นเอนติตี้จะต้องปฏิบัติตาม
โพรโตคอลเสมอ เช่น การรับส่งข้อมูลจะต้องทำอย่างไรบ้าง หรือเมื่อไหร่จึงจะทำการรับส่งข้อมูลกันได้ เป็นต้น องค์ประกอบหลักของโพรโตคอล จะประกอบไปด้วย

- Syntax หมายถึงรูปแบบ (format) หรือโครงสร้าง (Structure) ของข้อมูล เช่น กำหนดว่าใน 8 บิทแรกจะหมายถึงแอดเดรส (Address) ของผู้ส่งอีก 8 บิทถัดมาแอดเดรสของผู้รับ ส่วนที่เหลือจึงจะเป็นข้อมูลจริงๆถ้าไม่มีการกำหนด syntax แล้วเอนติตี้จะไม่สามารถทราบได้เลยว่าบิทแต่ละบิทที่ได้รับมานั้นคืออะไร

- Semantics หมายถึง ความหมายของข้อมูลที่ได้รับมา เช่น เมื่อได้รับข้อมูลมาแล้ว เอนติตี้รู้ syntax แล้ว แต่จะยังไม่รู้ว่าบิทแต่ละบิทนั้นทำอะไรได้บ้าง ดังนั้นจึงต้องมาทำการแปลความหมายของบิทเหล่านั้นเสียก่อน เช่น เมื่อทราบแอดเดรสของผู้รับแล้ว เอนติตี้จะสามารถทำการหาเส้นทางที่สั้นที่สุดในการส่งข้อมูลได้

- Timing เป็นข้อกำหนดของเวลาในการรับส่งข้อมูล เนื่องจากเอนติตี้แต่ละตัวนั้นมีความเร็วในการรับส่งที่ไม่เท่ากัน เช่น ตัวหนึ่งมีความเร็วของการส่ง 100 Mbps แต่อีกตัวมีความเร็วในการรับแค่ 1 Mbps ถ้าไม่มีโพรโตคอลแล้วข้อมูลโดยส่วนใหญ่จะหายไป เนื่องจากเอนติตี้ที่ทำงานช้ากว่าจะไม่สามารถรับข้อมูลได้ทัน

2.3.5 สัญญาณที่ใช้ในการสื่อสาร

สัญญาณที่ใช้ในระบบสื่อสารแบ่งออกได้เป็น 2 ประเภทคือสัญญาณอนาลอก(Analog)และสัญญาณดิจิตอล(Digital)สัญญาณอนาลอกเป็นสัญญาณที่มีขนาดเป็นค่าต่อเนื่องส่วนสัญญาณดิจิตอลเป็น

สัญญาณที่มีขนาดเปลี่ยนแปลง เป็นค่าของเลขลงตัว โดยปกติมักแทนด้วย ระดับแรงดันที่แสดงสถานะเป็น "0" และ "1" หรืออาจจะมีหลายสถานะ ซึ่งจะกล่าวถึงในเรื่องระบบสื่อสารดิจิตอล มีค่าที่ตั้งไว้ (threshold) เป็นค่าบอกสถานะ ถ้าสูงเกินค่าที่ตั้งไว้สถานะเป็น "1" ถ้าต่ำกว่าค่าที่ตั้งไว้ สถานะเป็น "0" ซึ่งมีข้อดีในการทำให้เกิดความผิดพลาดน้อยลง

เนื่องจากสัญญาณรบกวนต้องมีค่าสูงกว่าค่าที่ตั้งไว้สถานะจึงจะเปลี่ยน ตัวอย่างเช่น ในระบบดิจิตอลสถานะของข้อมูลเป็น "0" สัญญาณรบกวนมีค่า 0.2 โวลต์ แต่ค่าที่ตั้งไว้เท่ากับ 0.5 โวลต์ สถานะยังคงเดิมคือเป็น "0" ในขณะที่ระบบอนาลอก สัญญาณรบกวนจะเติมเข้าไปใน สัญญาณจริงโดยตรง กล่าวคือ สัญญาณจริงบวกสัญญาณรบกวนเป็นสัญญาณขณะนั้นถ้าให้สัญญาณรบกวนมีผลต่อสัญญาณ จริงและมีความผิดพลาดเกิดขึ้น

กระแสไฟฟ้าแบ่งออกได้เป็นไฟฟ้ากระแสตรงและกระแสสลับ หลายๆคนอาจจะคิดว่าไม่มีส่วนเกี่ยวข้องกับระบบสื่อสารโทรคมนาคม เมื่อกล่าวถึงสัญญาณในเชิงประยุกต์ก็อาจจะจำแนกในหมวดหมู่นี้ได้ การไหลของไฟฟ้ากระแสตรงในวงจรอย่างสม่ำเสมอไม่สามารถส่งข่าวสารได้ แต่เมื่อไรที่ถ้าการควบคุมกระแสให้เป็นพัลส์โดยการเปิดสวิตช์กระแสจะลดลงสู่ศูนย์และปิดสวิตช์ กระแสก็จะมีค่าค่าหนึ่ง พัลส์ของกระแสถูกผลิตตามรหัสที่ใช้แทนแต่ละตัวอักษรหรือตัวเลข โดยการรวมของพัลส์ การทำงานของสวิตช์สามารถส่งข้อความใดๆได้ ตัวอย่างที่เห็นได้เสมอได้แก่ รหัสมอร์ส เป็นต้น ส่วนไฟฟ้ากระแสสลับในรูปของคลื่นอยู่ในจำพวกคลื่นวิทยุมีการใช้งานอย่างกว้างขวางเป็นที่รู้จักกันดี

2.3.5.1. สัญญาณแบบอนาลอก

จะเป็นสัญญาณแบบต่อเนื่องที่ทุกๆค่าเปลี่ยนแปลงไปของระดับสัญญาณจะมีความหมาย การส่งสัญญาณแบบอนาลอก จะถูกรบกวนให้มีการแปลความหมายผิดพลาดได้ง่ายกว่าเนื่องจากค่าทุกค่าที่ถูกนำมาใช้งานนั้นเอง ซึ่งสัญญาณแบบอนาลอกนี้จะเป็นสัญญาณที่สื่อกลาง ในการสื่อสารส่วนมากใช้อยู่ เช่น สัญญาณเสียงในสายโทรศัพท์ เป็นต้น

2.3.5.2. สัญญาณแบบดิจิตอล

จะประกอบขึ้นจากระดับสัญญาณเพียง 2 ค่า คือสัญญาณระดับสูงสุดและสัญญาระดับต่ำสุด ดังนั้นจะมีประสิทธิภาพและ ความน่าเชื่อถือสูงกว่าแบบอนาลอก เนื่องจากมีการใช้งานเพียง 2 ค่าเพื่อนำมาตีความหมายเป็น On/Off หรือ 1/0 เท่านั้นซึ่งสัญญาณดิจิตอลนี้ จะเป็นสัญญาณที่คอมพิวเตอร์ใช้ในการทำงานและติดต่อสื่อสารกัน

ในทางปฏิบัติ จะสามารถใช้เครื่องมือในการแปลงระหว่างสัญญาณ ทั้งสองแบบได้เพื่อช่วยให้สามารถส่งสัญญาณดิจิตอลผ่านสัญญาณพาหะที่เป็นอนาล็อก เช่น สายโทรศัพท์หรือคลื่นวิทยุ การแปลงสัญญาณดิจิตอลเป็นอนาล็อก จะเรียกว่า โมดูเลชั่น (Modulation) เช่น การแปลงสัญญาณแบบ Amplitude modulation (AM) และ Frequency Modulation (FM) เป็นต้น ส่วนการแปลงสัญญาณ แบบอนาล็อกเป็นดิจิตอล จะเรียกว่า ดีโมดูเลชั่น (Demodulation) ตัวอย่างของเครื่องมือการแปลง เช่น MODEM(Modulation Demodulation) นั้นเอง

2.3.5.3 ลักษณะและความแตกต่างของอนาลอกและดิจิตอล

สัญญาณดิจิตอลมีลักษณะพิเศษที่แตกต่างจากสัญญาณอนาลอกเป็นอย่างมาก กล่าวคือ สัญญาณดิจิตอลเป็นสัญญาณที่ไม่ต่อเนื่อง แต่มีลักษณะเป็นขั้นๆหรือเป็นท่อนๆซึ่งเรามีศัพท์เรียกลักษณะดังกล่าวว่า ดิสครีต (Discrete) และสัญญาณจะมีค่าได้เฉพาะค่าที่กำหนดไว้ตายตัวเท่านั้น เช่น ใน

ระบบดิจิตอลที่ใช้กันทั่วไปนั้นใช้สัญญาณระบบเลขฐานสอง (Binary) ซึ่งมีค่าเพียงสองระดับเท่านั้น จากรูปแสดงลักษณะของสัญญาณดิจิตอล ในกรณีนี้สัญญาณสามารถมีค่าที่กำหนดไว้สองระดับคือ 0 กับ 5 โวลต์ ตามรูปนี้สัญญาณมีค่า 5 โวลต์ ในช่วงเวลาระหว่า t = 1 ถึง 2 กับในช่วงเวลาระหว่าง t = 3 ถึง 4 นอกจากช่วงเวลา ทั้งสองดังกล่าวสัญญาณมีค่า 0 โวลต์

ถ้าส่งสัญญาณนี้ไปตามสายหรือไปกับคลื่นวิทยุ เมื่อไปถึงปลายทางขนาดและรูปร่างของคลื่น อาจผิดเพี้ยนไปบ้าง ในทางปฏิบัติเรากำหนดค่าขั้นต่ำของสัญญาณระดับสูง (VH min) และค่าขั้นสูงของสัญญาณระดับต่ำ (VL max) ไว้ เพื่อให้สามารถแยกระดับแรงดันไฟฟ้า ทั้งสองจากสัญญาณดิจิตอลที่ผิดเพี้ยนไปได้ โดยวิธีนี้ระบบที่รับสัญญาณดิจิตอลก็ยังสามารถทำงานได้อย่างถูกต้อง แม้ว่าสัญญาณที่ รับเข้ามาจะมีความผิดเพี้ยนค่อนข้างมาก ซึ่งเป็นจุดเด่นของระบบดิจิตอลที่เหนือกว่าระบบอนาลอก

การเข้าใจความแตกต่างระหว่างสัญญาณอนาลอกและดิจิตอล ถือว่ามีความสำคัญมากในการที่จะเข้าใจการสื่อสารข้อมูล สัญญาณอนาลอกเป็นสัญญาณที่มีความต่อเนื่อง และมีค่าตลอดช่วงของสัญญาณ เช่น เสียงพูด, เสียงดนตรี, วีดีโอ บางครั้งเรียกว่าบอร์ดแบนด์ หรือ สัญญาณมอดูเลท สัญญาณดิจิตอลเป็นกลุ่มของสัญญาณที่มีค่าไม่ต่อเนื่อง ภายในช่วงสัญญาณมีรูปแบบเป็นสัญญาณและเป็นเลขฐานสอง คือ มีค่า 2 ค่า เป็น 1 และ 0 สัญญาณดิจิตอลอาจจะ
เรียกว่าเบสแบนด์

2.3.5.4 ประเภทของสัญญาณอนาลอกและดิจิตอล

คงเคยได้ยินคำว่า "อนาลอกและดิจิตอล" มาบ้างแล้ว ข้อมูลต่างๆที่เดินทางผ่านช่องสัญญาณในระบบโทรคมนาคมมีอยู่สองแบบ คือ สัญญาณอนาลอกกับสัญญาณดิจิตอล ลักษณะของสัญญาณอนาลอก (Analog Signal) อยู่ในรูปคลื่นแม่เหล็กไฟฟ้าที่เกิดขึ้นอย่างต่อเนื่องและถูกส่งผ่านสื่อนำสัญญาณไปยังจุดหมายที่ต้องการ เหมาะสำหรับการส่งสัญญาณเสียงสนทนา (Voice) ส่วนสัญญาณดิจิตอล (Digital Signal) มีลักษณะการแบ่งสัญญาณเป็นช่วงๆ อย่างไม่ต่อเนื่อง ออกเป็นสองระดับเพื่อแทนสถานะสองสถานะ คือ สถานะของบิท "0" กับสถานะของบิท "1" ซึ่งเปรียบได้กับการปิดเปิดสวิทช์ไฟฟ้า (0 หมายถึง ปิด 1 หมายถึง เปิด) เครื่องมืออิเลกทรอนิกส์ส่วนใหญ่ในปัจจุบันผลิตสัญญาณออกมาในรูปแบบสัญญาณดิจิตอลเกือบทั้งสิ้น แต่การส่งสัญญาณผ่านเครือข่ายโทรคมนาคมในปัจจุบัน ยังอยู่บน พื้นฐานของสายโทรศัพท์ธรรมดาซึ่งเป็นสัญญาณอนาลอกอยู่ ดังนั้นจึงต้องมีการแปลงสัญญาณผ่านอุปกรณ์ที่เรียกว่าโมเด็ม (Modulation/Demoldulation device ; MODEM) จึงจะสามารถสื่อสารข้อมูลในระยะไกลได้ แต่ในขณะเดียวกันทุกวันนี้ในองค์กรต่างๆ สามารถเชื่อมโยงข้อมูลระหว่างเครื่องมืออิเลกทรอนิกส์ในองค์กรได้ เช่น ระหว่างห้องทำงาน ระหว่างชั้น ระหว่างตึก การเดินสายสัญญาณ สามารถเลือกใช้สายที่นำสัญญาณดิจิตอลได้เลยก็ไม่จำเป็นต้องอาศัยโมเด็มช่วยแปลงสัญญาณอีก เช่น การสื่อสารในระบบ LAN

2.3.5.5 ข้อดีและข้อเสียของระบบอนาลอกและดิจิตอล

การพัฒนาเทคโนโลยีของไอซี ทำให้มีการพัฒนาทางด้านดิจิตอล และนำมาใช้ในเครื่องมือและอุปกรณ์ต่าง ๆ มากขึ้น อุปกรณ์ดิจิตอล มีข้อดีหลายประการดังต่อไปนี้

2.3.5.5.1. การแสดงผลทำให้เข้าใจได้ง่าย ตัวอย่างเช่น การแสดงผลของแรงดันไฟฟ้าเป็นตัวเลขจาก เครื่องวัดแรงดันไฟฟ้า

2.3.5.5.2. การควบคุมทำได้ง่าย ตัวอย่างเช่นระบบควบคุมอุณหภูมิของเตาเผาที่มีระบบดิจิตอลเข้ามาเกี่ยวข้อง การทำงานของระบบ มีตัวตรวจอุณหภูมิที่เปลี่ยนอุณหภูมิเป็นระดับแรงดันที่

เป็นสัญญาณ อนาลอก สัญญาณจะถูกเปลี่ยนเป็นสัญญาณดิจิตอล ด้วยวงจรเปลี่ยนสัญญาณอนาลอกเป็นดิจิตอล แล้วป้อนเข้าสู่ส่วนประมวลผล (Central Processing Unit : CPU) ซี พี ยู จะทำงานตามเงื่อนไขที่กำหนดไว้ ถ้ามีอุณหภูมิสูงหรือต่ำกว่าที่กำหนด จะส่งสัญญาณออกที่เอาท์พุท เพื่อควบคุมการปิดเปิดเชื้อเพลิงใน เตาเผา การเปลี่ยนสัญญาณดิจิตอลให้กลับมาเป็นสัญญาณอนาลอกใช้วงจร D/A คอนเวอร์เตอร์ (Digital to Analog Converter) สัญญาณอนาลอกจะไปควบคุมการปิดเปิด การฉีดน้ำมันเชื้อเพลิงในเตาเผา เพื่อให้ได้อุณหภูมิตามที่ตั้งไว้ การเปลี่ยนอุณหภูมิสามารถปรับได้ โดยการเปลี่ยนค่าที่เก็บไว้ใน ซี พี ยู

2.3.5.5.3. ความเที่ยงตรง วงจรอนาลอกทำให้มีความเที่ยงตรงสูงได้ยาก เพราะประกอบไปด้วยอุปกรณ์ที่มีค่าผิดพลาด และมีความไวต่อสิ่งแวดล้อม เช่น อุณหภูมิ ความชื้น จึงทำให้อุปกรณ์ต่างๆ เช่น ตัวต้านทาน ตัวเก็บประจุ มีคุณสมบัติเปลี่ยนไป เหมือนกับว่าปัญหาที่เกิดขึ้นในวงจรอนาลอก เป็นเพราะแรงดันไฟฟ้า ส่วนอุปกรณ์ในวงจรดิจิตอลก็มีปัญหาเช่นเดียวกัน แต่วงจรสามารถควบคุมการทำงานได้ ถึงแม้ว่าสัญญาณจะผิดเพี้ยนไปบ้าง ก็ไม่มีผลต่อการทำงานของวงจรเพราะสภาวะ 1 กับ 0 กำหนดจากระดับแรงดัน

2.3.5.5.4. ผลกระทบต่อการส่งในระยะไกล เมื่อมีการส่งสัญญาณออกไปในระยะไกล ๆ ตามสายส่งหรือเป็นคลื่นวิทยุ จะมีการรบกวนเกิดขึ้นได้ง่าย เรียกว่า นอยส์ (noise) ตัวอย่างเช่น การส่งสัญญาณไปยัง ดาวเทียมจะมีการรบกวนเนื่องจากการแผ่รังสี จากฟ้าแลบ หรือจุดดับบนดวงอาทิตย์ทำให้สัญญาณผิดเพี้ยนได้ง่าย ถ้าเป็นวงจรอนาลอกความเชื่อถือได้ขึ้นกับแรงดันที่ปลายทางว่าเบี่ยงเบนไปจากต้นทางมามากน้อยแค่ไหน เป็นปัญหาที่เกี่ยวกับความต่างศักย์ ถ้าส่งเป็นสัญญาณดิจิตอลจะไม่มีปัญหานี้ เพราะสัญญาณอาจผิดไปจากต้นทางได้บ้างแต่ยังคงสภาวะ 1 หรือ 0

2.3.5.5.5. ความเชื่อมั่น สัญญาณดิจิตอลมีค่าความผิดพลาดน้อยกว่าสัญญาณอนาลอก ทำให้วงจรที่ทำงานด้วยสัญญาณดิจิตอล มีความเชื่อถือได้มากกว่า เมื่อใช้แทนปริมาณต่าง ๆ ตัวอย่างเช่น การบวกสัญญาณ ถ้าทำงานในลักษณะอนาลอก

2.3.6 การส่งผ่านข้อมูล

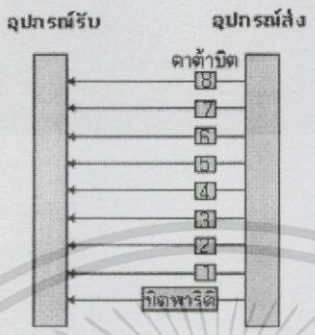
ในการส่งข้อมูลกันระหว่างคอมพิวเตอร์กับอุปกรณ์ภายนอกโดยใช้สายส่งสามารถทำได้ 2 วิธี คือ การส่งข้อมูลแบบ(Parallel Transmission) และการส่งข้อมูลแบบอนุกรม(Serial Transmission) โดยที่การส่งข้อมูลแบบขนานนั้น จะส่งข้อมูลไปคราวละหลายๆบิทพร้อมๆกัน ส่วนการส่งข้อมูลแบบอนุกรมนั้นจะส่งออกไปครั้งละบิทต่อเนื่องกัน นอกจากนั้นแล้วการส่งข้อมูลแบบอนุกรมยังสามารถแบ่งออกได้เป็น แบบซิงโครนัส(Synchronous) และอะซิงโครนัส(Asynchronous)

2.3.6.1 การส่งข้อมูลแบบขนาน (Parallel Transmission)

เป็นการส่งข้อมูลออกไปพร้อมๆกันได้คราวละหลายบิท หรือพูดอีกอย่างได้ว่า ณ เวลาใดๆสามารถที่จะส่งบิตข้อมูลจำนวน n บิทออกไปได้พร้อมกัน แต่วิธีการแบบนี้ถ้าต้องการที่จะส่งข้อมูลจำนวน n บิทแล้วจะต้องใช้สายจำนวน n สายด้วย

ข้อดีของวิธีการส่งแบบนี้คือ สามารถส่งข้อมูลได้เร็ว เนื่องจากจำนวนของข้อมูลที่ส่งออกไปได้ในแต่ละครั้งนั้นมีมาก ส่วนข้อเสียคือ ค่าใช้จ่ายจะสูง เพราะจะต้องใช้จำนวนของสายมาก นอกจากนั้นแล้วการส่งข้อมูลแบบขนานนี้จะเหมาะกับการส่งข้อมูลในระยะทางสั้นๆเท่านั้น เนื่องจากถ้าส่งไป

ในระยะทางไกลแล้วข้อมูลภายในสายอาจจะไปถึงปลายทางไม่พร้อมกัน ทำให้เกิดความผิดพลาดของการส่งข้อมูลได้



การถ่ายโอนข้อมูลแบบขนาน

2.3.6.2 การส่งข้อมูลแบบอนุกรม (Serial Transmission)

เป็นการส่งข้อมูลที่ใช้สายส่งเพียงเส้นเดียว ดังนั้นบิทของข้อมูลที่ถูกส่งออกจะเรียงตามลำดับกันไป โดยปกติแล้วการส่งข้อมูลกันภายในเครื่องคอมพิวเตอร์จะใช้วิธีการส่งข้อมูลแบบขนาน แต่การส่งข้อมูลกับอุปกรณ์ภายนอกแล้วอาจจะเป็นการส่งข้อมูลแบบอนุกรมก็ได้ ดังนั้น
ถ้าการติดต่อกับอุปกรณ์ภายนอกเครื่องคอมพิวเตอร์ต้องใช้การส่งข้อมูลแบบอนุกรมแล้ว จะต้องทำการแปลงข้อมูลจากแบบขนานไปเป็นแบบอนุกรม(parallel-to-serial) เสียก่อน ส่วนเครื่องคอมพิวเตอร์ฝั่งรับก็จะต้องแปลงข้อมูลแบบอนุกรมจากสายส่งไปเป็นแบบขนานด้วย(serial-to-parallel)

2.3.6.2.1 **แบบอะซิงโครนัส** (Asynchronous Transmission)

เป็นการส่งข้อมูลครั้งละไบต์ (1 byte = 8 bit) การส่งแบบอะซิงโครนัสนี้ไม่ต้องอาศัยสัญญาณนาฬิกาในการควบคุมจังหวะของการส่งข้อมูล ดังนั้นฝ่ายรับข้อมูลจึงไม่สามารถทราบเวลาที่แน่นอนของข้อมูลที่จะเข้ามาได้ ดังนั้นในแต่ละไบต์ที่ถูกส่งออกไปจึงต้องมีบิทเริ่มต้น ซึ่งจะกำหนดค่าเป็น “0” และบิทสิ้นสุด ซึ่งจะกำหนดค่าเป็น “1”

การที่ต้องมีการเพิ่มบิทเริ่มต้นและบิทสิ้นสุดเข้าไปด้วยนั้น เพื่อที่จะให้ฝ่ายรับข้อมูลทราบได้ว่าข้อมูลในแต่ละไบท์เริ่มต้นที่ใด และสิ้นสุดตรงไหน การส่งข้อมูลออกไปแต่ละครั้งนั้นจะส่งออกไปเป็นไบท์ และไม่จำเป็นว่าทุกๆไบท์จะต้องส่งออกไปอย่างต่อเนื่องกันอาจจะมีช่องว่างเกิดขึ้นระหว่างไบท์ของข้อมูลก็ได้

แสดงให้เห็นถึงไบท์ของข้อมูลที่ต้องมีบิทเริ่มต้นและบิทสิ้นสุด โดยแต่ละไบท์นั้นมีช่องว่างเกิดขึ้นระหว่างไบท์ของข้อมูลด้วย เนื่องจากการส่งข้อมูลแบบอะซิงโครนัสไม่ต้องอาศัยสัญญาณใดๆคอบควบคุมการส่งข้อมูล ดังนั้นวิธีนี้จึงเหมาะกับอุปกรณ์ที่มีความเร็วต่ำ เช่น คีย์บอร์ด เป็นต้น

2.3.6.2.2 **แบบซิงโครนัส** (Synchronous Transmission)

วิธีการส่งข้อมูลแบบนี้ไม่ต้องอาศัยบิทเริ่มต้น และบิทสิ้นสุด อีกทั้งยังสามารถส่งข้อมูลได้หลายๆไบท์ต่อเนื่องกันเป็นเฟรมได้โดยไม่จำเป็นที่จะต้องมีช่องว่างระหว่างไบท์ของข้อมูล ฝ่ายรับข้อมูลจะต้องทำหน้าที่ในการแยกเฟรมข้อมูลที่ได้มาบางออกเป็นไบท์เอง เนื่องจากไม่มีบิทเริ่มต้นและ

บิทสิ้นสุด ดังนั้นฝ่ายรับข้อมูลจึงต้องมีการนับจำนวนบิทของข้อมูลที่เข้ามาให้ได้อย่างถูกต้องและแม่นยำ การส่งแบบซิงโครนัสจะมีความเร็วของการรับส่งข้อมูลได้เร็วกว่าแบบอะซิงโครนัส เนื่องจากไม่ต้องมีบิทพิเศษเพิ่มเติมเข้ามา อีกทั้งยังไม่ต้องมีช่องว่างเกิดขึ้นระหว่างไบท์ของข้อมูลอีกด้วย ดังนั้นการส่งข้อมูลแบบนี้จึงเหมาะกับอุปกรณ์ที่ต้องการความเร็วสูง เช่น การส่งข้อมูลกันระหว่างคอมพิวเตอร์ เป็นต้น

2.3.7 ปัจจัยที่มีผลต่อการส่งสัญญาณ

ในการส่งสัญญาณโดยผ่านสื่อตัวกลางชนิดต่างๆกันนั้น มีปัจจัยอยู่หลายประการในระหว่างการส่งที่สามารถส่งผลกระทบทำให้สัญญาณที่ปลายทางไม่เหมือนกับที่ต้นทางได้ ซึ่งปัจจัยเหล่านี้จะประกอบไปด้วยความเบาบางของสัญญาณ การบิดเบี้ยวของสัญญาณ และสัญญาณรบกวน

- **ความเบาบางของสัญญาณ** เกิดจากการสูญเสียพลังงานในระหว่างการส่ง เนื่องจากสื่อตัวกลางชนิดต่างๆจะมีความต้านทานอยู่ในตัวเอง ดังนั้นเมื่อสัญญาณต้องผ่านสื่อเหล่านี้พลังงานไฟฟ้าจะถูกเปลี่ยนไปเป็นพลังงานความร้อน จึงทำให้เกิดการเบาบางของสัญญาณ ยิ่งส่งไปในระยะทางที่ไกลเท่าไหร่สัญญาณจะเกิดการเบาบางลงเท่านั้น ดังนั้นจึงต้องมีตัวขยายสัญญาณเพื่อทำหน้าที่ในการขยายสัญญาณให้กลับไปเป็นเหมือนเดิม

- **การบิดเบี้ยวของสัญญาณ** หมายถึง การที่สัญญาณมีลักษณะที่ผิดเพี้ยนไปจากเดิม ซึ่งเหตุการณ์แบบนี้จะเกิดขึ้นกับ composite signal เนื่องมาจากสัญญาณแบบนี้ผ่านตัวกลาง มีความเป็นไปได้ว่าอัตราเร็วในการแพร่สัญญาณของแต่ละสัญญาณนั้นจะไม่เท่ากัน บางสัญญาณอาจจะเดินทางได้เร็ว บางสัญญาณอาจจะช้ากว่า จึงทำให้เมื่อรวมสัญญาณที่ปลายทางแล้วจะเกิดการบิดเบี้ยวของสัญญาณขึ้น

