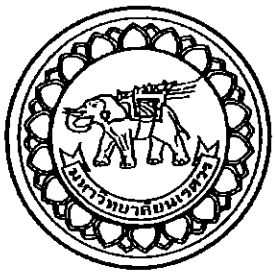


การเปิดและปิดมอเตอร์ปั๊มน้ำโดยการควบคุมทางไกลแบบไร้สาย

WIRELESS REMOTE ON-OFF CONTROL OF WATER PUMP MOTOR

นางสาวบุณยนุช กมลวรเดช รหัส 52361932
นางสาวภัณฑิลา เก่งพานิช รหัส 52362106
นางสาวอัญชิษฐา ปราสาททรัพย์ รหัส 52362380

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร
ปีการศึกษา 2555

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อโครงงาน	การเปิดและปิดมอเตอร์ปั๊มน้ำโดยการควบคุมทางไกลแบบไร้สาย	
ผู้ดำเนินโครงงาน	นางสาวบุณยนุช กมลวรเดช	รหัส 52361932
	นางสาวภัณฑิลา เก่งพานิช	รหัส 52362106
	นางสาวอัญชิษฐา ปราสาททรัพย์	รหัส 52362380
ที่ปรึกษาโครงงาน	ดร. นิพัทธ์ จันทรมินทร์	
สาขาวิชา	วิศวกรรมไฟฟ้า	
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์	
ปีการศึกษา	2555	

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

.....ที่ปรึกษาโครงงาน
(ดร. นิพัทธ์ จันทรมินทร์)

.....กรรมการ
(ดร. ศุภวรรณ พลพิทักษ์ชัย)

.....กรรมการ
(ดร. พนัส นัฏฤทธิ์)

ชื่อหัวข้อโครงงาน	การเปิดและปิดมอเตอร์ปั๊มน้ำโดยการควบคุมทางไกลแบบไร้สาย	
ผู้ดำเนินโครงงาน	นางสาวบุณยนุช กมลวรเดช	รหัส 52361932
	นางสาวภัณฑิลา เก่งพานิช	รหัส 52362106
	นางสาวอัญชิษฐา ปราสาททรัพย์	รหัส 52362380
ที่ปรึกษาโครงงาน	ดร. นิพัทธ์ จันทรมินทร์	
สาขาวิชา	วิศวกรรมไฟฟ้า	
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์	
ปีการศึกษา	2555	

บทคัดย่อ

ปริญญานิพนธ์นี้นำเสนอการพัฒนารูปแบบการควบคุมการเปิดและปิดมอเตอร์ปั๊มน้ำ ในที่นี้ได้ใช้เทคโนโลยีแบบไร้สายเพื่อรับและส่งสัญญาณระหว่างเครื่องคอมพิวเตอร์ซึ่งที่เป็นศูนย์กลางการควบคุมกับไมโครคอนโทรลเลอร์ที่ควบคุมการทำงานของรีเลย์ซึ่งเชื่อมต่อมอเตอร์ปั๊มน้ำกับแหล่งจ่ายไฟ สถานะการทำงานของมอเตอร์ขึ้นอยู่กับระดับน้ำในถังเก็บน้ำซึ่งถูกวัดค่าด้วยตัวรับรู้ความดันส่วนต่าง ทั้งนี้ผู้ใช้สามารถกำหนดขีดจำกัดล่างและขีดจำกัดบนของระดับน้ำได้ เมื่อเชื่อมต่อระบบให้เริ่มทำงาน การควบคุมจะเป็นแบบอัตโนมัติโดยปริยาย ถึงกระนั้นผู้ใช้ยังคงสามารถหยุดการทำงานของมอเตอร์ได้ทุกขณะที่ต้องการ ในขณะที่มอเตอร์จะเริ่มเดินเครื่องก็ต่อเมื่อระดับน้ำในถังอยู่ต่ำกว่าขีดจำกัดบน ในโครงงานนี้ได้เลือกใช้โมดูลสื่อสารไร้สายย่าน 2.4 GHz รุ่น XBee-PRO เพื่อรับและส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์กับไมโครคอนโทรลเลอร์รุ่น dsPIC30F4011 โดยเขียนชุดคำสั่งควบคุมการรับและส่งข้อมูลและสร้างส่วนต่อประสานกราฟิกกับผู้ใช้ด้วยโปรแกรมวิชวลเบสิก 2010 ร่วมทั้งสร้างแบบจำลองเพื่อทดสอบการเปิดและปิดมอเตอร์ร่วมกับการใช้ตัวรับรู้ความดันส่วนต่างเพื่อวัดระดับน้ำ

Project title	Wireless Remote On-Off Control of Water Pump Motor
Name	Ms. Bunyanuch Kamolvoradej ID. 52361932
	Ms. Punthila Kengphanich ID. 52362106
	Ms. Unchitta Prasatsap ID. 52362380
Project advisor	Mr. Niphat Jantharamin, Ph.D.
Major	Electrical Engineering
Department	Electrical and Computer Engineering
Academic year	2012

Abstract

This thesis presents development of an on-off control scheme for water pump motors. Hereby, wireless technology was adopted for signal reception and transmission between a computer which served as a control center and a microcontroller that regulated a relay connecting the motor to the line. The operating status of the motor was dictated by the water level in the storage tank, which was measured via a differential pressure sensor. The lower and upper limits of the water level in the tank were defined by the user. As the system was energized, it entered the 'Automatic' operating mode by default. In addition, the user could turn off the pump any time as needed. However, the motor started only if the water level was lower than the upper limit. In this project, a 2.4-GHz XBee-PRO OEM RF module was chosen for signal reception and transmission between the computer and the dsPIC30F4011 microcontroller. By means of the Visual Basic 2010, the computer was programmed and a graphical user interface (GUI) was created. Finally, a model of the water pump system with wireless remote control was built for testing along with the different pressure sensor.

กิตติกรรมประกาศ

โครงงานนี้สำเร็จลุล่วงไปได้ด้วยดีด้วยการดูแลจาก ดร. นิพัทธ์ จันทรมินทร์ ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงาน โดยให้คำแนะนำในการพัฒนาชิ้นงาน การทดสอบและปรับปรุงชิ้นงาน ตลอดจนสอนหลักการเขียนปริญญานิพนธ์ด้วยความใส่ใจในรายละเอียดของผลงาน จึงทำให้การดำเนินโครงงานและการเขียนปริญญานิพนธ์สำเร็จลุล่วงไปได้ ผู้ดำเนินโครงงานจึงขอขอบคุณเป็นอย่างสูง

ขอขอบคุณอาจารย์เศรษฐา ตั้งค้าวานิช ที่กรุณาให้คำปรึกษาเกี่ยวกับโมดูลสื่อสารไร้สายย่าน 2.4 GHz รุ่น XBee-PRO ตลอดจนแนะแนวทางการเขียนชุดคำสั่งด้วยโปรแกรมวิชวลเบสิก 2010 (Visual Basic 2010) เพื่อสร้างส่วนต่อประสานกราฟิกกับผู้ใช้

ขอขอบคุณว่าที่ร้อยตรี ธานี โกสุม (พี่ต้น) ที่แนะนำการใช้เครื่องออสซิลโลสโคปและให้คำปรึกษาเกี่ยวกับการเชื่อมต่อรีเลย์เพื่อเปิดและปิดมอเตอร์ปั๊มน้ำ รวมทั้งให้แนวคิดในการทดสอบการรับและส่งข้อมูลแบบไร้สาย

ขอขอบคุณคณาจารย์ทุกท่านที่ให้อบรมสั่งสอนตลอดการศึกษาเล่าเรียนในระดับปริญญาตรี ทำให้สามารถนำความรู้และทักษะในหลายๆด้านมาประยุกต์ใช้กับการดำเนินโครงงานนี้ รวมทั้งขอขอบคุณภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ที่ให้ยืมเครื่องมือวัดจนกระทั่งดำเนินโครงงานสำเร็จ

ขอขอบคุณรัฐบาลไทยที่จัดตั้งกองทุนกู้ยืมเพื่อการศึกษา (กยศ.) ซึ่งสนับสนุนให้ทุนการศึกษาแก่นางสาวอัญชิษฐา ปราสาททรัพย์ ในช่วงการศึกษาระดับปริญญาตรีชั้นปีที่ 1 และ 2

ขอขอบคุณบริษัท ปูนซีเมนต์เอเชีย จำกัด (มหาชน) ที่มอบทุนการศึกษาให้แก่นางสาวอัญชิษฐา ปราสาททรัพย์ ในช่วงการศึกษาระดับปริญญาตรีชั้นปีที่ 3 และ 4 รวมทั้งให้โอกาสได้ไปเรียนรู้และพัฒนาทักษะหลายด้านในช่วงฝึกงานจนได้แนวคิดในการพัฒนาชื่อหัวข้อโครงงาน

เหนือสิ่งอื่นใด ผู้ดำเนินโครงงานขอกราบขอบพระคุณบิดามารดาที่มอบความรัก ความเข้าใจ และคอยเป็นกำลังใจให้อยู่เสมอจนทำให้ประสบกับความสำเร็จอย่างทุกวันนี้

นางสาวบุณยนุช กมลวรเดช
นางสาวภัณฑิลา เก่งพานิช
นางสาวอัญชิษฐา ปราสาททรัพย์

สารบัญ

หน้า

ใบรับรองปริญญานิพนธ์ ..... ก

บทคัดย่อภาษาไทย ..... ข

บทคัดย่อภาษาอังกฤษ ..... ค

กิตติกรรมประกาศ ..... ง

สารบัญ ..... จ

สารบัญตาราง ..... ช

สารบัญรูป ..... ซ

บทที่ 1 บทนำ 1

1.1 ที่มาและความสำคัญของโครงงาน 1

1.2 วัตถุประสงค์ของโครงงาน 1

1.3 ขอบเขตของโครงงาน 2

1.4 ขั้นตอนและแผนการดำเนินงาน 2

1.5 ประโยชน์ที่คาดว่าจะได้รับจากโครงงาน 3

1.6 งบประมาณของโครงงาน 3

บทที่ 2 หลักการสื่อสารไร้สายด้วย XBee-PRO และการวัดระดับน้ำ 4

2.1 ระบบเครือข่ายไร้สาย 4

2.2 คุณสมบัติของโมดูลสื่อสารไร้สายย่าน 2.4 GHz รุ่น XBee-PRO 7

2.3 โมดูลสื่อสารไร้สายย่าน 2.4 GHz รุ่น XBee-PRO 10

2.3.1 การเชื่อมต่อ XBee-PRO กับคอมพิวเตอร์ 11

2.3.2 การเชื่อมต่อ XBee-PRO กับไมโครคอนโทรลเลอร์ 12

2.3.3 การจ่ายไฟเลี้ยงให้ XBee-PRO 13

2.4 ไมโครคอนโทรลเลอร์รุ่น PIC30F4011 14

2.5 ตัวรับรู้ความดันส่วนต่าง 16

บทที่ 3 การออกแบบและสร้างระบบควบคุมมอเตอร์ปั๊มน้ำแบบไร้สาย 18

3.1 การออกแบบขั้นตอนการทำงานของระบบ 18

สารบัญ (ต่อ)

หน้า

3.2 การออกแบบระบบควบคุมการทำงาน 19
3.2.1 การตั้งค่าโมดูลสื่อสารไร้สายย่าน 2.4 GHz รุ่น XBee-PRO 19
3.2.2 การตั้งค่าและควบคุมการทำงานด้วยคอมพิวเตอร์ 23
3.2.3 การประมวลผลและควบคุมรีเลย์ด้วยไมโครคอนโทรลเลอร์ 27
3.3 แบบจำลองของระบบมอเตอร์ปั๊มน้ำ 32

บทที่ 4 ผลการทดสอบและการวิเคราะห์ผล 34
4.1 การทดลองวัดแรงดันไฟฟ้า จากตัวรับรู้ความดันส่วนต่าง 34
4.2 การทดลองหาค่าความคลาดเคลื่อนของค่าระดับน้ำที่แสดงผล 35
4.3 การทดสอบระยะการส่งสูงสุดของ XBee-PRO 39

บทที่ 5 สรุปผลและข้อเสนอแนะ 40
5.1 สรุปผลการดำเนินงาน 40
5.2 ปัญหาและแนวทางการแก้ไข 40
5.3 แนวทางในการพัฒนาต่อไป 41

เอกสารอ้างอิง 42

ภาคผนวก ก คู่มือการติดตั้งและใช้งานโปรแกรม X-CTU ของ XBee-PRO 43
ภาคผนวก ข รหัสต้นฉบับเพื่อสร้างส่วนต่อประสานกราฟิกกับผู้ใช้โดยใช้วิชวลเบสิก 2010 51
ภาคผนวก ค รายละเอียดข้อมูลของไมโครคอนโทรลเลอร์รุ่น PIC30F4011 70
ภาคผนวก ง รหัสต้นฉบับเพื่อควบคุมไมโครคอนโทรลเลอร์ 88
ภาคผนวก จ รายละเอียดข้อมูลของตัวรับรู้ความดันส่วนต่าง รุ่น MPX5010DP 97

ประวัติผู้ดำเนินโครงงาน 104

สารบัญตาราง

ตารางที่ หน้า

2.1 การเปรียบเทียบคุณสมบัติของเทคโนโลยีการเข้าถึงไร้สายที่สำคัญ....6
2.2 การเปรียบเทียบคุณสมบัติของแต่ละรุ่น....8
4.1 ค่าแรงดันไฟฟ้าจากตัวรับรู้ความดันส่วนต่าง....34
4.2 ค่าระดับน้ำที่แสดงบนส่วนต่อประสานกราฟิกกับผู้ใช้ก่อนใช้ค่าชดเชย....36
4.3 ค่าระดับน้ำที่แสดงบนส่วนต่อประสานกราฟิกกับผู้ใช้หลังจากใช้ค่าชดเชย....38

สารบัญรูป

รูปที่ หน้า

2.1 ลักษณะและขนาดของโมดูลสื่อสารไร้สายย่าน 2.4 GHz รุ่น XBee-PRO 10
2.2 แผนภาพวงจรการเชื่อมต่อ XBee-PRO กับคอมพิวเตอร์ 12
2.3 แผนภาพวงจรการเชื่อมต่อโมดูล XBee-PRO กับอุปกรณ์ภายนอก 13
2.4 การจ่ายไฟเลี้ยงให้ XBee-PRO ผ่านตัวแบ่งแรงดันแบบตัวต้านทาน 14
2.5 การจ่ายไฟเลี้ยงให้ XBee-PRO ผ่านไอซีควบคุมค่าแรงดัน 14
2.6 รูปแบบแสดงการจัดขาสัญญาณของ dsPIC30F4011 15
2.7 กราฟคุณลักษณะของตัวรับรู้ความดันส่วนต่างรุ่น MPX5010DP 17
3.1 ขั้นตอนการทำงานระบบเปิดและปิดปั๊มน้ำโดยการควบคุมทางไกลแบบไร้สาย 18
3.2 ตัวรับและส่งสัญญาณของ XBee-PRO 19
3.3 แผงวงจร ZX-XBee 20
3.4 แผงวงจร ADX-XBee 20
3.5 หน้าต่างเริ่มต้นของโปรแกรม X-CTU 21
3.6 หมายเลขประจำอุปกรณ์ของ XBee-PRO 22
3.7 การลงหมายเลขประจำอุปกรณ์ของ XBee-PRO บน X-CTU 22
3.8 การตั้งค่าที่อยู่ของเป้าหมายที่จะส่งข้อมูลถึงกัน 23
3.9 หน้าต่างของส่วนต่อประสานกราฟิกกับผู้ใช้ 24
3.10 ลำดับการทำงานของส่วนควบคุมที่กำหนดด้วยส่วนต่อประสานกราฟิกกับผู้ใช้ 25
3.11 แผงวงจรไมโครคอนโทรลเลอร์ dsPIC30F4011 27
3.12 แผนผังการตรวจสอบข้อมูลที่รับมาจาก GUI 28
3.13 ลำดับการทำงานโปรแกรมหลักในไมโครคอนโทรลเลอร์ 30
3.14 แผงวงจรรีเลย์และวงจรขับรีเลย์ 31
3.15 แผงวงจรรีเลย์ 31
3.16 ตัวรับรู้ความดันส่วนต่างรุ่น MPX5010DP 32
3.17 มอเตอร์ปั๊มน้ำรุ่น AP2500 พิกัด 2,500 W 32
3.18 ส่วนประกอบและการเชื่อมต่ออุปกรณ์ในแบบจำลองของระบบควบคุมปั๊มน้ำ 33
4.1 ความสัมพันธ์ระหว่างการแรงดันไฟฟ้าจากตัวรับรู้ความดันส่วนต่างกับระดับน้ำ 35
4.2 ค่าความคลาดเคลื่อนของการแสดงค่าระดับน้ำบนส่วนต่อประสานกราฟิกกับผู้ใช้ 36
4.3 ความสัมพันธ์ระหว่างค่าระดับที่แสดงผลกับค่าระดับน้ำจริงหลังจากใช้ค่าชดเชย 37

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ในปัจจุบันเทคโนโลยีมีการพัฒนาให้ทันสมัยมากขึ้น จึงทำให้มีการนำเทคโนโลยีมาพัฒนาระบบควบคุมทางไกลในโรงงานอุตสาหกรรมที่ห้องควบคุมอยู่ไกลจากเครื่องจักรหรืออุปกรณ์ไฟฟ้าที่ต้องการควบคุมซึ่งมีการพัฒนาระบบควบคุมโดยผ่านทางเทคโนโลยีสื่อสารไร้สายแบบต่างๆ เช่น บลูทูธ (Bluetooth) วายฟาย (Wireless Fidelity) เป็นต้น เพื่อให้สามารถควบคุมได้จากระยะทางไกลแทนการใช้สายควบคุมเนื่องจากในระยะทางไกลจะมีสิ่งกีดขวางจากอุปกรณ์อื่นๆ ในโรงงาน ทำให้การเชื่อมต่อสายควบคุมนั้นเป็นไปได้ยุ่งยาก และการใช้สายควบคุมที่ยาวจะมีค่าใช้จ่ายในการติดตั้งสูง นอกจากนี้ระบบควบคุมทางไกลแบบไร้สายยังลดการใช้พลังงานให้น้อยลง มีอายุการใช้งานยาวนาน

โครงงานนี้จึงนำการควบคุมทางไกลแบบไร้สายมาพัฒนาระบบควบคุมเปิดและปิดมอเตอร์ปั๊มน้ำโดยผ่านอุปกรณ์โมดูลสื่อสารข้อมูลอนุกรมไร้สายย่าน 2.4 GHz รุ่น XBee-PRO โดยใช้ไมโครคอนโทรลเลอร์ในการประมวลผลและควบคุมการทำงานของอุปกรณ์ต่างๆ นั้นคือถ้าระดับน้ำในถังมีค่าต่ำกว่าที่กำหนด ซึ่งบ่งบอกว่าน้ำในถังมีปริมาณน้อยเกินไป