- **สัญญาณรบกวน** เป็นอีกหนึ่งปัจจัยที่เป็นอุปสรรคในการส่งสัญญาณระหว่างต้นทางและปลายทาง

2.4 เครือข่าย (Network)

เครือข่าย หมายถึง กลุ่มของอุปกรณ์ที่ใช้สื่อสาร(ปกติเรียกว่า โหนด)ที่ทำการเชื่อมโยงกันโดยใช้สื่อในการรับส่งข้อมูล ตัวอย่างของกลุ่มอุปกรณ์ เช่น คอมพิวเตอร์ พรินเตอร์ หรืออุปกรณ์ใดๆก็ตามที่สามารถทำการรับส่งข้อมูลกับอุปกรณ์อื่นๆได้

2.4.1 ประเภทของการเชื่อมโยง

ในการเชื่อมโยงอุปกรณ์กันเป็นเครือข่ายนั้น จะต้องมีสื่อที่ใช้ในการรับส่งข้อมูล การเชื่อมโยงกันระหว่างอุปกรณ์จะสามารถทำได้ 2 แบบคือ การเชื่อมโยงแบบจุดต่อจุด และการเชื่อมโยงแบบหลายจุด

2.4.1.1 การเชื่อมโยงแบบจุดต่อจุด

เป็นการเชื่อมโยงที่ต้องใช้สื่อในการส่งข้อมูลเพียงสายเดียวระหว่างโหนด 2 โหนด ถ้ามีโหนดมากกว่านั้นจะต้องเพิ่มสื่อที่ใช้ในการส่งข้อมูลด้วย เนื่องจากไม่สามารถใช้สื่อร่วมกันได้ ดังนั้นสื่อที่ใช้สำหรับการเชื่อมต่อกันของโหนดจะต้องถูกจองอยู่ตลอดเวลา ถึงแม้ว่าจะไม่มีการรับส่งข้อมูลกันในขณะนั้นก็ตาม

2.4.1.2 การเชื่อมโยงแบบหลายจุด

หรือเรียกอีกชื่อว่า มัลติดรอป เป็นการเชื่อมโยงกันระหว่างโหนดหลายๆโหนดโดยใช้สื่อเพียงเส้นเดียว หรือเป็นการใช้สื่อในการส่งข้อมูลร่วมกันนั่นเอง การเชื่อมโยงแบบนี้จะสิ้นเปลืองน้อยกว่าแบบจุดต่อจุด

2.4.2 ประเภทของเครือข่าย

ในการแบ่งประเภทของเครือข่ายสามารถแบ่งได้หลายแบบ แต่ถ้าต้องการแบ่งประเภทตามขนาดของเครือข่ายแล้ว สามารถแบ่งออกได้เป็น 3 ประเภทดังนี้

2.4.2.1 แลน (Local Area Network : LAN)

เป็นเครือข่ายขนาดเล็กซึ่งเชื่อมโยงคอมพิวเตอร์และอุปกรณ์สื่อสารที่อยู่ในท้องที่บริเวณเดียวกันเข้าด้วยกัน เช่น ภายในอาคาร หรือภายในองค์การที่มีระยะทางไม่ไกลมากนัก เป็นต้น โดยคอมพิวเตอร์แต่ละเครื่องจะต่อเข้ากับอุปกรณ์เครือข่าย เช่น ฮับ สวิทช์ เป็นต้น ซึ่งอุปกรณ์เครือข่ายแต่ละตัวจะเชื่อมต่อกันโดยเข้าสายตีเกลียวคู่ สายใยแก้วนำแสงหรือคลื่นวิทยุ และเครือข่ายแลนจะเชื่อมต่อถึงกันด้วยอุปกรณ์จัดเส้นทาง (router) การสร้างเครือข่ายแลนนี้แต่ละองค์กร สามารถดำเนินการเองได้โดยการวางสายสัญญาณสื่อสารภายในอาคาร หรือภายในพื้นที่ของตนเอง เครือข่ายแลนมีตั้งแต่เครือข่ายขนาดเล็กที่เชื่อมโยงคอมพิวเตอร์ตั้งแต่สองเครื่องขึ้นไปภายในห้องเดียวกัน จนถึงเชื่อมโยงระหว่างห้องหรือองค์กรขนาดใหญ่ เช่น ภายในสำนักงาน ภายในโรงเรียนหรือมหาวิทยาลัย เป็นต้น ทำให้เครื่องคอมพิวเตอร์หลายๆ เครื่องที่เชื่อมต่อกัน สามารถส่งข้อมูลแลกเปลี่ยนกันได้อย่าง สะดวก รวดเร็ว และยังสามารถใช้ทรัพยากรร่วมกันได้อีกด้วย

2.4.2.2 แมน (Metropolitan Area Network : MAN)

เป็นเครือข่ายที่เชื่อมโยงแลนที่อยู่ห่างกัน เช่น ระหว่างสำนักงานที่อยู่คนละอาคาร ระบบเคเบิ้ลทีวีตามบ้านในปัจจุบัน เป็นต้นโดยมีลักษณะหารเชื่อมโยงคอมพิวเตอร์ที่มีระยะห่างไกลกันในช่วง 5-40 กิโลเมตร ผ่านสายสื่อสารประเภทสายใยแก้วนำแสง สายโคแอกเซียล หรืออาจใช้คลื่นไมโครเวฟ

2.4.2.3 แวน (Wide Area Network : WAN)

เป็นเครือข่ายคอมพิวเตอร์ขนาดใหญ่ที่เชื่อมโยงระบบคอมพิวเตอร์ในระยะห่างไกล มีการติดต่อสื่อสารกันในระยะห่างไกล มีการติดต่อสื่อสารในบริเวณกว้างเช่น เชื่อมโยงระหว่างจังหวัด ระหว่างประเทศ เป็นต้น

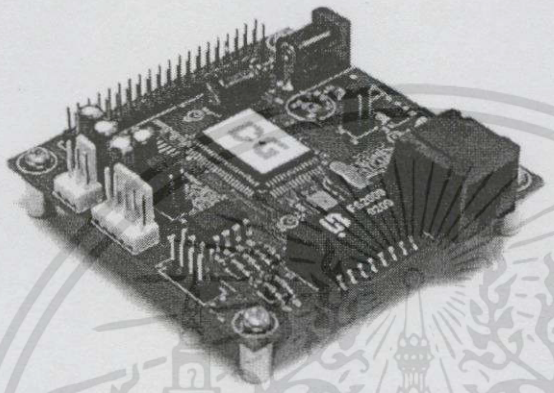
2.4.3 เครือข่ายอินเทอร์เน็ต

อินเทอร์เน็ต (Internet)คือ เครือข่ายของคอมพิวเตอร์ขนาดใหญ่ที่เชื่อมโยงเครือข่ายคอมพิวเตอร์ทั่วโลกเข้าด้วยกันโดยอาศัยเครือข่ายโทรคมนาคมเป็นตัวเชื่อมเครือข่ายภายใต้มาตรฐานการเชื่อมโยงด้วยโปรโตคอลเดียวกันคือ TCP/IP (Transmission Control Protocol / Internet Protocol) เพื่อให้คอมพิวเตอร์ทุกเครื่องในอินเทอร์เน็ตสามารถสื่อสารระหว่างกันได้ นับว่าเป็นเครือข่ายที่กว้างขวางที่สุดในปัจจุบัน เนื่องจากมีผู้นิยมใช้ โปรโตคอลอินเทอร์เน็ตจากทั่วโลกมากที่สุด

อินเทอร์เน็ตจึงมีรูปแบบคล้ายกับเครือข่ายคอมพิวเตอร์ระบบแวนแต่มีโครงสร้างการทำงานที่แตกต่างกันมากพอสมควร เนื่องจากระบบแวนเป็นเครือข่ายที่ถูกสร้างโดยองค์กรๆ เดียวหรือกลุ่มองค์กรเพื่อวัตถุประสงค์ด้านใดด้านหนึ่ง และมีผู้ดูแลระบบที่รับผิดชอบแน่นอน แต่อินเทอร์เน็ตจะเป็น

การเชื่อมโยงกันระหว่างคอมพิวเตอร์นับล้านๆ เครื่องแบบไม่ถาวรขึ้นอยู่กับเวลานั้นๆ ว่าใครต้องการเข้าสู่ระบบอินเทอร์เน็ตบ้าง ใครจะติดต่อสื่อสารกับใครก็ได้ จึงทำให้ระบบอินเทอร์เน็ตไม่มีผู้ใดรับผิดชอบหรือดูแลทั้งระบบ

2.5 Ethernet I/O Board



รูปที่ 2.5 อุปกรณ์สำหรับเชื่อมต่อผ่านระบบเครือข่าย

Ethernet I/O Board เป็นบอร์ด Embedded System ที่ใช้สำหรับควบคุมการใช้งานผ่านระบบเครือข่าย(LAN) ซึ่งเป็นอุปกรณ์แบบ Embedded Network Controller โดยสั่งการควบคุมจากเครื่องคอมพิวเตอร์ที่เชื่อมต่ออยู่กับ LAN วงเดียวกันกับ Ethernet I/O Board ในการรับส่งข้อมูลผ่าน LAN ซึ่งใช้โปรโตคอล TCP/IP Stack สามารถควบคุม GPIO และสามารถวัดสัญญาณอนาลอกมาเป็นดิจิตอลด้วย ADC และเปลี่ยนการรับส่งข้อมูลจาก RS232 เป็น LAN

2.5.1 การนำ Embedded Network Controller ไปใช้งาน

สามารถนำไปใช้งานได้ 3 แบบดังนี้

2.5.1.1 นำไปใช้เป็นตัวควบคุมฮาร์ดแวร์บนบอร์ด Embedded Network Controller โดยทำการเชื่อมต่อเข้ากับ LAN และเรียกใช้งานผ่าน PC ที่เชื่อมต่ออยู่ในวง LAN โดยจะใช้เทอร์มินอลต่างๆเช่น Hyperterminal , Terateam เป็นตัวควบคุมการรับส่งข้อมูลเพื่อนำไปใช้ในการควบคุมการทำงานของพอร์ทต่างๆ

2.5.1.2 ใช้เป็นโมดูลในการเปลี่ยนการเชื่อมต่อกับอุปกรณ์ที่มีการเชื่อมต่อแบบต่างๆ เช่น แปลงการเชื่อมต่อแบบ RS232 (Comport) ไปเป็นการรับส่งแบบ LAN หรือจะเป็น USB, Parallel ให้สามารถเชื่อมต่อผ่าน LAN ได้โดยไม่ต้องทำการเปลี่ยนแปลงอุปกรณ์ที่มีอยู่เดิม ซึ่งจะใช้ตัว Embedded Network Controller เป็นตัวแปลงการเชื่อมต่อดังกล่าวแทน

2.5.1.3 ใช้เป็นตัว Web Server คือเป็นการใช้งานตัว Embedded Network Controller ผ่าน Web Browser จากตัว Embedded Network Controller ทั้งในการควบคุมและการรับส่งข้อมูลระหว่าง PC กับ Embedded Network Controller โดยใช้ Web Browser เป็นตัวแสดงผลและสั่งการควบคุม

2.5.2 คุณสมบัติเด่นของ Ethernet I/O Board

- ลดระยะเวลาในการพัฒนาไดร์ฟเวอร์, ซอท์ฟแวร์ และในส่วนของการจัดการเกี่ยวกับ TCP/IP Protocol Stack

- สามารถควบคุมอุปกรณ์ผ่าน Ethernet ได้โดยไม่ต้องรู้ลึกเกี่ยวกับ Protocol Stack และ Socket Programming

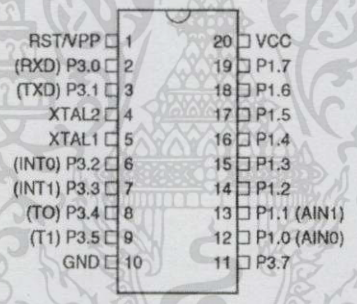
- สามารถเปลี่ยนการเชื่อมต่อแบบ RS232/RS485 เป็นการเชื่อมต่อแบบ LAN ได้เมื่อต้องการเปลี่ยนการเชื่อมต่อของอุปกรณ์จาก RS232 หรือ RS4585

- สามารถควบคุมลอจิกของ GPIO และอ่านค่า ADC จาก Ethernet I/O Board ผ่านแลนเน็ตเวิร์กได้โดยใช้โปรแกรม VB6.0 และ VC++6.0 เรียกใช้งานผ่าน DLL ของ Ethernet I/O

- สามารถใช้งานแบบ Virtual Comport

- อุปกรณ์มีขนาดเล็กและประหยัดไฟ

2.6 Microcontroller AT89C4051



รูปที่ 2.6 ไมโครคอนโทรลเลอร์ AT89C4051

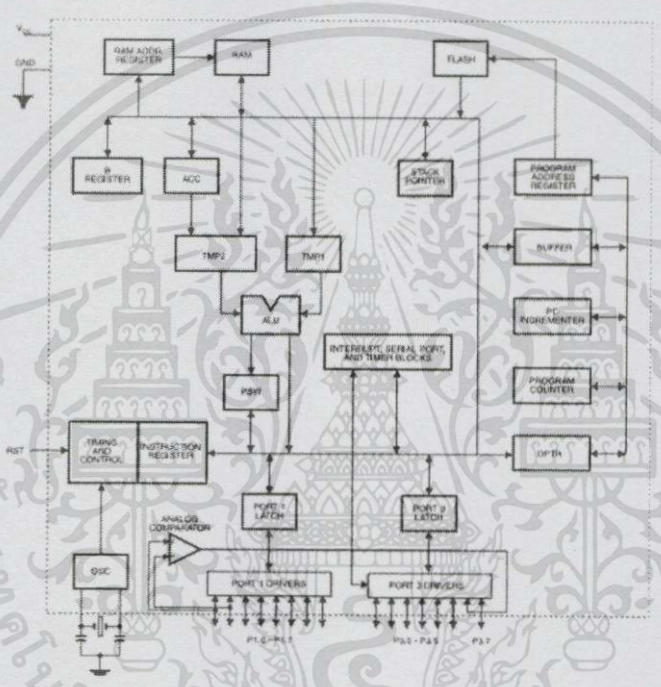
AT89C4051ใช้ระดับแรงดันต่ำ มีประสิทธิภาพสูง สามารถเขียนและลบข้อมูลในหน่วยความจำแบบอ่านอย่างเดียว ซึ่งอุปกรณ์ที่ใช้ผลิตมีความหนาแน่นสูงในเทคโนโลยีหน่วยความจำและเข้ากันได้กับมาตรฐานอุตสาหกรรมชุดคำสั่ง MCS-51

AT89C4051 เป็นไมโครคอนโทรลเลอร์ที่มีประสิทธิภาพซึ่งให้โซลูชั่นที่มีความยืดหยุ่นสูงและค่าใช้จ่ายที่มีประสิทธิภาพเพื่อการควบคุมการฝังตัวจำนวนมาก

❖ คุณสมบัติ

- 4K ไบต์ของการโปรแกรมใหม่ หน่วยความจำแฟลช – สามารถวงจร เขียน / ลบได้ : 10,000 รอบ
- ย่านการทำงาน 2.7V ถึง 6V

- 28 x 8 บิท แรมภายใน
- 15 Programmable I/O Lines
- 6 แหล่งอินเตอร์รัพ
- ขับLEDเอาท์พุตได้โดยตรง
- สามารถโปรแกรมช่อง UART ได้



รูปที่2.61 แสดงฟังก์ชันบล็อกไดอะแกรม

ตารางที่ 2.6 แสดงรายละเอียดขา

	Description
V_{CC}	Supply voltage
GND	Ground
Port 1	Port 1 is an 8-bit bi-directional I/O port. Port pins P1.2 to P1.7 provide internal pullups. P1.0 and P1.1 require external pullups. P1.0 and P1.1 also serve as the positive input (AIN0) and the negative input (AIN1), respectively, of the on-chip precision analog

<table>
<tr><td></td><td>comparator. The Port 1 output buffers can sink 20 mA and can drive LED displays directly. When 1s are written to Port 1 pins, they can be used as inputs. When pins P1.2 to P1.7 are used as inputs and are externally pulled low, they will source current (IIL) because of the internal pullups. Port 1 also receives code data during Flash programming and verification.</td></tr>
<tr><td>Port 3</td><td>Port 3 pins P3.0 to P3.5, P3.7 are seven bi-directional I/O pins with internal pullups. P3.6 is hard-wired as an input to the output of the on-chip comparator and is not accessible as a gen- eral-purpose I/O pin. The Port 3 output buffers can sink 20 mA. When 1s are written to Port 3 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (IIL) because of the pullups. Port 3 also serves the functions of various special features of the AT89C4051 as listed below:
<table>
<tr><th>Port pin</th><th>Alternate Functions</th></tr>
<tr><td>P3.0</td><td>RXD (serial input port)</td></tr>
<tr><td>P3.1</td><td>TXD (serial output port)</td></tr>
<tr><td>P3.2</td><td>INT0 (external interrupt 0)</td></tr>
<tr><td>P3.3</td><td>INT1 (external interrupt 1)</td></tr>
<tr><td>P3.4</td><td>T0 (timer 0 external input)</td></tr>
<tr><td>P3.5</td><td>T1 (timer 1 external input)</td></tr>
</table></td></tr>
<tr><td>RST</td><td>Reset input. All I/O pins are reset to 1s as soon as RST goes high. Holding the RST pin high for two machine cycles while the</td></tr>
</table>

	oscillator is running resets the device. Each machine cycle takes 12 oscillator or clock cycles
XTAL1	Input to the inverting oscillator amplifier and input to the internal clock operating circuit
XTAL2	Output from the inverting oscillator amplifier

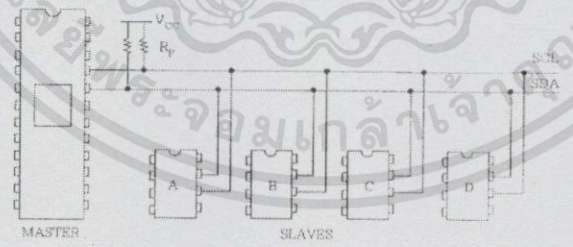
2.7 วงจรออกแบบ

2.7.1 I2C Bus

I2C Bus ย่อมาจาก Inter Integrate Circuit Bus (IIC) นิยมเรียกสั้นๆว่า I2C BUS (ไอ-แสคว-ซี-บัส) เป็นการสื่อสารอนุกรม แบบซิงโครนัส (Synchronous) เพื่อใช้ ติดต่อสื่อสาร ระหว่างไมโครคอนโทรลเลอร์ (MCU) กับอุปกรณ์ภายนอก ซึ่งถูกพัฒนาขึ้นโดยบริษัท Philips Semiconductors โดยใช้สายสัญญาณเพียง 2 เส้นเท่านั้น คือสายข้อมูลอนุกรม serial data (SDA) และสายสัญญาณนาฬิกา serial clock (SCL) ซึ่งสามารถ เชื่อมต่ออุปกรณ์จำนวนหลายๆ ตัว เข้าด้วยกันได้ ทำให้ MCU ใช้พอร์ตเพียง 2 พอร์ตเท่านั้น

2.7.1.1คุณสมบัติโดยทั่วไปของบัสI2C

สาย SDA และ SCL เป็นสายสัญญาณ 2 ทิศทาง(bi-directional line) ต้องมีการต่อตัวต้านทานพูลอัพกับแรงดัน +5V ไว้ตลอดเวลา เพื่อให้สายมีสถานะลอจิกสูงในขณะที่ไม่มีการติดต่อใช้งาน ทั้งยังช่วยป้องกันสัญญาณรบกวนที่อาจมีเข้ามาในสายสัญญาณทั้งสอง วงจรเอาท์พุทของอุปกรณ์ที่ต่ออยู่บนบัส I2C ต้องมีลักษณะเป็นวงจรเดรนเปิด(Open-drain) หรือ คอลเล็กเตอร์เปิด(Open-collector) อัตราการถ่ายทอดข้อมูลบนบัส I2C สูงถึง 100 กิโลบิทต่อวินาทีในโหมดปกติ และ สูงถึง 400 กิโลบิทต่อวินาทีในโหมดความเร็วสูง อุปกรณ์ที่ต่อร่วมบนบัส I2C จะต้องมีความจุไฟฟ้ารวมที่เกิดขึ้นระหว่างสาย SDA และ SCL ไม่เกิน 400pf การเข้าถึงอุปกรณ์บนบัส I2C ใช้ข้อมูลสำหรับการเข้าถึงสองค่าคือ 7 บิท(7-bit addressing) หรือ 10 บิท(10-bit addressing)



รูปที่ 2.7 การเชื่อมต่อด้วยระบบบัสแบบ I2C

2.7.1.2 หลักการของบัส I2C

บัส I2C ประกอบด้วยสายสัญญาณ 2 เส้นคือ SDA และ SCL อุปกรณ์ที่ต่อพ่วงบนบัสสามารถมีได้มากมาย ดังนั้นจึงต้องมีการกำหนดรูปแบบของการติดต่อบนบัส เพื่อให้ผู้ใช้งานทราบว่าขณะนี้อุปกรณ์ใดติดต่อกันอยู่ และอุปกรณ์ใดเป็นตัวรับหรือส่ง อุปกรณ์บนบัส I2C สามารถเป็นได้ทั้งตัวรับ

และส่ง บางอุปกรณ์ทำหน้าที่เป็นตัวรับอย่างเดียว จะไม่มีอุปกรณ์ใดบนบัส I2C ที่ทำหน้าที่เป็นตัวส่งอย่างเดียว อุปกรณ์ที่ทำหน้าที่ควบคุมจังหวะการติดต่อบนบัส I2C เรียกว่า มาสเตอร์(master) อุปกรณ์ที่ถูกควบคุมหรืออุปกรณ์ที่ต่อพ่วงเข้าไปบนบัส I2C เรียกว่า สเลฟ(slave) อุปกรณ์ที่เป็นผู้สร้างข้อมูลหรือส่งข้อมูล เรียกว่า ตัวส่ง (transmitter) อุปกรณ์ที่เป็นผู้รับข้อมูล เรียกว่า ตัวรับ (receiver)

2.7.1.3 ข้อกำหนดการติดต่อบนบัสI2C

2.7.1.3.1. การถ่ายทอดข้อมูลจะเกิดขึ้นได้เมื่อบัสว่างเท่านั้น

2.7.1.3.2. ในระหว่างการถ่ายทอดข้อมูล เมื่อใดก็ตามที่สาย SCL มีสถานะลอจิกสูง สายข้อมูลต้องรักษาข้อมูลไว้ อย่าให้เกิดความเปลี่ยนแปลงเด็ดขาด มิฉะนั้นสัญญาณที่เกิดขึ้นจะได้รับการแปลความหมายเป็นสัญญาณควบคุมแทน

2.7.1.4สภาวะที่เกิดขึ้นบนบัสI^{2}Cมีดังนี้

2.7.1.4.1. บัสว่าง(Bus not busy) สภาวะนี้เกิดขึ้นเมื่อ สถานะลอจิกบนสาย SDA และSCLมีลอจิกสูงทั้งคู่นั้นหมายความว่าการถ่ายทอดข้อมูลสามารถเริ่มต้นขึ้นได้

2.7.1.4.2. เริ่มต้นการถ่ายทอดข้อมูล(start data transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงลอจิกจากสูงไปต่ำ ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะนี้ว่า สภาวะเริ่มต้น (START)

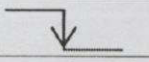
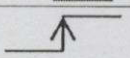
2.7.1.4.3. ข้อมูลดำรงอยู่บนบัส(data valid) สภาวะนี้เกิดขึ้นถัดจากสภาวะเริ่มต้น โดยสถานะลอจิกที่เกิดขึ้นบนสาย SDA ก็คือข้อมูลที่ทำการถ่ายทอด เมื่อสาย SCL มีลอจิกสูง สถานะที่สาย SDA ต้องคงที่ เพื่อให้อุปกรณ์รับข้อมูลในจังหวะนั้นว่า เป็น"0" หรือ "1" ข้อมูลอาจเกิดความเปลี่ยนแปลงได้ในขณะที่สาย SCL เป็นลอจิกต่ำ แต่เมื่อใดก็ตามที่ต้องการให้เกิดการถ่ายทอดข้อมูลอย่างสมบูรณ์ สถานะลอจิกที่ขา SDA ต้องคงที่ตลอดช่วงเวลาที่สาย SCL มีสถานะลอจิกสูง หากเกิดการเปลี่ยนแปลงสถานะลอจิกในขณะที่สาย SCL มีลอจิกสูงอยู่นั้น อุปกรณ์มาสเตอร์ที่ควบคุมการถ่ายทอดข้อมูลจะแปลความหมายเป็นสภาวะหยุดหรือสภาวะเริ่มต้นก็ได้ทำให้ข้อมูลที่ทำการถ่ายทอดเกิดความผิดพลาดเกิดขึ้น

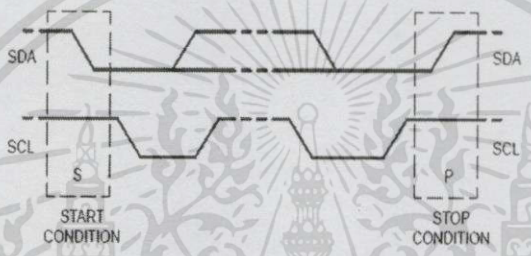
2.7.1.4.4. รับรู้ข้อมูล(acknowledge) เกิดขึ้นหลังจากการถ่ายทอดข้อมูลจากตัวส่งมายังตัวรับเกิดขึ้นอย่างสมบูรณ์ โดยตัวส่งจะทำการส่งข้อมูลมา 1 บิทเรียกว่า บิทรับรู้(acknowledge bit) มีถานะเป็นลอจิกสูง หลังการส่งข้อมูลมาครบถ้วน ส่วนอุปกรณ์มาสเตอร์จะทำการส่งสัญญาณรับรู้พิเศษซึ่งสัมพันธ์กับสัญญาณนาฬิกาอุปกรณ์สเลฟที่ถูกอ้างอิงในการติดต่อ หรือ กำลังติดต่ออยู่ในขณะนั้นก็จะกำเนิดบิทรับรู้ที่มีสถานะลอจิกต่ำเพื่อตอบสนองให้ทราบว่าได้รับข้อมูลเรียบร้อยแล้ว

2.7.1.4.5. หยุดการถ่ายทอดข้อมูล(stop data transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากต่ำไปสูง ในขณะที่สาย SCL มีสถานะลอจิกสูงเรียกสภาวะที่เกิดขึ้นนี้ว่า สภาวะหยุด(STOP)

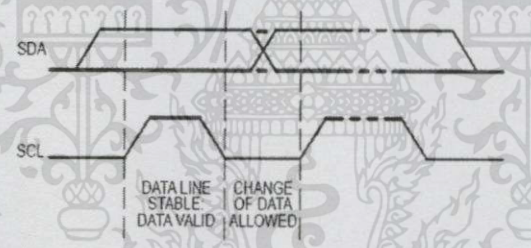
ตารางที่ 2.7 แสดงสภาวะบนบัส

สภาวะบนการสื่อสาร	SDA	SCL
1.ไม่มีการสื่อสาร	High	High

2.เริ่มส่งข้อมูล(Start condition)		High
3.หยุดส่งข้อมูล(Stop condition)		High
4.การรับส่งข้อมูล 1 บิทใช้ clock 1 ลูกขณะรับส่งข้อมูล ข้อมูลบน SDA ตองคงที่และ SDL เป็น High ข้อมูลบน SDA เปลี่ยนแปลงไดเมื่อ SCL เป็นLow	No change	High Low
5.สง Acknowledge bit เมื่อรับข้อมูลครบ 1 ไบท์ แลว	Low	High



รูปที่ 2.71 แสดงสภาพเริ่มและหยุดสงขอมูล

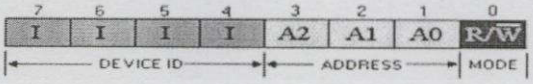


รูปที่ 2.72 แสดง Acknowledge / NOT-Acknowledge

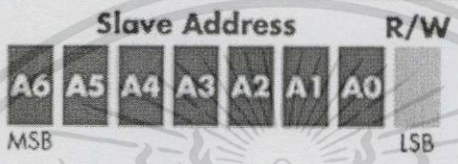
2.7.1.5 การเข้าถึงอุปกรณ์บนบัส I^2C

2.7.1.5.1. การเข้าถึงแบบ 7 บิท(7-bit addressing) ข้อมูลไบท์แรกที่เกิดขึ้นหลังจากสภาวะเริ่มต้นคือ ข้อมูลที่ใช้อ้างอุปกรณ์ที่ต้องการติดต่อโดยมีรูปแบบแสดงในรูปที่ 2.73 ใน 7 บิทบน รวมทั้งบิท LSB ด้วยจะเป็นข้อมูลแอดเดรสของอุปกรณ์สเลฟที่ต้องการติดต่อ โดยแบ่งเป็น บิทกำหนดแอดเดรสคงที่(fix address bit) จำนวน 4 บิท ซึ่งข้อมูลนี้อุปกรณ์แต่ละตัวจะถูกกำหนดมาจากผู้ผลิต ไม่สามารถเปลี่ยนแปลงแก้ไขได้ ถัดมาอีก 3 บิทเป็นบิทกำหนดแอดเดรสที่สามารถโปรแกรมได้(programmable address bit) โดยผู้ใช้งานต้องกำหนดสถานะลอจิกให้แก่ขา A0-A2 ของอุปกรณ์ที่มีการเชื่อมต่อแบบบัส I2C ส่วนในบิต LSB ที่ใช้กำหนดการอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวนั้น ๆ หากบิต LSB เป็น "0" หมายถึงต้องการเขียนข้อมูลไปยังอุปกรณ์นั้น ถ้าเป็น "1" จะเป็นการอ่านข้อมูลจากอุปกรณ์สเลฟ ข้อมูลในไบท์ต่อมาคือข้อมูลควบคุม(control byte) ในอุปกรณ์แต่ละตัวจะมีการกำหนดข้อมูลควบคุมที่แตกต่างกันไป ยกตัวอย่างเช่น ไอซีเมมโมรีของทีวีตระกูล 24Cxx จะต้องส่งข้อมูลแอดเดรสของหน่วยความจำ

ก่อนที่จะทำการส่งข้อมูลไป ข้อมูลในไบท์ต่อมาคือ ข้อมูลที่ทำการถ่ายทอดจริง(data) หลังจากการถ่ายทอดข้อมูลในแต่ละไบท์ อุปกรณ์สเลฟที่ได้รับการติดต่อต้องส่งสัญญาณรับรู้ตอบกลับมาด้วยทุกครั้ง



รูปที่ 2.73 การกำหนดบิตแอดเดรสแบบ7บิท

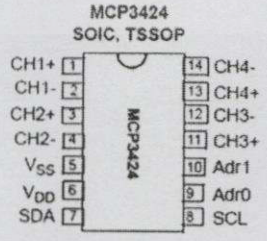


รูปที่ 2.74 แสดงรูปแบบข้อมูลอนุกรมของการอ้างถึงแบบ7บิท

2.7.1.5.2.การเข้าถึงแบบ 10 บิท(10-bit addressing) จะมีข้อมูลเพิ่มเติมขึ้นมาเล็กน้อย โดยในไบท์แรกหลักจากสภาวะเริ่มต้น ต้องกำหนดให้ 5 บิทบนมีข้อมูลเป็น 11110 ส่วนอีก 2 บิทถัดมาเป็นบิทแอดเดรสของอุปกรณ์ที่ต้องการติดต่อ ในบิท LSB ของข้อมูลไบท์แรกยังคงเป็นการกำหนดว่าต้องการอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวที่ต้องการติดต่อด้วย ข้อมูลไบต์ต่อมาเป็นข้อมูลแอดเดรสในไบต์ที่ 2 ของอุปกรณ์ที่ต้องการติดต่อด้วย ข้อมูลไบท์ถัดไปจึงเป็นข้อมูลควบคุมข้อมูล หลังจากนี้ก็จะเป็นข้อมูลจริงที่ใช้ในการติดต่อเช่นเดียวกันกับการเข้าถึงแบบ 7 บิทหลังจากถ่ายทอดข้อมูลครบทุกไบต์ต้องมีสภาวะรับรู้เกิดขึ้น เพื่อให้ขบวนการถ่ายทอดข้อมูลสามารถดำเนินต่อไปได้ดังรูปแบบข้อมูลอนุกรมของการอ้างถึงแบบ10บิท

รูปที่ 2.75 แสดงรูปแบบข้อมูลอนุกรมของการอ้างถึงแบบ10บิท

2.8 ไอซี MCP 3424



รูปที่ 2.8 ไอซี MCP3424

MCP3424 เป็นตัวแปลงสัญญาณอนาลอกเป็นดิจิตอล ที่มีสัญญาณรบกวนน้อยและมี ความเที่ยงตรงสูง อุปกรณ์ชิ้นนี้สามารถแปลงสัญญาณพรอมใหเอาท์พุท ขนาด 18 บิท ในอัตรา 3.75,15,60 หรือ 240 ต อวินาที ขึ้นอยู่กับการตั้งค่าผ่านทาง I2C อุปกณ์สามารถสอบเทียบค่า offset และค่า gain ไดอัตโนมัติอุปกรณ ชิ้นนี้มามารถส่งข้อมูลไดอย่างถูกต้องในสภาวะที่มีรูปแบบความผันผวนของอุณหภูมิและแหลงจ่ายไฟฟ้า

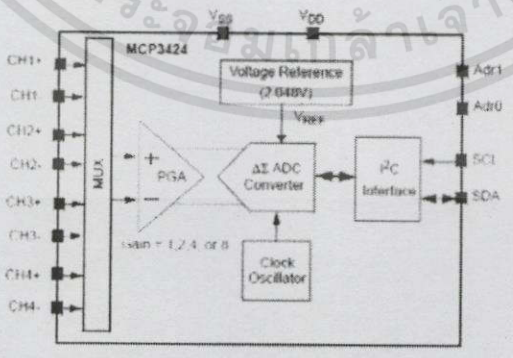
2.8.1 MCP3424 มีการแปลงสัญญาณสองแบบ

2.8.1.1. การแปลงสัญญาณแบบส่ง output ครั้งเดียว อุปกรณ์จะดำเนินการแปลงสัญญาณเพียงครั้งเดียวและจะส่งกระแสต่ำสแตนด์บายไวอัตโนมัติจนกว่าจะมีคำสั่งแปลงสัญญาณครั้งต่อไป วิธีนี้จะลดการใช้กระแสไฟฟ้าลงอย่างมากในช่วงที่ไมไดมีการใช้งาน

2.8.1.2. การแปลงสัญญาณแบบส่งเอาท์พุทอย่างต่อเนื่องจะแปลงสัญญาณอย่างต่อเนื่องโดยไมหยุดตามความเร็วที่ไดตั้งเอาไว การแปลงสัญญาณจะเลือกสัญญาณที่เขามาลาสุดแปลงออกไปทางเอาท์พุต อุปกรณ์นี้จะดำเนินการเมื่อมีแหล่งจ่ายไฟฟ้า 2.7 V ถึง 5.5 V และมีสาย 2 เสนในการตอกับพอร์ต I2C ตำแหนง I2C address bit สำหรับ MCP3424 เลือกโดยใช้การเชื่อมต่อภายนอก (เลือกจาก pin Adr0 and Adr1)

❖ **คุณสมบัติ**

- มีอินพุต 4 ชอง
- I2C Interface
- มีโหมดมาตรฐาน, รวดเร็วและความเร็วสูง
- กำหนดแอดเดรสสำหรับเชื่อมต่อกับอุปกรณ์ภายนอก
- ใช้แหลงจ่ายไฟ 2.7V to 5.5V
- ชวงของอุณหภูมิในการใช้งาน -40°C to +125°C
- Voltage Reference ของบอร์ด (VREF): ความเที่ยงตรง: 2.048V ± 0.05%
- โปรแกรมเพื่อกำหนดคา Gain ของบอรด (PGA):
- คา Gains 1, 2, 4 หรือ 8

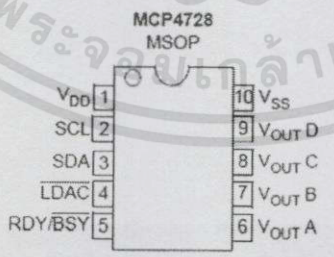


รูปที่ 2.81 แสดงฟังก์ชันบล็อกไดอะแกรม

ตารางที่ 2.8 แสดงการทำงานของแต่ละขา

MCP3424	Sym	Description
1	CH1+	Positive Differential Analog Input Pin of Channel 1
2	CH1-	Negative Differential Analog Input Pin of Channel 1
3	CH2+	Positive Differential Analog Input Pin of Channel 2
4	CH2-	Negative Differential Analog Input Pin of Channel 2
5	V_{SS}	Ground Pin
6	V_{DD}	Positive Supply Voltage Pin
7	SDA	Bidirectional Serial Data Pin of the I2C Interface
8	SCL	Serial Clock Pin of the I2C Interface
9	Adr0	I2 C Address Selection Pin
10	Adr1	I2C Address Selection Pin
11	CH3+	Positive Differential Analog Input Pin of Channel 3
12	CH3-	Negative Differential Analog Input Pin of Channel 3
13	CH4+	Positive Differential Analog Input Pin of Channel 4
14	CH4-	Negative Differential Analog Input Pin of Channel 4

2.9 ไอซี MCP 4728



รูปที่ 2.9 ไอซี MCP4728

อุปกรณ์ MCP4728 เป็นตัวแปลงสัญญาณดิจิตอลเป็นอนาลอก (DAC) ขนาด 12 บิต 32 ด้วย หน่วยความจำ (EEPROM) ค่า input code DAC, การกำหนดค่าบิตของอุปกรณ์, และบิทตำแหน่งของ I2C จะ ถูกโปรแกรมลงที่หน่วยความจำ (EEPROM) โดยการใช้คำสั่งอินเตอร์เฟสของ I2C คุณสมบัติของหน่วยความจำ

คือจะช่วยให้อุปกรณ์เก็บค่า input code ไว้ได้ในระหว่างที่ปิดไฟ, เรียกใช้ค่าเอาท์พุทได้ทันทีหลังจากเปิดไฟ พร้อมกับบันทึกค่าที่เคยตั้งไว้ คุณลักษณะนี้มีประโยชน์อย่างมากในตอนที่อุปกรณ์ DAC นี้ถูกใช้เป็นตัวสนับสนุนอุปกรณ์อื่นในด้านเนตเวิร์ก

❖ คุณสมบัติ

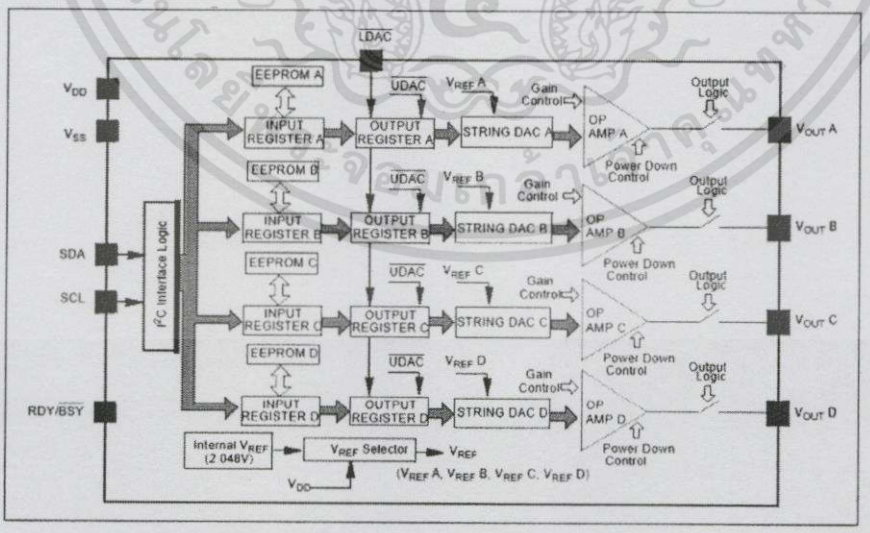
- Voltage output DAC ขนาด 12-บิท
- เลือกใช้แรงดันไฟฟ้าอ้างอิงได้ทั้งภายในและภายนอก
- ย่านการใช้งานของ Voltage output : ใช้ VREF ภายใน (2.048V) :
- 0.000V ถึง 2.048V กำหนดค่า Gain = 1
- 0.000V ถึง 4.096V กำหนดค่า Gain = 2

❖ ถ้าใช้ VREF ภายนอก (VDD) : 0.000V ถึง VDD

- เลือกโหมด Normal หรือ Power-Down
- ใช้พลังงานน้อย
- ทำงานโดยใช้แหล่งจ่ายไฟขนาด: 2.7V to 5.5V

❖ อินเตอร์เฟสของ I2C :

- บิทแอดเดรส: ผู้ใช้สามารถโปรแกรมเข้าสู่ EEPROM
- โหมดมาตรฐาน (100 kbps), เร็ว (400 kbps) และ ความเร็วสูง (HS) (3.4 Mbps)
- ย่านอุณหภูมิที่ใช้งาน: -40°C ถึง +125°C

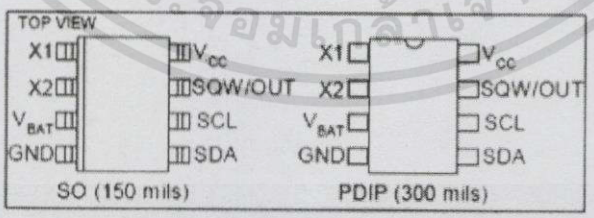


รูปที่ 2.91 แสดงฟังก์ชันบล็อกไดอะแกรม

ตารางที่ 2.9 แสดงการทำงานของแต่ละขา

Pin No.	Name	Function
1	V_{DD}	Supply voltage
2	SCL	I^2C serial Clock Input
3	SDA	I^2C serial Data Input and Output
4	LDAC	This pin is used for two purposes: (a)Synchronization Input. It is used to transfer the content of the DAC input registers to the output registers (V_{OUT}) (b) Select the device for reading and writing I^2C address bit
5	RDY/BSY	This pin is a status indicator of EEPROM programming activity. An external pull-up resistor(about 100Ω)
6	$V_{OUT}A$	Buffered analog voltage of channel A. The output amplifier has rail-to-rail operation.
7	$V_{OUT}B$	Buffered analog voltage of channel B. The output amplifier has rail-to-rail operation.
8	$V_{OUT}C$	Buffered analog voltage of channel C. The output amplifier has rail-to-rail operation.
9	$V_{OUT}D$	Buffered analog voltage of channel D. The output amplifier has rail-to-rail operation.
10	V_{SS}	Ground reference

2.10 DS1307 (Serial Real Time Clock)



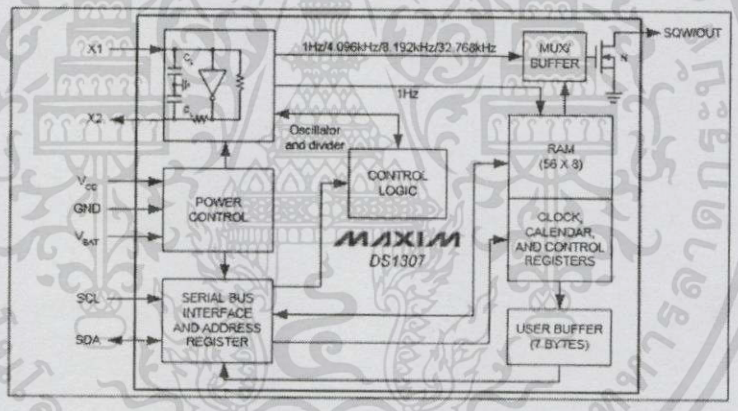
รูปที่ 2.10 ไอซี DS1307 Serial Real Time

DS1307 เป็น IC ฐานเวลาของดัลลัสเซมิคอนดักเตอร์ (Dallas Semiconductor) มีบัสรับส่งข้อมูลแบบ I2C ซึ่งเป็นแบบ 2 wire สามารถสื่อสารได้ 2 ทิศทาง (bi-direction bus) ฐานเวลาของ DS1307 นั้นสามารถเก็บข้อมูล วินาที, นาที, ชั่วโมง, วัน, วันที่, เดือน และปี ได้ ระบบเวลาสามารถทำงานโหมดรูปแบบ

24 ชั่วโมง หรือ 12 ชั่วโมง AM/PM ก็ได้ ภายมีระบบตรวจจับแหล่งจ่ายไฟ โดยถ้าแหล่งจ่ายไฟหลักถูกตัดไป DS1307 สามารถสวิตซ์ไปใช้ไฟจากแบตเตอรี่ และทำงานต่อไป โดยที่ยังสามารถรักษาข้อมูลไว้ได้ โครงสร้างมีขาทั้งหมด 8 ขา และมีรายละเอียดการทำงานของขาต่าง ๆ ดังนี้

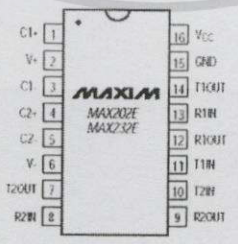
คุณสมบัติ

- VCC: ใช้ต่อไฟเลี้ยง +5V
- GND: ใช้ต่อกราวด์
- VBAT: ใช้ต่อกับแบตเตอรี่ 3V เพื่อรักษาการทำงาน ในกรณีที่ไม่มีไฟเลี้ยงจ่าย
- SDA: ขารับส่งข้อมูลด้วยระบบบัส I2C
- SCL: ขาสัญญาณนาฬิกาสำหรับการรับส่งข้อมูลด้วยระบบบัส I2C
- SQW/OUT: ขาเอาต์พุตสัญญาณสแควร์เวฟสามารถเลือกความถี่ได้
- X1, X2: ใช้ต่อกับคริสตอลความถี่มาตรฐาน 32.768 kHz เพื่อสร้างฐานเวลาจริงให้กับไอซี



รูปที่ 2.10(ก) แสดงฟังก์ชันบล็อกไดอะแกรม

2.11 ไอซี MAX 232

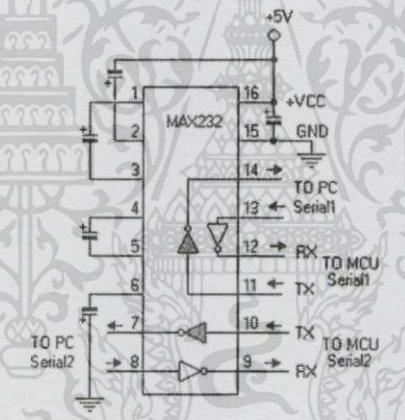


รูปที่ 2.11 ไอซี MAX 232

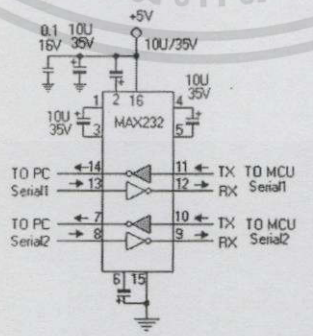
เป็นไอซีที่แปลงระดับสัญญาณจากระดับ TTL ไปเป็นระดับของ RS-232 และในทำนองเดียวกันก็รับระดับสัญญาณจากRS-232เพื่อแปลงเป็นระดับสัญญาณจากระดับ TTL ให้กับไมโครคอนโทรลเลอร์ได้ ใช้แรงดันไฟฟ้า5โวลต์ MAX 232 เป็นไอซีขนาด 16 ขา ใช้ไฟเลี้ยง5โวลต์ ภายในมีวงจรแปลง RS-232 เป็น TTL สองชุด และวงจรแปลง TTLเป็น RS-232 อีกสองชุดและภายใน MAX 232 มีวงจรทวีแรงดันและวงจรกลับขั้วแรงดันซึ่งต้องอาศัยอิเลคโทรไลติก คาปาซิเตอร์ภายนอกสี่ตัวจึงจะทำงานได้

❖ คุณสมบัติ

- ทำงานที่ระดับแรงดันไฟฟ้าแหล่งจ่าย 5V
- ทำงานได้ถึง 120 kbit/s สองตัวขับและสองตัวรับ
- ±30โวลต์แรงดันอินพุท
- กระแสแหล่งจ่าย 8 mA



รูปที่2.11(ก) รูปวงจรภายในMAX232



รูปที่ 2.11(ข) รูปการต่อใช้งานMAX232กับไมโครคอนโทรลเลอร์

2.12 หน้าจอแสดงผล LCD 4x20

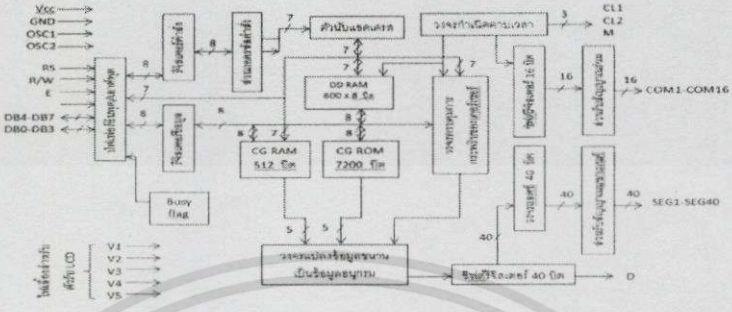
LCD รายละเอียดเกี่ยวกับโมดูล LCDในโมดูล LCD จะมีส่วนประกอบหลักๆ 3 ส่วนดังนี้

- ตัวแสดงผล(display)ภายในเป็นผลึกเหลวที่สามารถแสดงผลให้เห็นโดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองข้อมูลที่แสดงผลบนจอ LCD
- ตัวควบคุม(controller)เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของโมดูล LCD เช่น ลบจอภาพ แสดงตัวอักษร หรือเลื่อนเคอร์เซอร์ เป็นต้น ตัวควบคุมนี้ใช้ชิปควบคุมโดยเฉพาะชิปที่นิยมใช้คือเบอร์ HD44780 และ HD61830 โดย HD44780 ใช้ควบคุม LCD แบบอักขระ ส่วน HD61830 ใช้ควบคุม LCD กราฟฟิก
- ตัวขับ(driver)เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผล แสดงข้อมูลตามที่กำหนดชิปที่ใช้ทำหน้าที่เป็นตัวขับนี้ได้แก่ เบอร์ HD44100H , MSM5259 เป็นต้น

2.12.1 โครงสร้างภายในของตัวควบคุมโมดูล LCD

ในการใชงานโมดูล LCD จำเปนตองทำความเขาใจเกี่ยวกับโครงสราง และคำสั่งที่ใชในการควบคุมใหดีเสียกอน ยกตัวอยาง LCD แบบอักขระ เพราะสามารถเขาใจไดงาย ใหรูปที่ P15-1 เปนบล็อกไดอะแกรมภายในของชิปควบคุม LCD เบอร HD44780 ซึ่งใชในโมดูล LCD แบบอักขระ ประกอบดวย

- **บัฟเฟอรอินพุตเอาตพุต** เปนสวนที่ใชติดตอรับสงขอมูลกับตัวอุปกรณภายนอกเพื่อถายทอดขอมูลเขาออกภายในตัวควบคุม
- **รีจิสเตอรคำสั่ง(Instruction Registor:IR)** เปนรีจิสเตอรใชรับขอมูลคำสั่งจาก อุปกรณภายนอกเพื่อนำไปควบคุมการแสดงผล
- **รีจิสเตอรขอมูล (Data Registor : DR)** เปนรีจิสเตอรใชรับขอมูลจากอุปกรณ ภายนอกเพื่อสงตอไปยังหนวยความจำที่ทำหนาที่เก็บขอมูลแสดงผลหรือนำขอมูลไปสรางตัวอักษรเพิ่มเติมในแรมเก็บตัวอักษร
- **แรมเก็บขอมูลแสดงผล(display Data RAM : DDRAM)** เปนหน่วยความจำแรมทำหน้าที่เก็บขอมูลที่มาจากรีจิสเตอร DR ตัวควบคุมจะนำขอมูลใน DDRAM นี้ไปเปดตาราง (Look up-lable) ของตัวอักษรที่เก็บไวในหนวยความจำรอมและแรมเก็บตัวอักษรเพื่อนำไปแสดงที่ตัวแสดงผล

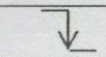
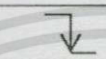


รูปที่ 2.12 แสดงไดอะแกรมการทำงานของโมดูล LCD แบบอักขระ

- **รอมเก็บตัวอักษร** (Character Generator ROM : CDROM)
  เป็นหน่วยความจำรอมที่ใช้เก็บข้อมูลตัวอักษรหรือสัญลักษณ์ที่สามารถอ่านออกไปแสดงที่ตัวแสดงผลได้ มีขนาด 7200 บิต โดยจะถูกอ่านได้ด้วยค่าของข้อมูลใน DDRAM
- **แรมเก็บตัวอักขระ** (Character Generator RAM : CGRAM)
  เป็นหน่วยความจำแรมความจุ 512 บิตใช้เก็บอักษรที่มีการสร้างเพิ่มเติมขึ้นใหม่ ในกรณีที่ตัวอักษรใน CGROM ไม่เพียงพอ การเขียนและอ่านค่าไปใช้นั้นทำได้เช่นเดียวกับ CGROM คือ เขียนข้อมูลลงใน DDRAM แล้วตัวข้อมูลจะมาอ่านค่าจาก CGRAM เอง แฟลก BUSY เป็นส่วนที่ทำหน้าที่แจ้งสถานะการทำงานของตัวควบคุมให้อุปกรณ์ภายนอกทราบว่าตัวควบคุมพร้อมที่จะรับข้อมูลหรือคำสั่งหรือไม่ดังนั้นก่อนการส่งข้อมูลหรือคำสั่งมายังตัวควบคุมต้องตรวจสอบสถานะของ แฟลก BUSY นี้เสียก่อน

รูปที่ 2.12 (ก) แสดง LCD 4x20

ตารางที่ 2.10 รูปรางและการจัดขาโมดูล LCD แบบอักขระ

RS	R/W	E	การทำงาน
0	0		เขียนคำสั่ง
0	1		อ่านสถานะของโมดูล LCD
1	0		เขียนข้อมูล
1	1		อ่านข้อมูล