ไมโครคอนโทรลเลอร์จะสั่งให้มอเตอร์ทำงานเพื่อปั๊มน้ำเข้ามาเก็บในถัง จนกระทั่งระดับน้ำในถังมีค่าสูงถึงค่าที่กำหนด ไมโครคอนโทรลเลอร์จะสั่งให้มอเตอร์หยุดทำงาน และจะสั่งให้มอเตอร์ทำงานอีกครั้งเมื่อระดับน้ำในถังลดลงจนต่ำกว่าที่กำหนดอีกครั้ง

1.2 วัตถุประสงค์ของโครงงาน

เพื่อพัฒนาระบบควบคุมการเปิดและปิดมอเตอร์ปั๊มน้ำที่อยู่ห่างไกลจากห้องควบคุม โดยสั่งการด้วยเครื่องคอมพิวเตอร์ที่เป็นศูนย์กลางในการควบคุมซึ่งมีการรับและส่งข้อมูลแบบไร้สายกับไมโครคอนโทรลเลอร์ที่ควบคุมการทำงานของรีเลย์สำหรับเปิดและปิดมอเตอร์ปั๊มน้ำ

1.3 ขอบเขตของโครงงาน

1) การเปิดและปิดมอเตอร์ปั๊มน้ำขึ้นอยู่กับระดับน้ำต่ำสุดและสูงสุดที่กำหนดตามลำดับ
2) ใช้ไมโครคอนโทรลเลอร์ควบคุมการทำงานของรีเลย์เพื่อเปิดและปิดมอเตอร์ปั๊มน้ำ
3) รับและส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ที่เป็นศูนย์กลางในการควบคุมกับไมโครคอนโทรลเลอร์โดยใช้โมดูลสื่อสารไร้สาย ย่าน 2.4 GHz รุ่น XBee-PRO
4) เขียนคำสั่งควบคุมการรับและส่งข้อมูลผ่าน XBee-PRO และสร้างหน้าต่างส่วนต่อประสานกราฟิกกับผู้ใช้ (Graphic User Interface: GUI) โดยใช้โปรแกรมวิชวลเบสิก (Visual Basic)

1.4 ขั้นตอนและแผนการดำเนินงาน

รายละเอียด	พ.ศ. 2555							พ.ศ. 2556		
	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.
1) ศึกษาหลักการทำงานของ XBee-PRO										
2) ศึกษาการเขียนชุดคำสั่งด้วยโปรแกรมวิชวลเบสิก 2010										
3) เขียนคำสั่งควบคุมการรับและส่งข้อมูลระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์										
4) ใช้ไมโครคอนโทรลเลอร์ควบคุมรีเลย์										
5) สร้างแบบจำลองเพื่อทดสอบการเปิดและปิดมอเตอร์										
6) สรุปผลและจัดทำรูปเล่มปริญญานิพนธ์										

1.5 ประโยชน์ที่คาดว่าจะได้รับจากโครงงาน

ระบบควบคุมการเปิดและปิดมอเตอร์ปั๊มน้ำโดยการควบคุมทางไกลแบบไร้สายที่พัฒนาขึ้นในโครงงานนี้ สามารถนำไปประยุกต์ใช้กับการควบคุมอุปกรณ์ไฟฟ้าอื่นๆที่อยู่ไกลจากห้องควบคุม เพื่อเป็นการอำนวยความสะดวกแก่ผู้ที่นำไปใช้ในงานต่างๆมีอายุการใช้งานที่ยาวนาน และเป็นการลดต้นทุนจากระบบเดิมที่ใช้สายไฟในการเดินสายไปยังห้องควบคุม

1.6 งบประมาณของโครงงาน

1) โมดูลสื่อสารไร้สายย่าน 2.4 GHz รุ่น XBee-PRO 2 ตัว	4,200 บาท
2) แผงวงจรไมโครคอนโทรลเลอร์ dsPIC30F4011	700 บาท
3) แผงวงจรรีเลย์	200 บาท
4) มอเตอร์ปั๊มน้ำ พิกัด 2,500 W	300 บาท
5) แบบจำลองถังน้ำ	1,000 บาท
6) ตัวรับรู้ความดันส่วนต่างรุ่น MPX5010DP	350 บาท
7) ค่าถ่ายเอกสารและเข้าเล่มปริญญานิพนธ์	1,000 บาท
รวมเป็นเงินทั้งสิ้น (เจ็ดพันเจ็ดร้อยห้าสิบบาทถ้วน)	7,750 บาท

หมายเหตุ: ถัวเฉลี่ยทุกรายการ

บทที่ 2
หลักการสื่อสารไร้สายด้วย XBee-PRO และการวัดระดับน้ำ

2.1 ระบบเครือข่ายไร้สาย

เนื่องจากในโครงงานนี้ได้นำโมดูลสื่อสารไร้สายย่าน 2.4 GHz รุ่น XBee-PRO มาใช้ควบคุมการเปิดและปิดมอเตอร์ปั๊มน้ำแบบไร้สายในระยะทางไกล โดยการเขียนชุดคำสั่งให้ไมโครคอนโทรลเลอร์ประมวลผลข้อมูลที่ได้รับ เพื่อควบคุมการปิดและเปิดมอเตอร์ปั๊มน้ำ และใช้ตัวรับรู้ความดันส่วนต่างเพื่อวัดระดับน้ำ จึงควรมีความรู้เบื้องต้นเกี่ยวกับระบบเครือข่ายไร้สาย (Wireless local area network: WLAN) ความเข้าใจคุณลักษณะและหลักการทำงานทั้ง XBee-PRO ซึ่งใช้สื่อสารข้อมูลทางไกลแบบไร้สาย คุณสมบัติของไมโครคอนโทรลเลอร์ที่นำมาใช้ในโครงงานนี้เพื่อควบคุมการทำงานของมอเตอร์ปั๊มน้ำ รวมทั้งชนิดและหลักการทำงานของตัวรับรู้ความดันส่วนต่างที่เลือกใช้

ระบบเครือข่ายไร้สายเป็นระบบการสื่อสารข้อมูลที่มีความคล่องตัวอย่างมาก ซึ่งอาจจะนำมาใช้ทดแทนหรือเพิ่มต่อกับระบบเครือข่ายแลนใช้สายแบบดั้งเดิมโดยใช้การส่งคลื่นความถี่ในย่านวิทยุและคลื่นอินฟราเรดในการรับและส่งข้อมูลระหว่างคอมพิวเตอร์แต่ละเครื่อง ผ่านอากาศ ทะลุกำแพง เพดานหรือสิ่งก่อสร้างอื่นๆ โดยปราศจากความต้องการของการเดินสาย นอกจากนั้นระบบเครือข่ายไร้สายยังมีคุณสมบัติครอบคลุมทุกอย่างเหมือนกับระบบแบบใช้สายแต่ความแตกต่างที่สำคัญคือไม่ต้องใช้สายตัวนำจึงทำให้การเคลื่อนย้ายในการใช้งานทำได้สะดวก ในขณะที่ระบบแบบใช้สายต้องใช้เวลาและการลงทุนในการปรับเปลี่ยนตำแหน่งการใช้งานเครื่องคอมพิวเตอร์

เทคโนโลยีการเข้าถึงข้อมูลเฉพาะในส่วนที่เป็นการเข้าถึงไร้สาย (Broadband wireless access: BWA) นั้นสามารถแบ่งออกตามลักษณะของการเข้าถึงได้ดังต่อไปนี้

1) ระบบการติดต่อไร้สายส่วนบุคคล (Personal area network: PAN) คือเทคโนโลยีการเข้าถึงไร้สายในพื้นที่เฉพาะส่วนบุคคลบริเวณไม่กว้างมากนัก ระยะทางไม่เกิน 10 m และมีอัตราการรับและส่งข้อมูลความเร็วสูงไม่เกิน 1 Mbps เทคโนโลยีเหล่านี้ติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์และอุปกรณ์ต่อพ่วง (Peripherals) ให้สามารถรับและส่งข้อมูลถึงกันได้

2) ระบบเครือข่ายท้องถิ่น (Local area network: LAN) คือเทคโนโลยีการเข้าถึงไร้สายในพื้นที่เฉพาะมีระยะทางที่ส่งไม่เกิน 100 m มีอัตราการรับและส่งข้อมูล

ความเร็วในช่วง 11–54 Mbps และติดตั้งสถานีเรียกว่าการต่อจุดทั้งหมดเพื่อทำหน้าที่เชื่อมต่อสัญญาณระหว่างอุปกรณ์ปลายทาง (Terminal equipment) ที่เป็นเซลล์ขนาดเล็กมาก (Pico cells) ข้อจำกัดการใช้งานเทคโนโลยีนี้คือจำนวนของผู้ใช้งานในขณะใดขณะหนึ่งพร้อมกัน ระยะห่างระหว่างจุดทั้งหมดกับอุปกรณ์ปลายทาง (Terminal equipment) และความพอเพียงของคลื่นความถี่ เนื่องจากส่วนใหญ่เป็นการใช้งานในลักษณะที่ได้รับการยกเว้นใบอนุญาต (Unlicensed) จึงต้องใช้คลื่นความถี่ร่วมกันกับผู้ประกอบการ

3) ระบบเครือข่ายระดับเมือง (Metropolitan area network: MAN) คือเทคโนโลยีการเข้าถึงไร้สายในพื้นที่เขตเมืองหรือพื้นที่ขนาดใหญ่ ซึ่งมีระยะการส่งอยู่ในช่วงประมาณ 10 – 50 km ทั้งนี้ขึ้นอยู่กับคลื่นความถี่ที่ใช้งาน และมีอัตราการรับและส่งข้อมูลที่ความเร็วสูงในช่วง 11 – 100 Mbps ขึ้นไปโดยขึ้นอยู่กับการใช้งานว่าเป็นเอ็นแอลโอเอส (Non-line-of-sight: NLOS) หรือแอลโอเอส (Line-of-sight: LOS) แต่เดิมนั้นเทคโนโลยีการเข้าถึงดังกล่าวมุ่งเน้นที่การใช้งานแบบประจำที่ (Fixed) ซึ่งอุปกรณ์ของผู้ใช้บริการมักติดตั้งอยู่กับที่ภายนอกอาคาร (Outdoor) ก่อนมีการพัฒนาไปเป็นการใช้งานภายในอาคาร (Indoor) แล้วจึงพัฒนาออกแบบให้สามารถใช้งานแบบเคลื่อนที่ (Mobile) ได้

4) ระบบเครือข่ายระหว่างประเทศ (Wide area network: WAN) คือเทคโนโลยีการเข้าถึงไร้สายบริเวณกว้างมากโดยครอบคลุมพื้นที่ในเขตภูมิภาคหรือทั่วประเทศมีอัตราการรับและส่งข้อมูลที่มีความเร็วไม่เกิน 1.5 Mbps เพราะเน้นการใช้งานในแบบเคลื่อนที่

ปัจจุบันระบบเครือข่ายแบบไร้สายถูกสร้างและพัฒนาเทคโนโลยีต่างๆขึ้นมาเพื่อให้เกิดความสะดวกสบายในการใช้งานหลายด้านมากขึ้น แต่ละเทคโนโลยีของระบบเครือข่ายแบบไร้สายนี้มีคุณสมบัติแตกต่างกันขึ้นอยู่กับการนำไปใช้งาน ดังนั้นก่อนเลือกใช้เทคโนโลยีต่างๆ เราจำเป็นต้องรู้คุณสมบัติของแต่ละเทคโนโลยีว่ามีข้อดีหรือข้อจำกัดเพื่อเลือกให้เหมาะสมกับงานที่นำไปใช้ อาทิ เทคโนโลยีการเข้าถึงข้อมูลเฉพาะแบบไร้สาย ระยะทางการรับส่งสัญญาณ ความถี่ เป็นต้น ดังแสดงในตารางที่ 2.1 ซึ่งในโครงงานนี้ได้เลือกใช้เทคโนโลยีแบบระบบสื่อสารไร้สาย (Zigbee) และเลือกใช้อุปกรณ์ในการรับและส่งสัญญาณคือ โมดูลสื่อสารไร้สายย่าน 2.4 GHz รุ่น XBee-PRO [1]

ตารางที่ 2.1 การเปรียบเทียบคุณสมบัติของเทคโนโลยีการเข้าถึงไร้สายที่สำคัญ [1]

เทคโนโลยี	เทคโนโลยีแบบไร้สาย	ระยะการทำงาน (m)	ความถี่ (GHz)
ยูดับเบิลยูบี (UWB)	ระบบการติดต่อไร้สายส่วนบุคคล	10	เปลี่ยนแปลงได้
บลูทูธ (Bluetooth)	ระบบการติดต่อไร้สายส่วนบุคคล	10	2.4
ระบบสื่อสารไร้สาย (Zigbee)	ระบบการติดต่อไร้สายส่วนบุคคล	10	868 MHz, 915 MHz, 2.4 GHz
วายฟาย (WiFi)	ระบบเครือข่ายท้องถิ่น	100	5
	ระบบเครือข่ายท้องถิ่น	100	2.4
	ระบบเครือข่ายท้องถิ่น	100	2.4
วายแม็กซ์ (WiMAX)	ระบบเครือข่ายระดับเมือง	6,400 – 9,600	11
	ระบบเครือข่ายระดับเมืองแบบเคลื่อนที่	1,600 – 4,800	2 – 6
วายด์แบนด์โคดดิวิชันมัลติเพิลแอ็คเซส	ระบบเครือข่ายระหว่างประเทศ	1,600 – 8,000	1.8, 1.9, 2.1
โคดดิวิชันมัลติเพิลแอ็คเซส (Cdma 2000)	ระบบเครือข่ายระหว่างประเทศ	1,600 – 8,000	0.4, 0.8, 0.9, 1.7, 1.8, 1.9, 2.1
เอ็มบีดับเบิลยูเอ (MBWA)	ระบบเครือข่ายระหว่างประเทศแบบเคลื่อนที่	4,000 – 12,000	3.5

จากตารางที่ 2.1 ยูดับเบิลยูบี (Ultra wide band: UWB) เป็นเทคโนโลยีการสื่อสารไร้สายรูปแบบใหม่ใช้การส่งผ่านข้อมูลแบบพัลซ์ (Pulse) สั้นๆผ่านคลื่นวิทยุความถี่กว้างต่างจากการส่งผ่านข้อมูลผ่านคลื่นความถี่วิทยุแบบระบบวิทยุแบบแถบความถี่แคบ และการส่งผ่านข้อมูลผ่านคลื่นความถี่วิทยุแบบกว้างในขณะที่วายแม็กซ์ (WiMax) คือเครือข่ายบริการอินเตอร์เน็ตไร้สายความเร็วสูง ที่มีพื้นที่ครอบคลุมบริเวณกว้างมากส่วนวายด์แบนด์ โคด ดิวิชัน มัลติเพิล แอ็คเซส (Wideband code division multiple access: WCDMA) คือระบบเครือข่ายมาตรฐานใหม่ที่พัฒนามาจากสามจี (3G)

2.2 คุณสมบัติของโมดูลสื่อสารไร้สายย่าน 2.4 GHz รุ่น XBee-PRO

การเลือกใช้งานโมดูลสื่อสารข้อมูลอนุกรมไร้สายควรพิจารณาจากคุณสมบัติต่อไปนี้

1) ระยะการส่งสัญญาณ

โมดูลสื่อสารข้อมูลอนุกรมไร้สายจะแยกรุ่นสำหรับระยะการรับส่งอย่างชัดเจนด้วยคำว่าโปร (PRO) โดยรุ่นระยะสั้นกำลังการส่ง 1 – 2 mW มีระยะรับส่งประมาณ 100 – 120 m ส่วนรุ่นระยะไกลแบบโปร (PRO) กำลังส่งจะอยู่ในช่วงระหว่าง 50 – 60 mW โดยมีระยะประมาณ 1500 m ซึ่งระยะทางการส่งสัญญาณขึ้นอยู่กับสภาพแวดล้อมของระบบ และสายอากาศที่ใช้งาน เนื่องจากเป็นความถี่ 2.4 GHz ซึ่งเป็นย่านความถี่สูง อัตราการลดทอนสัญญาณจึงสูงและสิ่งกีดขวางส่งผลอย่างมากต่อระยะทางการส่งสัญญาณ

2) ซีรีย์ (Series)

โมดูลสื่อสารข้อมูลอนุกรมไร้สายแบ่งออกเป็น 2 ซีรีย์ คือซีรีย์ 1 และซีรีย์ 2 โดยซีรีย์ 2 สามารถแบ่งเป็น 2 รุ่นย่อยคือ แซดเน็ต (Znet) 2.5 และแซดบี (ZB) แต่ในปัจจุบันโมดูลสื่อสารข้อมูลอนุกรมไร้สายซีรีย์ 2 เหลือแค่รุ่นแซดบีเท่านั้นเพื่อเพิ่มความสามารถในการยกระดับเฟิร์มแวร์ (Firmware) ของโมดูลสื่อสารข้อมูลอนุกรมไร้สายผ่านอากาศได้ แต่โดยส่วนมากไม่ใช้ฟังก์ชันนี้ ข้อแตกต่างระหว่างซีรีย์ 1 และซีรีย์ 2 ที่สำคัญคือซีรีย์ 1 ทำโครงข่ายแบบเมช (Mesh Network) ไม่ได้ แต่ซีรีย์ 2 ทำได้ นอกจากนี้ยังสามารถเปรียบเทียบคุณสมบัติของแต่ละรุ่นได้ดังตารางที่ 2.2

จากตารางที่ 2.2 เพอริเฟอรัล (Peripheral) คือ ค่าที่ใช่เรียกอุปกรณ์คอมพิวเตอร์ใดๆที่ไม่ใช่ส่วนประกอบหลักของคอมพิวเตอร์ในส่วนของเอดีซี (Analog to digital convertor: ADC) คือไอซีแปลงแอนะล็อกเป็นดิจิตอลที่ความละเอียดต่างๆ แล้วแต่ชนิดส่วนการยกระดับเฟิร์มแวร์ (Firmware) คือ ซอฟต์แวร์ (Software) ที่บริษัทผู้ผลิตพัฒนาขึ้นเพื่อเพิ่มสมรรถนะในการทำงานและโครงข่ายแบบเมช (Mesh network) คือโครงข่ายไร้สายที่สามารถส่งผ่านข้อมูลถึงทุกจุดได้ต่อกันไป

ตารางที่ 2.2 การเปรียบเทียบคุณสมบัติของแต่ละรุ่น [2]