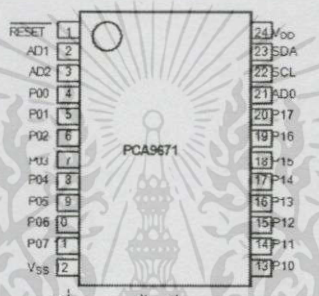
ตารางที่ 2.11 แสดงการหนาที่การทำงานของแตละขา

ขา	สัญลักษณ์	คำอธิบาย	ระดับ	หน้าที่	
1	VSS	Ground	-	0V	Ground
2	VDD	Power Supply	-	+5V	ต่อกับแรงดันไฟเลี้ยง +5V
3	V0	LCD Control	-	-	ต่อกับแรงดันเพื่อปรับความเข้มของการแสดงผล
4	RS	Register Select	H/L	RS=0 หมายถึงต้องการติดต่อกับรีจิสเตอร์คำสั่ง (Instruction Register) RS=1 หมายถึงต้องการติดต่อกับรีจิสเตอร์ข้อมูล (Data Register)	
5	R/W	Read/Write	H/L		
6	E	Enable	H,H->L	Enable Signal	
7-14	DB0-DB7	Data Bus	H/L	Data Bus Line	
15	A	Back Light A	-	Back Light +5V (สำหรับรุ่นที่มี Back Light)	
16	K	Back Light K	-	Back Light 0V (สำหรับรุ่นที่มี Back Light)	

- Vss (ขา1) : ตอกราวด
- V_{DD} (ขา2): ตอไฟเลี้ยง +5V
- Vo (ขา3) : เปนขาอินพุตรับแรงดันเพื่อนปรับความเขมของการแสดงผล
- RS(ขา4): เปนขาอินพุตใชในการแยกชนิดของขอมูลที่ทำการประมวลผลในขณะนั้น วาเปนคำสั่งสำหรับรีจิสเตอร IR หรือเปนขอมูลสำหรับรีจิสเตอร DR โดยถาขานี้เปน “00” ขอมูลที่สงมาจะเป็นคำสั่ง แตถาขานี้เปน “1” ขอมูลที่สงมาจะเปนขอมูลที่จะแสดงผล

- R/W(ขา5): เปนขาที่ใชเลือกการอานหรือเขียนขอมูลกับโมดูล LCD ถาเปน “0” เปนการกำหนดใหเขียนขอมูล แตถาเปน “1” จะเปนการอานขอมูล
- E (ขา6) : เปนขาสำหรับรับสัญญาณพัลสเอ็นเอเบิลโมดูล LCD ใหทำงาน D0-D7 (ขา 7-14): เปนขาที่ใชเปนทางผานของขอมูลระหวาง LCD กับอุปกรณภายนอกขนาด 8 บิท อนึ่งขา RS,R/W และ E จะใชรวมกัน

2.13 ไอซี PCA9671

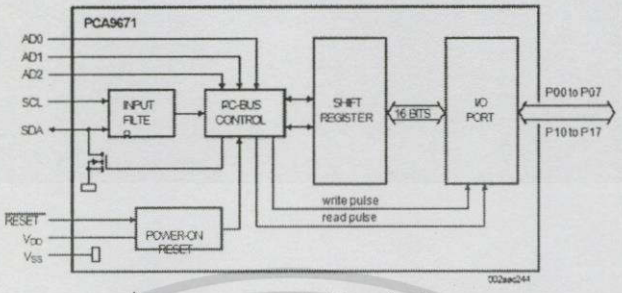


รูปที่ 2.13 ไอซี PCA9671

มีวัตถุประสงค์ทั่วไปเพื่อขยาย remote I/O สำหรับไมโครคอนโทรลเลอร์ผานบัสI2Cและเป็นส่วนหนึ่งของFast-mode Plus (Fm+)

❖ **คุณสมบัติ**

- อินเตอร์เฟซ I2C บัส 1 MHz
- สอดคล้องกับ I2C บัสอย่างรวดเร็วและโหมดมาตรฐาน
- SDA ด้วย30mA มีความสามารถในการซิงค์4000 pF buses
- ย่านการทำงาน2.3 V ถึง5.5 V
- 16 บิทขาremote I/O
- ย่านอุณหภูมิ 40-80 °C
- แลชเอาท์พุตด้วย25mA สามารถซิงค์สำหรับการขับLEDได้โดยตรง
- ใช้กระแสสแตนด์บายต่ำ
- Active Low reset input

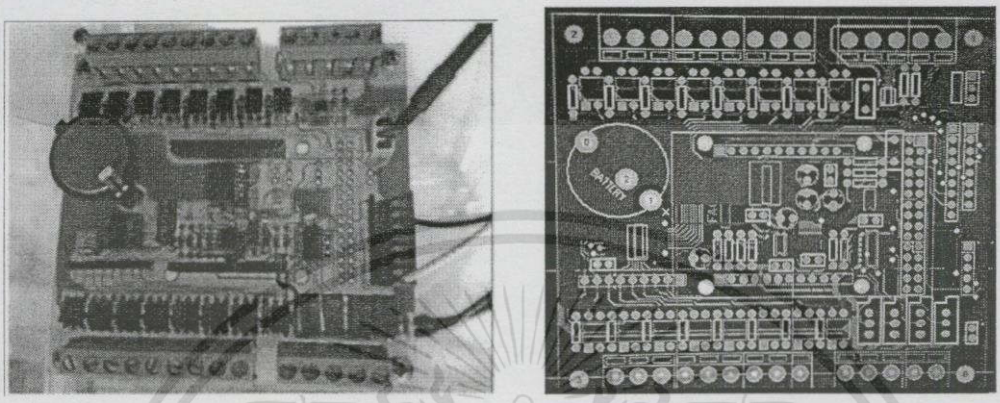


รูปที่ 2.13(ก) แสดงฟังก์ชันบล็อกไดอะแกรม

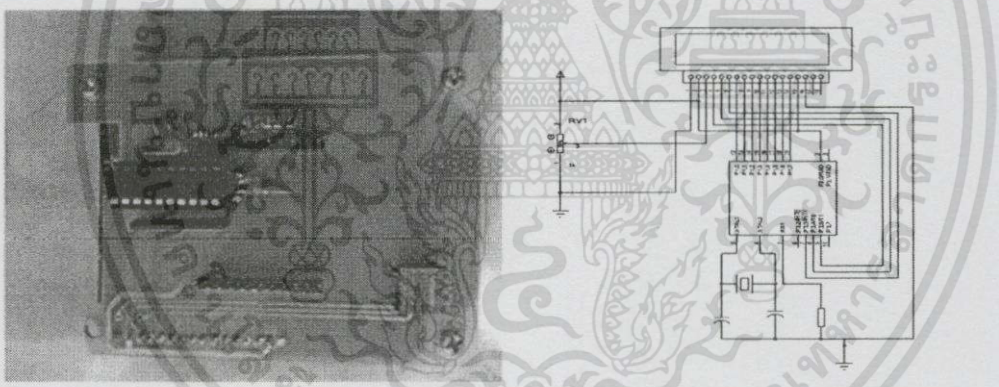
ตารางที่ 2.13 แสดงการหน้าที่การทำงานของแต่ละขา

Pin	Symbol	Description
1	RESET	Reset input (activeLow)
2	AD1	address input 1
3	AD2	address input 2
4	P00	quasi-bidirectional I/O 00
5	P01	quasi-bidirectional I/O 01
6	P02	quasi-bidirectional I/O 02
7	P03	quasi-bidirectional I/O 03
8	P04	quasi-bidirectional I/O 04
9	P05	quasi-bidirectional I/O 05
10	P06	quasi-bidirectional I/O 06
11	P07	quasi-bidirectional I/O 07
12	VSS	supply ground
13	P10	quasi-bidirectional I/O 10
14	P11	quasi-bidirectional I/O 11
15	P12	quasi-bidirectional I/O 12
16	P13	quasi-bidirectional I/O 13
17	P14	quasi-bidirectional I/O 14
18	P15	quasi-bidirectional I/O 15
19	P16	quasi-bidirectional I/O 16
20	P17	quasi-bidirectional I/O 17

2.14 การออกแบบแผ่นวงจรพิมพ์



รูปที่ 2.14 แผงวงจร



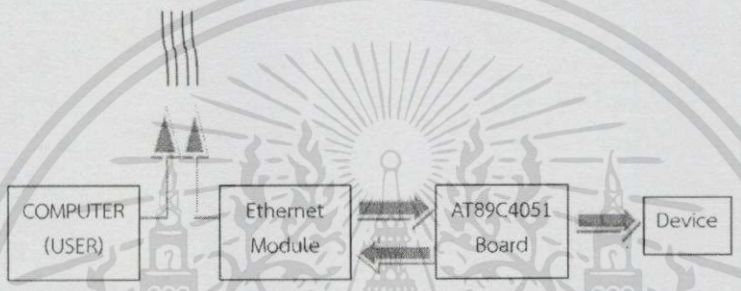
รูปที่ 2.15 แผงLCDที่มีAT89C4051ประกอบอยู่ข้างหลัง

บทที่ 3

วิธีการดำเนินงาน

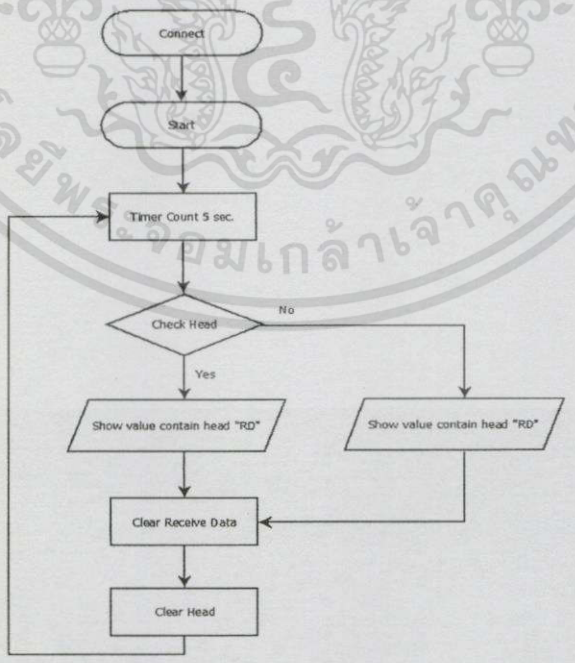
3.1 การออกแบบวงจร

จากการศึกษาปริญญานิพนธ์อย่างละเอียดเพื่อต้องการทราบถึงอุปกรณ์ใดบ้างที่จำเป็นต้องใช้ แต่ละตัวมีคุณสมบัติในการใช้งานอย่างไร ระบบมีลำดับขั้นตอนการทำงานอย่างไรบ้าง เพื่อให้สามารถออกแบบวงจรที่จะใช้กับระบบได้อย่างมีประสิทธิภาพและเกิดความผิดพลาดน้อยที่สุด โดยได้ออกแบบแผนภาพการทำงานของทั้งระบบไว้ดังนี้

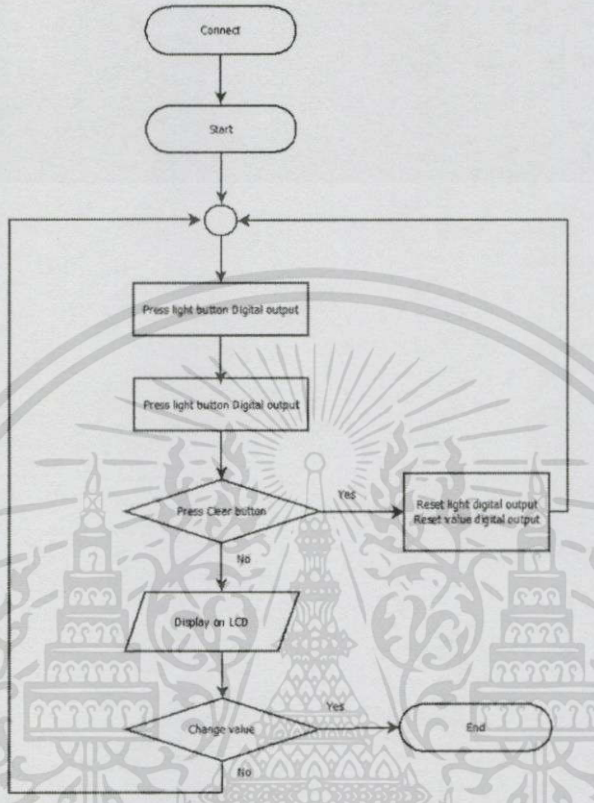


รูปที่3.11 บล็อกการทำงานของทั้งระบบ

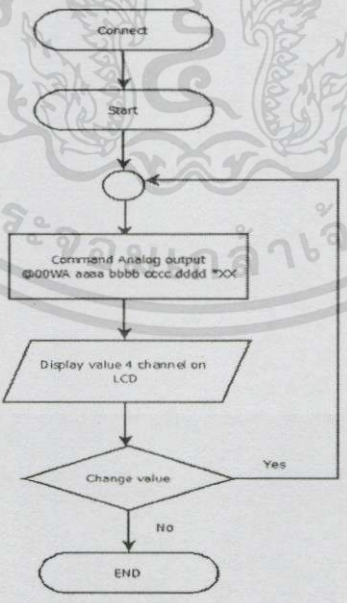
นอกจากแผนภาพการทำงานของระบบโดยรวมแล้ว ยังต้องมีการออกแบบในส่วนของแอพพลิเคชั่นที่จะใช้เพื่อเป็นตัวกลางในการติดต่อสื่อสารระหว่างผู้ใช้งานกับอุปกรณ์ โดยได้เลือกเป็นโปรแกรม Visual Basic เนื่องจากเป็นโปรแกรมที่ง่ายต่อการใช้งาน มีการออกแบบขั้นตอนการทำงานของโปรแกรมไว้ดังนี้



รูปที่ 3.12 แผนภาพการทำงานของอนาลอกอินพุทและดิจิตอลอินพุท



รูปที่ 3.13 แผนภาพการทำงานโปรแกรมของข้อมูลดิจิตอลเอาท์พุท



รูปที่ 3.14 แผนภาพการทำงานของอนาลอกเอาท์พุท

3.2 การเขียนโปรแกรมภาษาแอสเซมบลี

เนื่องจากอุปกรณ์ที่ใช้นั้นมีไมโครคอนโทรลเลอร์ AT89C4051 เป็นตัวหลักในการควบคุมการติดต่อสื่อสารทั้งหมด โดยเป็นตระกูลเดียวกันกับไมโครคอนโทรลเลอร์ MCS-51 ซึ่งสามารถเขียนโปรแกรมได้ 2 ภาษาคือ ภาษาซีและภาษาแอสเซมบลี แต่มักจะนิยมเขียนด้วยภาษาแอสเซมบลีเพราะมีคำสั่งที่สั้นกว่าและรวดเร็วในการประมวลผล เพื่อให้ไอซีแต่ละตัวบนอุปกรณ์ทำงานร่วมกันได้และให้ระบบทำงานได้นั้นจึงจำเป็นที่จะต้องมีการเขียนโปรแกรมเพื่อควบคุมการทำงานของไอซี ซึ่งไอซีแต่ละตัวก็มีหน้าที่ต่างกันไปตามคุณสมบัติตามที่กล่าวไว้ในบทที่2 โดยทำการเขียนโปรแกรมเป็นภาษาแอสเซมบลีแล้ว compile ด้วยโปรแกรม asm51 จะได้ Hex file เพื่อนำไป write ลง ship ซึ่งมีการเขียนโปรแกรมให้กับส่วนต่างๆที่สำคัญตามขั้นตอนดังต่อไปนี้

3.2.1 การประกาศตัวแปรต่างๆ

3.2.2 ส่วนของหน้าจอแสดงผล(LCD)

3.2.3 ส่วนนาฬิกาของระบบ

3.2.4 ส่วนของไอซีต่างๆ

3.2.5 ส่วนของการติดต่อสื่อสารแบบอนุกรม

3.2.6 ส่วนของตัวเลขในการแปลง (เลขฐานสิบหก, เลขฐานสอง, ASCII) สำหรับค่าข้อมูลดิจิตอล

3.2.7 ส่วนของ Command ที่ใช้เป็นโปรโตคอลในการติดต่อสื่อสาร

หมายเหตุ : ซึ่งส่วนคำสั่งการเขียนโปรแกรมทั้งหมดทั้งส่วนของอุปกรณ์ที่เป็นภาษาแอสเซมบลีและส่วนของโปรแกรม Visual Basic ที่เป็นภาษาซีอยู่ในส่วนของภาคผนวก

3.3 การออกแบบโปรโตคอล

ในการติดต่อสื่อสารข้อมูลกันระหว่างคอมพิวเตอร์สองเครื่อง หรือระหว่างคอมพิวเตอร์กับอุปกรณ์อื่นๆจำเป็นที่จะต้องมีโปรโตคอลเพื่อใช้เป็นตัวกลางในการสื่อสารข้อมูลระหว่างปลายทางกับต้นทางได้อย่างถูกต้อง โดยผลตอบสนองของระบบจะอยู่ในบทที่ 4 ซึ่งโปรโตคอลหลักๆที่ใช้ในการทดลองมีดังนี้

3.3.1 “@00RA*XX” เป็นโปรโตคอลที่ออกแบบมาเพื่อใช้ในการอ่านค่าข้อมูลอนาลอกอินพุท โดย R มาจากคำว่า Read และ A มากจากคำว่า Analog

3.3.2 “@00RD*XX” เป็นโปรโตคอลที่ออกแบบมาเพื่อใช้ในการอ่านค่าข้อมูลดิจิตอลอินพุท โดย R มาจากคำว่า Read และ D มากจากคำว่า Digital

3.3.3 “@00WAaaaa bbbb cccc dddd*XX” เป็นโปรโตคอลที่ออกแบบมาเพื่อใช้ในการเขียนค่าอนาลอกเอาท์พุท โดยที่ aaaa bbbb cccc dddd เป็นตัวเลขตั้งแต่ 0-9 ที่ต้องการ

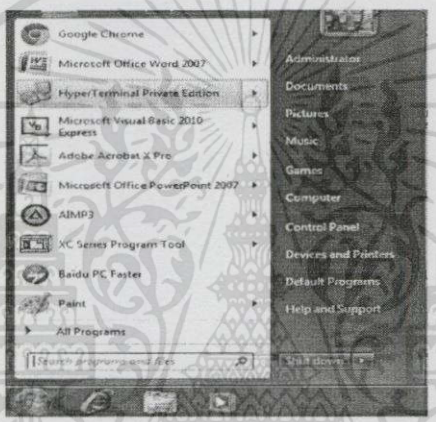
3.3.4 “@00WD00AA” เป็นโปรโตคอลที่ออกแบบมาเพื่อใช้ในการเขียนค่าดิจิตอลเอาท์พุท โดยที่ AA เป็นเลขฐานสิบหก 0-F ที่เราต้องการและแสดงผลเป็นเลขฐานสองลอจิก 0 และ 1 ตามการติดดับของไฟ

3.4 ขั้นตอนการทดสอบโดยใช้ RS232 Communication

เมื่อทำการประกอบส่วนต่างๆเข้าด้วยกันเรียบร้อยแล้วในส่วนของฮาร์ดแวร์ จะต้องทำการทดสอบการใช้งานว่าสามารถรับส่งค่าได้จริงหรือไม่ โดยเริ่มต้นทำการทดลองที่ตัวอุปกรณ์เพียงอย่างเดียวก่อน ทำได้โดยการกดปุ่มดูค่าข้อมูลที่เข้ามาในระบบทั้งอนาลอกและดิจิตอลว่าสามารถแสดงผลตอบสนองที่จอแสดงผลได้หรือไม่

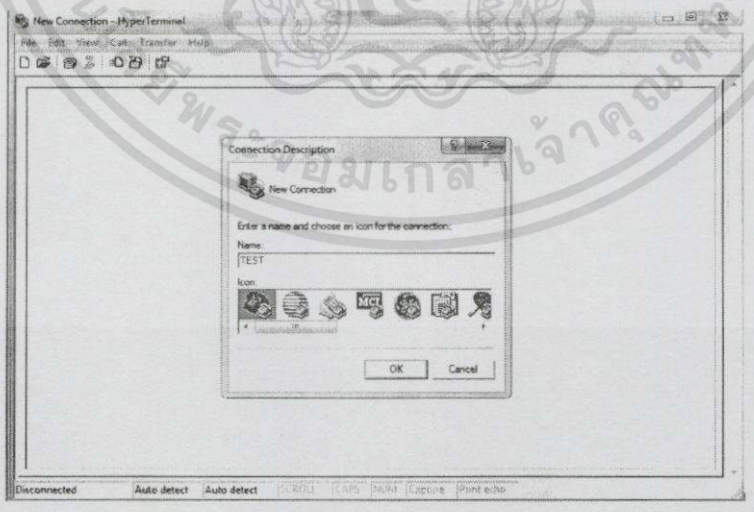
ต่อจากนั้นเมื่อทดสอบที่ตัวอุปกรณ์เพียงอย่างเดียวเสร็จเรียบร้อย ให้ทดลองว่าอุปกรณ์สามารถเชื่อมต่อกับคอมพิวเตอร์ได้หรือไม่และสามารถอ่านหรือเขียนค่าลงไปได้หรือไม่ โดยมีขั้นตอนดังต่อไปนี้

3.4.1 เปิดโปรแกรม HyperTerminal



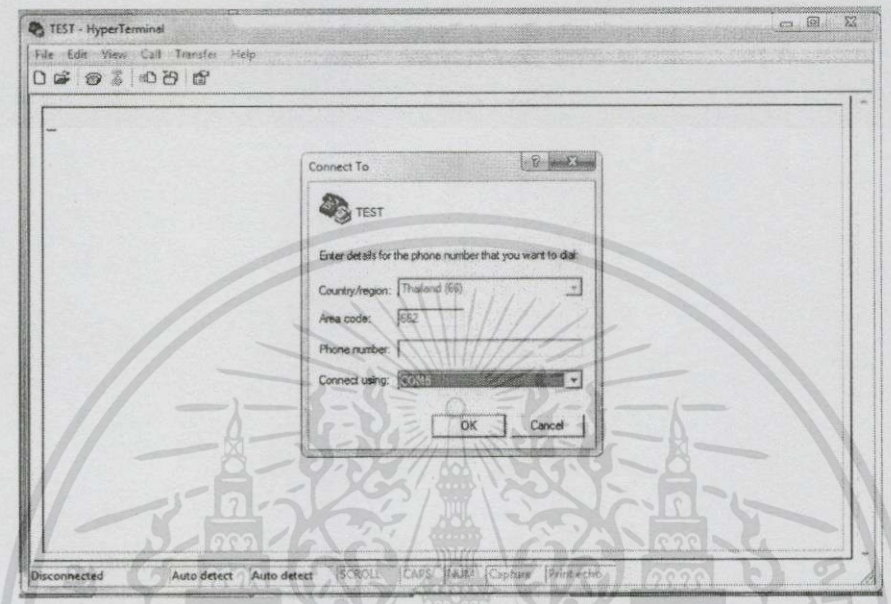
รูปที่ 3.41 ขั้นตอนการเปิดโปรแกรม HyperTerminal

3.4.2 ตั้งชื่อไฟล์ที่จะทำการทดสอบว่า Test แล้วกด OK



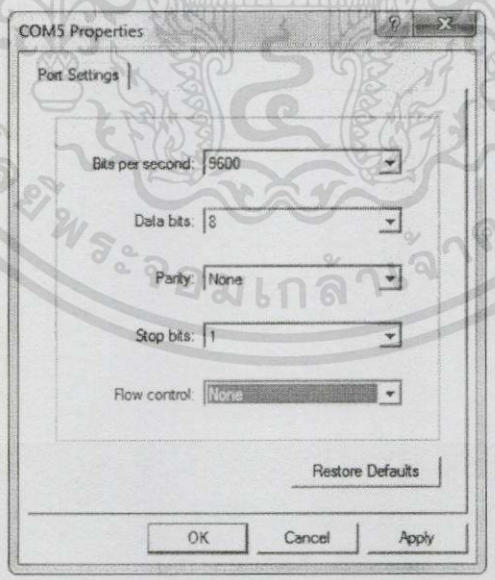
รูปที่ 3.42 สร้างไฟล์เพื่อทำการทดสอบ

3.4.3 เลือก comport ที่จะใช้ติดต่อให้ตรงกับ comport ของ RS232 ที่เราใช้โดยสามารถดูช่อง comportได้ที่ Device Manager ในคอมพิวเตอร์



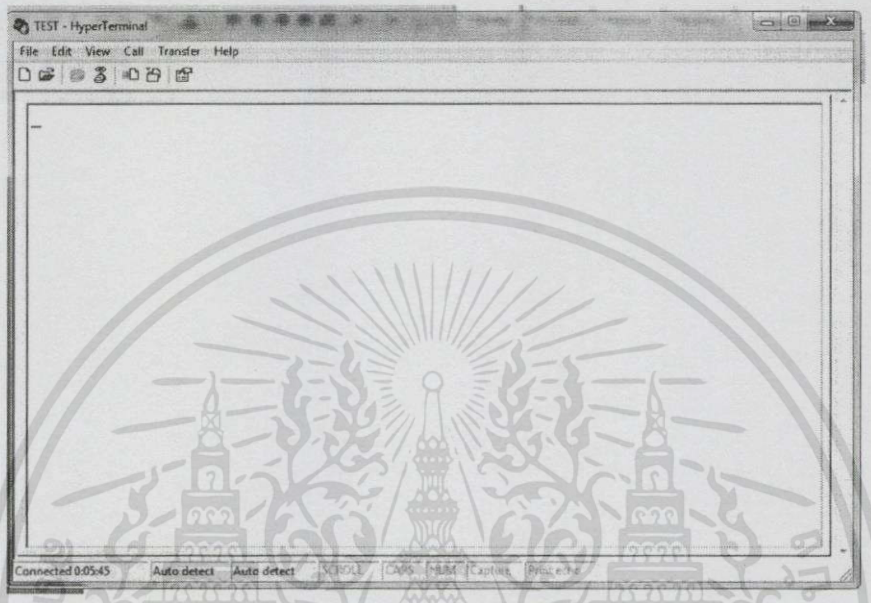
รูปที่ 3.43 เลือกช่องการติดต่อสื่อสารของ RS232

3.4.4 ตั้งค่าPort setting ตามรูปภาพ



รูปที่ 3.44 การตั้งค่า port

3.4.5 จะได้หน้าจอพร้อมที่จะอ่านค่าและส่งค่าไปที่ Microcontroller



รูปที่ 3.45 หน้าจอสำหรับทดสอบระบบ

3.4.6 ทำการทดสอบโดยการพิมพ์โปรโตคอลที่ใช้ในการติดต่อสื่อสาร เพื่อเป็นการทดสอบการรับส่งข้อมูลทั้งอนาลอกและดิจิตอล โดยมีโปรโตคอลที่ใช้และผลตอบสนองเป็นดังนี้

ตารางที่3.4 แสดงโปรโตคอลที่ใช้ในการทดสอบและผลตอบสนองของระบบ

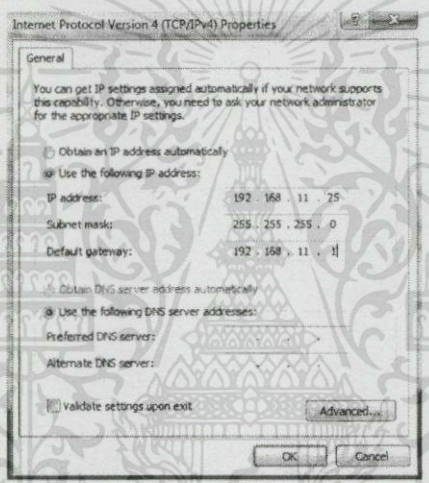
โปรโตคอลที่ใช้	ผลตอบสนองของระบบ
@00RA*XX	@00RA 1234 1200 1000 0005 *XX
@00RD*XX	@00RDllOO*XX
@00WA aaaa bbbb cccc dddd *XX	aaaa bbbb cccc dddd
@00WD00AC*XX	1010 1100

หมายเหตุ : RA คือ การอ่านค่าอนาลอกอินพุทจากอุปกรณ์
RD คือ การอ่านค่าดิจิตอลอินพุทจากอุปกรณ์
WA คือ การเขียนค่าอนาลอกเอาท์พุทลงไปในอุปกรณ์
WD คือ การเขียนค่าดิจิตอลเอาท์พุทลงไปในอุปกรณ์

3.5 การเชื่อมต่อกับเครือข่าย LAN-RS232 port

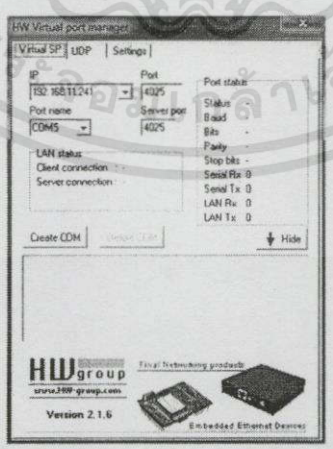
ตามขั้นตอนการดำเนินงานที่ได้กล่าวมาทั้งหมดในข้อ 3.1 ถึง 3.4 นั้นเป็นขั้นตอนในการทดลองโดยใช้ RS232 เป็นพอร์ทในการเชื่อมต่อและรับส่งข้อมูลระหว่างอุปกรณ์กับผู้ใช้งาน(คอมพิวเตอร์) แต่เมื่อต้องการให้ระบบมีการติดต่อสื่อสารที่ไกลมากขึ้นและสะดวกมากขึ้นเราจึงจำเป็นต้องเปลี่ยนจากสายส่งข้อมูล RS232 มาเป็นการเชื่อมต่อกับเครือข่ายเพื่อให้ได้ระยะในการติดต่อสื่อสารข้อมูลที่ไกลมากขึ้นโดยมีอุปกรณ์ที่รองรับการเชื่อมต่อกับเครือข่ายคือ Ethernet IO Board โดยใช้สายแลนในการรับส่งข้อมูลและมีสวิตช์แยกสำหรับแยกสายแลน(Hub)เพื่อเป็นจุดกลางในการสื่อสารของสายข้อมูลในระบบ โดยมีขั้นตอนการดำเนินงานดังนี้

3.5.1 เริ่มจากการตั้งค่า IP Address ของเครื่องคอมพิวเตอร์ โดยตั้งค่าเป็น 198.162.11.25 (จุดหลังสุดจะเป็นเลขอะไรก็ได้ยกเว้นเลข 20 และห้ามเกิน 255) ตั้งค่า subnet mark เป็น 255.255.255.0 และค่า Default Address เป็น 192.168.11.1



รูปที่ 3.51 การตั้งค่า IP Address ของคอมพิวเตอร์

3.5.2 จากนั้นเปิดโปรแกรม HW Virtual Serial Port >>> ค่า IP Address ที่เห็นนั้นเป็นค่า IP Address ของ Ethernet IO Board



รูปที่ 3.52 โปรแกรม HW Virtual Serial Port

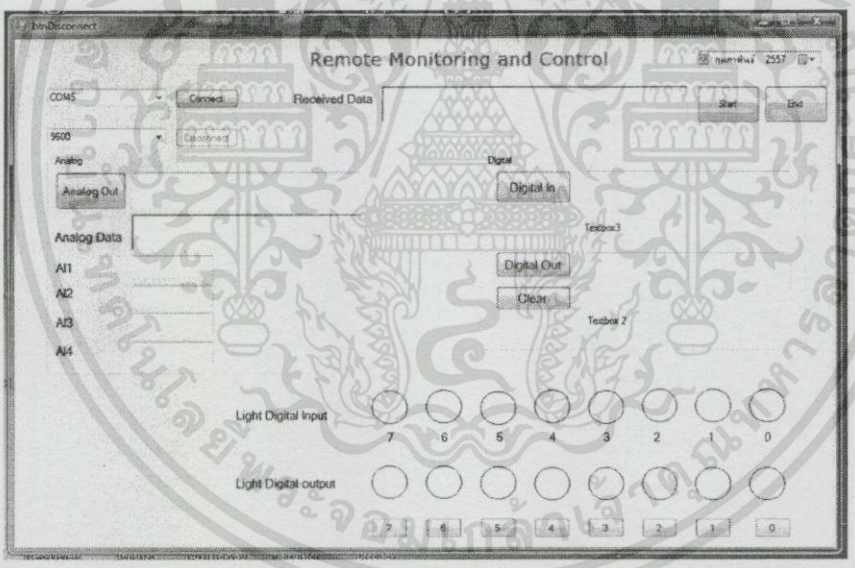
3.5.3 จากนั้นเลือก comport ที่ใช้ให้ตรงกันกับ comport ของ Ethernet IO Board แล้วกดตรงคำว่า Create com เพื่อเป็นการดูว่าในขณะนี้คอมพิวเตอร์กับ Ethernet IO Board อยู่ภายในวงแลนเดียวกันหรือยัง

3.5.4 ต่อมาทำการเปิดโปรแกรม HyperTerminal ขึ้นมาเพื่อเป็นการทดสอบระบบว่าคอมพิวเตอร์สามารถติดต่อสื่อสารรับส่งค่าข้อมูลกับอุปกรณ์ได้หรือไม่ โดยทำตามขั้นตอนเดิมในหัวข้อที่ 3.4 ถ้ามีผลตอบสนองกลับมาแสดงว่าคอมพิวเตอร์กับอุปกรณ์อยู่ในวงแลนเดียวกันและสามารถสื่อสารข้อมูลกันได้ ก็จะทำให้ระบบมีการติดต่อสื่อสารที่ไกลขึ้นกว่าเดิม

3.6 ขั้นตอนการทดสอบการรับส่งข้อมูลกับอุปกรณ์ทั้งหมด

หลังจากที่ทำการเขียนโปรแกรมทั้งในส่วนของหน้าคอนโทรล(กราฟิก)และหน้าคำสั่งการทำงานเสร็จเรียบร้อยแล้วก็มาถึงขั้นตอนในการทดลองการใช้งานว่าจะสามารถใช้งานได้ตามคำสั่งโปรแกรมที่เขียนไว้หรือไม่ ทำการอธิบายขั้นตอนการทดลองโดยสรุปได้ดังนี้

3.6.1 ทำการต่อสายพอร์ทอนุกรม RS232 เพื่อเป็นการเชื่อมต่ออุปกรณ์เข้ากับเครื่องคอมพิวเตอร์และเมื่อทำการเปิดโปรแกรม Visual Basic ขึ้นมาแล้วจะเจอหน้าโปรแกรมดังรูป



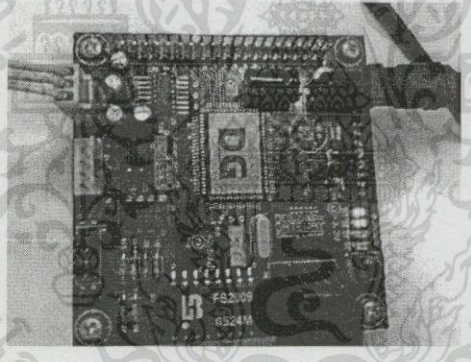
รูปที่ 3.61 หน้ากราฟิกโปรแกรม Visual Basic สำหรับผู้ใช้งาน

3.6.2 ทำการกดปุ่ม connect เพื่อเป็นการเชื่อมต่อ จากนั้นเริ่มต้นการทดลองโดยกดปุ่ม start หลังจากกดปุ่มไปแล้วจะหน่วงเวลาไป 5 วินาทีก่อนที่จะอ่านค่าข้อมูล >>เมื่อครบ 5 วินาทีแล้วสังเกตที่ช่อง Received Data จะทำการอ่านค่าข้อมูลอนาลอกอินพุทและดิจิตอลอินพุทสลับกันทุกๆ5วินาที >>โดยค่าของข้อมูลอนาลอกจะมีทั้งหมด 4 ช่องจะปรากฏค่าของแต่ละช่องแยกกันอยู่ที่ AI1 AI2 AI3 และAI4 ตามลำดับ ส่วนค่าข้อมูลดิจิตอลอินพุทนั้นต้องกดที่ปุ่ม Digital In ก่อนจึงจะแสดงค่าข้อมูลที่ช่อง Textbox 3

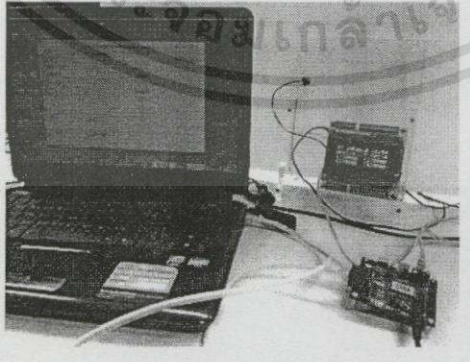
3.6.3 ในส่วนของการเขียนค่าข้อมูล(Write)ให้กับอุปกรณ์นั้นทำได้โดย >>ค่าข้อมูลอนาลอกเอาท์พุท ให้ทำการเขียนลงไปที่ช่องรับข้อมูลข้างๆคำว่า Analog Out โดยเขียนตามแบบฟอร์มของโปรโตคอลคือ @00WA aaaa bbbb cccc dddd *XX (โดยที่ a,b,c เป็นค่า 0-9)

3.6.4 การเขียนค่าข้อมูลดิจิตอลเอาท์พุทนั้นให้ดูจากด้านล่างของหน้ากราฟิก โดยข้อมูลดิจิตอลจะมีทั้งหมด 16 บิท อินพุท 8 บิท เอาท์พุท 8 บิท และจะแบ่งพิจารณาเป็น2ชุด ชุดละ4บิท จะสั่งค่าข้อมูลโดยปุ่มกด 0-7 ซึ่งแทนจำนวนบิตของข้อมูลดิจิตอล ค่าของแต่ละปุ่มจะเป็นไปตามเลขฐานสิบหก 0-F และเมื่อเขียนค่าข้อมูลลงไปแล้ว กดปุ่มDigital Out ค่าจะแสดงตามที่ผู้ใช้ต้องการที่จอแสดงผลของอุปกรณ์ โดยให้แสดงผลเป็นเลขฐานสองจะแสดงโดยการติดของไฟ ติด=1 , ไม่ติด=0 และทำการอ่านค่ากลับมาที่ Digital In ถ้าค่าที่อ่านได้ตรงกันไฟจะติดเหมือนกับ Digital Out

3.6.5 ถ้าต้องการให้ได้ระยะไกลมากขึ้นโดยการต่อกับเครือข่ายแลน ให้เปลี่ยนจากสาย RS232 มาเป็นสายแลน(โดยสายแลนจะต่ออยู่กับบอร์ด Ethernet IO ซึ่งเป็นอุปกรณ์ส่วนที่ใช้เชื่อมต่อกับแลน)และทำการตั้งค่าที่โปรแกรม Visual Basic ให้ตรงกันกับสายแลน จากนั้นก็ทำการทดลองตามขั้นตอนเดิมในข้อที่ 3.6.2 ใหม่อีกครั้ง ผลการทดลองที่ได้ก็จะเหมือนกันกับการใช้สาย RS232 ในการติดต่อสื่อสาร



รูปที่3.65 Ethernet IO Board



รูปที่ 3.66 ระบบโดยรวมเมื่อทำการเชื่อมต่อกับเครือข่ายแลน

บทที่ 4

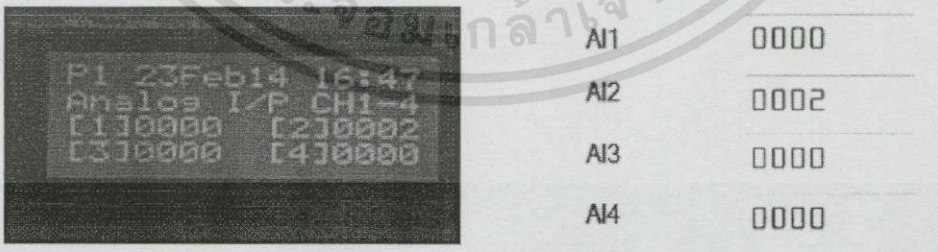
ผลการทดลอง

4.1 ผลการทดลองกับสัญญาณอนาลอก

จากผลการทดลองในการรับส่งค่าที่เป็นสัญญาณอนาลอก จะสังเกตพบว่าค่าข้อมูลของสัญญาณอนาลอกนั้นค่อนข้างที่จะถูกรบกวนได้ง่ายจากภายนอก จึงทำให้ค่าที่อ่านได้มีการแกว่งอยู่บ้างเล็กน้อยและเป็นค่าข้อมูลที่ไม่นิ่ง แต่ไม่ได้เป็นปัญหาในการทดลองเท่าไหร่นักเพราะเนื่องจากเน้นไปที่การติดต่อสื่อสารข้อมูลและสังเกตผลตอบสนองของระบบมากกว่า ส่วนค่าอนาลอกเอาท์พุทที่ได้จากการเขียนค่าลงไปไม่ค่อยโดนรบกวนเท่าไหร่นัก ทำให้ค่าที่เขียนลงไปค่อนข้างจะตรงตามที่ผู้ใช้งานต้องการและมีผลการทดลองในการรับและส่งค่าของสัญญาณอนาลอกดังนี้

ตารางที่ 4.1 : ผลการอ่านค่าสัญญาณอนาลอกอินพุทจากทั้ง 4 ช่องสัญญาณ

ครั้งที่	ผลตอบสนอง
1	0000 0003 0001 0000
2	0001 0000 0000 0002
3	0003 0000 0000 0001
4	0001 0001 0002 0000
5	0000 0000 0001 0001
6	0000 0003 0002 0001
7	0002 0000 0001 0002
8	0000 0000 0001 0003
9	0000 0003 0001 0000
10	0001 0002 0001 0001

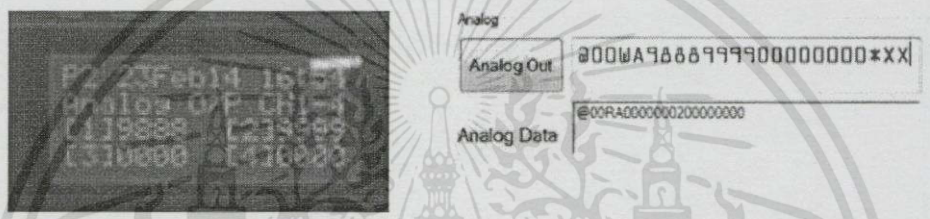


(ก) (ข)

รูปที่ 4.1 ผลการอ่านและแสดงค่าอนาลอกอินพุท
(ก) จากอุปกรณ์
(ข) จากโปรแกรม

ตารางที่ 4.2 : ผลการอ่านค่าสัญญาณอนาลอกอินพุทจากทั้ง 4 ช่องสัญญาณ

ครั้งที่	ค่าข้อมูลที่ส่งไป	ผลตอบสนอง
1	1111 2222 3333 4444	1111 2222 3333 4444
2	5555 6666 7777 8888	5555 6666 7777 8888
3	1010 2020 3030 4040	1010 2020 3030 4040
4	3434 5656 6767 7878	3434 5656 6767 7878
5	0099 8877 6655 4433	0099 8877 6655 4433

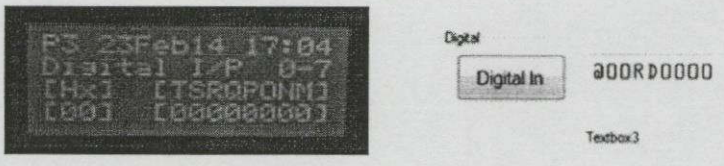


(ก) (ข)

รูปที่ 4.2 ผลการเขียนและแสดงค่าอนาลอกเอาท์พุท
(ก) จากอุปกรณ์
(ข) จากโปรแกรม

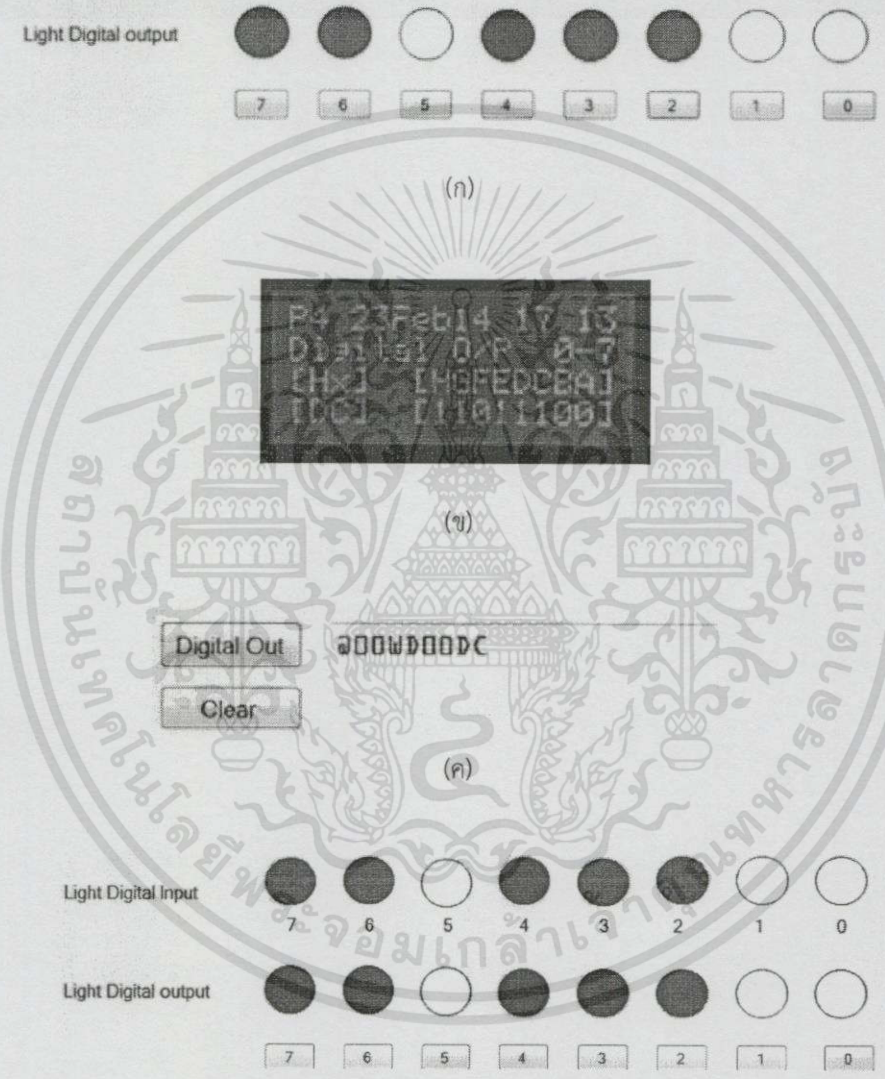
4.2 ผลการทดลองกับสัญญาณดิจิตอล

จากผลการทดลองในการรับส่งค่าของสัญญาณดิจิตอล จะพบว่าค่าข้อมูลของสัญญาณดิจิตอลที่ทำการอ่านค่าเข้ามาจะเป็น “00000000” เนื่องจากว่าสัญญาณดิจิตอลนั้นจะมีค่าแค่ “1” กับ “0” ดังนั้นสัญญาณรบกวนที่จะมาทำให้เกิดการเปลี่ยนแปลงข้อมูลของสัญญาณดิจิตอลจึงทำได้ยากกว่าสัญญาณอนาลอก และด้วยการทดลองนี้จะเน้นการรับส่งข้อมูลที่เป็นสัญญาณมาตรฐานเพื่อให้มีความอเนกประสงค์ในการนำไปประยุกต์ใช้ในด้านอื่นประกอบกับง่ายต่อการนำไปพัฒนาต่อ ค่าอินพุทของสัญญาณดิจิตอลจึงมีแต่ค่า “0” เพราะยังไม่มีการประยุกต์ใช้งานกับอุปกรณ์ใด แต่ค่าข้อมูลที่สั่งการทำงานหรือเป็นค่าข้อมูลดิจิตอลที่เขียนโดยผู้ใช้ สามารถเขียนลงไปในอุปกรณ์ให้ได้ค่าตามที่ต้องการได้ โดยสัญญาณดิจิตอลทั้งอินพุทและเอาท์พุทจะมีอยู่ 16 บิท ซึ่งมีผลการทดลองดังนี้



(ก) (ข)

รูปที่ 4.3 ผลการอ่านและแสดงค่าดิจิตอลอินพุท
(ก) จากอุปกรณ์
(ข) จากโปรแกรม



รูปที่ 4.4 ผลการเขียนและแสดงค่าดิจิตอลอินเอาท์พุท
(ก) ไฟแสดงค่าตามตำแหน่งบิท (ค) ผลจากโปรแกรม
(ข) ผลจากอุปกรณ์ (ง)ไฟแสดงค่ากลับมาที่ DI

บทที่ 5

สรุปผลการทดลองและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

ในปริญญานิพนธ์ฉบับนี้ ได้นำเสนอเรื่องการศึกษาระบบควบคุมระยะไกล โดยเน้นไปที่การติดต่อสื่อสารข้อมูลและการสังเกตผลตอบสนองของระบบผ่านอุปกรณ์บอร์ดทดลองโดยมีไมโครคอนโทรลเลอร์ Atmel AT89C4051 เป็นอุปกรณ์ตัวหลักควบคุมการติดต่อสื่อสารข้อมูลโดยใช้โปรแกรม Visual Basic 2010 เป็นส่วนแสดงผลและติดต่อกับส่วนผู้ใช้งาน เพื่อบอกสถานะการทำงานและเป็นการศึกษาเกี่ยวกับการติดต่อสื่อสารของข้อมูล เนื่องจากว่าค่าอินพุทและเอาท์พุทที่ทำการรับส่งในระบบเป็นสัญญาณมาตรฐานทั้งสัญญาณอนาลอกและสัญญาณดิจิตอล จึงทำให้สามารถที่จะนำไปพัฒนาต่อในด้านต่างๆได้ง่ายเพราะเป็นพื้นฐานในการติดต่อสื่อสารข้อมูล โดยในชุดอุปกรณ์นี้จะทำการควบคุมการติดต่อสื่อสารเพื่อทำการรับส่งค่าข้อมูลที่ได้จากภายนอกและจากผู้ใช้โดยผ่านสายที่เป็นพอร์ทอนุกรม RS232 และสร้างแอพพลิเคชั่นโดยใช้โปรแกรม Visual Basic 2010 ทำให้สามารถดูการทำงานของระบบได้ นอกจากนั้นยังสามารถพัฒนาต่อไปเพื่อให้เป็นการควบคุมระยะไกลมากขึ้นและสะดวกในการใช้งานเมื่อไม่ได้อยู่ในบริเวณอุปกรณ์ โดยการนำอุปกรณ์และคอมพิวเตอร์ไปเชื่อมต่อกับเครือข่ายแลน ทำให้มีความสะดวกสบายและรวดเร็วในการรับส่งข้อมูลมากขึ้น

จากผลการศึกษาและทดลองการรับส่งค่าข้อมูลผ่านอุปกรณ์คอนโทรลเลอร์ทำให้พบว่า ชุดอุปกรณ์และโปรแกรมสามารถใช้งานได้จริงและมีประสิทธิภาพในการติดต่อสื่อสารข้อมูลได้ค่อนข้างดี และสามารถนำโปรแกรมควบคุมนี้ไปพัฒนาต่อและประยุกต์ใช้งานในการรับส่งค่าเพื่อควบคุมอุปกรณ์ในลักษณะอื่นอีกได้หลายอย่าง

5.2 ข้อเสนอแนะ

ในขั้นตอนของการค้นคว้าศึกษาหาข้อมูลนั้นควรที่จะทราบก่อนว่าในชุดอุปกรณ์ทดลองมีอะไรเป็นส่วนประกอบบ้าง และแต่ละตัวทำหน้าที่อะไร เพื่อให้เรียนรู้การทำงานของชุดอุปกรณ์ได้รวดเร็วยิ่งขึ้น และเมื่อได้ทำการทดลองแล้วจะพบว่า ในบางครั้งการติดต่อสื่อสารข้อมูลระหว่างชุดอุปกรณ์กับโปรแกรม Visual Basic 2010 นั้นเมื่อทำการเชื่อมต่อกันแล้วด้วยพอร์ทอนุกรม RS232 ยังพบว่ามีช่วงเวลาที่หน่วงของผลตอบสนองอยู่ อาจเนื่องด้วยว่ามีการประมวลผลได้ไม่เร็วนัก อาจทำการแก้ไขปัญหานี้โดยการเพิ่มระยะเวลาให้กับ Timer ในโปรแกรมเพื่อให้มีช่วงเวลาในการรับผลตอบสนองที่ชัดมากขึ้น

5.3 อุปสรรคและปัญหา

เนื่องด้วยว่าปริญญานิพนธ์เรื่องนี้ว่าด้วยเรื่องการศึกษาระบบของระบบการควบคุมระยะไกล (Remote Control) จึงทำให้การทำงานส่วนมากจะอยู่ที่การศึกษาหาข้อมูล บางครั้งข้อมูลที่ต้องการนั้นค่อนข้างหาได้ยากทั้งจากหนังสือและอินเทอร์เน็ต และด้วยการรับส่งข้อมูลที่มีทั้งแบบอนาลอกและดิจิตอล ในส่วนของข้อมูลอนาลอกนั้นจะสังเกตได้ว่ามีการเปลี่ยนแปลงของข้อมูลอยู่ตลอดเวลา เพราะสัญญาณอนาลอกจะถูกรบกวนได้ง่ายจากสิ่งภายนอก ทำให้ค่าที่อ่านได้จะค่อนข้างแกว่งอยู่

เล็กน้อยแต่ไม่ถึงกับเป็นปัญหาเท่าไหร่นัก ในการทดลองบางครั้งโปรแกรมไม่สามารถประมวลผลได้ หรือถ้าประมวลผลได้ก็จะมีการหน่วงเวลาของช่วงเวลาในการรอผลตอบสนอง อาจด้วยเพราะว่าเปิดใช้งานเป็นเวลานานติดต่อกันโดยไม่มีช่วงเวลาพักหรืออาจเพราะว่าในการส่งสัญญาณแต่ละครั้งนั้นต้องการเวลาในการตรวจสอบสายสัญญาณ เลยอาจทำให้โปรแกรมเกิดการรวนอยู่บ้าง และปัญหาหลักๆส่วนมากจะอยู่ที่การเขียนโปรแกรม เนื่องจากเป็นโปรแกรมที่ผู้ทำปริญญานิพนธ์ไม่เคยศึกษาและทดลองใช้งานมาก่อน จึงทำให้ต้องใช้ระยะเวลานานในการศึกษา ทดลองผิดลองถูกในการเขียนโปรแกรมเพื่อให้เชื่อมต่อกับชุดอุปกรณ์รวมถึงการเขียนให้รับและส่งค่าข้อมูล