คุณสมบัติ	ซีรีย์ 1	ซีรีย์ 2	ซีรีย์ 1 แบบโปร	ซีรีย์ 2 แบบโปร
กำลังขาเข้า	3.3 V ที่ 50 mA	3.3 V ที่ 40 mA	3.3 V ที่ 215 mA	3.3 V ที่ 295 mA
ความเร็วในการส่งผ่านข้อมูลในอากาศ (kbps)	250 kbps	250 kbps	250 kbps	250 kbps
กำลังขาออก	1 mW (+0 dBm)	2 mW (+3 dBm)	60 mW (+18 dBm)	50 mW (+17 dBm)
ระยะทาง (m)	100 m	120 m	1,500 m	1,600 m
เพอริเฟอรัล (Peripheral)	ขาเข้าแอลดีซี 6 – 10 บิต 8 ดิจิตอลของขา I/O pins	ขาเข้าแอลดีซี 6 – 10 บิต 8 ดิจิตอลของขา I/O pins	ขาเข้าแอลดีซี 6 – 10 บิต 8 ดิจิตอลของขา I/O pins	ขาเข้าแอลดีซี 6 – 10 บิต 8 ดิจิตอลของขา I/O pins
การยกระดับเฟิร์มแวร์ (Firmware)	ที่ติดตั้ง (Local)	การกำหนดค่าโอเวอร์แอร์ (Over-air) ของแซดบี	ที่ติดตั้ง (Local)	การกำหนดค่าโอเวอร์แอร์ (Over-air) ของแซดบี
รูปแบบโครงข่าย (Network)	แบบจุดต่อจุดและจุดต่อหลายจุด	แบบจุดต่อจุดหรือจุดต่อหลายจุดหรือโครงข่ายเมช	แบบจุดต่อจุดและจุดต่อหลายจุด	แบบจุดต่อจุดหรือจุดต่อหลายจุดหรือโครงข่ายเมช

3) สายอากาศของโมดูลสื่อสารข้อมูลอนุกรมไร้สาย

สายอากาศของโมดูลสื่อสารข้อมูลไร้สายสามารถแบ่งได้ 4 แบบ ดังต่อไปนี้

ก) สายอากาศแบบชิป (Chip) เหมาะกับการใช้งานในโครงงานที่ต้องการขนาดเล็ก เพราะการใช้สายอากาศแบบนี้ สายอากาศไม่เกะกะนำไปใส่กล่องได้ แต่ได้เฉพาะกล่องพลาสติก ไม่สามารถใช้กล่องเหล็กได้ เนื่องจากใส่กล่องเหล็กสัญญาณจะไม่สามารถส่งออกมานอกกล่องเหล็กได้ หากต้องใช้กล่องเหล็กควรเลือกใช้สายอากาศที่ต่อออกมานอกกล่องเหล็ก

ข) สายอากาศแบบขดลวดสายไฟ (Wire) จะมีระยะการส่งสัญญาณและความเสถียรได้ตามลักษณะเฉพาะของแต่ละรุ่น และด้วยสายอากาศที่ยื่นออกมาลักษณะนี้ บางทีผู้ใช้อาจจะรู้สึกเกะกะ ทำให้ใส่กล่องที่ออกแบบมาไม่ได้ เหมาะกับการใช้งานแบบทั่วไป

ค) สายอากาศแบบยูเอฟแอล (UFL) จะมีระยะการส่งสัญญาณและความเสถียรได้ตามลักษณะเฉพาะของแต่ละรุ่น เหมาะกับงานที่ออกแบบใส่ในกล่อง และต้องการให้สายอากาศยื่นออกมานอกกล่อง เนื่องจากการที่ต้องต่อสายยูเอฟแอล (UFL) ถึง เอสเอ็มเอ (SMA) ออกมาเพิ่มเติมตรงจุดนี้จะทำให้เกิดการลดทอนสัญญาณบางส่วน แต่ก็จะมีการขยายสัญญาณที่สายอากาศอีกที จึงต้องไปพิจารณาอัตราขยายที่สายอากาศต่อด้วย (อัตราขยายเรียกว่า เกน (Gain) จะมีระยะและความเสถียรได้ตามลักษณะเฉพาะของแต่ละรุ่น มีหน่วยเป็นเดซิเบล (dB))

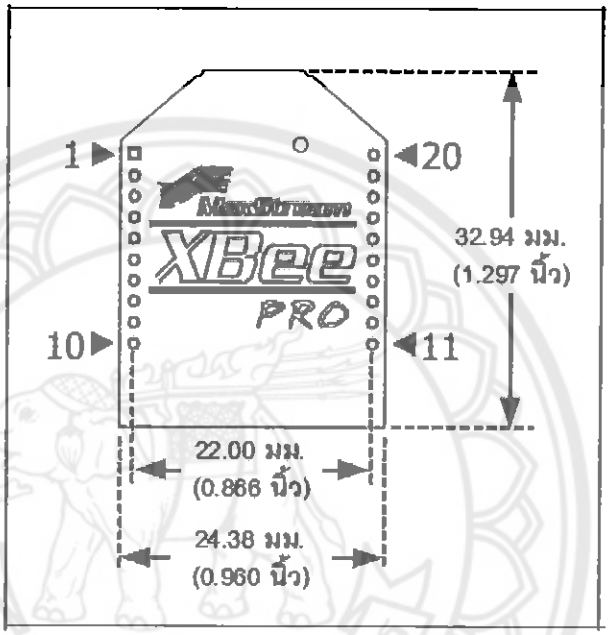
ง) สายอากาศแบบเอสเอ็มเอ (SMA) จะมีระยะการส่งสัญญาณและความเสถียรได้ตามลักษณะเฉพาะของแต่ละรุ่นสูงสุด ต่องานร่วมกับสายอากาศ จะมีการขยายสัญญาณที่สายอากาศอีกที ซึ่งในการต่อใช้งานจริงการออกแบบใส่กล่อง จะต้องออกแบบให้มีตำแหน่งของโมดูลสื่อสารข้อมูลอนุกรมแบบไร้สายให้ใกล้กับรูเจาะของกล่อง

4) การเลือกอุปกรณ์มาใช้งานร่วมกับ XBee-PRO

โมดูลสื่อสารข้อมูลอนุกรมไร้สายมีพิน (Pin) 20 ขา โดยมีระยะห่างประมาณ 2.0 mm ซึ่งไม่สามารถต่อกับแผงการทดลองได้เพราะโดยทั่วไปในการออกแบบวงจรอิเล็กทรอนิกส์ มักวางขาไอซีด้วยระยะห่าง 2.54 mm อย่างไรก็ตามยังมีทางเลือกอื่น ได้แก่ รุ่น ESOC028 หากไม่อยากนำโมดูลสื่อสารข้อมูลอนุกรมไร้สายมาบัดกรีลงแผงวงจรโดยตรง สามารถใช้ช่องเสียบ (Socket) สำหรับซีพียูแทน [2]

2.3 โมดูลสื่อสารไร้สายย่าน 2.4 GHz รุ่น XBee-PRO

หลังจากพิจารณาคุณสมบัติของโมดูลสื่อสารข้อมูลอนุกรมไร้สายแต่ละรุ่น ในโครงงานนี้ได้เลือกใช้โมดูลสื่อสารไร้สายย่าน 2.4 GHz รุ่น XBee-PRO ซึ่งสามารถส่งสัญญาณระยะทางไกลสุดถึง 1,500 m (เมื่ออยู่กลางแจ้งและไม่มีสิ่งกีดขว้าง) มีสายอากาศแบบขดลวดสายไฟ จึงไม่เกะกะและเหมาะกับการทำโครงงานขนาดเล็ก ลักษณะและขนาดของ XBee-PRO แสดงดังรูปที่ 2.1



รูปที่ 2.1 ลักษณะและขนาดของโมดูลสื่อสารไร้สายย่าน 2.4 GHz รุ่น XBee-PRO [3]

คุณสมบัติโดยทั่วไปของโมดูลสื่อสารไร้สายย่าน 2.4 GHz รุ่น XBee-PRO มีดังต่อไปนี้

1) ความถี่ในการทำงานสูง: 2.4 GHz
2) สายอากาศ: มีสายอากาศแบบขดลวดสายไฟ
3) ระยะทำการในร่ม: สูงสุดประมาณ 100 m
4) ระยะทำการกลางแจ้ง (แบบ Line-of-sight): สูงสุดประมาณ 1,500 m
5) กำลังส่ง: 60 mW (18 dBm)
6) ความไวในการรับส่งสัญญาณ: -100 dBm
7) การทำงานของขาพอร์ต: สามารถกำหนดผ่านช่องซอฟต์แวร์เอ็กซ์ซีทียู (X-CTU) เพื่อทำให้สัญญาณขาเข้าที่เข้ามาเป็นแอนะล็อกหรือดิจิตอล เมื่อออกไปจากขาออกจะเป็นสัญญาณดิจิตอล โดยมีวงจรแปลงสัญญาณแอนะล็อกเป็นดิจิตอลมีความละเอียด 10 บิต
8) ไฟเลี้ยง 2.8 ถึง 3.4 V

9) กระแสไฟฟ้า: เมื่อส่งข้อมูล 215 mA, รับข้อมูล 55 mA, น้อยกว่า 10 μA ในโหมดลดพลังงานที่ไฟเลี้ยง +3.3 V

10) อุณหภูมิใช้งาน: -40 ถึง 85°C

คุณสมบัติด้านการสื่อสารข้อมูลของ XBee-PRO มีดังต่อไปนี้

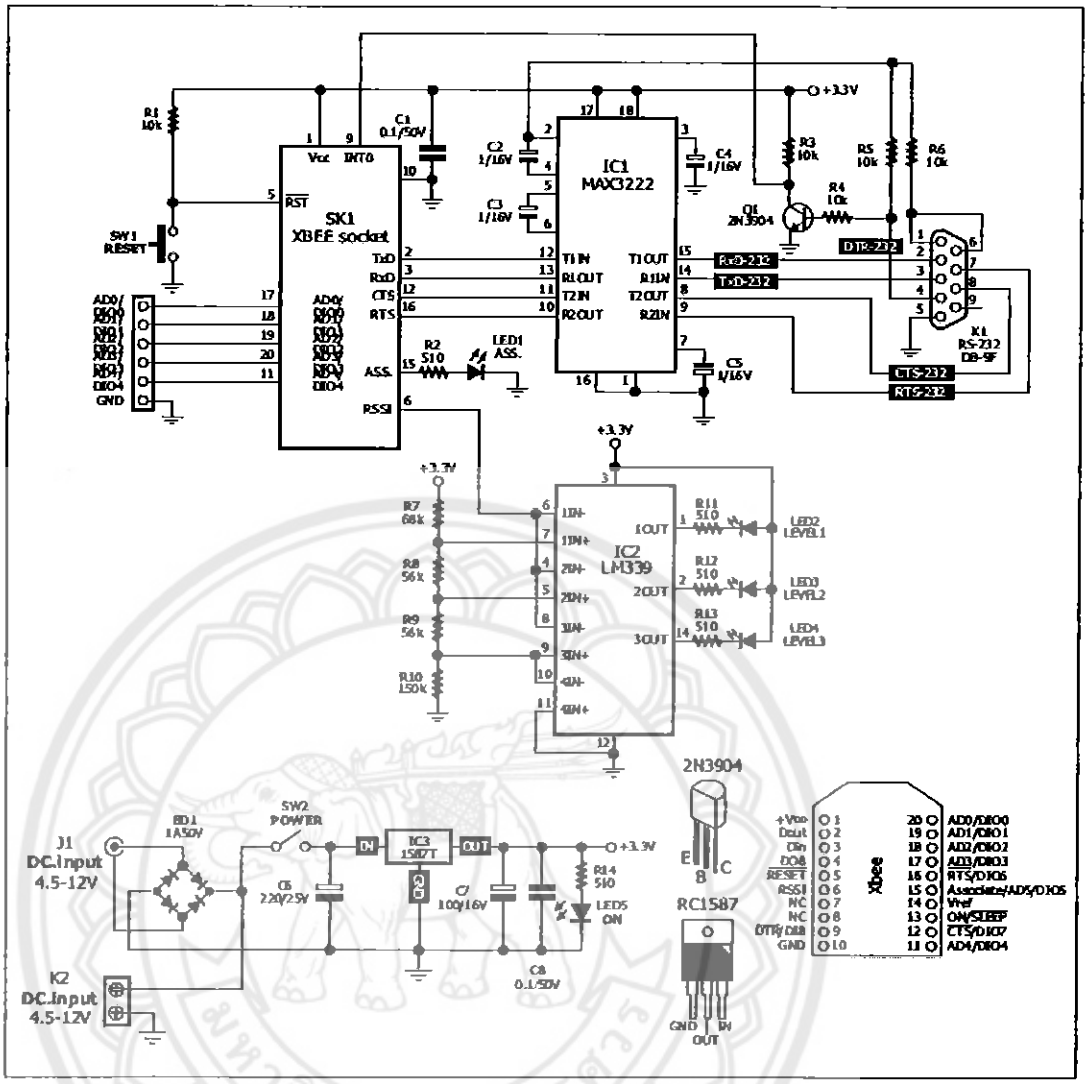
1) สามารถทำงานเป็นอุปกรณ์มาสเตอร์และสเลฟได้
2) อัตราถ่ายทอดข้อมูลผ่านคลื่นวิทยุ: 250,000 บิตต่อวินาที
3) อัตราการถ่ายทอดข้อมูลอนุกรม (Baud rate): 1,200 – 115,200 บิตต่อวินาที
4) รูปแบบโครงข่ายข้อมูลที่รองรับ: แบบจุดต่อจุด (Point-to-point), จุดต่อหลายจุด (Point-to-multipoint) และเข้ากันได้กับอุปกรณ์ตามมาตรฐานรหัส 802.15.4
5) ทางเลือกแอดเดรส: PAN ID, ช่อง (Channel) และแอดเดรส (Addresses) สำหรับแอดเดรสสามารถกำหนดรหัสได้มากถึง 65,000 รหัส
6) เทคโนโลยีในการกระจายคลื่น: ดีเอสเอสเอส (Direct sequence spread spectrum: DSSS)
7) รองรับการทำงานทั้งแบบเอพีไอ (Application programming interface: API) และคำสั่งเอที สามารถกำหนดได้ผ่านทางซอฟต์แวร์เอ็กซ์-ซีทียู (X-CTU) [3]

2.3.1 การเชื่อมต่อ XBee-PRO กับคอมพิวเตอร์

ในการตั้งค่าพารามิเตอร์ต่างๆของ XBee-PRO จำเป็นต้องกระทำผ่านการเชื่อมต่อกับคอมพิวเตอร์ โดยในโครงงานนี้ได้เลือกใช้แผงวงจร ZX-XBee ซึ่งมีแผนภาพวงจรดังรูปที่ 2.2

คุณสมบัติที่สำคัญของแผงวงจร ZX-XBee มีดังต่อไปนี้

1) มีตัวต่อตัวเมียเพื่อติดตั้ง XBee-PRO
2) มีจุดต่อขาพอร์ต D0 ถึง D4 ของ XBee-PRO ซึ่งสามารถกำหนดลักษณะการทำงานได้ผ่านทางโปรแกรมเอ็กซ์ซีทียู (X-CTU)
3) เชื่อมต่อคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม
4) ใช้ร่วมกับโปรแกรม X-CTU เพื่อกำหนดค่าพารามิเตอร์และยกระดับเฟิร์มแวร์
5) ใช้ไฟเลี้ยงในย่าน 4.5 – 12 V บนแผงวงจรมีวงจรควบคุมไฟเลี้ยงคงที่ที่ 3.3 V
6) มีแอลอีดีแสดงความพร้อมในการติดต่อสื่อสารข้อมูลเอเอสเอส และแอลอีดี แสดงความแรงของการกระจายคลื่น 3 ระดับ โดยแสดงผลเมื่อรับส่งข้อมูลเท่านั้น

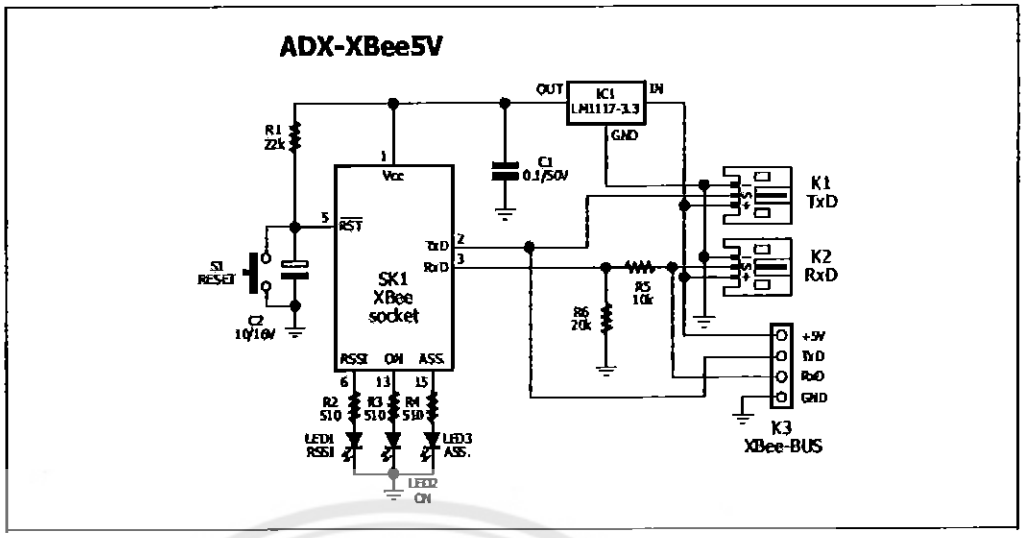


รูปที่ 2.2 แผนภาพวงจรการเชื่อมต่อ XBee-PRO กับคอมพิวเตอร์ [3]

2.3.2 การเชื่อมต่อ XBee-PRO กับไมโครคอนโทรลเลอร์

การเชื่อมต่อ XBee-PRO กับอุปกรณ์ภายนอก เช่น ไมโครคอนโทรลเลอร์ ได้เลือกใช้แผงวงจร ADX-XBee ซึ่งมีการต่อวงจรดังรูปที่ 2.3 และมีคุณสมบัติที่สำคัญดังนี้

1) มีตัวต่อตัวเมีย เพื่อติดตั้ง XBee-PRO
2) มีจุดต่อขาพอร์ต TxD และ RxD สำหรับเชื่อมต่ออุปกรณ์ภายนอก โดยจัดสัญญาณผ่านตัวต่อเจเอสที 3 (JST 3) ขาและไอดีซี (IDC) ทั้งแบบตัวผู้ และตัวเมีย ทำให้อุปกรณ์ภายนอกไม่ว่าจะเป็นแผงวงจรไมโครคอนโทรลเลอร์ หรือเอฟพีจีเอ สามารถติดต่อใช้งานกับ XBee-PRO ได้ง่ายและสะดวกขึ้น