บรรณานุกรม

http://koonkrujiraporn.blogspot.com/2011/07/2_20.html

http://opsdapp.mod.go.th/autoweb/autopost/upload/100-010-moring 2556-02-14_1405.pdf

http://www.il.mahidol.ac.th/e-media/computer/network/net_datacom3.htm

https://sites.google.com/site/websitraywichang3204ldawaly/hnwy-kar-reiyn-ru-thi-2/2-1prapheth-khxng-kherux-khay-lan-wan-pan-man

ภาคผนวก

เกี่ยวกับภาคผนวก

Code Program การทำงานของอุปกรณ์

```
VARIABLE DECLARATION:   MCON            EQU 0C6H
                        TA              EQU 0C7H
                        MIN             EQU 20          ; @0.5 SEC
                        LCD_DTA         EQU P1
                        LCD_RS          EQU P1.5
                        LCD_EN          EQU P1.4
                        I2CDAT          EQU P1.7
                        I2CCLK          EQU P1.6
                        LD4728          EQU P3.3
                        SECOND_ADDR     EQU 00H         ;1BYTE
                        MINUTE_ADDR     EQU 01H         ;1BYTE
                        HOUR_ADDR       EQU 02H         ;1BYTE
                        DAY_ADDR        EQU 03H         ;1BYTE
                        DATE_ADDR       EQU 04H         ;1BYTE
                        MONTH_ADDR      EQU 05H         ;1BYTE
                        YEAR_ADDR       EQU 06H         ;1BYTE
STATION                 EQU 00H                         ;BYTE
PCA9671                 EQU 40H                         ;BYTE
MCP4728                 EQU 0C0H                        ;BYTE
VARIABLE INTERNAL RAM:  CNTL    EQU 10H
                        CNTH    EQU 11H
                        BYTE    EQU 12H
                        LINEPOI EQU 13H
                        SUM     EQU 14H
                        MINUTE  EQU 15H
                        HEXBUF  EQU 16H             ;2 BYTES
                        MULFG   EQU 18H             ;BYTE
                        SEC_FLG EQU 19H     ;BYTE
                        MONI_PAGE EQU 1AH   ;BYTE MONITORING PAGE
                        DIGITIN EQU 1BH     ;BYTE
                        DIGITOUT EQU 1CH    ;BYTE
                        I2CBYTE1 EQU 1DH    ;BYTE
```

```
I2CBYTE2    EQU 1EH     ;BYTE
I2CBYTE3    EQU 1FH     ;BYTE
ANIN1       EQU 20H     ;2BYTE
ANIN2       EQU 22H     ;2BYTE
ANIN3       EQU 24H     ;2BYTE
ANIN4       EQU 26H     ;2BYTE
ANOUT1      EQU 28H     ;2BYTE
ANOUT2      EQU 2AH     ;2BYTE
ANOUT3      EQU 2CH     ;2BYTE
ANOUT4      EQU 2EH     ;2BYTE
CLOCK_BUF   EQU 30H     ;ARRAY[1..8] OF BYTE
AUXREG1     EQU 38H     ;BYTE
AUXREG2     EQU 39H     ;BYTE
ANAHEX1     EQU 3AH     ;BYTE
ANAHEX2     EQU 3BH     ;BYTE
SECFLG      EQU 3CH     ;BYTE
NBHI        EQU 3DH     ;BYTE
NBLO        EQU 3EH     ;BYTE
INP_RSLT    EQU 3FH     ;BYTE
LINEIN      EQU 40H     ;ARRAY [1..32] OF BYTE
INP_PRS     EQU 5EH     ;BYTE
INP_LST     EQU 5FH     ;BYTE
STACK       EQU 60H
            ORG 0000H
            LJMP START
            ORG 0003H
EX0_INT:    RETI
            ORG 000BH
TF0_INT:    JMP INT0_SERV
            ORG 0013H
EX1_INT:    RETI
            ORG 001BH
TF1_INT:    RETI
            ORG 0023H
SR_INT:     JMP SR_SERV
            ORG 0030H
EX1_SERV:   RETI
            ORG 0100H
```

```
TIMER [0] INTERRUPT SERVICE ROUTINE
INT0_SERV:      PUSH ACC
                PUSH DPH
                PUSH DPL
                SETB RS0
                MOV DPL,CNTL
                MOV DPH,CNTH
                INC DPTR
                MOV A,DPL
                ORL A,DPH
                JZ SEC_OPR
                MOV CNTL,DPL
                MOV CNTH,DPH
SEC_RET:        CLR RS0
                POP DPL
                POP DPH
                POP ACC
                RETI
SEC_OPR:        CALL SECFLAG
                MOV DPTR,#0FFFFH-1800     ;1800 @1 SEC  54000 @15 SEC
                MOV CNTL,DPL
                MOV CNTH,DPH
                CLR RS0
                POP DPL
                POP DPH
                POP ACC
                RETI
SECFLAG:        MOV A,SEC_FLG
                XRL A,#0FFH
                JZ SEC0
                MOV SEC_FLG,#0FFH
                CALL READ_RTCDT
                CALL READ_DIGITAL
                CALL WRITE_DIGITAL
                CALL READ_ANALOG
                CALL WRITE_ANALOG
                CALL MONITOR
                RET
```

```
SEC0:           MOV SEC_FLG,#0
                CALL READ_DIGITAL
                CALL WRITE_DIGITAL
                CALL READ_ANALOG
                CALL WRITE_ANALOG
                CALL MONITOR
                CALL BLINKOFF
                RET
BLINKOFF:       MOV A,MONI_PAGE
                XRL A,#0
                JZ BLINKRET
                MOV A,#8DH
                CALL LCDCNT
                MOV A,#' '
                CALL CHAR_WR
BLINKRET:       RET
LCD SUBROUTINE & PARAMETER MONITORING
INITLCD:        CALL DLY10MS
                CALL DLY10MS
                MOV A,#03H          ;INITIAL SET 8 BIT
                CALL LCDSND
                CALL DLY10MS
                MOV A,#03H
                CALL LCDSND
                CALL DLY1MS
                MOV A,#03H
                CALL LCDSND
                CALL DLY1MS
                MOV A,#02H
                CALL LCDSND
                CALL DLY40US
                MOV A,#28H
                CALL LCDCNT
                CALL DLY40US
                MOV A,#0CH
                CALL LCDCNT
                CALL DLY5MS
                MOV A,#06H
```

```
                CALL LCDCNT
                CALL DLY5MS
                MOV A,#80H
                CALL LCDCNT
                CALL DLY100US
                RET
DLY40US:        MOV R7,#40          ;DELAY USEC
                DJNZ R7,$
                RET
DLY100US:       MOV R7,#1           ;DELAY USEC
D100U:          MOV R6,#100
                DJNZ R6,$
                DJNZ R7,D100U
                RET
DLY1MS:         MOV R7,#10          ;DELAY 10X100=1000 USEC
D1MS:           MOV R6,#100
                DJNZ R6,$
                DJNZ R7,D1MS
                RET
DLY5MS:         MOV R7,#50          ;DELAY 50X100=5000 USEC
D5MS:           MOV R6,#100
                DJNZ R6,$
                DJNZ R7,D5MS
                RET
DLY10MS:        MOV R7,#100         ;DELAY 100X100=10000 USEC
D10MS:          MOV R6,#100
                DJNZ R6,$
                DJNZ R7,D10MS
                RET
MASGE_WR:       MOV A,#0H
                MOVC A,@A+DPTR
                CJNE A,#0DH,LCDWR
                RET
LCDWR:          CALL CHAR_WR
                INC DPTR
                JMP MASGE_WR
CHAR_WR:        MOV NBLO,A          ;LO NIBBLE
                SWAP A
```

```
                MOV NBHI,A          ;HI NIBBLE
                ANL NBHI,#0FH
                ANL NBLO,#0FH
                SETB LCD_RS         ;4BIT DATA WR
                NOP
                NOP
                MOV A,P1
                ANL A,#0F0H
                ORL A,NBHI
                MOV P1,A
                NOP
                SETB LCD_EN
                NOP                 ;\ Positive age(+) to enable
                NOP
                NOP
                CLR  LCD_EN         ;\ OUTPUT PORT.
                CALL DLY40US
                NOP
                NOP
                SETB LCD_RS         ;4BIT DATA WR
                NOP
                NOP
                MOV A,P1
                ANL A,#0F0H
                ORL A,NBLO
                MOV P1,A
                NOP
                SETB LCD_EN
                NOP                 ;\ Positive age(+) to enable
                NOP
                NOP
                CLR  LCD_EN         ;\ OUTPUT PORT
                CALL DLY100US
                RET
LCDCNT:         MOV NBLO,A          ;LO NIBBLE
                SWAP A
                MOV NBHI,A          ;HI NIBBLE
                ANL NBHI,#0FH
```

```
        ANL NBLO,#0FH
        CLR LCD_RS          ;4BIT INSTRUCTION WR
        NOP
        NOP
        MOV A,P1
        ANL A,#0F0H
        ORL A,NBHI
        MOV P1,A
        NOP
        SETB LCD_EN         ;\ Positive age(+) to enable
        NOP
        NOP
        NOP
        CLR  LCD_EN         ;\ OUTPUT PORT.
        CALL DLY40US
        NOP
        NOP
        CLR LCD_RS          ;4BIT INSTRUCTION WR
        NOP
        NOP
        MOV A,P1
        ANL A,#0F0H
        ORL A,NBLO
        MOV P1,A
        NOP
        SETB LCD_EN
        NOP                 ;\ Positive age(+) to enable
        NOP
        NOP
        CLR  LCD_EN         ;\ OUTPUT PORT
        CALL DLY100US
        RET
LCDSND: MOV NBLO,A          ;FIRST NIBBLE
        SWAP A
        MOV NBHI,A          ;SECOND NIBBLE
        ANL NBHI,#0FH
        ANL NBLO,#0FH
        CLR LCD_RS          ;4BIT INSTRUCTION WR
```

```
                NOP
                NOP
                MOV A,P1
                ANL A,#0F0H
                ORL A,NBLO
                MOV P1,A
                NOP
                SETB LCD_EN             ;\ Positive age(+) to enable
                NOP
                NOP
                NOP
                CLR  LCD_EN             ;\ OUTPUT PORT.
                RET
SETDD1:         MOV A,#10000000B        ;SET LINE1 (80)
                CALL LCDCNT
                RET
SETDD2:         MOV A,#11000000B        ;SET LINE2 (C0)
                CALL LCDCNT
                RET
SETDD3:         MOV A,#10010000B        ;SET LINE3 (90)
                CALL LCDCNT
                RET
SETDD4:         MOV A,#11010000B        ;SET LINE4 (D0)
                CALL LCDCNT
                RET
LCD_CLOCK:      MOV A,CLOCK_BUF+2       ;HR
                CALL HEXASCI
                MOV R3,A
                MOV A,R2
                CALL CHAR_WR
                MOV A,R3
                CALL CHAR_WR
                MOV A,#':'
                CALL CHAR_WR
                MOV A,CLOCK_BUF+1       ;MIN
                CALL HEXASCI
                MOV R3,A
                MOV A,R2
```

```
                CALL CHAR_WR
                MOV A,R3
                CALL CHAR_WR
                RET
LCD_DATE:       MOV A,CLOCK_BUF+4       ;DATE
                CALL HEXASCI
                MOV R3,A
                MOV A,R2
                CALL CHAR_WR
                MOV A,R3
                CALL CHAR_WR
                MOV A,CLOCK_BUF+5       ;MONTH
                CALL BCDBIN
                DEC A
                MOV B,#3
                MUL AB
                MOV R2,A
                MOV DPTR,#MONTHCHR
                MOVC A,@A+DPTR
                CALL CHAR_WR
                INC R2
                MOV A,R2
                MOVC A,@A+DPTR
                CALL CHAR_WR
                INC R2
                MOV A,R2
                MOVC A,@A+DPTR
                CALL CHAR_WR
                MOV A,CLOCK_BUF+6       ;YEAR
                CALL HEXASCI
                MOV R3,A
                MOV A,R2
                CALL CHAR_WR
                MOV A,R3
                CALL CHAR_WR
                RET
ANALOG DISPLAY 4 DIGIT xxxx
CHKMINUS:       MOV R6,DPH
```

```
                MOV R7,DPL
                MOV R2,#7FH
                MOV R3,#0FFH
                CALL DPSUB
                JNC ANA_MINUS           ;IF > 7FFFH THEN MINUS
                MOV DPH,R6
                MOV DPL,R7
                RET
ANA_MINUS:      MOV A,R6
                CPL A
                MOV DPH,A
                MOV A,R7
                CPL A
                MOV DPL,A
                RET
ANA_DISP:       CALL CHKMINUS
                CALL HTOD
                MOV A,HEXBUF
                CALL HEXASCI
                MOV R3,A
                MOV A,R2
                CALL CHAR_WR
                MOV A,R3
                CALL CHAR_WR
                MOV A,HEXBUF+1
                CALL HEXASCI
                MOV R3,A
                MOV A,R2
                CALL CHAR_WR
                MOV A,R3
                CALL CHAR_WR
                RET
LCD_ANALOG1:    MOV DPH,ANIN1
                MOV DPL,ANIN1+1
                CALL ANA_DISP
                RET
LCD_ANALOG2:    MOV DPH,ANIN2
                MOV DPL,ANIN2+1
```

```
                CALL ANA_DISP
                RET
LCD_ANALOG3:    MOV DPH,ANIN3
                MOV DPL,ANIN3+1
                CALL ANA_DISP
                RET
LCD_ANALOG4:    MOV DPH,ANIN4
                MOV DPL,ANIN4+1
                CALL ANA_DISP
                RET

ANALOG OUT LCD DISPLAY
LCD_ANALOG5:    MOV DPH,ANOUT1
                MOV DPL,ANOUT1+1
                CALL ANA_DISP
                RET
LCD_ANALOG6:    MOV DPH,ANOUT2
                MOV DPL,ANOUT2+1
                CALL ANA_DISP
                RET
LCD_ANALOG7:    MOV DPH,ANOUT3
                MOV DPL,ANOUT3+1
                CALL ANA_DISP
                RET
LCD_ANALOG8:    MOV DPH,ANOUT4
                MOV DPL,ANOUT4+1
                CALL ANA_DISP
                RET
HEADER_WR:      CALL SETDD1
                MOV DPTR,#MASG_HD1
                CALL MASGE_WR
                CALL SETDD2
                MOV DPTR,#MASG_HD2
                CALL MASGE_WR
                CALL SETDD3
                MOV DPTR,#MASG_HD3
                CALL MASGE_WR
                CALL SETDD4
```

```
                MOV DPTR,#MASG_HD4
                CALL MASGE_WR
                RET
MASG_HD1:   DB '**** FACCON ****',0DH
MASG_HD2:   DB '    REMOTE    ',0DH
MASG_HD3:   DB 'ANALOG & DIGITAL',0DH
MASG_HD4:   DB 'RS232--->9600N81',0DH
MONI_PAGE1:     CALL SETDD1
                MOV DPTR,#MASG_HD5
                CALL MASGE_WR
                CALL LCD_DATE
                MOV A,#' '
                CALL CHAR_WR
                CALL LCD_CLOCK
                CALL SETDD2
                MOV DPTR,#MASG_HD6
                CALL MASGE_WR
                CALL SETDD3
                MOV DPTR,#MASG_HD7
                CALL MASGE_WR
                CALL LCD_ANALOG1
                MOV DPTR,#MASG_HD71
                CALL MASGE_WR
                CALL LCD_ANALOG2
                CALL SETDD4
                MOV DPTR,#MASG_HD8
                CALL MASGE_WR
                CALL LCD_ANALOG3
                MOV DPTR,#MASG_HD81
                CALL MASGE_WR
                CALL LCD_ANALOG4
                RET
MASG_HD5:       DB 'P1 ',0DH
MASG_HD6:       DB 'Analog I/P CH1-4',0DH
MASG_HD7:       DB '[1]',0DH
MASG_HD71:      DB '  [2]',0DH
MASG_HD8:       DB '[3]',0DH
MASG_HD81:      DB '  [4]',0DH
```

```
MONI_PAGE2:     CALL SETDD1
                MOV DPTR,#MASG_HD9
                CALL MASGE_WR
                CALL LCD_DATE
                MOV A,#' '
                CALL CHAR_WR
                CALL LCD_CLOCK
                CALL SETDD2
                MOV DPTR,#MASG_HD10
                CALL MASGE_WR
                CALL SETDD3
                MOV DPTR,#MASG_HD11
                CALL MASGE_WR
                CALL LCD_ANALOG5
                MOV DPTR,#MASG_HD111
                CALL MASGE_WR
                CALL LCD_ANALOG6
                CALL SETDD4
                MOV DPTR,#MASG_HD12
                CALL MASGE_WR
                CALL LCD_ANALOG7
                MOV DPTR,#MASG_HD121
                CALL MASGE_WR
                CALL LCD_ANALOG8
                RET
MASG_HD9:       DB 'P2 ',0DH
MASG_HD10:      DB 'Analog O/P CH1-4',0DH
MASG_HD11:      DB '[1]',0DH
MASG_HD111:     DB '  [2]',0DH
MASG_HD12:      DB '[3]',0DH
MASG_HD121:     DB '  [4]',0DH
MONI_PAGE3:     CALL SETDD1
                MOV DPTR,#MASG_HD13
                CALL MASGE_WR
                CALL LCD_DATE
                MOV A,#' '
                CALL CHAR_WR
                CALL LCD_CLOCK
```

```
                CALL SETDD2
                MOV DPTR,#MASG_HD14
                CALL MASGE_WR
                CALL SETDD3
                MOV DPTR,#MASG_HD15
                CALL MASGE_WR
                CALL SETDD4
                MOV A,#'['
                CALL CHAR_WR
                MOV A,DIGITIN
                CALL HEXASCI
                MOV R3,A
                MOV A,R2
                CALL CHAR_WR
                MOV A,R3
                CALL CHAR_WR
                MOV DPTR,#MASG_HD16
                CALL MASGE_WR
                MOV A,DIGITIN
                CALL LCD_DISPBIT
                MOV A,#']'
                CALL CHAR_WR
                RET
MASG_HD13:      DB 'P3 ',0DH
MASG_HD14:      DB 'Digital I/P  0-7',0DH
MASG_HD15:      DB '[Hx]  [TSRQPONM]',0DH
MASG_HD16:      DB ']  [',0DH
LCD_DISPBIT:    MOV R4,#8
LCD_BITLP:      RLC A
                JC LCD_BIT1
                PUSH ACC
                MOV A,#'0'
                CALL CHAR_WR
                POP ACC
                DJNZ R4,LCD_BITLP
                RET
LCD_BIT1:       PUSH ACC
                MOV A,#'1'
```

```
                CALL CHAR_WR
                POP ACC
                DJNZ R4,LCD_BITLP
                RET
MONI_PAGE4:     CALL SETDD1
                MOV DPTR,#MASG_HD17
                CALL MASGE_WR
                CALL LCD_DATE
                MOV A,#' '
                CALL CHAR_WR
                CALL LCD_CLOCK
                CALL SETDD2
                MOV DPTR,#MASG_HD18
                CALL MASGE_WR
                CALL SETDD3
                MOV DPTR,#MASG_HD19
                CALL MASGE_WR
                CALL SETDD4
                MOV A,#'['
                CALL CHAR_WR
                MOV A,DIGITOUT
                CALL HEXASCI
                MOV R3,A
                MOV A,R2
                CALL CHAR_WR
                MOV A,R3
                CALL CHAR_WR
                MOV DPTR,#MASG_HD16
                CALL MASGE_WR
                MOV A,DIGITOUT
                CALL LCD_DISPBIT
                MOV A,#']'
                CALL CHAR_WR
                RET
MASG_HD17:      DB 'P4 ',0DH
MASG_HD18:      DB 'Digital O/P  0-7',0DH
MASG_HD19:      DB '[Hx]  [HGFEDCBA]',0DH
MASG_HD20:      DB ']  [',0DH
```

```
MONI_PAGE5:     CALL SETDD1
                MOV DPTR,#MASG_HD21
                CALL MASGE_WR
                CALL LCD_DATE
                MOV A,#' '
                CALL CHAR_WR
                CALL LCD_CLOCK
                CALL SETDD2
                MOV DPTR,#MASG_HD22
                CALL MASGE_WR
                CALL SETDD3
                MOV DPTR,#MASG_HD23
                CALL MASGE_WR
                CALL LCD_ANALOG1
                MOV DPTR,#MASG_HD231
                CALL MASGE_WR
                CALL LCD_ANALOG5
                CALL SETDD4
                MOV DPTR,#MASG_HD24
                CALL MASGE_WR
                CALL LCD_ANALOG2
                MOV DPTR,#MASG_HD241
                CALL MASGE_WR
                CALL LCD_ANALOG6
                RET
MASG_HD21:      DB 'P5 ',0DH
MASG_HD22:      DB 'Ana I/P |Ana O/P',0DH
MASG_HD23:      DB '[1]',0DH
MASG_HD231:     DB ' |[1]',0DH
MASG_HD24:      DB '[2]',0DH
MASG_HD241:     DB ' |[2]',0DH
MONI_PAGE6:     CALL SETDD1
                MOV DPTR,#MASG_HD25
                CALL MASGE_WR
                CALL LCD_DATE
                MOV A,#' '
                CALL CHAR_WR
                CALL LCD_CLOCK
```

```
                CALL SETDD2
                MOV DPTR,#MASG_HD26
                CALL MASGE_WR
                CALL SETDD3
                MOV DPTR,#MASG_HD27
                CALL MASGE_WR
                CALL LCD_ANALOG3
                MOV DPTR,#MASG_HD271
                CALL MASGE_WR
                CALL LCD_ANALOG7
                CALL SETDD4
                MOV DPTR,#MASG_HD28
                CALL MASGE_WR
                CALL LCD_ANALOG4
                MOV DPTR,#MASG_HD281
                CALL MASGE_WR
                CALL LCD_ANALOG8
                RET
MASG_HD25:      DB 'P6 ',0DH
MASG_HD26:      DB 'Ana I/P |Ana O/P',0DH
MASG_HD27:      DB '[3]',0DH
MASG_HD271:     DB ' |[3]',0DH
MASG_HD28:      DB '[4]',0DH
MASG_HD281:     DB ' |[4]',0DH
MONI_PAGE7:     CALL SETDD1
                MOV DPTR,#MASG_HD29
                CALL MASGE_WR
                CALL LCD_DATE
                MOV A,#' '
                CALL CHAR_WR
                CALL LCD_CLOCK
                CALL SETDD2
                MOV DPTR,#MASG_HD30
                CALL MASGE_WR
                CALL SETDD3
                MOV DPTR,#MASG_HD31
                CALL MASGE_WR
                MOV A,DIGITIN
```

```
                CALL LCD_DISPBIT
                MOV A,#'['
                CALL CHAR_WR
                MOV A,DIGITIN
                CALL HEXASCI
                MOV R3,A
                MOV A,R2
                CALL CHAR_WR
                MOV A,R3
                CALL CHAR_WR
                MOV A,#']'
                CALL CHAR_WR
                CALL SETDD4
                MOV DPTR,#MASG_HD32
                CALL MASGE_WR
                MOV A,DIGITOUT
                CALL LCD_DISPBIT
                MOV A,#'['
                CALL CHAR_WR
                MOV A,DIGITOUT
                CALL HEXASCI
                MOV R3,A
                MOV A,R2
                CALL CHAR_WR
                MOV A,R3
                CALL CHAR_WR
                MOV A,#']'
                CALL CHAR_WR
                RET
MASG_HD29:      DB 'P7 ',0DH
MASG_HD30:      DB 'Digital I/O   ',0DH
MASG_HD31:      DB 'I/P ',0DH
MASG_HD32:      DB 'O/P ',0DH
MONITOR:        MOV A,MONI_PAGE
                CJNE A,#0,MONI1
                JMP MONI_PAGE0
MONI1:          CJNE A,#1,MONI2
                JMP MONI_PAGE1
```

```
MONI2:      CJNE A,#2,MONI3
            JMP MONI_PAGE2
MONI3:      CJNE A,#3,MONI4
            JMP MONI_PAGE3
MONI4:      CJNE A,#4,MONI5
            JMP MONI_PAGE4
MONI5:      CJNE A,#5,MONI6
            JMP MONI_PAGE5
MONI6:      CJNE A,#6,MONI7
            JMP MONI_PAGE6
MONI7:      CJNE A,#7,MONI8
            JMP MONI_PAGE7
MONI8:      MOV MONI_PAGE,#0
            RET
```