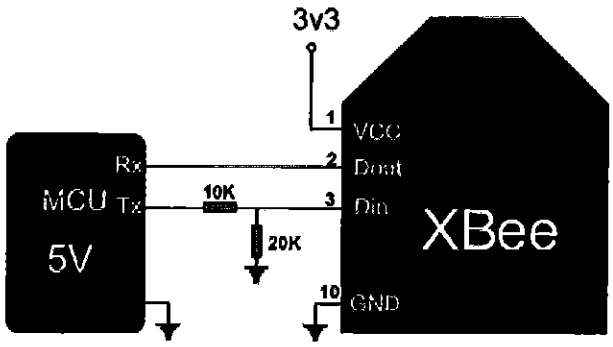
รูปที่ 2.3 แผนภาพวงจรการเชื่อมต่อโมดูล XBee-PRO กับอุปกรณ์ภายนอก [3]

3) ใช้ไฟเลี้ยง +5 V และขนาดของแผงวงจร 1.8 x 2 in
4) มีแอลอีดีแสดงความพร้อมในการติดต่อสื่อสารข้อมูลเอเอสเอส และแจ้งสภาวะการรับส่งสัญญาณ

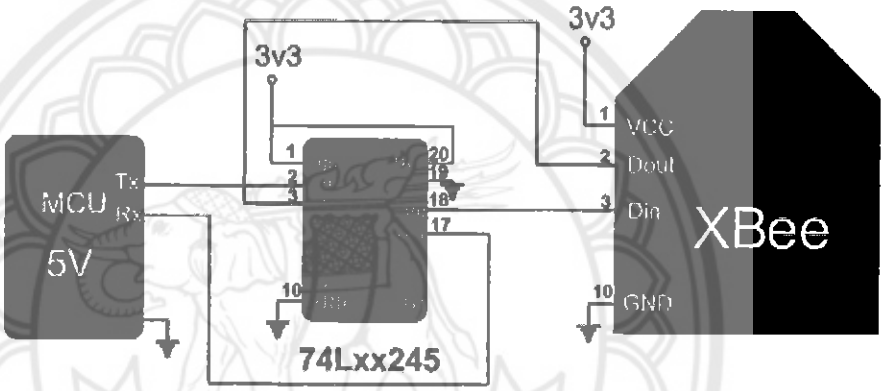
2.3.3 การจ่ายไฟเลี้ยงให้ XBee-PRO

เนื่องจาก XBee-PRO ต้องการไฟเลี้ยงในย่าน 2.8 – 3.4 V และขาสัญญาณทั้งหมดทำงานในระบบบัส 3 V ดังนั้นเมื่อนำไปเชื่อมต่อกับไมโครคอนโทรลเลอร์หรือชุดอุปกรณ์ภายนอกที่ใช้ระบบบัส 5 V จำเป็นต้องมีการลดแรงดันที่ขาพอร์ตลง ตัวอย่างวงจรการเชื่อมต่อเพื่อจ่ายไฟเลี้ยงให้ XBee-PRO [3] มี 2 ลักษณะคือ

1) การใช้ตัวแบ่งแรงดันแบบตัวต้านทาน โดยต่อตัวต้านทาน 2 ตัว ระหว่าง XBee-PRO กับไมโครคอนโทรลเลอร์ ตัวต้านทาน 2 ตัวนี้จะแบ่งแรงดันจาก 5 V เป็น 3.3 V โดยใช้หลักการแบ่งแรงดันดังรูปที่ 2.4 ซึ่งเป็นวิธีที่ง่ายและราคาถูก แต่ไม่ควรใช้กับอัตราบอด (Baud rate) สูงๆ
2) การคุมค่าแรงดัน โดยต่อไอซีคุมค่าแรงดัน (IC regulator) ระหว่าง XBee-PRO กับไมโครคอนโทรลเลอร์ ซึ่งไอซีคุมค่าแรงดันทำหน้าที่รักษาแรงดันเอาท์พุตจากแหล่งจ่ายของไมโครคอนโทรลเลอร์ให้คงที่เท่ากับ 3.3 V ดังรูปที่ 2.5 ซึ่งเป็นวิธีที่ปลอดภัย มีเสถียรภาพกว่าแบบแรก และสามารถใช้กับอัตราบอดสูงๆ ได้ดี



รูปที่ 2.4 การจ่ายไฟเลี้ยงให้ XBee-PRO ผ่านตัวแบ่งแรงดันแบบตัวต้านทาน [2]



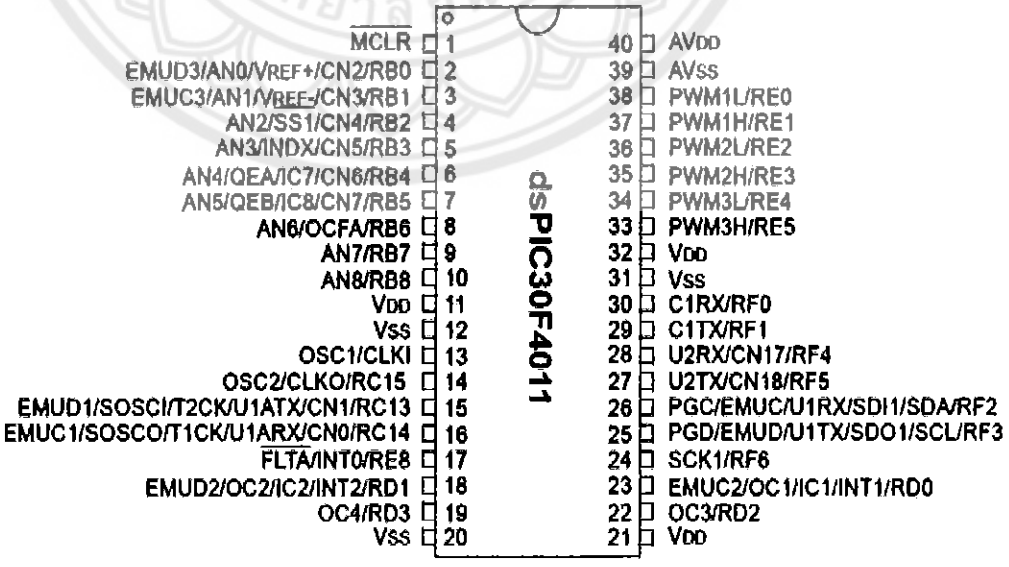
รูปที่ 2.5 การจ่ายไฟเลี้ยงให้ XBee-PRO ผ่านไอซีควบคุมค่าแรงดัน [2]

2.4 ไมโครคอนโทรลเลอร์รุ่น PIC30F4011

dsPIC30F4011 เป็นแผงวงจรไมโครคอนโทรลเลอร์ในตระกูล dsPIC30F ซึ่งใช้ในการประมวลผลข้อมูลแบบ 16 บิต จากค่ายไมโครชิป (Microchip) ซึ่งมีความสามารถในการประมวลผลข้อมูลสัญญาณแบบดิจิตอลทำให้เหมาะอย่างยิ่งสำหรับนำไปประยุกต์ใช้ในงานควบคุมต่างๆ โครงสร้างภายในเป็นการผสมผสานระหว่างไมโครคอนโทรลเลอร์และวงจรประมวลผลสัญญาณดิจิตอล (Digital Signal Processing: DSP) หรืออาจเรียกไมโครคอนโทรลเลอร์ตระกูล dsPIC30F4011 ว่าเป็นดีเอสซี (Digital Signal Controller: DSC) โครงสร้างของ dsPIC30F4011 ได้รับการออกแบบให้มีขนาดเล็ก เหมาะต้องการนำไปประยุกต์ใช้งาน ภายในแผงวงจรมีส่วนที่จำเป็นต้องการใช้งานและสะดวกต่อการพัฒนาโปรแกรม มีความยืดหยุ่น สามารถปรับเปลี่ยนสัญญาณอินพุตและเอาท์พุต เพื่อนำไปประยุกต์ใช้งานในลักษณะต่างๆตามความต้องการ

แผงวงจร dsPIC30F4011 มีคุณสมบัติที่เหมาะสมกับการนำมาใช้ในโครงงาน [4] ดังนี้

1) มีตัวแปลงสัญญาณแอนะล็อกเป็นดิจิตอล (Analog-to-Digital Converter: ADC) 10 บิต/500 Ksps จำนวน 6 ช่อง (dsPIC30F2010) หรือ 9 ช่อง (dsPIC30F4011) และมียูอาร์ต (Universal Asynchronus Receiver/Transmitte: UART) จำนวน 1 ช่อง (dsPIC30F2010) หรือ 2 ช่อง (dsPIC30F4011)
2) ใช้คริสตัล (Crystal) ความถี่ 7.3728 MHz สามารถใช้เฟสล็อกลูป (Phase Locked Loop: PLL) คูณความถี่เพื่อสั่งการความถี่ 29.4912 MHz ได้
3) มีพอร์ตสื่อสารอนุกรมยูอาร์ตแบบอาร์เอสสองสามสอง (RS232) จำนวน 1 ช่อง สำหรับ dsPIC30F2010 และ 2 ช่อง สำหรับ dsPIC30F4011 พร้อมจัมเปอร์ (Jumper) สำหรับเลือกใช้งานยูอาร์ตหรือจีพีไอโอ (General Purpose Input/Output: GPIO) ได้ตามต้องการโดยใช้ขั้วต่อยูอาร์ต แบบ CPA-4 ขา
4) มีขั้วไอซีเอสพี (In circuit Serial Programming: ICSP) มาตรฐานไอซีดีทู (In Circuit Debugging: ICD2) แบบ RJ11 สำหรับใช้ร่วมกับชุดพัฒนาโปรแกรมและชุดตรวจสอบโปรแกรมที่รองรับการทำงานตามมาตรฐานไอซีดีทูของไมโครชิป
5) มีสวิตช์รีเซ็ตสำหรับสั่งปรับตั้งการทำงานของไมโครคอนโทรลเลอร์ใหม่
6) Power AC/DC Input พร้อมตัวคุมค่าแรงดันแบบสวิตชิงหมายเลข LM2575 ขนาด 5V/1A ลดปัญหาความร้อนจากวงจรตัวคุมค่าแรงดันและหลอดแอลอีดีแสดงสถานะแหล่งจ่ายไฟ



รูปที่ 2.6 รูปแบบการจัดขาสัญญาณของ dsPIC30F4011 [4]

2.5 ตัวรับรู้ความดันส่วนต่าง

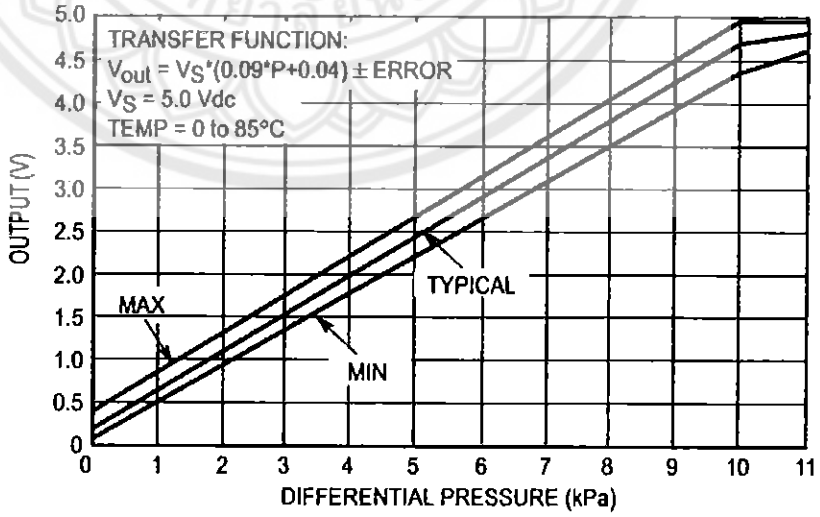
โครงงานนี้ได้นำตัวรับรู้จำแนกตามข้อมูลหรือวัตถุประสงค์ในการวัดเข้ามาใช้งาน เนื่องจากเป็นการวัดระดับน้ำในระดับต่างๆเท่านั้น โดยการแบ่งชนิดของตัวรับรู้อาศัยหลักเกณฑ์ต่างๆ [5] ดังต่อไปนี้

1) การจำแนกตามความต้องการพลังงาน
   ก) แบบแอกทีฟ (Active sensors) เป็นทรานสดิวเซอร์ที่สามารถปล่อยพลังงานเองได้ เช่น เทอร์โมคัปเปิ้ล (Thermocouple) เพียซโซ (Piezo) และเซลล์แสงอาทิตย์ (Solar Cell) เป็นต้น อุปกรณ์เหล่านี้ไม่ต้องมีแหล่งจ่ายกำลังจากภายนอกที่สอดคล้องกับปริมาณที่วัดโดยให้ก็สามารถให้สัญญาณแรงดันหรือกระแส
   ข) แบบพาสซีฟ (Passive sensors) ต้องใช้แหล่งจ่ายจากภายนอกเพื่อให้ตรวจรู้ได้ เช่น ตัวรับรู้ที่ใช้หลักการเปลี่ยนค่าความต้านทาน ค่าความจุ ค่าความเหนี่ยวนำ เป็นต้น
2) การจำแนกตามลักษณะกลไกในการทำงาน
   ก) การเปลี่ยนแปลงค่าความจุ (Variable capacitance transducer)
   ข) การเปลี่ยนแปลงค่าความเหนี่ยวนำ (Variable inductance transducer)
   ค) การเปลี่ยนแปลงค่าความต้านทาน (Variable resistance transducer)
3) การจำแนกตามชนิดของการเปลี่ยนแปลงพลังงาน
   ก) เปลี่ยนพลังงานกลเป็นพลังงานไฟฟ้า
   ข) เปลี่ยนพลังงานไฟฟ้าเป็นพลังงานกล
   ค) เปลี่ยนพลังงานแสงเป็นพลังงานไฟฟ้า
   ง) เปลี่ยนพลังงานความร้อนเป็นพลังงานไฟฟ้า
4) การจำแนกตามชนิดของสัญญาณที่ใช้
   ก) แบบแอนะล็อก ซึ่งให้สัญญาณเป็นแบบต่อเนื่อง
   ข) แบบไบนารี ซึ่งให้สัญญาณแบบเปิด-ปิด (ON-OFF)
   ค) แบบดิจิตอล ซึ่งให้สัญญาณเป็นแบบดิจิตอล
5) การจำแนกตามตำแหน่งที่ใช้ในระบบ
   ก) ทรานสดิวเซอร์ด้านเข้า (Input transducer) อยู่ทางด้านเข้าของระบบเครื่องมือ เช่น ไมโครโฟน เป็นต้น
   ข) ทรานสดิวเซอร์ด้านออก (Output transducer) เช่น ลำโพงของระบบเครื่องขยายเสียง
6) การจำแนกตามข้อมูลหรือวัตถุประสงค์ในการวัด ได้แก่ ทรานสดิวเซอร์ที่ใช้วัดการเคลื่อนที่ อุณหภูมิ ความดัน อัตราการไหล และตำแหน่ง เป็นต้น

ในโครงงานนี้ได้เลือกใช้ตัวรับรู้ความดันส่วนต่างรุ่น MPX5010DP มาใช้วัดระดับน้ำ เนื่องจากตัวรับรู้ความดันส่วนต่างมีเสถียรภาพและความแม่นยำสูงในการวัดระดับน้ำ โดยใช้หลักการที่ความลึกของระดับน้ำเดียวกันมีค่าความดันเท่ากันทุกทิศทางและตั้งฉากกับผิวภาชนะที่บรรจุอยู่ รูปร่างของภาชนะที่บรรจุน้ำจึงไม่ส่งผลต่อค่าความดันของน้ำ ทำให้สามารถวัดความดันน้ำได้ทั้งในน้ำนิ่งหรือน้ำไหล อย่างไรก็ตามความพลิ้วบริเวณผิวน้ำอาจส่งผลต่อความคลาดเคลื่อนของค่าความดันที่วัดโดยเฉพาะในขณะสูบน้ำเข้าในถังน้ำ คุณสมบัติหลักของ MPX5010DP มีดังนี้

1) ใช้หลักการวัดความดันส่วนต่าง (Differential pressure)
2) จำนวนขาทั้งหมด 6 ขา
3) ใช้งานในช่วงความดัน 0 – 10 kPa
4) ช่วงแรงดันไฟฟ้า 4.75 – 5.25 V

ตัวรับรู้ความดันส่วนต่าง รุ่น MPX5010DP เปรียบเทียบค่าความดันของน้ำกับความดันบรรยากาศภายในตัวรับรู้แล้วให้ค่าออกมาเป็นแรงดันไฟฟ้าตามกราฟคุณลักษณะดังรูปที่ 2.7 โดยแกนตั้งเป็นแรงดันเอาท์พุตของตัวรับรู้ ส่วนแกนนอนเป็นผลต่างระหว่างค่าความดันของน้ำกับความดันบรรยากาศ โดยผลต่างมีค่าอยู่ในช่วง 10 kPA ซึ่งสัมพันธ์กับความสูงของระดับน้ำ 1 m ค่าต่ำสุดของกราฟอยู่ในช่วง 0 – 0.2 V อันเกิดจากความคลาดเคลื่อนในการทดลอง ในขณะที่ค่าสูงสุดอยู่ในช่วง 4.75 – 5 V อย่างไรก็ตามเนื่องจากตัวรับรู้ความดันส่วนต่างรุ่น MPX5010DP ได้รับแรงดันไฟเลี้ยง +5 V ดังนั้นจากกราฟคุณลักษณะจะเห็นว่า MPX5010DP สามารถให้แรงดันเอาท์พุตสูงสุดไม่เกินค่าไฟเลี้ยงนั่นเอง [6]



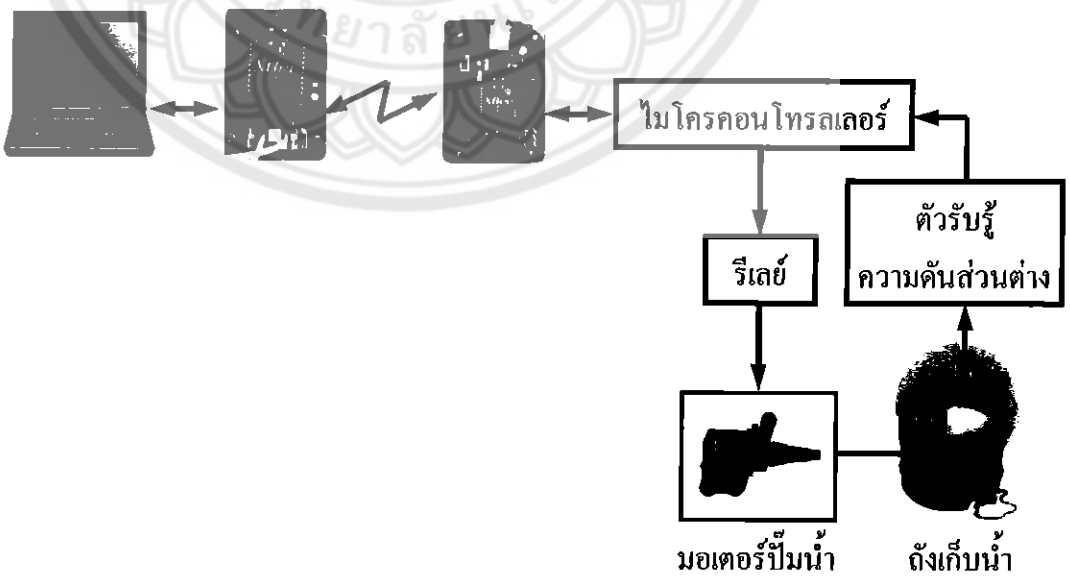
รูปที่ 2.7 กราฟคุณลักษณะของตัวรับรู้ความดันส่วนต่างรุ่น MPX5010DP [6]

บทที่ 3
การออกแบบและสร้างระบบควบคุมมอเตอร์ปั๊มน้ำแบบไร้สาย

หลังจากศึกษาหลักการทำงานของส่วนโมดูลสื่อสารไร้สายย่าน 2.4 GHzรุ่น XBee-PRO และไมโครคอนโทรลเลอร์รวมทั้งวิธีใช้ตัวรับรู้ความดันส่วนต่างในการวัดระดับน้ำแล้วจึงเริ่มการกำหนดขั้นตอนการดำเนินงานด้วยการเขียนผังงานเพื่อออกแบบการทำงานของระบบทั้งหมด หลังจากนั้นจึงใช้โปรแกรม X-CTU เพื่อกำหนดค่าให้ XBee-PRO ทั้ง 2 ตัวทำหน้าที่รับและส่งข้อมูลถึงกัน และเขียนชุดคำสั่งด้วยโปรแกรมวิชวลเบสิก 2010 เพื่อสร้างส่วนต่อประสานกราฟิกกับผู้ใช้สำหรับควบคุมการรับและส่งข้อมูลกับไมโครคอนโทรลเลอร์ ในส่วนของไมโครคอนโทรลเลอร์มีการเขียนโปรแกรมด้วยภาษาซีเพื่อรับและส่งข้อมูลกับคอมพิวเตอร์ ประมวลผลและควบคุมการทำงานของรีเลย์ รวมถึงการออกแบบจำลองของระบบมอเตอร์ปั๊มน้ำโดยใช้ตัวรับรู้ความดันส่วนต่างเพื่อวัดระดับน้ำ

3.1 การออกแบบขั้นตอนการทำงานของระบบ

ขั้นตอนการทำงานของระบบควบคุมการเปิดและปิดมอเตอร์ปั๊มน้ำด้วยวิธีการควบคุมทางไกลแบบไร้สายแสดงด้วยแผนภาพในรูปที่ 3.1



รูปที่ 3.1 ขั้นตอนการทำงานระบบเปิดและปิดปั๊มน้ำโดยการควบคุมทางไกลแบบไร้สาย

ผู้ใช้สามารถเชื่อมต่อให้ระบบเริ่มทำงานด้วยหน้าต่างส่วนต่อประสานกราฟิกกับผู้ใช้ที่คอมพิวเตอร์ซึ่งเชื่อมต่อกับ XBee-PRO ตัวที่ 1 ข้อมูลถูกแปลงเป็นสัญญาณและส่งจาก XBee-PRO ตัวที่ 1 ไปยัง XBee-PRO ตัวที่ 2 ซึ่งเชื่อมต่อและส่งข้อมูลให้กับไมโครคอนโทรลเลอร์เพื่อประมวลผลเมื่อเข้าเงื่อนไขการเปิดมอเตอร์ไมโครคอนโทรลเลอร์จะสั่งให้รีเลย์ทำงานโดยต่อหน้าสัมผัสเพื่อเริ่มเดินเครื่องมอเตอร์ปั๊มน้ำ น้ำถูกสูบเข้าไปยังถังเก็บน้ำซึ่งมีตัวรับรู้ความดันส่วนต่างคอยตรวจวัดระดับน้ำโดยเปรียบเทียบความดันที่ได้จากภายในสายยางกับความดันบรรยากาศแล้วแปลงค่าออกมาเป็นค่าแรงดันไฟฟ้า เพื่อส่งให้ไมโครคอนโทรลเลอร์ประมวลผลและส่งข้อมูลกลับไปแสดงผลยังคอมพิวเตอร์โดยผ่าน XBee-PRO ตัวที่ 2 และตัวที่ 1 ตามลำดับหลังจากที่ระดับน้ำเพิ่มสูงจนถึงขีดจำกัดบน (Upper limit) ที่ผู้ใช้กำหนด ไมโครคอนโทรลเลอร์จะสั่งให้รีเลย์เปิดวงจรเพื่อให้มอเตอร์หยุดปั๊มน้ำ รวมทั้งส่งข้อมูลไปแสดงผลที่คอมพิวเตอร์ และจะสั่งให้รีเลย์ต่อวงจรอีกครั้งเมื่อระดับน้ำลดต่ำกว่าขีดจำกัดล่าง (Lower limit) ที่ผู้ใช้กำหนด

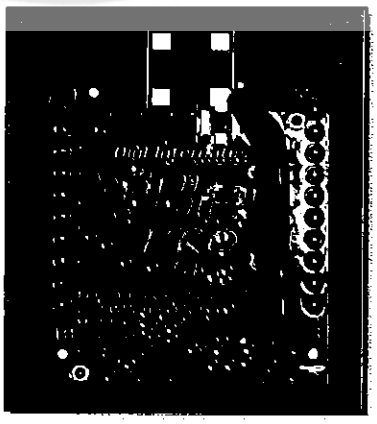
3.2 การออกแบบระบบควบคุมการทำงาน

ระบบควบคุมการทำงานประกอบด้วย 3 ส่วนหลักคือโมดูลสื่อสารไร้สายย่าน 2.4 GHz รุ่น XBee-PRO คอมพิวเตอร์ และไมโครคอนโทรลเลอร์

3.2.1 การตั้งค่าโมดูลสื่อสารไร้สายย่าน 2.4 GHz รุ่น XBee-PRO

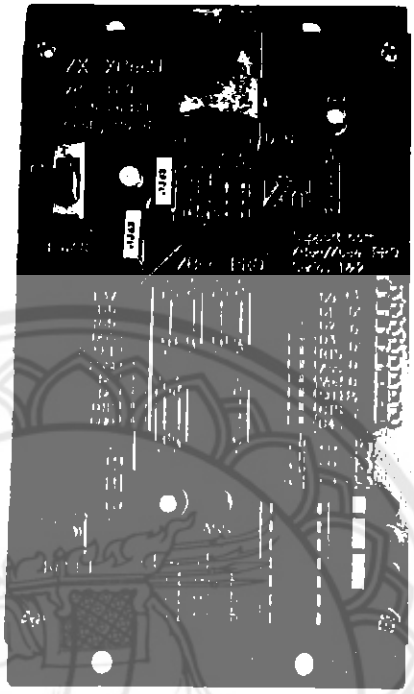
การทำงานของ XBee-PRO จำเป็นต้องใช้ส่วนประกอบสำคัญดังต่อไปนี้

1) โมดูลสื่อสารไร้สายย่าน 2.4 GHz รุ่น XBee-PRO (รูปที่ 3.2) เป็นตัวรับและตัวส่งข้อมูลไปยังคอมพิวเตอร์และแผงวงจรไมโครคอนโทรลเลอร์



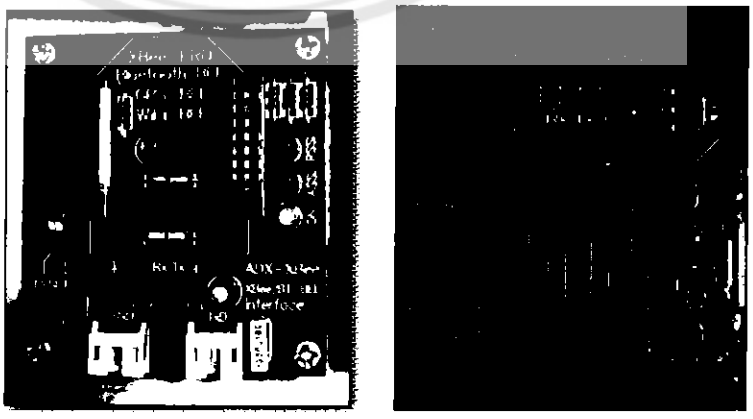
รูปที่ 3.2 ตัวรับและส่งสัญญาณของ XBee-PRO

2) แผงวงจร ZX-XBee (รูปที่ 3.3) ใช้ในการเชื่อมต่อ XBee-PRO กับคอมพิวเตอร์ เพื่อตั้งค่าพารามิเตอร์ต่างๆของ XBee-PRO



รูปที่ 3.3 แผงวงจร ZX-XBee

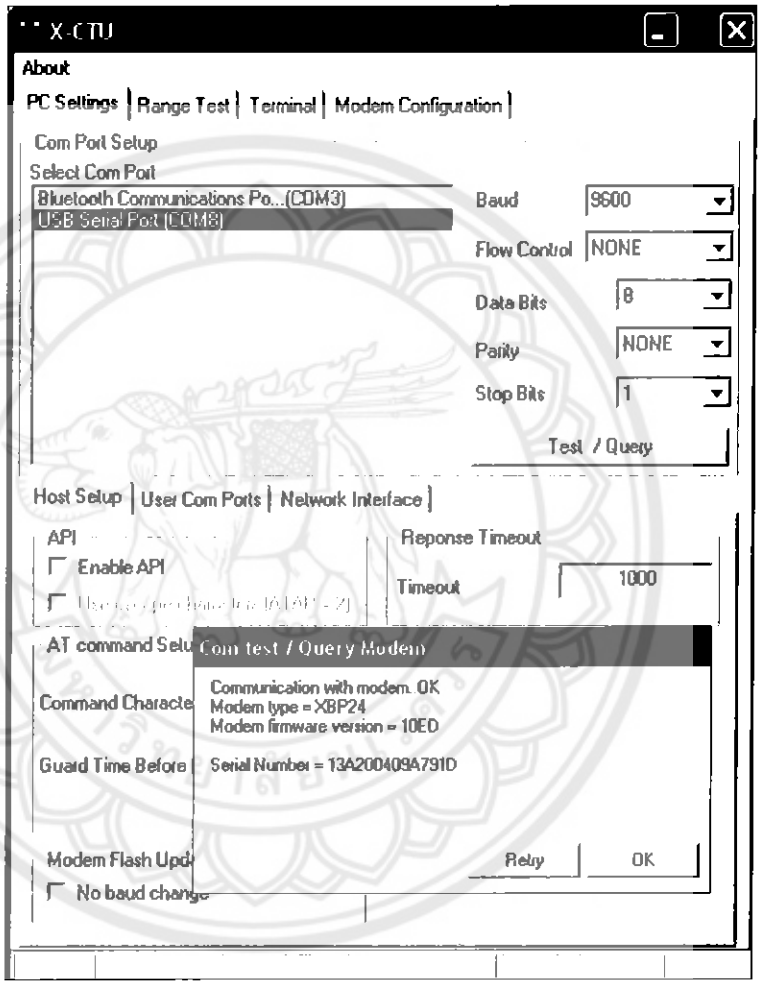
3) แผงวงจร ADX-XBee (รูปที่ 3.4) ใช้เชื่อมต่อกับ XBee-PRO เข้ากับอุปกรณ์ภายนอก เช่น ไมโครคอนโทรลเลอร์ เพื่อช่วยให้ไมโครคอนโทรลเลอร์สามารถสื่อสารกับคอมพิวเตอร์โดยรับและส่งข้อมูลแบบไร้สายได้



(ก) ด้านหน้า (ข) ด้านหลัง

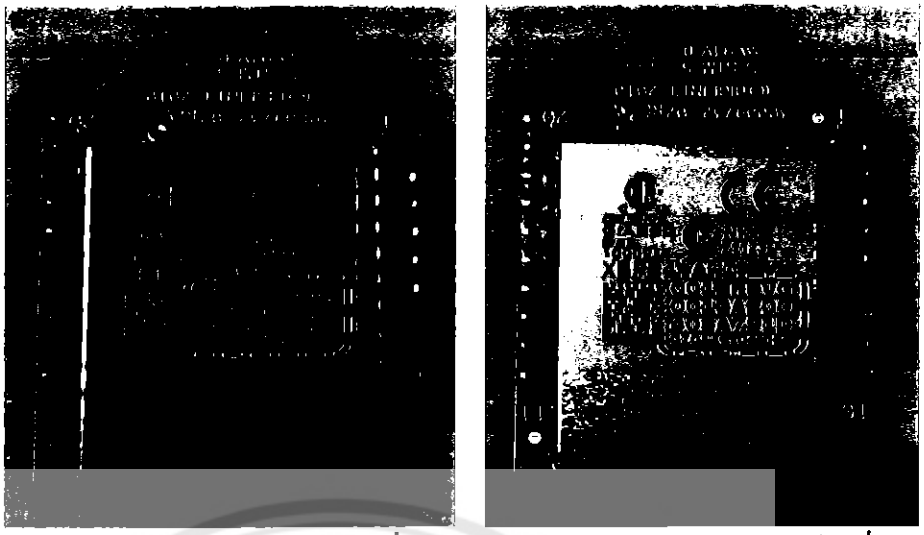
รูปที่ 3.4 แผงวงจร ADX-XBee

การกำหนดค่า XBee-PRO เพื่อให้สามารถใช้งานในการรับและส่งข้อมูลได้ มีวิธีตั้งค่าพารามิเตอร์ต่างๆโดยเริ่มต้นจากการเปิดโปรแกรม X-CTU จากนั้นเลือกคอมพอร์ตที่ต่อกับแผงวงจรที่เชื่อม XBee-PRO กับคอมพิวเตอร์ แล้วกด Test เพื่อตรวจสอบการเชื่อมต่อคอมพอร์ตกับ XBee-PRO ถ้าปรากฏหน้าต่างดังรูปที่ 3.5 แสดงว่าคอมพอร์ตกับ XBee-PRO สื่อสารกันได้แล้วจึงกด OK



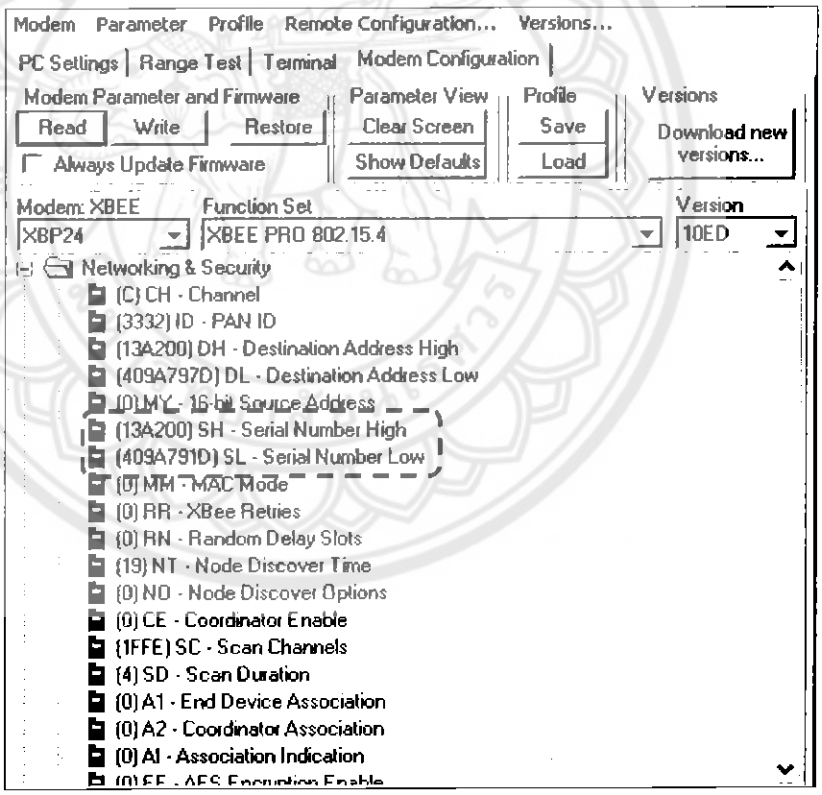
รูปที่ 3.5 หน้าต่างเริ่มต้นของโปรแกรม X-CTU

หลังจากที่กด OK แล้วกดแถบ Modem Configuration แล้วกด Read โปรแกรม X-CTU จะทำการยกระดับข้อมูล โดยการตั้งค่าเป้าหมายในการส่งข้อมูลซึ่งตั้งเป็นหมายเลขประจำอุปกรณ์ (Serial Number) ของ XBee-PRO เป้าหมายดังแสดงในรูปที่ 3.6 และ 3.7 หลังจากการยกระดับเสร็จสิ้นแล้วจึงกด Write