GENERAL I2C SUBROUTINE

```
I2C_START:  CLR I2CDAT          ;START CONDITION
            CALL I2CDLY
            CLR  I2CCLK
            CALL I2CDLY2
            RET
I2C_STOP:   SETB I2CCLK         ;STOP CONDITION
            CALL I2CDLY
            SETB I2CDAT
            CALL I2CDLY2
            RET
I2C_IDLE:   CALL I2CDLY
            SETB I2CDAT         ;IDEL CONDITION
            SETB I2CCLK
            CALL I2CDLY2
            RET
I2C_RESTART: SETB I2CCLK        ;STOP CONDITION
            CALL I2CDLY
            SETB I2CDAT
            CALL I2CDLY2
            CALL I2CDLY2
            CLR I2CDAT          ;START CONDITION
            CALL I2CDLY
```

```
                CLR  I2CCLK
                CALL I2CDLY2
                RET
I2C_RESET:      CALL I2C_START          ;RESET ALL DEVICE
                MOV A,#0
                CALL I2C_WRITE
                MOV A,#06H
                CALL I2C_WRITE
                CALL I2C_STOP
                RET
I2C_READ:       SETB I2CDAT             ;FLOAT I2CDATA BIT
                MOV R4,#8               ;READ DATA
                CLR A
I2CRDLP:        CALL I2CDLY
                SETB I2CCLK
                CALL I2CDLY
                MOV C,I2CDAT
                RLC A
                CLR I2CCLK
                CALL I2CDLY
                DJNZ R4,I2CRDLP
                CLR I2CDAT              ;ACKNOWLEDGE BY MASTER
                CALL I2CDLY
                SETB I2CCLK
                CALL I2CDLY
                CLR I2CCLK
                CALL I2CDLY2
                RET
I2C_READNAK:    SETB I2CDAT             ;FLOAT I2CDATA BIT
                MOV R4,#8               ;READ DATA + NAK
                CLR A
I2CRDNAKLP:     CALL I2CDLY
                SETB I2CCLK
                CALL I2CDLY
                MOV C,I2CDAT
                RLC A
                CLR I2CCLK
                CALL I2CDLY
```

```
                DJNZ R4,I2CRDNAKLP
                SETB I2CDAT
                CALL I2CDLY
                SETB I2CCLK             ;NOT ACKNOWLEDGE BY MASTER
                CALL I2CDLY
                CLR I2CCLK
                CALL I2CDLY
                CLR I2CDAT
                CALL I2CDLY2
                RET
I2C_WRITE:      MOV R4,#8
RTCWRLP:        RLC A
                MOV I2CDAT,C
                CALL I2CDLY
                SETB I2CCLK             ;RISING EDGE CLOCK
                CALL I2CDLY
                CLR I2CCLK
                CALL I2CDLY
                DJNZ R4,RTCWRLP
                CALL I2CDLY
                SETB I2CCLK             ;ACKNOWLEDGE BY SLAVE
                CALL I2CDLY
                CLR I2CCLK
                CALL I2CDLY2
                RET
I2CDLY:         MOV R5,#10              ;DELAY
                DJNZ R5,$
                RET
I2CDLY2:        MOV R5,#30              ;DELAY
                DJNZ R5,$
                RET
MCP3424 I2C SUBROUTINE
ANALOG TO DIGITAL CONVERSTION SUBROUINE
ADC_CFG1:       CALL I2C_START      ;I2C_Start()    Generate Start condition
                MOV A,#0DCH
                CALL I2C_WRITE      ;I2C_Write(MPC3424)   send slave address
                MOV A,#80H          ;CHANNEL1 @12BITS
                CALL I2C_WRITE      ;I2C_Write      12 bits Send register address
```

```
                CALL I2C_STOP       ;I2C_Stop();
                RET
ADC_CFG2:       CALL I2C_START      ;I2C_Start()
                MOV A,#0DCH
                CALL I2C_WRITE      ;I2C_Write(MPC3424)  send slave address
                MOV A,#0A0H         ;CHANNEL2 @12BITS
                CALL I2C_WRITE      ;I2C_Write      12 bits Send register addres
                CALL I2C_STOP       ;I2C_Stop();
                RET
ADC_CFG3:       CALL I2C_START      ;I2C_Start()
                MOV A,#0DCH
                CALL I2C_WRITE      ;I2C_Write(MPC3424)  send slave address
                MOV A,#0C0H         ;CHANNEL3 @12BITS
                CALL I2C_WRITE      ;I2C_Write      12 bits Send register address
                CALL I2C_STOP       ;I2C_Stop();
                RET
ADC_CFG4:       CALL I2C_START      ;I2C_Start()
                MOV A,#0DCH
                CALL I2C_WRITE      ;I2C_Write(MPC3424)  send slave address
                MOV A,#0E0H         ;CHANNEL4 @12BITS
                CALL I2C_WRITE      ;I2C_Write      12 bits Send register address
                CALL I2C_STOP       ;I2C_Stop();
                RET
READ_ADC1216:
                CALL I2C_START      ;I2C_Start()    Generate Start condition
                MOV A,#0DDH         ;I2C_Write(MPC3424+1)send slave address
                CALL I2C_WRITE
                CALL I2C_READ
                MOV I2CBYTE2,A      ;nd2 := I2C_Read()    read the 2nd byte
                CALL I2C_READNAK
                MOV I2CBYTE3,A      ;rd3 := I2C_Read()    read the 3nd byte
                CALL I2C_STOP       ;I2C_Stop()           Send Stop Condition
                RET
READ_ADC18:     CALL I2C_START      ;I2C_Start()          Generate Start condition
                MOV A,#0DDH         ;I2C_Write(MPC3424+1)send slave address
                CALL I2C_WRITE
                CALL I2C_READ
                MOV I2CBYTE1,A      ;st1 := I2C_Read()    read the 1st byte
```

```
                CALL I2C_READ
                MOV I2CBYTE2,A      ;nd2 := I2C_Read()    read the 2nd byte
                CALL I2C_READNAK
                MOV I2CBYTE3,A      ;rd3 := I2C_Read()    read the 3nd byte
                CALL I2C_STOP       ;I2C_Stop()           Send Stop Condition
                RET
READ_ANALOG1:   CALL ADC_CFG1
                CALL DLY5MS_AN
                CALL READ_ADC1216  ;CH4 12BIT
                MOV A,I2CBYTE2
                MOV ANIN2,A
                MOV A,I2CBYTE3
                MOV ANIN2+1,A
                RET
READ_ANALOG2:   CALL ADC_CFG2
                CALL DLY5MS_AN
                CALL READ_ADC1216  ;CH4 12BIT
                MOV A,I2CBYTE2
                MOV ANIN1,A
                MOV A,I2CBYTE3
                MOV ANIN1+1,A
                RET
READ_ANALOG3:   CALL ADC_CFG3
                CALL DLY5MS_AN
                CALL READ_ADC1216  ;CH4 12BIT
                MOV A,I2CBYTE2
                MOV ANIN4,A
                MOV A,I2CBYTE3
                MOV ANIN4+1,A
                RET
READ_ANALOG4:   CALL ADC_CFG4
                CALL DLY5MS_AN
                CALL READ_ADC1216  ;CH4 12BIT
                MOV A,I2CBYTE2
                MOV ANIN3,A
                MOV A,I2CBYTE3
                MOV ANIN3+1,A
                RET
```

```
READ_ANALOG:    CALL READ_ANALOG1
                CALL READ_ANALOG2
                CALL READ_ANALOG3
                CALL READ_ANALOG4
                CALL DLY5MS
                RET
DLY5MS_AN:      MOV R7,#50          ;A TO D CONVERSE DELAY TIME
D5MS_AN:        MOV R6,#100
                DJNZ R6,$
                DJNZ R7,D5MS_AN
                RET
```

MCP 4728 : DIGITAL TO ANALOG CONVERSION SUBROUTINE

```
WRITE_ANALOG:   CALL I2C_START
                MOV A,#0C0H
                CALL I2C_WRITE
                MOV A,#01010001B    ;sequential write command UDAC = 1
                CALL I2C_WRITE
                MOV A,ANOUT1                ;HI BYTE
                ANL A,#0FH
                ORL A,#00010000B            ;Vref=5V & Gx= x2
                CALL I2C_WRITE
                MOV A,ANOUT1+1              ;LOW BYTE
                CALL I2C_WRITE
                MOV A,ANOUT2
                ANL A,#0FH
                ORL A,#00010000B    ;Vref=5V & Gx= x2
                CALL I2C_WRITE
                MOV A,ANOUT2+1
                CALL I2C_WRITE
                MOV A,ANOUT3
                ANL A,#0FH
                ORL A,#00010000B    ;Vref=5V & Gx= x2
                CALL I2C_WRITE
                MOV A,ANOUT3+1
                CALL I2C_WRITE
                MOV A,ANOUT4
                ANL A,#0FH
```

```
        ORL A,#00010000B    ;Vref=5V & Gx= x2
        CALL I2C_WRITE
        MOV A,ANOUT4+1
        CALL I2C_WRITE
        CALL I2C_STOP
        CLR LD4728
        CALL DLY1MS
        SETB LD4728
        RET

PCA9671 I2C SUBROUTINE : DIGITAL INPUT OUTPUT READ/WRITE SUBROUINE
READ_DIGITAL:   CALL I2C_START
        MOV A,#PCA9671+1
        CALL I2C_WRITE
        CALL I2C_READ
        CPL A
        MOV I2CBYTE3,A      ;PORT 0 OUTPUT NOT USE FOR READ
        CALL I2C_READ
        CPL A
        MOV DIGITIN,A       ;PORT 1 INPUT
        CALL I2C_STOP
        RET
WRITE_DIGITAL:  CALL I2C_START
        MOV A,#PCA9671
        CALL I2C_WRITE
        MOV A,DIGITOUT          ;PORT 0 OUTPUT
        CPL A
        CALL I2C_WRITE
        MOV A,#0FFH         ;PULL UP PORT 1 DIGITAL IN
        CALL I2C_WRITE
        CALL I2C_STOP
        RET

DS1307  I2C SUBROUTINE
RTC_CFG:    CALL I2C_START      ;I2C_Start()    Generate Start condition
        MOV A,#0D0H         ;SLAVE ADDRESS
        CALL I2C_WRITE
        MOV A,#SECOND_ADDR      ;SECOND ADDRESS
```

```
        CALL I2C_WRITE
        CALL I2C_RESTART
        MOV A,#0D1H          ;SLAVE ADDRESS
        CALL I2C_WRITE
        CALL I2C_READ
        ANL A,#7FH
        MOV @R0,A
        CALL I2C_STOP
        CALL I2C_IDLE
        CALL I2CDLY2
        CALL I2C_START
        MOV A,#0D0H          ;SLAVE ADDRESS WR
        CALL I2C_WRITE
        MOV A,#SECOND_ADDR       ;SECOND ADDRESS
        CALL I2C_WRITE
        MOV A,@R0
        CALL I2C_WRITE
        CALL I2C_STOP
        RET
READ_RTCDT: CALL I2C_START       ;I2C_Start()    Generate Start condition
        MOV A,#0D0H
        CALL I2C_WRITE       ;I2C_Write(0xD0)        send slave address
        MOV A,#SECOND_ADDR
        CALL I2C_WRITE       ;I2C_Write(0x02)        Send register address
        CALL I2C_RESTART
        MOV A,#0D1H
        CALL I2C_WRITE
        CALL I2C_READ                ;I2C_Read()     read the byte
        MOV CLOCK_BUF+0,A            ;SEC
        CALL I2C_READ                ;I2C_Read()     read the byte
        MOV CLOCK_BUF+1,A            ;MIN
        CALL I2C_READ                ;I2C_Read()     read the byte
        MOV CLOCK_BUF+2,A            ;HR
        CALL I2C_READ                ;I2C_Read()     read the byte
        MOV CLOCK_BUF+3,A            ;DAY
        CALL I2C_READ                ;I2C_Read()     read the byte
        MOV CLOCK_BUF+4,A            ;DATE
        CALL I2C_READ                ;I2C_Read()     read the byte
```

```
                MOV CLOCK_BUF+5,A       ;MONTH
                CALL I2C_READNAK        ;I2C_Read()     read the byte
                MOV CLOCK_BUF+6,A       ;YEAR
                CALL I2C_STOP           ;I2C_Stop()     Send Stop Condition
                CALL I2C_IDLE
                CALL DLY5MS
                RET
WRITE_RTCDT:    CALL I2C_START          ;I2C_Start()    Generate Start condition
                MOV A,#0D0H
                CALL I2C_WRITE          ;I2C_Write(0xD0) send slave address
                MOV A,#SECOND_ADDR
                CALL I2C_WRITE          ;I2C_Write(0x02) Send register address
                MOV A,#0
                CALL I2C_WRITE
                MOV A,CLOCK_BUF+1
                CALL I2C_WRITE
                MOV A,CLOCK_BUF+2
                CALL I2C_WRITE
                MOV A,CLOCK_BUF+3
                CALL I2C_WRITE
                MOV A,CLOCK_BUF+4
                CALL I2C_WRITE
                MOV A,CLOCK_BUF+5
                CALL I2C_WRITE
                MOV A,CLOCK_BUF+6
                CALL I2C_WRITE
                CALL I2C_STOP
                CALL I2C_IDLE
                CALL DLY5MS
                RET
S E R I A L   COMMUNICATION SUBROUTINE
INIT_SERIAL:    CPL A
                CLR TR1
                MOV SCON,#01010000B     ;set serial N81
                MOV TMOD,#00100010B
                ANL PCON,#01111111B
                MOV TH0,#0
                MOV TH1,#0FDH
```

```
                RET

SERIAL CODE MESSAGE
SER_MESG:       MOV R3,#0
SER_NEXT:       MOV A,R3
                MOVC A,@A+DPTR
                CJNE A,#0DH,SER_APP
                RET
CRLF:           MOV A,#0DH
                CALL SERIAL_TX
                MOV A,#0AH
                CALL SERIAL_TX
                RET
SER_APP:        CALL SERIAL_TX
                INC R3
                JMP SER_NEXT
SERIAL_TX:      MOV SBUF,A   ;INPUT IN A
                JNB TI,$
                CLR TI
                RET

MATHEMATIC UTILITY SUBROUTINE CODE CONVERSION
BCDBIN:         MOV R2,A              ;1 BYTE A=I/P  A=O/P
                ANL A,#0F0H
                RR A
                MOV R3,A
                RR A
                RR A
                ADD A,R3
                MOV R3,A
                MOV A,R2
                ANL A,#0FH
                ADD A,R3
                RET
ASCHEX:         MOV A,R2      ;R2[L] & R3[H]=I/P  A=O/P
                CALL A2HEX
                MOV R4,A
                MOV A,R3
```

```
        CALL A2HEX
        RR A
        RR A
        RR A
        RR A
        ORL A,R4
        RET
A2HEX:  CLR C
        SUBB A,#'0'
        MOV R5,A
        CLR C
        SUBB A,#10
        JC,A2HEX1
        MOV A,R5
        SUBB A,#7
        RET
A2HEX1: MOV A,R5
        RET
HEXASCI: MOV R3,A          ;A=I/P   R2[H] & A[L]=O/P
        ANL A,#0F0H
        RR A
        RR A
        RR A
        RR A
        CALL NASCII
        MOV R2,A
        MOV A,R3
        ANL A,#0FH
        CALL NASCII
        RET
NASCII: MOV R4,A
        SUBB A,#10
        JC NAS1
        MOV A,R4
        ADD A,#7
        MOV R4,A
NAS1:   MOV A,R4
        ADD A,#'0'
```

```
                RET
********** HTOD SUB **********
HEX TO DECIMAL
IN  = DPTR      , OUT = HEXBUF,HEXBUF+1    , REG = A,R0,R1,R2,R3,R4,R5,DPTR
HTOD:           CLR A                   ;CLEAR OUTPUT
                MOV HEXBUF,A
                MOV HEXBUF+1,A
                MOV R4,#16              ;SHIFT 16 BIT
HTOD1:          MOV A,DPL
                RLC A
                MOV DPL,A
                MOV A,DPH
                RLC A
                MOV DPH,A
                MOV R5,#2               ;2 BYTE ADD DECIMAL
                MOV R0,#HEXBUF+1    ;INDEX O/P
HTOD2:          MOV A,@R0
                ADDC A,ACC
                DA A
                MOV @R0,A
                DEC R0
                DJNZ R5,HTOD2
                DJNZ R1,HTOD1
                RET
********** DTOH SUB **********
DECIMAL TO HEX
IN  = R1,R2,R3          OUT = DPTR          REG = A,R0,R1,R2,R3,R4,R5,DPTR
DTOH:           MOV R4,#16
DTOH1:          MOV R5,#3     ;SHIFT & SUB
                MOV R0,#1     ;INDEX TO R1
                CLR C
DTOH2:          MOV A,@R0
                RRC A
                PUSH PSW      ;-[
                JNB ACC.7,DTOH3
                CLR C
                SUBB A,#30H
DTOH3:          JNB ACC.3,DTOH4
```

```
                CLR C
                SUBB A,#03H
DTOH4:          MOV @R0,A
                INC R0
                POP PSW      ;-]
                DJNZ R5,DTOH2
                MOV A,DPH
                RRC A
                MOV DPH,A
                MOV A,DPL
                RRC A
                MOV DPL,A
                DJNZ R4,DTOH1
                RET
********** DPMUL SUB **********
IN  = DPTR  MULTIPLER       OUT = DPTR  RESULT
    = R2,R3 MULTIPLICAND        = CARRY FLAG SET WHEN RESULT OVERFLOW
REG = A,R1,R2,R3,R4,R5,DPTR
DPMUL:          MOV  R4,#0          ;CLEAR RESULT
                MOV  R5,#0
                CLR  MULFG
                MOV  R1,#16
DPMUL1:         MOV  A,R5           ;*2
                ADD  A,R5
                MOV  R5,A
                MOV  A,R4
                ADDC A,R4
                MOV  R4,A
                JNC  DPMUL2         ;NO CARRY
                SETB MULFG          ;OVERFLOW
DPMUL2:         MOV  A,R3
                RLC  A
                MOV  R3,A
                MOV  A,R2
                RLC  A
                MOV  R2,A
                JNC  DPMUL3
                MOV  A,R5           ;R4R5=R4R5+DPTR
```

```
                ADD   A,DPL
                MOV   R5,A
                MOV   A,R4
                ADDC  A,DPH
                MOV   R4,A
                JNC   DPMUL3          ;NO CARRY
                SETB  MULFG           ;OVERFLOW
DPMUL3:         DJNZ  R1,DPMUL1
                MOV   DPH,R4
                MOV   DPL,R5
                MOV   C,MULFG         ;LOAD MULFG TO CARRY
                RET
********** DPDIV SUB **********
IN  = DPTR  DIVIDEND          OUT  = DPTR  RESULT
    = R2,R3 DIVISOR                  = R4,R5 REMAINDER
                                     = CARRY FLAG SET WHEN HAS REMAINDER
REG = A,R1,R2,R3,R4,R5,DPTR
DPDIV:          CLR   C
                MOV   R4,#0           ;CLEAR RESULT
                MOV   R5,#0
                MOV   R1,#16
DPDIV1:         MOV   A,DPL
                RLC   A
                MOV   DPL,A
                MOV   A,DPH
                RLC   A
                MOV   DPH,A
                MOV   A,R5            ;*2 (WITH CARRY)
                ADDC  A,R5
                MOV   R5,A
                MOV   A,R4
                ADDC  A,R4
                MOV   R4,A
                MOV   A,R5            ;R4R5=R4R5-R2R3 (WITH CARRY)
                SUBB  A,R3
                MOV   R5,A
                MOV   A,R4
                SUBB  A,R2
```

```
                MOV   R4,A
                JNC   DPDIV2
                MOV   A,R5              ;R4R5=R4R5+R2R3
                ADD   A,R3
                MOV   R5,A
                MOV   A,R4
                ADDC  A,R2
                MOV   R4,A
DPDIV2:         CPL   C
                DJNZ  R1,DPDIV1
                MOV   A,DPL             ;*2 (WITH CARRY)
                ADDC  A,DPL
                MOV   DPL,A
                MOV   A,DPH
                ADDC  A,DPH
                MOV   DPH,A
                MOV   A,R4
                ORL   A,R5
                JZ    DPDIV3
                SETB  C                 ;HAS REMAINDER->CARRY SET
                RET
DPDIV3:         CLR   C                 ;NO REMAINDET->CARRY CLEAR
                RET

********** DPADD SUB **********
DPTR = DPTR + R2,R3
IN  = DPTR,R2,R3        OUT = DPTR          REG = A,DPTR
DPADD:          MOV   A,DPL
                ADD   A,R3
                MOV   DPL,A
                MOV   A,DPH
                ADDC  A,R2
                MOV   DPH,A
                RET
********** DPSUB SUB *********
DPTR = DPTR - R2,R3
IN  = DPTR,R2,R3        OUT  = DPTR         REG  = A,DPTR
DPSUB:          CLR C
```

```
        MOV A,DPL
        SUBB A,R3
        MOV DPL,A
        MOV A,DPH
        SUBB A,R2
        MOV DPH,A
        RET
SERVICE ROUTINE FOR SERIAL COMMUNICATION
SR_SERV:    PUSH ACC
            PUSH DPH
            PUSH DPL
            SETB RS0            ;BANK1
            CLR RS1
            JNB RI,$
            MOV A,SBUF
            MOV BYTE,A
            MOV A,LINEPOI
            XRL A,#30           ;CHK LINEPOI
            JZ RST_LINEPOI
            MOV A,LINEPOI
            INC LINEPOI
            ADD A,#LINEIN
            MOV R0,A            ;R0=LINEPOI+LINEIN
            MOV @R0,BYTE
            MOV A,BYTE
            CJNE A,#0DH,SKIP2  ;CHK #0DH
            CALL COM_DECODE
SKIP2:      CLR RI
            CLR RS0             ;BANK0
            CLR RS1
            POP DPL
            POP DPH
            POP ACC
            RETI
RST_LINEPOI:    MOV LINEPOI,#0
                JMP SKIP2
COM_DECODE:     MOV LINEPOI,#0
                MOV R0,#LINEIN
```

```
                MOV A,@R0
                CJNE A,#'@',COMSKIP
CHK_STATION:    INC R0
                MOV A,@R0
                MOV R3,A
                INC R0
                MOV A,@R0
                MOV R2,A
                CALL ASCHEX
                XRL A,#00
                JNZ COMSKIP
                MOV R2,#30              ;CHK '*' 30 CHR LIMIT
CHKEND:     INC R0
            MOV A,@R0
            XRL A,#'*'
            JZ CHKSUM
            DJNZ R2,CHKEND
            JMP COMSKIP
CHKSUM:     INC R0
            MOV A,@R0
            CJNE A,#'X',CHKSUM_APP
            JMP CHK_COM
CHKSUM_APP:     MOV R2,A
                INC R0
                MOV A,@R0
                MOV R3,A
                CALL ASCHEX         ;R2&R3=I/P ASCII *'XX' TO HEX=(SUM)
                MOV SUM,A
                MOV R0,#LINEIN
                MOV A,#'@'
CHKSUM1:        INC R0
                XRL A,@R0
                MOV R2,A
                MOV A,@R0
                CJNE A,#0DH,CHKSUM1
                MOV A,R2
                XRL A,SUM
                JNZ COMSKIP
```

```
CHK_COM:    MOV DPTR,#COM_DEF
            MOV R3,#0
CHK_LOOP:   MOV R0,#LINEIN+3
            MOV A,#0
            MOVC A,@A+DPTR
            XRL A,@R0
            JNZ CHKCOM1
            INC DPTR
            INC R0
            MOV A,#0
            MOVC A,@A+DPTR
            XRL A,@R0
            JNZ CHKCOM2
            JZ COM_DECT
CHKCOM1:    INC DPTR
CHKCOM2:    INC DPTR
            INC R3              ;COMMAND COUNT <= 10
            MOV A,R3
            XRL A,#10
            JNZ CHK_LOOP
            CALL COM_ERR
            RET
COMSKIP:    RET
COM_DECT:   MOV A,R3
            CJNE A,#0,COMD_1
            JMP COMMAND_0
COMD_1:     CJNE A,#1,COMD_2
            JMP COMMAND_1
COMD_2:     CJNE A,#2,COMD_3
            JMP COMMAND_2
COMD_3:     CJNE A,#3,COMD_4
            JMP COMMAND_3
COMD_4:     CJNE A,#4,COMD_5
            JMP COMMAND_4
COMD_5:     CJNE A,#5,COMD_6
            JMP COMMAND_5
COMD_6:     CJNE A,#6,COMD_7
            JMP COMMAND_6
```

```
COMD_7:         CJNE A,#7,COMD_8
                JMP COMMAND_7
COMD_8:         CJNE A,#8,COMD_9
                JMP COMMAND_8
COMD_9:         CJNE A,#9,COMD_10
                JMP COMMAND_9
COMD_10:        RET

COM_DEF:    DB 'RA'  ;READ ANALOG IN 4 CH
                ;COMMAND @00RA*XX
                ;RESPOND @00RA 1234 1200 1000 0005 *XX
            DB 'RB'  ;READ ANALOG OUT 4 CH
                ;COMMAND @00RB*XX
                ;RESPOND @00RB 1234 2048 1234 0050 *XX
            DB 'RC'  ;READ ANALOG IN CHANNEL
                ;COMMAND @00RC01*XX READ ANALOG IN CH01
                ;RESPOND @00RC011234*XX
            DB 'RD'  ;READ DIGITAL 16 BIT INPUT AND OUTPUT
                ;COMMAND @00RD*XX
                ;RESPOND @00RDIIOO*XX
            DB 'RE'  ;READ ANALOG OUT CHANNEL
                ;COMMAND @00RE01*XX READ ANALOG OUT CH01
                ;RESPOND @00RE011234*XX
            DB 'RT'  ;READ TIME KEEPER
                ;COMMAND @00RT*XX
                ;RESPOND @00RT1935230713*XX
                ;19->HOUR REG.
                ;35->MIN  REG.
                ;23->DATE REG
                ;07->MONTH REG.
                ;13->YEAR REG.
            DB 'WA'  ;WRITE ANALOG OUT ALL CHANNEL
                ;COMMAND @00WA aaaa bbbb cccc dddd *XX
            DB 'WC'  ;WRITE ANALOG OUT CHANNEL
                ;COMMAND @00WA00 aaaa*XX
                ;00=CHANNEL
                ;DATA aaaa  (DATA MAY FROM 0000-4096)
            DB 'WD'  ;WRITE DIGITAL 16 BIT
```

```
                ;COMMAND @00WDIIOO*XX
        DB 'WT'  ;WRITE TIME KEEPER HOUR:MIN
                ;COMMAND @00WT1935230713*XX
                ;19->HOUR REG.
                ;35->MIN  REG.
                ;23->DATE REG
                ;07->MONTH REG.
                ;13->YEAR REG.
MESG5:      DB '-CMD*ERR',0DH
MESG6:      DB '-CHN*ERR',0DH
MESG7:      DB '-WA*TX',0DH
MESG8:      DB '-WC*TX',0DH
MESG9:      DB '-WD*TX',0DH
MESG10:     DB '-WT*TX',0DH
    COM_ERR:    MOV DPTR,#MESG5
                CALL SER_MESG
                CALL CRLF
                RET

OPERATION   SECTION
GETHEAD:    MOV A,@R0
            CALL SERIAL_TX
            INC R0
            DJNZ R2,GETHEAD
            RET

READ ANALOG 4 CH 00-04  [FUN00]
    DB 'RA'  ;READ ANALOG 4 CH
            ;COMMAND @00RA*XX
            ;RESPOND @00RA 1234 1200 1000 0005 *XX
COMMAND_0:      CALL DELAY
                CALL DELAY
                MOV R0,#LINEIN
                MOV R2,#5
                CALL GETHEAD
                CALL RDAI_CH1
                CALL RDAI_CH2
                CALL RDAI_CH3
```

```
                CALL RDAI_CH4
                CALL CRLF
                RET
        DB 'RB'  ;READ ANALOG OUT 4 CH
                ;COMMAND @00RB*XX
                ;RESPOND @00RB 1234 2048 1234 0050 *XX
COMMAND_1:      CALL DELAY
                CALL DELAY
                MOV R0,#LINEIN
                MOV R2,#5
                CALL GETHEAD
                CALL RDAO_CH1
                CALL RDAO_CH2
                CALL RDAO_CH3
                CALL RDAO_CH4
                CALL CRLF
                RET
DB 'RC'  ;READ ANALOG CHANNEL
                ;COMMAND @00RC01*XX READ ANALOG CH 1
                ;RESPOND @00RA011234*XX
COMMAND_2:      CALL DELAY
                CALL DELAY
                MOV R0,#LINEIN
                MOV R2,#5
                CALL GETHEAD
                MOV R3,LINEIN+5     ;HI
                MOV R2,LINEIN+6     ;LO
                CALL ASCHEX
                MOV R3,A
                CLR C
                SUBB A,#5           ;IF > CH4
                JNC COM1_ERR
                MOV A,R3
CHKRD_CH1:      CJNE A,#1,CHKRD_CH2
                CALL RDAI_CH1
                CALL CRLF
                RET
CHKRD_CH2:      CJNE A,#2,CHKRD_CH3
```

```
                CALL RDAI_CH2
                CALL CRLF
                RET
CHKRD_CH3:      CJNE A,#3,CHKRD_CH4
                CALL RDAI_CH3
                CALL CRLF
                RET
CHKRD_CH4:      CJNE A,#4,CHKRD_RET
                CALL RDAI_CH4
                CALL CRLF
CHKRD_RET:      RET
COM1_ERR:       MOV DPTR,#MESG6
                CALL SER_MESG
                CALL CRLF
                RET
RDAI_CH1:       MOV A,ANIN1
                MOV DPH,A
                MOV A,ANIN1+1
                MOV DPL,A
                CALL CHKMINUS
                CALL HTOD
                CALL ADC_TRMESG
                RET
RDAI_CH2:       MOV A,ANIN2
                MOV DPH,A
                MOV A,ANIN2+1
                MOV DPL,A
                CALL CHKMINUS
                CALL HTOD
                CALL ADC_TRMESG
                RET
RDAI_CH3:       MOV A,ANIN3
                MOV DPH,A
                MOV A,ANIN3+1
                MOV DPL,A
                CALL CHKMINUS
                CALL HTOD
                CALL ADC_TRMESG
```

```
                RET
RDAI_CH4:       MOV A,ANIN4
                MOV DPH,A
                MOV A,ANIN4+1
                MOV DPL,A
                CALL CHKMINUS
                CALL HTOD
                CALL ADC_TRMESG
                RET
RDAO_CH1:       MOV A,ANOUT1
                MOV DPH,A
                MOV A,ANOUT1+1
                MOV DPL,A
                CALL HTOD
                CALL ADC_TRMESG
                RET
RDAO_CH2:       MOV A,ANOUT2
                MOV DPH,A
                MOV A,ANOUT2+1
                MOV DPL,A
                CALL HTOD
                CALL ADC_TRMESG
                RET
RDAO_CH3:       MOV A,ANOUT3
                MOV DPH,A
                MOV A,ANOUT3+1
                MOV DPL,A
                CALL HTOD
                CALL ADC_TRMESG
                RET
RDAO_CH4:       MOV A,ANOUT4
                MOV DPH,A
                MOV A,ANOUT4+1
                MOV DPL,A
                CALL HTOD
                CALL ADC_TRMESG
                RET
ADC_TRMESG:     MOV A,HEXBUF            ;[H] BYTE
```

```
                    CALL HEXASCI
                    MOV R3,A
                    MOV A,R2
                    CALL SERIAL_TX
                    MOV A,R3
                    CALL SERIAL_TX
                    MOV A,HEXBUF+1    ;[H] BYTE
BYTE_TR:            CALL HEXASCI
                    MOV R3,A
                    MOV A,R2
                    CALL SERIAL_TX
                    MOV A,R3
                    CALL SERIAL_TX
                    RET
DB 'RD'  ;READ DIGITAL 16 BIT INPUT AND OUTPUT
          ;COMMAND @00RD*XX
          ;RESPOND @00RDIIOO*XX
COMMAND_3:          CALL DELAY
                    CALL DELAY
                    MOV R0,#LINEIN
                    MOV R2,#5
                    CALL GETHEAD
                    MOV A,DIGITIN
                    CALL BYTE_TR
                    MOV A,DIGITOUT
                    CALL BYTE_TR
                    CALL CRLF
                    RET
DB 'RE'   ;READ ANALOG OUT CHANNEL
          ;COMMAND @00RE01*XX READ ANALOG OUT CH01
          ;RESPOND @00RE011234*XX
COMMAND_4:          CALL DELAY
                    CALL DELAY
                    MOV R0,#LINEIN
                    MOV R2,#5
                    CALL GETHEAD
                    MOV R3,LINEIN+5          ;HI
                    MOV R2,LINEIN+6          ;LO
```

```
                    CALL ASCHEX
                    MOV R3,A
                    CLR C
                    SUBB A,#5                   ;IF > CH4
                    JNC COM3_ERR
                    MOV A,R3
CHKRDAO_CH1:        CJNE A,#1,CHKRDAO_CH2
                    CALL RDAO_CH1
                    CALL CRLF
                    RET
CHKRDAO_CH2:        CJNE A,#2,CHKRDAO_CH3
                    CALL RDAO_CH2
                    CALL CRLF
                    RET
CHKRDAO_CH3:        CJNE A,#3,CHKRDAO_CH4
                    CALL RDAO_CH3
                    CALL CRLF
                    RET
CHKRDAO_CH4:        CJNE A,#4,CHKRDAO_RET
                    CALL RDAO_CH4
                    CALL CRLF
CHKRDAO_RET:        RET
COM3_ERR:           MOV DPTR,#MESG6
                    CALL SER_MESG
                    CALL CRLF
                    RET
DB 'RT'  ;READ TIME KEEPER
                ;COMMAND @00RT*XX
                ;RESPOND @00RT1935230713*XX
                ;19->HOUR REG.
                ;35->MIN  REG.
                ;23->DATE REG
                ;07->MONTH REG.
                ;13->YEAR REG.
COMMAND_5:          CALL DELAY
                    CALL DELAY
                    MOV R0,#LINEIN
                    MOV R2,#5
```

```
                CALL GETHEAD
                MOV A,CLOCK_BUF+2          ;HR
                CALL BYTE_TR
                MOV A,CLOCK_BUF+1          ;MIN
                CALL BYTE_TR
                MOV A,CLOCK_BUF+4          ;DATE
                CALL BYTE_TR
                MOV A,CLOCK_BUF+5          ;MONTH
                CALL BYTE_TR
                MOV A,CLOCK_BUF+6          ;YEAR
                CALL BYTE_TR
                CALL CRLF
                RET
DB 'WA'  ;WRITE ANALOG OUT ALL CHANNEL
                ;COMMAND @00WA aaaa bbbb cccc dddd *XX
                ;RESPOND @00WA
COMMAND_6:      MOV R3,LINEIN+5            ;HI
                MOV R2,LINEIN+6            ;LO
                CALL ASCHEX
                CALL BCDBIN
                MOV B,#100
                MUL AB
                MOV DPH,B
                MOV DPL,A
                MOV R3,LINEIN+7            ;HI
                MOV R2,LINEIN+8            ;LO
                CALL ASCHEX
                CALL BCDBIN
                MOV R2,#0
                MOV R3,A
                CALL DPADD
                MOV ANOUT1,DPH
                MOV ANOUT1+1,DPL
                MOV R3,LINEIN+9            ;HI
                MOV R2,LINEIN+10           ;LO
                CALL ASCHEX
                CALL BCDBIN
                MOV B,#100
```

```
MUL AB
MOV DPH,B
MOV DPL,A
MOV R3,LINEIN+11          ;HI
MOV R2,LINEIN+12          ;LO
CALL ASCHEX
CALL BCDBIN
MOV R2,#0
MOV R3,A
CALL DPADD
MOV ANOUT2,DPH
MOV ANOUT2+1,DPL
MOV R3,LINEIN+13          ;HI
MOV R2,LINEIN+14          ;LO
CALL ASCHEX
CALL BCDBIN
MOV B,#100
MUL AB
MOV DPH,B
MOV DPL,A
MOV R3,LINEIN+15          ;HI
MOV R2,LINEIN+16          ;LO
CALL ASCHEX
CALL BCDBIN
MOV R2,#0
MOV R3,A
CALL DPADD
MOV ANOUT3,DPH
MOV ANOUT3+1,DPL
MOV R3,LINEIN+17          ;HI
MOV R2,LINEIN+18          ;LO
CALL ASCHEX
CALL BCDBIN
MOV B,#100
MUL AB
MOV DPH,B
MOV DPL,A
MOV R3,LINEIN+19          ;HI
```

```
                MOV R2,LINEIN+20        ;LO
                CALL ASCHEX
                CALL BCDBIN
                MOV R2,#0
                MOV R3,A
                CALL DPADD
                MOV ANOUT4,DPH
                MOV ANOUT4+1,DPL
                MOV R0,#LINEIN
                MOV R2,#5
                CALL GETHEAD            ;@00WA
                MOV DPTR,#MESG7         ;-WA*TX
                CALL SER_MESG
                CALL CRLF
                RET
DB 'WC' ;WRITE ANALOG OUT CHANNEL
        ;COMMAND @00WA00 aaaa*XX
        ;00=CHANNEL
        ;DATA aaaa  (DATA MAY FROM 0000-4096)
COMMAND_7:      MOV R0,#LINEIN
                MOV R2,#5
                CALL GETHEAD
                MOV R3,LINEIN+5         ;HI
                MOV R2,LINEIN+6         ;LO
                CALL ASCHEX
                MOV R3,A
                CLR C
                SUBB A,#5               ;IF > CH4
                JNC COM7_ERR
                MOV A,R3
WR_CH1:     CJNE A,#1,WR_CH2
            CALL WRAI_CH1
            CALL CRLF
            RET
WR_CH2:     CJNE A,#2,WR_CH3
            CALL WRAI_CH2
            CALL CRLF
            RET
```

```
WR_CH3:     CJNE A,#3,WR_CH4
            CALL WRAI_CH3
            CALL CRLF
            RET
WR_CH4:     CJNE A,#4,WR_RET
            CALL WRAI_CH4
            CALL CRLF
WR_RET:     RET
COM7_ERR:       MOV DPTR,#MESG6
                CALL SER_MESG
                CALL CRLF
                RET
WRAI_CH1:       CALL WRAI_CHN
                MOV ANOUT1,DPH
                MOV ANOUT1+1,DPL
                MOV R0,#LINEIN
                MOV R2,#5
                CALL GETHEAD            ;@00WA
                MOV DPTR,#MESG8         ;-WC*TX
                CALL SER_MESG
                CALL CRLF
                RET
WRAI_CH2:       CALL WRAI_CHN
                MOV ANOUT2,DPH
                MOV ANOUT2+1,DPL
                MOV R0,#LINEIN
                MOV R2,#5
                CALL GETHEAD            ;@00WA
                MOV DPTR,#MESG8         ;-WC*TX
                CALL SER_MESG
                CALL CRLF
                RET
WRAI_CH3:       CALL WRAI_CHN
                MOV ANOUT3,DPH
                MOV ANOUT3+1,DPL
                MOV R0,#LINEIN
                MOV R2,#5
                CALL GETHEAD            ;@00WA
```

```
                MOV DPTR,#MESG8         ;-WC*TX
                CALL SER_MESG
                CALL CRLF
                RET
WRAI_CH4:       CALL WRAI_CHN
                MOV ANOUT4,DPH
                MOV ANOUT4+1,DPL
                MOV R0,#LINEIN
                MOV R2,#5
                CALL GETHEAD            ;@00WA
                MOV DPTR,#MESG8         ;-WC*TX
                CALL SER_MESG
                CALL CRLF
                RET
WRAI_CHN:       MOV R3,LINEIN+7         ;HI
                MOV R2,LINEIN+8         ;LO
                CALL ASCHEX
                CALL BCDBIN
                MOV B,#100
                MUL AB
                MOV DPH,B
                MOV DPL,A
                MOV R3,LINEIN+9         ;HI
                MOV R2,LINEIN+10        ;LO
                CALL ASCHEX
                CALL BCDBIN
                MOV R2,#0
                MOV R3,A
                CALL DPADD
                RET
DB 'WD'  ;WRITE DIGITAL 16 BIT
                ;COMMAND @00WDIIOO*XX
COMMAND_8:      MOV R3,LINEIN+5         ;HI
                MOV R2,LINEIN+6         ;LO
                CALL ASCHEX
                MOV DIGITIN,A
                MOV R3,LINEIN+7         ;HI
                MOV R2,LINEIN+8         ;LO
```

```
                CALL ASCHEX
                MOV DIGITOUT,A
                MOV R0,#LINEIN
                MOV R2,#5
                CALL GETHEAD            ;@00WA
                MOV DPTR,#MESG10  ;-WD*TX
                CALL SER_MESG
                CALL CRLF
                RET
DB 'WT'  ;WRITE TIME KEEPER HOUR:MIN
        ;COMMAND @00WT1935230713*XX
        ;19->HOUR REG.
        ;35->MIN  REG.
        ;23->DATE REG
        ;07->MONTH REG.
        ;13->YEAR REG.
COMMAND_9:      MOV R3,LINEIN+5         ;HI
                MOV R2,LINEIN+6         ;LO
                CALL ASCHEX
                MOV CLOCK_BUF+2,A
                MOV R3,LINEIN+7         ;HI
                MOV R2,LINEIN+8         ;LO
                CALL ASCHEX
                MOV CLOCK_BUF+1,A
                MOV R3,LINEIN+9         ;HI
                MOV R2,LINEIN+10        ;LO
                CALL ASCHEX
                MOV CLOCK_BUF+4,A
                MOV R3,LINEIN+11        ;HI
                MOV R2,LINEIN+12        ;LO
                CALL ASCHEX
                MOV CLOCK_BUF+5,A
                MOV R3,LINEIN+13        ;HI
                MOV R2,LINEIN+14        ;LO
                CALL ASCHEX
                MOV CLOCK_BUF+6,A
                MOV CLOCK_BUF+0,#0
                CALL WRITE_RTCDT
```

```
                MOV R0,#LINEIN
                MOV R2,#5
                CALL GETHEAD            ;@00WA
                MOV DPTR,#MESG10        ;-WT*TX
                CALL SER_MESG
                CALL CRLF
                RET
DELAY:          MOV R7,#05
DLY:            MOV R6,#0FFH
                DJNZ R6,$
                DJNZ R7,DLY
                RET
GETINPUT:       MOV A,P3
                MOV R2,A
                CALL DELAY
                MOV A,P3
                XRL A,R2
                JNZ GETINPUT
                CALL DELAY
                MOV A,R2                ;CHK STATE
                CPL A
                MOV INP_PRS,A
                MOV A,INP_LST
                XRL A,INP_PRS
                MOV R2,A
                MOV A,INP_LST
                CPL A
                ANL A,R2
                MOV INP_RSLT,A
                MOV A,INP_PRS
                MOV INP_LST,A
                MOV A,INP_RSLT
                JNB ACC.7,CHKINP2
                INC MONI_PAGE
CHKINP2:        RET
SETDT:          MOV CLOCK_BUF+0,#0      ;SEC
                MOV CLOCK_BUF+1,#45H    ;MIN
                MOV CLOCK_BUF+2,#16H    ;HR
```

```
        MOV CLOCK_BUF+3,#3          ;DAY0-7
        MOV CLOCK_BUF+4,#23H        ;DATE
        MOV CLOCK_BUF+5,#07H        ;MONTH
        MOV CLOCK_BUF+6,#13H        ;YEAR
        CALL WRITE_RTCDT
        RET

START PROGRAM
START:  MOV R7,#100
STRDLY: MOV R6,#250
        DJNZ R6,$
        DJNZ R7,STRDLY
        MOV IE,#0
        MOV SP,#STACK
        CALL INIT_SERIAL            ;set serial communication
        CALL INITLCD                ;INITIAL LCD 4X16
        CALL I2C_RESET              ;RESET ALL DEVICE
        CALL I2C_IDLE
        CALL RTC_CFG
        MOV P1,#0FFH
        MOV P3,#0FFH
        CLR TI
        CLR RI
        MOV LINEPOI,#0              ;LINE POINTER
        MOV ANAHEX1,#0              ;ANALOG HEX
        MOV ANAHEX2,#0              ;ANALOG HEX
        MOV ANOUT1,#0
        MOV ANOUT1+1,#0
        MOV ANOUT2,#0
        MOV ANOUT2+1,#0
        MOV ANOUT3,#0
        MOV ANOUT3+1,#0
        MOV ANOUT4,#0
        MOV ANOUT4+1,#0
        MOV DIGITIN,#0
        MOV DIGITOUT,#0
        MOV MINUTE,#MIN             ;START DEFAULT 60 SEC
        MOV SECFLG,#0               ;SEC FLG = 0
```

```
        MOV MONI_PAGE,#0
        MOV DPTR,#0FFFFH-1800          ;3600 @1 SEC  54000 @15 SEC
        MOV CNTL,DPL
        MOV CNTH,DPH
        SETB TR0                       ;T0 START
        SETB TR1                       ;T1 START
        SETB ET0                       ;INT T0
        SETB ES                        ;INT SERIAL
        SETB EA                        ;INT ALL ENABLE

OPER:   CALL GETINPUT
        JMP OPER
        END
```