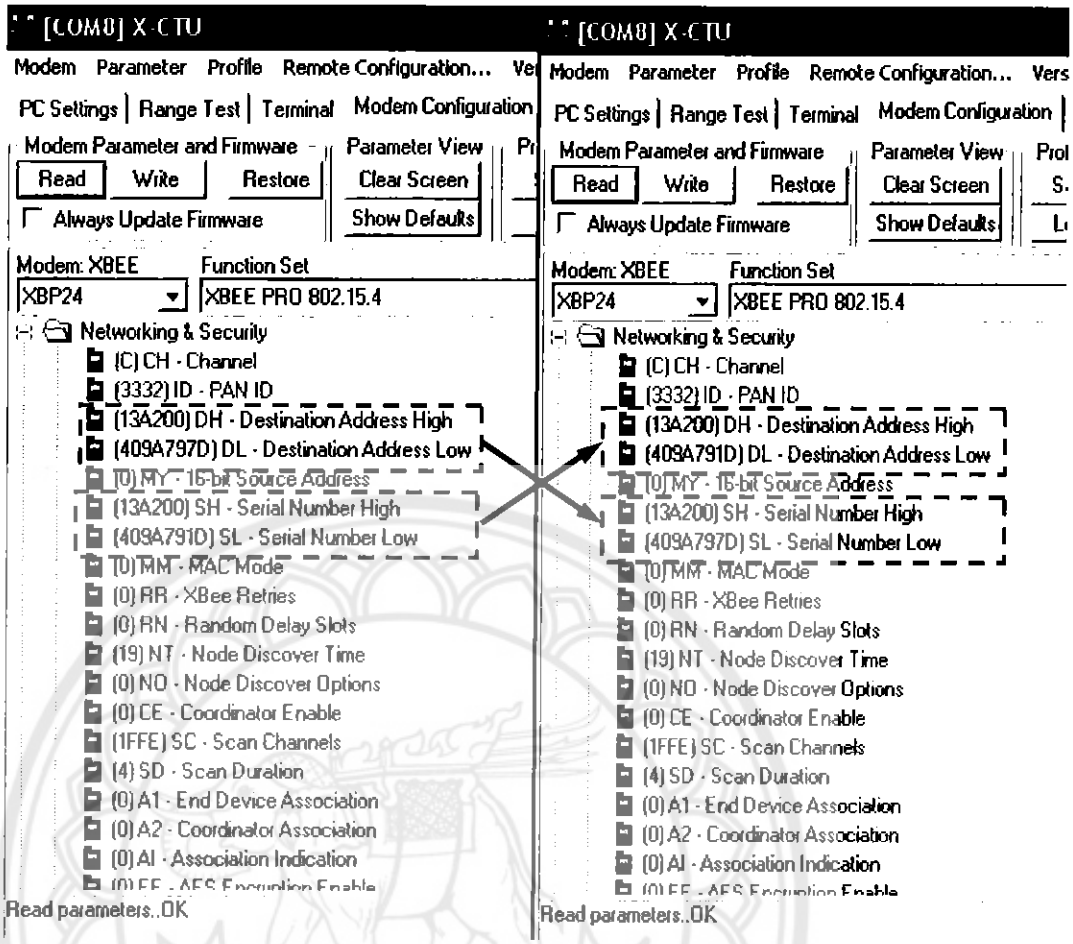
(ก) หมายเลขประจำอุปกรณ์ตัวที่ 1 (ข) หมายเลขประจำอุปกรณ์ตัวที่ 2

รูปที่ 3.6 หมายเลขประจำอุปกรณ์ของ XBee-PRO



รูปที่ 3.7 การลงหมายเลขประจำอุปกรณ์ของ XBee-PRO บน X-CTU

หลังจากนั้นให้ทำการกรอกที่อยู่ของเป้าหมาย (Destination Address) ดังรูปที่ 3.8 เพื่อให้ XBee-PRO ทั้ง 2 ตัวสามารถรับและส่งข้อมูลถึงกันได้



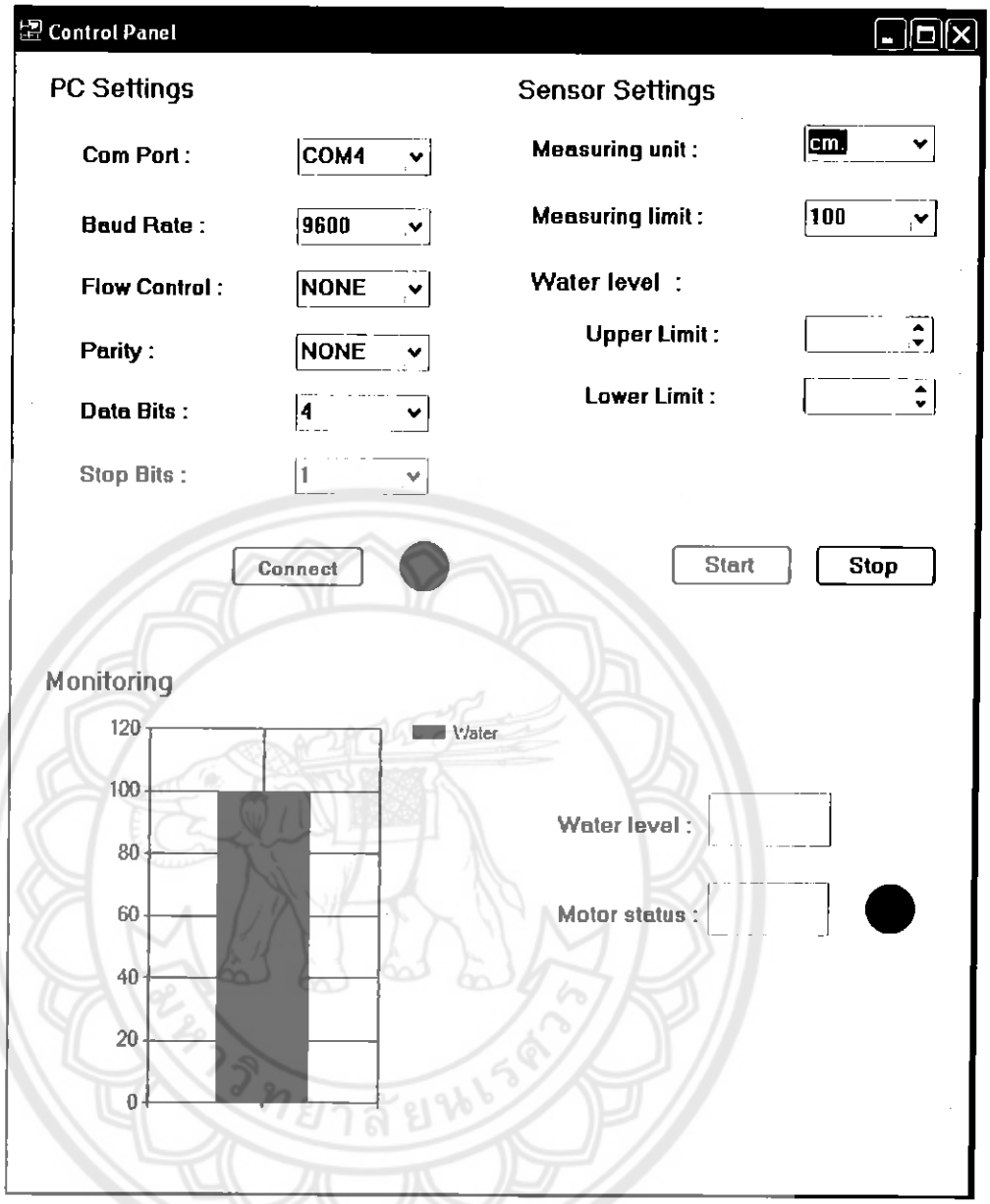
รูปที่ 3.8 การตั้งค่าที่อยู่ของเป้าหมายที่จะส่งข้อมูลถึง

3.2.2 การตั้งค่าและควบคุมการทำงานด้วยคอมพิวเตอร์

ในส่วนของคอมพิวเตอร์มีหน้าต่างของส่วนต่อประสานกราฟิกกับผู้ใช้ ซึ่งมีลักษณะดังรูปที่ 3.9 เพื่อให้ผู้ใช้สามารถใช้งานได้สะดวก โดยบนหน้าต่างนี้ช่วยให้ผู้ใช้สามรถตั้งค่าต่างๆในการเชื่อมต่อ รวมทั้งแสดงผลระดับน้ำและสถานะการทำงานของมอเตอร์ปั๊มน้ำ

หน้าต่างส่วนต่อประสานกราฟิกกับผู้ใช้ประกอบด้วย 3 ส่วนดังนี้

ส่วนที่ 1 คือ PC Settings มีไว้เพื่อกำหนดค่าพารามิเตอร์ของมาตรฐาน RS-232 โดยผู้ใช้ต้องกำหนดค่าพารามิเตอร์อยู่ 6 ตัว คือ คอมพอร์ต (Com Port) พาริตี้ (Parity) อัตราบอด (BaudRate) บิตข้อมูล (Data Bits) บิตจบ (Stop Bits) และการควบคุมการไหลของข้อมูล (Flow control) หลังจากนั้นกดปุ่ม Connect เพื่อเปิดพอร์ตอนุกรม

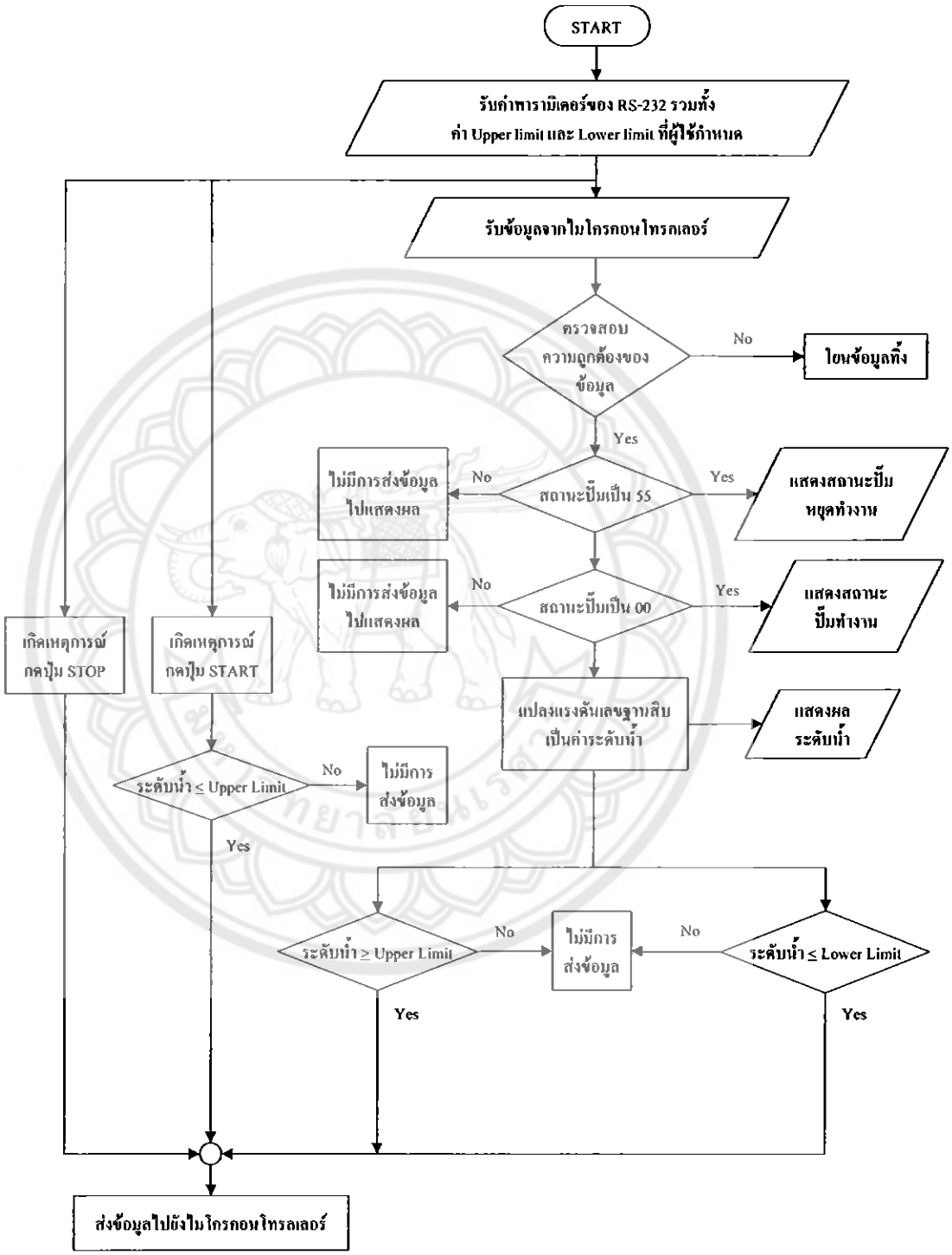


รูปที่ 3.9 หน้าต่างของส่วนต่อประสานกราฟิกกับผู้ใช้

ส่วนที่ 2 คือ Sensor Settings มีไว้เพื่อให้ผู้ใช้กำหนดตั้งค่าการใช้งานดังต่อไปนี้การเลือกหน่วยของการวัดระดับน้ำเป็นเซนติเมตร (cm) หรือเมตร (m) เลือกระดับน้ำสูงสุดที่ตัวรับรู้จะวัดได้ในแต่ละรุ่น โดยที่การควบคุมทำงานของมอเตอร์ปั๊มน้ำจะเป็นแบบระบบอัตโนมัติ (Automatic system) ผู้ใช้จึงต้องกำหนดค่าขีดจำกัดบน (Upper Limit) และขีดจำกัดล่าง (Lower Limit) อย่างไรก็ตามผู้ใช้สามารถกดปุ่มเพื่อหยุดการทำงานของมอเตอร์ได้ทุกขณะตามต้องการ

ส่วนที่ 3 คือ Monitoring เป็นหน้าต่างแสดงผลระดับน้ำ และแสดงสถานะการทำงานของมอเตอร์ปั๊มน้ำ โดยที่ไฟสีแดงหมายถึงมอเตอร์ปั๊มน้ำไม่ทำงาน ไฟสีเขียวหมายถึงมอเตอร์ปั๊มน้ำทำงาน

ลำดับการทำงานของส่วนควบคุมที่กำหนดด้วยส่วนต่อประสานกราฟิกกับผู้ใช้แสดงใน รูปที่ 3.10



รูปที่ 3.10 ลำดับการทำงานของส่วนควบคุมที่กำหนดด้วยส่วนต่อประสานกราฟิกกับผู้ใช้

จากแผงผังลำดับการทำงานของส่วนควบคุมในรูปที่ 3.10 ดังนี้คือ เริ่มต้นให้ผู้ใช้เลือกค่าพารามิเตอร์ของ RS-232 แล้วกดปุ่ม Connect เพื่อเปิดพอร์ตอนุกรมให้คอมพิวเตอร์สามารถส่งข้อมูลทั้ง 10 ตัวออกไปทาง XBee-PRO ทั้งสองตัว จากนั้นข้อมูลถูกส่งออกไปยังแผงวงจรไมโครคอนโทรลเลอร์ โดยข้อมูล 10 ตัวที่ส่งไปประกอบไปด้วย

ข้อมูลตัวที่ 1 แบ่งออกเป็น 2 กรณีคือ ถ้าตัว "P" แสดงว่าเป็นข้อมูลที่ส่งมาจากคอมพิวเตอร์ ถ้าเป็นตัว "D" แสดงว่าเป็นข้อมูลที่ส่งมาจากไมโครคอนโทรลเลอร์

ข้อมูลตัวที่ 2 มี 2 กรณีคือ ถ้าเป็นตัว "A" แสดงว่าเป็นการสั่งไปให้ควบคุมการทำงานของมอเตอร์ปั๊มน้ำแบบอัตโนมัติ ถ้าเป็นตัว "B" แสดงว่าเป็นการสั่งไปควบคุมการทำงานของมอเตอร์ปั๊มน้ำแบบควบคุมด้วยมือ

ข้อมูลตัวที่ 3 และ 4 แบ่งเป็น 2 กรณี ถ้าข้อมูลตัวที่ 3 เป็น "0" และข้อมูลตัวที่ 4 เป็น "0" แสดงว่ามอเตอร์ปั๊มน้ำกำลังทำงานอยู่ ถ้าข้อมูลตัวที่ 3 เป็น "5" และข้อมูลตัวที่ 4 เป็น "5" แสดงว่ามอเตอร์ปั๊มน้ำไม่ทำงาน

ข้อมูลตัวที่ 5-9 เป็นค่าแรงดันไฟฟ้าที่แปลงมาเป็นเลขฐานสิบ โดยมีค่าอยู่ในช่วง 0-1024

ข้อมูลตัวที่ 10 แบ่งเป็น 2 กรณีคือ ถ้าตัว "P" แสดงว่าเป็นข้อมูลที่ส่งมาจากคอมพิวเตอร์ ถ้าเป็นตัว "D" เป็นข้อมูลที่ส่งมาจากไมโครคอนโทรลเลอร์

เมื่อไมโครคอนโทรลเลอร์ทำการประมวลผลเสร็จสิ้นแล้วจะส่งข้อมูลกลับมา 10 ตัวที่คอมพิวเตอร์ ส่วนต่อประสานกราฟิกกับผู้ใช้ทำการตรวจสอบข้อมูลที่ได้รับ หลังจากเสร็จสิ้นขั้นตอนการตรวจสอบข้อมูลและแปลงข้อมูลจากค่าแรงดันน้ำในเลขฐานสิบเป็นค่าความสูงของระดับน้ำแล้ว จึงนำข้อมูลนี้มาแสดงผลบนหน้าจอ พร้อมทั้งสถานะการทำงานของปั๊ม และนำค่าระดับน้ำนั้นมาตรวจสอบตาม 2 เงื่อนไขดังนี้

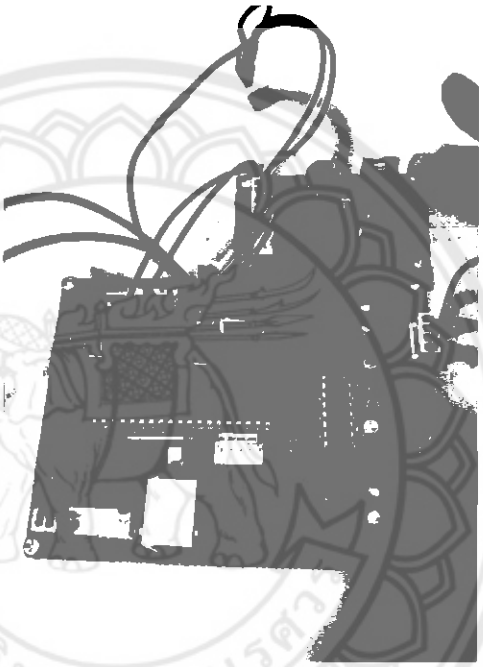
กรณีที่ 1 ถ้าระดับน้ำต่ำสุดที่ผู้ใช้ตั้งไว้มีค่ามากกว่าหรือเท่ากับค่าระดับน้ำที่แสดงทางจอคอมพิวเตอร์ จะมีการส่งข้อมูลไปให้ไมโครคอนโทรลเลอร์ แล้วสั่งให้รีเลย์ควบคุมมอเตอร์ปั๊มน้ำให้ทำงาน

กรณีที่ 2 ถ้าค่าระดับน้ำสูงสุดที่ผู้ใช้ตั้งค่าไว้มีค่าน้อยกว่าหรือเท่ากับค่าระดับน้ำที่แสดงทางจอคอมพิวเตอร์ จะมีการส่งข้อมูลไปให้ไมโครคอนโทรลเลอร์ แล้วสั่งให้รีเลย์ควบคุมมอเตอร์ปั๊มน้ำให้หยุดทำงาน

3.2.3 การประมวลผลและควบคุมรีเลย์ด้วยไมโครคอนโทรลเลอร์

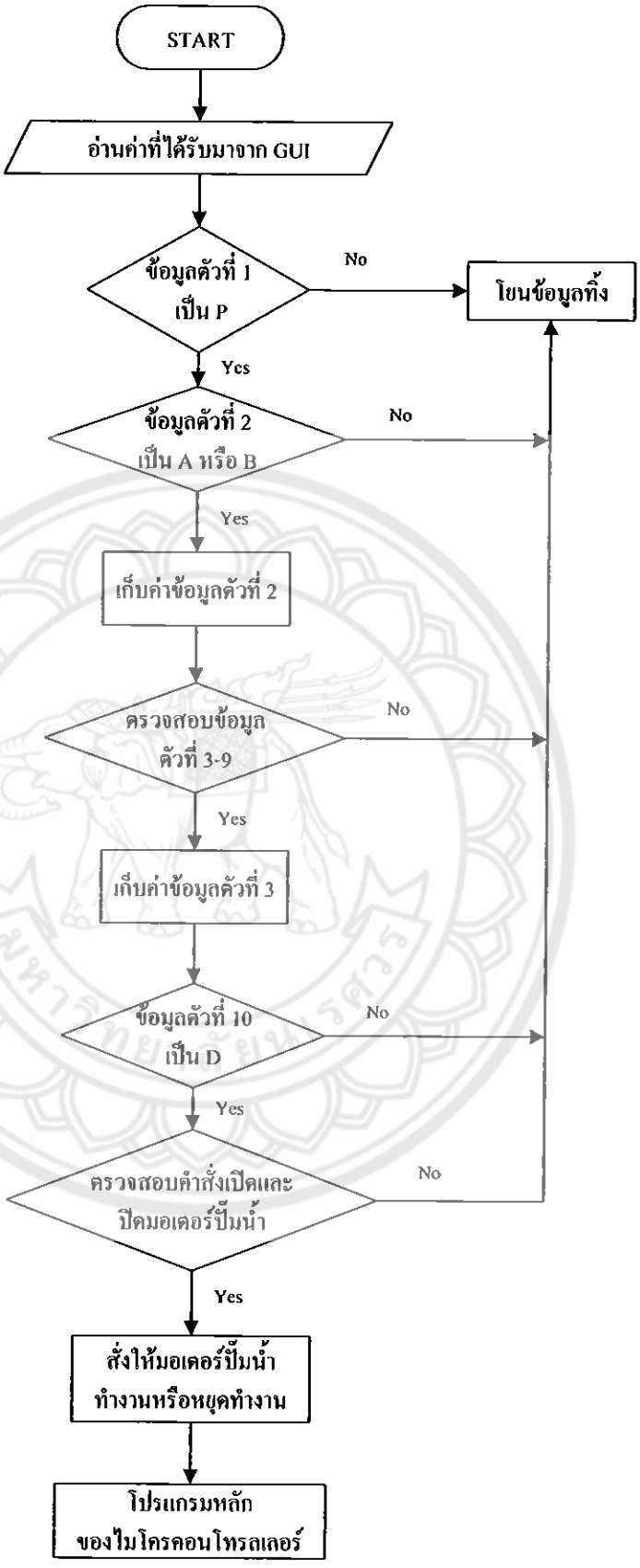
1) แผงวงจรไมโครคอนโทรลเลอร์ dsPIC30F4011

ในโครงงานนี้ใช้ไมโครคอนโทรลเลอร์ควบคุมการทำงานของรีเลย์โดยรับค่าแรงดันไฟฟ้าจากตัวรับรู้ความดันส่วนต่างซึ่งสอดคล้องกับระดับน้ำขณะใดขณะหนึ่งมาประมวลผล รวมทั้งการรับและส่งค่าข้อมูลกลับไปแสดงผลทางจอคอมพิวเตอร์ แผงวงจรไมโครคอนโทรลเลอร์ dsPIC30F4011 ที่ใช้ในโครงงานแสดงดังรูปที่ 3.11



รูปที่ 3.11 แผงวงจรไมโครคอนโทรลเลอร์ dsPIC30F4011

ในที่นี้ได้กำหนดให้ขาที่ 2 ทำหน้าที่รับข้อมูลที่อ่านค่าได้จากตัวรับรู้ความดันส่วนต่าง รุ่น MPX5010DP โดยรับเข้ามาเป็นสัญญาณแอนะล็อก ขาที่ 19 ทำหน้าที่ส่งสัญญาณลอจิก 1 หรือ ลอจิก 0 ไปควบคุมการทำงานของรีเลย์ ส่วนขาที่ 27 ทำหน้าที่ส่งคำสั่งที่ได้จากการประมวลผลทั้งหมดออกไปยัง XBee-PRO ตัวที่ 2 แล้วส่งสัญญาณข้อมูลไปยัง XBee-PRO ตัวที่ 1 เพื่อนำไปแสดงผลทางจอคอมพิวเตอร์ และขาที่ 28 ทำหน้าที่รับคำสั่งที่ส่งจาก XBee-PRO ตัวที่ 1 ไปยัง XBee-PRO ตัวที่ 2 เพื่อให้ไมโครคอนโทรลเลอร์ประมวลผลตามคำสั่งที่เขียนไว้



รูปที่ 3.12 แผนผังการตรวจสอบข้อมูลที่รับมาจาก GUI

จากรูปที่ 3.12 แสดงการอธิบายหลักการทำงานการตรวจสอบข้อมูล เมื่อผู้ใช้สั่งการไมโครคอนโทรลเลอร์จะรับค่าคำสั่งนั้นมาตรวจสอบข้อมูลทีละตัวโดยแบ่งออกเป็น 10 กรณีดังนี้

กรณีที่ 1 ตรวจสอบข้อมูลว่าเป็นตัว P หรือไม่ ถ้าเป็นตัว P เก็บข้อมูลและนับเป็นข้อมูลตัวที่ 1 ถ้าไม่ใช่ตัว P ให้ไปรับค่าใหม่เข้ามา

กรณีที่ 2 ตรวจสอบข้อมูลว่าเป็นตัว A หรือ B ถ้าใช่ตัว A หรือ B ให้เก็บข้อมูลและนับเป็นข้อมูลตัวที่ 2 ถ้าไม่ใช่ตัว A หรือ B ให้ไปรับค่าใหม่เข้ามา

กรณีที่ 3 ตรวจสอบข้อมูลที่รับมา จากนั้นจัดเก็บข้อมูลนั้นและนับเป็นข้อมูลตัวที่ 3

กรณีที่ 4 ตรวจสอบข้อมูลที่รับมา จากนั้นจัดเก็บข้อมูลนั้นและนับเป็นข้อมูลตัวที่ 4

กรณีที่ 5 ตรวจสอบข้อมูลที่รับเข้ามา และนับเป็นข้อมูลตัวที่ 5

กรณีที่ 6 ตรวจสอบข้อมูลที่รับเข้ามา และนับเป็นข้อมูลตัวที่ 6

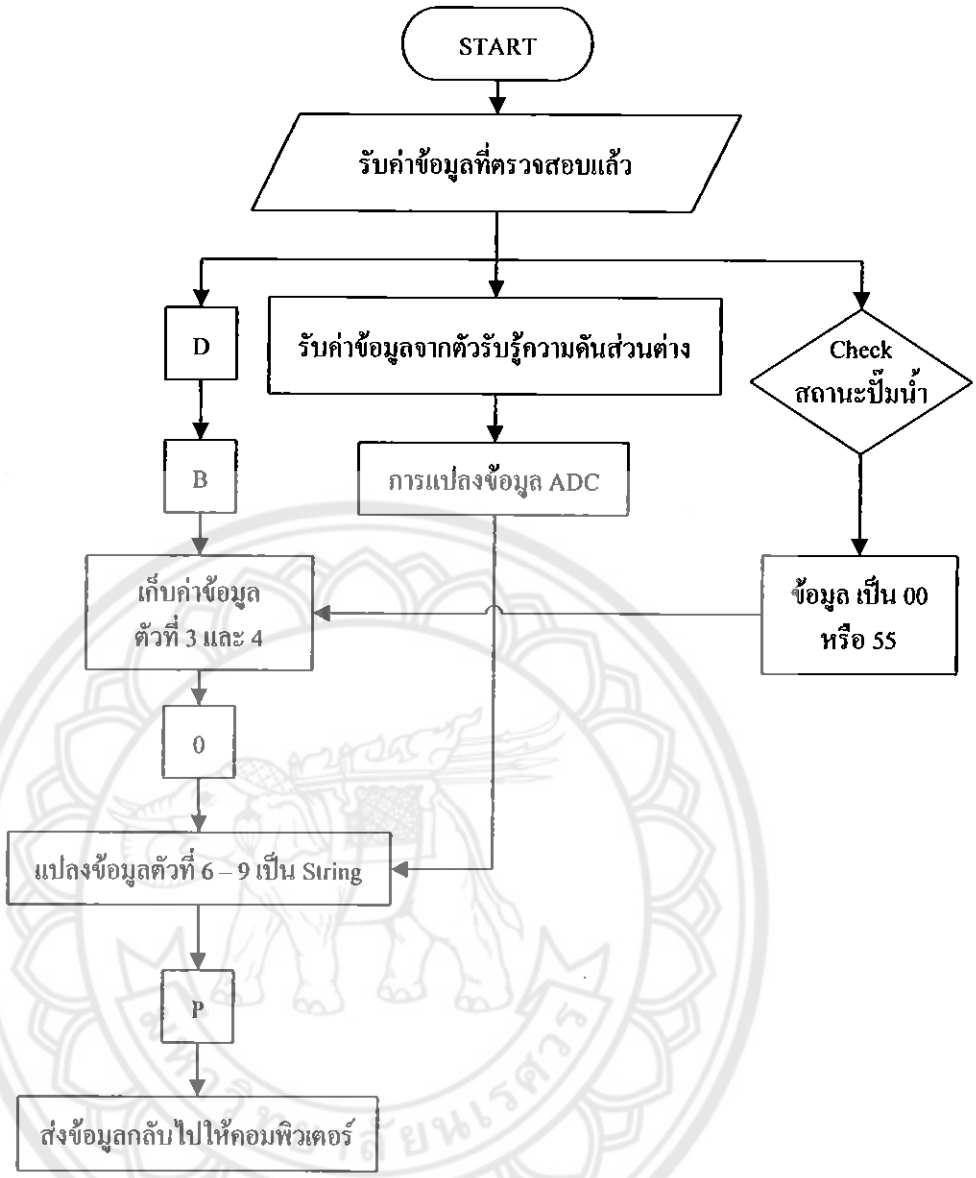
กรณีที่ 7 ตรวจสอบข้อมูลที่รับเข้ามา และนับเป็นข้อมูลตัวที่ 7

กรณีที่ 8 ตรวจสอบข้อมูลที่รับเข้ามา และนับเป็นข้อมูลตัวที่ 8

กรณีที่ 9 ตรวจสอบข้อมูลที่รับเข้ามา และนับเป็นข้อมูลตัวที่ 9

กรณีที่ 10 ตรวจสอบข้อมูลว่าเป็นตัว D หรือไม่ ถ้าเป็นตัว D ให้ตรวจสอบว่าข้อมูลตัวที่ 2 ที่รับมาเป็น A หรือ B ถ้าเป็น A ให้ไปทำโปรแกรมหลักต่อไป แต่ถ้าเป็น B ให้ไปเปิดหรือปิดมอเตอร์ปั๊มน้ำ ถ้าไม่ใช่ตัว D ให้ไปรับข้อมูลค่าใหม่เข้ามา

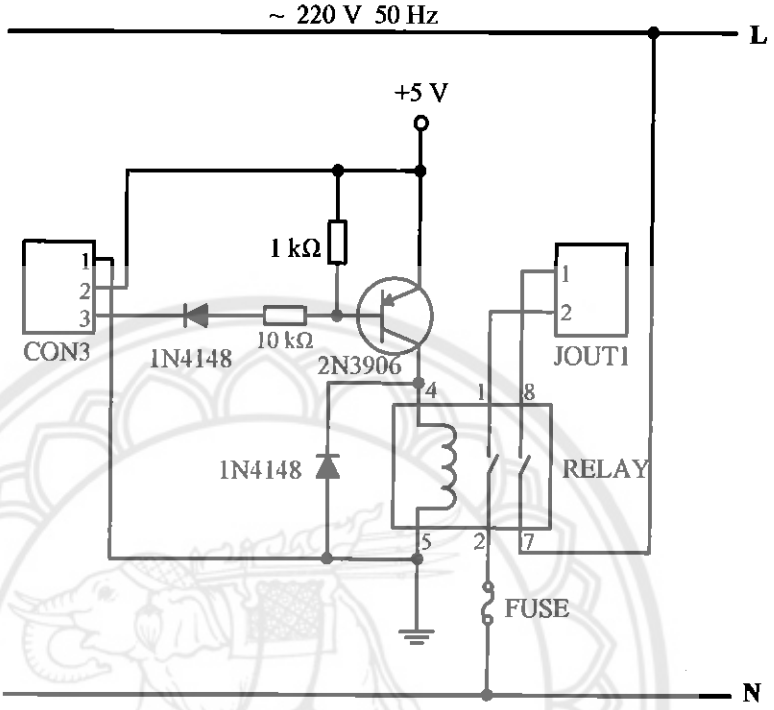
หลักการทำงานโปรแกรมหลักของไมโครคอนโทรลเลอร์แสดงดังรูปที่ 3.13 หลังจากที่ไมโครคอนโทรลเลอร์ได้ทำการตรวจสอบข้อมูลข้างต้นแล้วจึงเข้าสู่การทำงานของโปรแกรมหลักของไมโครคอนโทรลเลอร์โดยตรวจสอบสถานะของมอเตอร์ปั๊มน้ำหลังจากตรวจสอบเรียบร้อยแล้วจะนำค่าที่ตรวจสอบได้ไปเก็บค่าเป็นข้อมูลตัวที่ 3 และ 4 จากนั้นเป็นการรับค่าข้อมูลจากตัวรับรู้ความดันส่วนต่างแล้วแปลงข้อมูลที่ได้รับจากตัวรับรู้ความดันส่วนต่างเป็นแอนะล็อกเป็นดิจิตอลแล้วนำไปเก็บเป็นข้อมูลตัวที่ 6 ถึง 9 หลังจากนั้นทำการแปลงดิจิตอลเป็นสตริงอีกครั้งเมื่อส่งข้อมูลกลับไปยังคอมพิวเตอร์โดยส่งผ่าน XBee-PRO ตัวที่ 2 และตัวที่ 1 ตามลำดับ โดยการตอบกลับของไมโครคอนโทรลเลอร์นั้นได้ถูกออกแบบให้ส่งค่า D ออกก่อนต่อด้วย B จากนั้นเป็นข้อมูลที่ได้เก็บค่าไว้ที่ตัวที่ 3 และ 4 แล้วคั่นด้วยเลข 0 ต่อด้วยค่าข้อมูลตัวที่ 6 ถึง 9 ที่แปลงเป็นสตริงจากนั้นตามด้วย P โดยข้อมูลทั้งหมดถูกส่งผ่านไปยัง XBee-PRO เพื่อนำข้อมูลที่ประมวลผลเสร็จสิ้นไปแสดงผลทางจอคอมพิวเตอร์



รูปที่ 3.13 ลำดับการทำงานของโปรแกรมหลักในไมโครคอนโทรลเลอร์

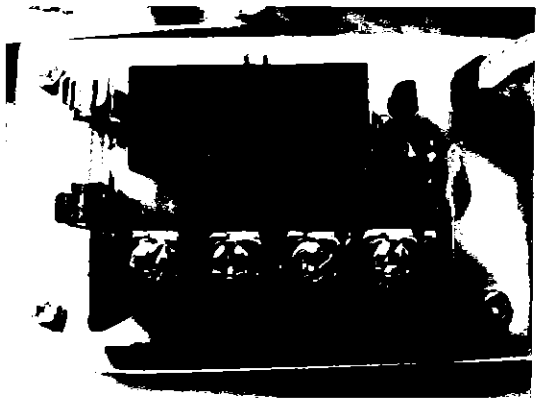
2) แผงวงจรรีเลย์และวงจรขับ

ในโครงงานนี้ใช้รีเลย์ทำหน้าที่เป็นสวิตช์ควบคุมการเปิดและปิดมอเตอร์มอเตอร์ปั๊มน้ำ โดยรับคำสั่งจากไมโครคอนโทรลเลอร์ แผงวงจรรีเลย์ที่ออกแบบและสร้างขึ้นแสดงดังรูปที่ 3.14



รูปที่ 3.14 แผนภาพวงจรรีเลย์และวงจรขับ

จากรูปที่ 3.14 สามารถอธิบายการทำงานของรีเลย์ได้ดังนี้คือ วงจรรีเลย์มีการนำทรานซิสเตอร์มาต่อเพื่อเพิ่มกระแสขับรีเลย์ เมื่อต้องการให้มอเตอร์ปั๊มน้ำทำงาน เราต้องให้ไมโครคอนโทรลเลอร์ส่งค่าลอจิก 0 มาเพื่อให้ทรานซิสเตอร์นำกระแส และถ้าต้องการให้มอเตอร์ปั๊มน้ำหยุดทำงาน เราต้องให้ไมโครคอนโทรลเลอร์ส่งค่าลอจิก 1 มาทำให้ทรานซิสเตอร์หยุดนำกระแส



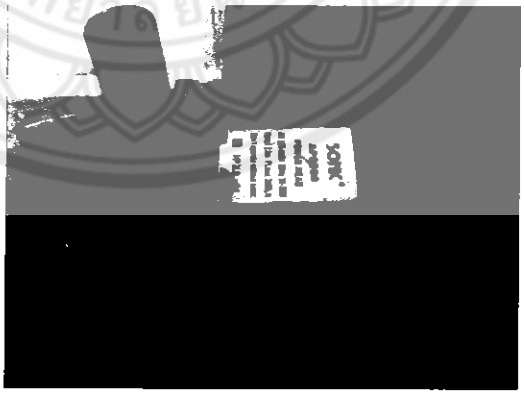
รูปที่ 3.15 แผงวงจรรีเลย์

3.3 แบบจำลองของระบบมอเตอร์ปั๊มน้ำ

1) ตัวรับรู้ความดันส่วนต่าง รุ่น MPX5010DP มีขั้วต่อทางไฟฟ้าจำนวน 6 ขา ซึ่งขาที่ 1 เป็นขาเอาท์พุต ขาที่ 2 ต่อกับกราวด์ และขาที่ 3 ต่อกับไฟเลี้ยงซึ่งต้องการใช้ 5 V เรานำสายยางต่อเข้ากับจุกของ MPX5010DP เพื่อใช้ผิวความดันของน้ำมาเปรียบเทียบกับความดันบรรยากาศ โดยป้องกันไม่ให้ตัวรับรู้สัมผัสกับน้ำโดยตรงเพื่อหลีกเลี่ยงอันตรายที่อาจเกิดจากไฟฟ้ารั่วได้