คำสั่งการทำงานของโปรแกรม Visual Basic 2010

```
Imports System
Imports System.ComponentModel
Imports System.Threading
Imports System.IO.Ports
Imports System.Text

Public Class Form1
    Dim myPort As Array  'COM Ports detected on the system will be stored
    Dim Digital_Out As Byte
    Dim Digital_OutStr As String

    Dim receivedData As String
    Dim header As String = ""
    Dim timerTikCount As Integer = 0

Delegate Sub SetTextCallback(ByVal [text] As String) 'Added to prevent threading
errors during receiveing of data
    Dim _myPort As String
    Private Sub SerialPort1_DataReceived(ByVal sender As Object, ByVal e As
System.IO.Ports.SerialDataReceivedEventArgs) Handles SerialPort1.DataReceived
    ReceivedText(SerialPort1.ReadExisting())
    End Sub
```

```
Private Sub ReceivedText(ByVal [text] As String)
If Me.rtbReceived.InvokeRequired Then
        Dim x As New SetTextCallback(AddressOf ReceivedText)
        Me.Invoke(x, New Object() {(text)})
Else
        Me.rtbReceived.Text &= [text]
        receivedData = rtbReceived.Text
        If receivedData.Contains("RA") Then
        header = "RA"
End If
        If receivedData.Contains("RD") Then
        header = "RD"
End If
        If header.Equals("RA") Then
        Me.rtbAnalogData.Text = ""
        Me.rtbAnalogData.Text = receivedData
        Me.Textbox1.Text &= Mid(rtbReceived.Text, 6, 4)
        Me.RACH2.Text &= Mid(rtbReceived.Text, 10, 4)
        Me.RACH3.Text &= Mid(rtbReceived.Text, 14, 4)
        Me.RACH4.Text &= Mid(rtbReceived.Text, 18, 4)
        receivedData = ""
        header = ""
        ElseIf header.Equals("RD") Then
        Me.TextBox3.Text = ""
        Me.TextBox3.Text = receivedData
        receivedData = ""
        header = ""

        'Change light
        Dim inDataStr As String = ""
        inDataStr &= Mid(TextBox3.Text, 8, 2)
        If String.IsNullOrEmpty(inDataStr) Then
        Exit Sub
End If
        Dim inDataInt As Integer
        inDataInt = Convert.ToUInt32(inDataStr, 16)
        Dim inDataByte As Byte
        inDataByte = CByte(inDataInt)
```

```
        If (inDataByte And 128) > 0 Then
        bit7.FillColor = Color.Red
Else
        bit7.FillColor = Color.White
End If
        If (inDataByte And 64) > 0 Then
        bit6.FillColor = Color.Red
Else
        bit6.FillColor = Color.White
End If
        If (inDataByte And 32) > 0 Then
        bit5.FillColor = Color.Red
Else
        bit5.FillColor = Color.White
End If
        If (inDataByte And 16) > 0 Then
        bit4.FillColor = Color.Red
Else
        bit4.FillColor = Color.White
End If
        If (inDataByte And 8) > 0 Then
        bit3.FillColor = Color.Red
Else
        bit3.FillColor = Color.White
End If
        If (inDataByte And 4) > 0 Then
        bit2.FillColor = Color.Red
Else
        bit2.FillColor = Color.White
End If
        If (inDataByte And 2) > 0 Then
        bit1.FillColor = Color.Red
Else
        bit1.FillColor = Color.White
End If
        If (inDataByte And 1) > 0 Then
        bit0.FillColor = Color.Red
```

```
        Else
                bit0.FillColor = Color.White
        End If
         End If
       End If
End Sub
Private Sub ReceivedText1(ByVal [text] As String)
End Sub
    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Timer1.Tick
    timerTikCount += 1
    If timerTikCount = 5 Then
            SerialPort1.Write("@00RD*XX" & vbCr)
            rtbReceived.Text = ""
            TextBox3.Text = ""
    ElseIf timerTikCount = 10 Then
            SerialPort1.Write("@00RA*XX" & vbCr)
            rtbReceived.Text = ""
            rtbAnalogData.Text = ""
            timerTikCount = 0
    End If
End Sub
    Private Sub Textbox1_TextChanged(ByVal sender As System.Object, ByVal e
    As System.EventArgs) Handles Textbox1.TextChanged
            Textbox1.Text = Mid(rtbReceived.Text, 6, 4)
End Sub
    Private Sub RACH2_TextChanged(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles RACH2.TextChanged
            RACH2.Text = Mid(rtbReceived.Text, 10, 4)
End Sub
    Private Sub RACH3_TextChanged(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles RACH3.TextChanged
            RACH3.Text = Mid(rtbReceived.Text, 14, 4)
End Sub
    Private Sub RACH4_TextChanged(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles RACH4.TextChanged
            RACH4.Text = Mid(rtbReceived.Text, 18, 4)
End Sub
```

```
    Private Sub frmMain_Load(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles MyBase.Load
    'When our form loads, auto detect all serial ports in the system and populate
the cmbPort Combo box.
    myPort = IO.Ports.SerialPort.GetPortNames() 'Get all com ports available
    cmbBaud.Items.Add(9600)    'Populate the cmbBaud Combo box to common
baud rates used
    cmbBaud.Items.Add(19200)
    cmbBaud.Items.Add(38400)
    cmbBaud.Items.Add(57600)
    cmbBaud.Items.Add(115200)

    For i = 0 To UBound(myPort)
        cmbPort.Items.Add(myPort(i))
    Next
    cmbPort.Text = cmbPort.Items.Item(0)    'Set cmbPort text to the first COM
port detected
    cmbBaud.Text = cmbBaud.Items.Item(0)    'Set cmbBaud text to the first Baud
rate on the list
    btnDisconnect.Enabled = False          'Initially Disconnect Button is Disabled
End Sub
    Private Sub btnConnect_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles btnConnect.Click
        SerialPort1.PortName = cmbPort.Text       'Set SerialPort1 to the
selected COM port at startup
    SerialPort1.BaudRate = cmbBaud.Text       'Set Baud rate to the selected
value on

    'Other Serial Port Property
    SerialPort1.Parity = IO.Ports.Parity.None
    SerialPort1.StopBits = IO.Ports.StopBits.One
    SerialPort1.DataBits = 8            'Open our serial port
    SerialPort1.Open()

    btnConnect.Enabled = False         'Disable Connect button
    btnDisconnect.Enabled = True       'and Enable Disconnect button
End Sub
```

```
    Private Sub btnDisconnect_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles btnDisconnect.Click
            SerialPort1.Close()          'Close our Serial Port
    btnConnect.Enabled = True
    btnDisconnect.Enabled = False
End Sub
    Private Sub btnStart_Click_1(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles btnStart.Click
                Timer1.Start()
                btnStart.Enabled = False
End Sub
    Private Sub WACH1_TextChanged(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles WACH1.TextChanged
        'SerialPort1.Write(WACH1.Text & vbCr)
        'Textbox1.Text = Mid(rtbReceived.Text, 6, 4)
End Sub
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button1.Click
    End
End Sub
    Private Sub btnDigitalOut_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles btnDigitalOut.Click
        'TextBox2.Text = TextBox2.Text + "*xx"
        SerialPort1.Write(TextBox2.Text & vbCr)
End Sub
    Private Sub btnClear_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles btnClear.Click
                Me.TextBox2.Text = ""
                OvalShape1.FillColor = Color.White
                OvalShape2.FillColor = Color.White
                OvalShape3.FillColor = Color.White
                OvalShape4.FillColor = Color.White
                OvalShape5.FillColor = Color.White
                OvalShape6.FillColor = Color.White
                OvalShape7.FillColor = Color.White
                OvalShape8.FillColor = Color.White
                Digital_Out = 0
End Sub
```

```
    Private Sub btnDigitalIn_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles btnDigitalIn.Click
                rtbReceived.Text = ""
                TextBox3.Text = ""
    SerialPort1.Write("@00RD*XX" & vbCr)
End Sub
    Private Sub btn0_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles btn0.Click
    If (Digital_Out And 1) > 0 Then                'Button 3 คือ ปุ่ม 0
        Digital_Out = Digital_Out And 254
        OvalShape8.FillColor = Color.White
    Else
        Digital_Out = Digital_Out Or 1
        OvalShape8.FillColor = Color.Red
    End If
        Digital_OutStr = Hex(Digital_Out)
    If Len(Digital_OutStr) = 1 Then
        Digital_OutStr = "0" + Digital_OutStr
    End If
        TextBox2.Text = "@00WD00" + Digital_OutStr + vbCr
End Sub
    Private Sub btn1_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles btn1.Click
        If (Digital_Out And 2) > 0 Then                'Button 1
            Digital_Out = Digital_Out And 253
            OvalShape7.FillColor = Color.White
    Else
            Digital_Out = Digital_Out Or 2
            OvalShape7.FillColor = Color.Red
    End If
            Digital_OutStr = Hex(Digital_Out)
    If Len(Digital_OutStr) = 1 Then
            Digital_OutStr = "0" + Digital_OutStr
    End If
            TextBox2.Text = "@00WD00" + Digital_OutStr + vbCr
End Sub
```

```
    Private Sub btn2_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles btn2.Click
    If (Digital_Out And 4) > 0 Then                    'Button 2
            Digital_Out = Digital_Out And 251
            OvalShape6.FillColor = Color.White
    Else
            Digital_Out = Digital_Out Or 4
            OvalShape6.FillColor = Color.Red
    End If
            Digital_OutStr = Hex(Digital_Out)
    If Len(Digital_OutStr) = 1 Then
            Digital_OutStr = "0" + Digital_OutStr
    End If
            TextBox2.Text = "@00WD00" + Digital_OutStr + vbCr
End Sub
    Private Sub btn3_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles btn3.Click
    If (Digital_Out And 8) > 0 Then                    'Button 3
            Digital_Out = Digital_Out And 247
            OvalShape5.FillColor = Color.White
    Else
            Digital_Out = Digital_Out Or 8
            OvalShape5.FillColor = Color.Red
    End If
            Digital_OutStr = Hex(Digital_Out)
    If Len(Digital_OutStr) = 1 Then
            Digital_OutStr = "0" + Digital_OutStr
    End If
            TextBox2.Text = "@00WD00" + Digital_OutStr + vbCr
End Sub
    Private Sub btn4_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles btn4.Click
    If (Digital_Out And 16) > 0 Then                   'Button 4
            Digital_Out = Digital_Out And 239
            OvalShape4.FillColor = Color.White
    Else
            Digital_Out = Digital_Out Or 16
            OvalShape4.FillColor = Color.Red
```

```
        End If
            Digital_OutStr = Hex(Digital_Out)
            TextBox2.Text = "@00WD00" + Digital_OutStr + vbCr
    End Sub
        Private Sub btn5_Click(ByVal sender As System.Object, ByVal e As
        System.EventArgs) Handles btn5.Click
        If (Digital_Out And 32) > 0 Then                'Button 5
            Digital_Out = Digital_Out And 223
            OvalShape3.FillColor = Color.White
        Else
            Digital_Out = Digital_Out Or 32
            OvalShape3.FillColor = Color.Red
        End If
            Digital_OutStr = Hex(Digital_Out)
            TextBox2.Text = "@00WD00" + Digital_OutStr + vbCr
    End Sub
        Private Sub btn6_Click(ByVal sender As System.Object, ByVal e As
        System.EventArgs) Handles btn6.Click
        If (Digital_Out And 64) > 0 Then                'Button 6
            Digital_Out = Digital_Out And 191
            OvalShape2.FillColor = Color.White
        Else
            Digital_Out = Digital_Out Or 64
            OvalShape2.FillColor = Color.Red
        End If
            Digital_OutStr = Hex(Digital_Out)
            TextBox2.Text = "@00WD00" + Digital_OutStr + vbCr
    End Sub
        Private Sub btn7_Click(ByVal sender As System.Object, ByVal e As
        System.EventArgs) Handles btn7.Click
        If (Digital_Out And 128) > 0 Then               'Button 7
            'Close light
            Digital_Out = Digital_Out And 127
            OvalShape1.FillColor = Color.White
        Else
            'Open light
            Digital_Out = Digital_Out Or 128
            OvalShape1.FillColor = Color.Red
```

```
        End If
            Digital_OutStr = Hex(Digital_Out)
            TextBox2.Text = "@00WD00" + Digital_OutStr + vbCr
    End Sub
        Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
        System.EventArgs) Handles Button2.Click
            SerialPort1.Write(WACH1.Text & vbCr)
    End Sub
End Class
```