รูปที่ 3.16 ตัวรับรู้ความดันส่วนต่างรุ่น MPX5010DP

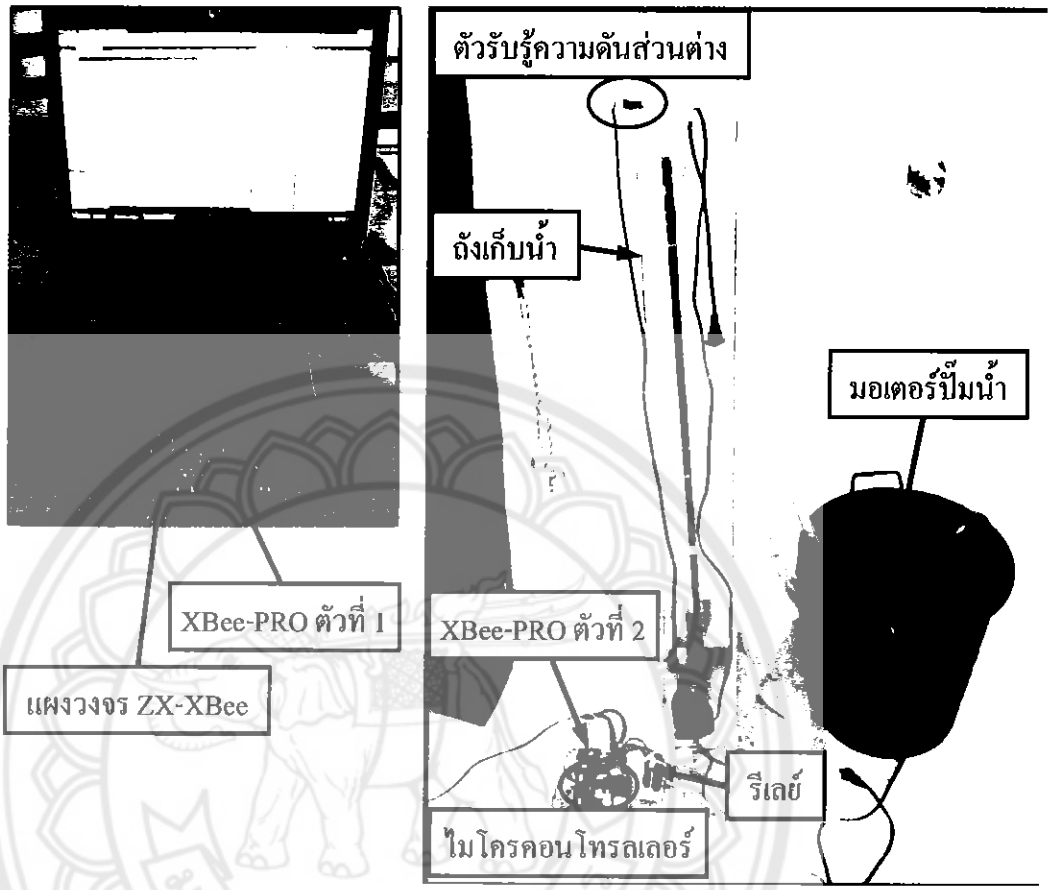
2) มอเตอร์ปั๊มน้ำ (รูปที่ 3.17) ซึ่งใช้กับไฟบ้าน (220 V 50 Hz) พิกัด 2,500 W สูบน้ำได้สูงสุด 2 m ซึ่งเพียงพอที่จะใช้งานกับความสูงของถังเก็บน้ำที่สร้างขึ้นในแบบจำลอง



รูปที่ 3.17 มอเตอร์ปั๊มน้ำรุ่น AP 2500 พิกัด 2,500 W

3) ถังเก็บน้ำ (รูปที่ 3.18) ซึ่งถูกออกแบบโดยเน้นความสูงมากกว่าความจุ โดยสร้างมีขนาด ฐาน 14 cm x 14 cm และสูง 110 cm

การเชื่อมต่ออุปกรณ์ในแบบจำลองการควบคุมมอเตอร์ปั๊มน้ำแสดงได้ดังรูปที่ 3.18



(ก) คอมพิวเตอร์และ XBee-PRO ตัวที่ 1

(ข) ไมโครคอนโทรลเลอร์ XBee-PRO ตัวที่ 2 และแบบจำลองระบบปั๊มน้ำ

รูปที่ 3.18 ส่วนประกอบและการเชื่อมต่ออุปกรณ์ในแบบจำลองของระบบควบคุมปั๊มน้ำ

บทที่ 4
ผลการทดสอบและการวิเคราะห์ผล

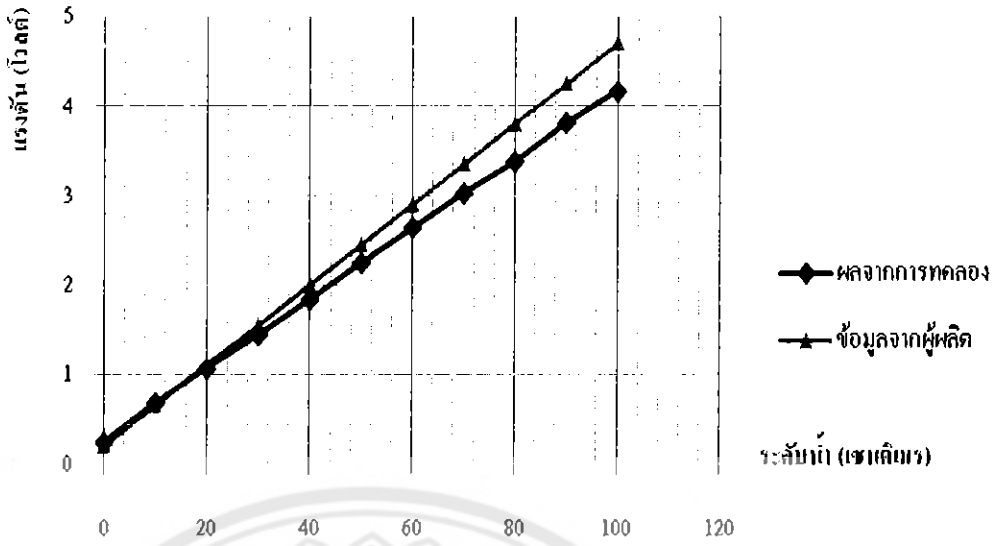
4.1 การทดลองวัดแรงดันไฟฟ้าจากตัวรับรู้ความดันส่วนต่าง

การทดลองนี้มีจุดประสงค์เพื่อวัดค่าแรงดันไฟฟ้าที่ได้จากตัวรับรู้ความดันส่วนต่างในแต่ละระดับน้ำทุกๆ 10 cm ตั้งแต่ 0 – 100 cm โดยทำการทดลอง 5 ครั้งสำหรับแต่ละค่าระดับน้ำแล้วหาค่าเฉลี่ยของแรงดันที่แต่ละระดับน้ำ ผลการทดลองแสดงดังตารางที่ 4.1 แล้วนำไปเปรียบเทียบกับข้อมูลที่ได้จากผู้ผลิตต่อไป

ตารางที่ 4.1 ค่าแรงดันไฟฟ้าจากตัวรับรู้ความดันส่วนต่าง

ระดับน้ำ (cm)	ค่าแรงดันไฟฟ้าของตัวรับรู้ความดันส่วนต่าง (V)					
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	ค่าเฉลี่ย
0	0.204	0.285	0.223	0.209	0.285	0.242
10	0.691	0.652	0.720	0.730	0.622	0.683
20	1.110	1.022	1.150	1.070	1.000	1.070
30	1.472	1.452	1.368	1.516	1.410	1.444
40	1.888	1.837	1.788	1.870	1.822	1.841
50	2.294	2.253	2.169	2.365	2.188	2.254
60	2.605	2.690	2.568	2.747	2.630	2.648
70	3.014	3.100	2.901	3.114	3.023	3.030
80	3.425	3.404	3.250	3.416	3.423	3.384
90	3.791	3.776	3.825	3.872	3.840	3.821
100	4.190	4.200	4.000	4.230	4.230	4.170

จากตารางที่ 4.1 สามารถนำค่าแรงดันไฟฟ้าที่วัดได้ มาเขียนเป็นกราฟได้ดังรูปที่ 4.1 จากการทดลองจะพบว่าเมื่อระดับน้ำเพิ่มสูงขึ้น ค่าแรงดันไฟฟ้าที่ได้จากตัวรับรู้ความดันส่วนต่างจะมีค่าเพิ่มสูงขึ้นตามกันแบบเชิงเส้น จากรูปที่ 4.1 จะเห็นได้ว่าที่ระดับน้ำสูงขึ้นค่าแรงดันที่วัดได้จากตัวรับรู้มีค่าแตกต่างจากข้อมูลที่ได้จากผู้ผลิตเพิ่มมากขึ้นโดยมีสาเหตุมาจากความผิดพลาดภายในตัวรับรู้ความดันส่วนต่าง และการติดตั้งสายยางที่ต่อมาจากหัวจุกของตัวรับรู้



รูปที่ 4.1 ความสัมพันธ์ระหว่างแรงดันไฟฟ้าจากตัวรับรู้ความดันส่วนต่างกับระดับน้ำ

4.2 การทดลองหาค่าความคลาดเคลื่อนของค่าระดับน้ำที่แสดงผล

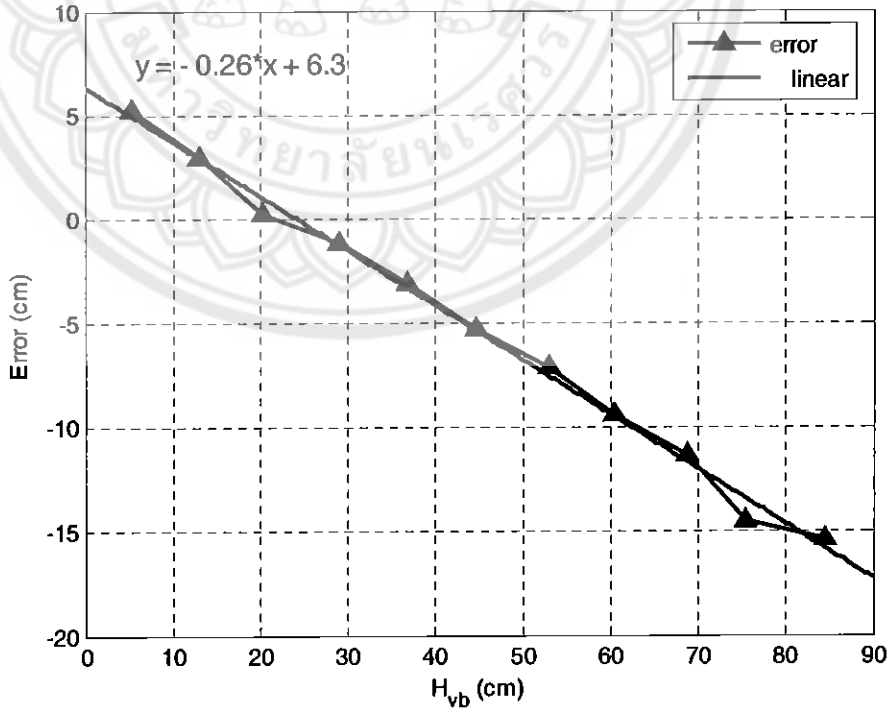
การทดลองนี้มีวัตถุประสงค์ 2 ข้อคือ

1) เพื่อคำนวณหาค่าความคลาดเคลื่อนของค่าระดับน้ำที่แสดงผลบนส่วนต่อประสานกราฟิกกับผู้ใช้ โดยเปรียบเทียบกับค่าระดับน้ำจริง
2) เพื่อตรวจสอบค่าความใกล้เคียงของระดับน้ำที่แสดงผลบนส่วนต่อประสานกราฟิกกับผู้ใช้กับค่าระดับน้ำจริงเมื่อพิจารณาใช้ค่าชดเชยแล้ว

การทดลองนี้แบ่งออกเป็น 2 ขั้นตอนคือ ในขั้นที่ 1 ทำการอ่านค่าระดับน้ำที่แสดงผลบนส่วนต่อประสานกราฟิกกับผู้ใช้สำหรับระดับน้ำแต่ละค่าซึ่งห่างกัน 10 cm โดยทำการทดลอง 5 ครั้งสำหรับแต่ละค่าระดับน้ำ ผลการทดลองแสดงดังตารางที่ 4.2 ซึ่งพบว่าความคลาดเคลื่อนระหว่างค่าระดับน้ำที่แสดงผลบนส่วนต่อประสานกราฟิกกับผู้ใช้กับค่าระดับน้ำจริงมีความสัมพันธ์กับค่าระดับน้ำจริงในรูปแบบเชิงเส้นสามารถแสดงด้วยกราฟเส้นตรงได้ดังรูปที่ 4.2

ตารางที่ 4.2 ค่าระดับน้ำที่แสดงบนส่วนต่อประสานกราฟิกกับผู้ใช้ก่อนใช้ค่าชดเชย

ระดับน้ำ	ค่าระดับน้ำที่แสดงบนGUI (cm)						ค่าความคลาดเคลื่อน	
(cm)	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	ค่าเฉลี่ย	(cm)	(%)
0	4.49	4.30	4.69	7.51	5.08	5.21	5.21	-
10	12.35	13.87	12.05	13.38	12.99	12.93	2.93	29.3
20	17.80	21.58	19.96	21.02	20.90	20.25	0.25	1.25
30	29.69	29.64	28.52	29.30	27.05	28.84	-1.16	3.87
40	37.11	36.43	36.33	37.21	37.43	36.90	-3.1	7.75
50	45.31	45.02	43.75	44.68	44.53	44.66	-5.34	10.68
60	52.05	53.02	51.56	54.79	53.03	52.89	-7.11	11.85
70	60.06	61.62	58.50	61.62	60.84	60.53	-9.47	13.53
80	69.24	69.65	66.11	69.24	69.14	68.68	-11.32	14.15
90	76.56	75.29	72.83	75.49	77.05	75.44	-14.56	16.18
100	84.77	86.04	81.64	86.04	84.38	84.57	-15.43	15.43



รูปที่ 4.2 ค่าความคลาดเคลื่อนของการแสดงค่าระดับน้ำบนส่วนต่อประสานกราฟิกกับผู้ใช้

จากรูปที่ 4.2 เราสามารถหาสมการค่าชดเชยความคลาดเคลื่อนในการแสดงผลโดยอาศัยหลักการเลือกเส้นกราฟที่เหมาะสมกับข้อมูล (Curve fitting) จะได้

$$\text{Comp} = -0.26 \cdot H_{VB} + 6.3 \tag{4.1}$$

โดยที่ Comp คือ ค่าชดเชยความคลาดเคลื่อนในการแสดงผล (cm)

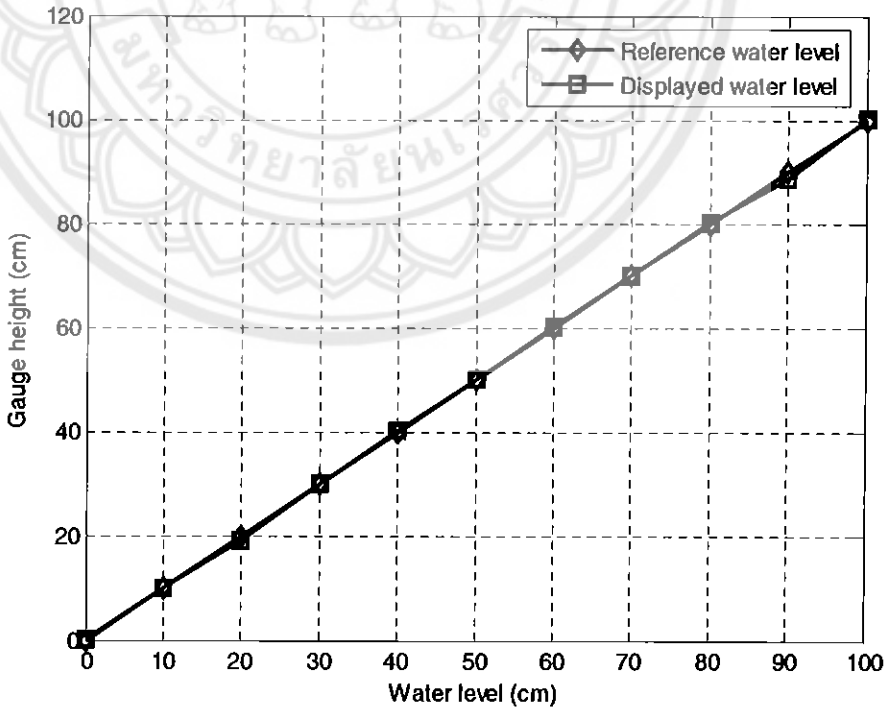
H_{VB} คือ ค่าระดับน้ำที่แปลงมาจากค่าแรงดันไฟฟ้าในเลขฐานสิบที่ได้รับจากไมโครคอนโทรลเลอร์ (cm)

จากนั้นทำการเขียนโปรแกรมในวิชวลเบสิก 2010 ให้แสดงค่าระดับน้ำบนส่วนต่อประสานกราฟิกกับผู้ใช้ โดยนำค่าแรงดันไฟฟ้าในรูปของเลขฐานสิบที่ได้รับจากไมโครคอนโทรลเลอร์ลบด้วยค่าชดเชยตามสมการที่ (4.1) แล้วจะได้ค่าระดับน้ำจึงได้สมการดังนี้

$$H_{disp} = 1.26 \cdot H_{VB} - 6.3 \tag{4.2}$$

โดยที่ H_{disp} คือ ค่าระดับน้ำที่แสดงบนส่วนต่อประสานกราฟิกกับผู้ใช้ (cm)

เมื่อนำผลจากการใช้สมการที่ (4.2) มาเปรียบเทียบกับระดับน้ำจริง สามารถแสดงได้ดังรูปที่ 4.3 ซึ่งพบว่าค่าระดับน้ำที่แสดงผลบนส่วนต่อประสานกราฟิกกับผู้ใช้นั้นใกล้เคียงกับค่าระดับน้ำจริงอย่างมาก



รูปที่ 4.3 ความสัมพันธ์ระหว่างค่าระดับน้ำที่แสดงผลกับค่าระดับน้ำจริงหลังจากใช้ค่าชดเชย

ในขั้นที่ 2 ของการทดลองเราทำการทดลองวัดระดับน้ำและอ่านค่าระดับน้ำที่แสดงบนส่วนต่อประสานกราฟิกกับผู้ใช้ซึ่งเกิดจากการใช้ค่าชดเชยตามสมการที่ (4.2) เรียบร้อยแล้ว โดยในการวัดระดับน้ำแต่ละค่ามีการทดลองซ้ำ 3 ครั้งแล้วหาค่าเฉลี่ยผลการทดลองแสดงดังตารางที่ 4.3

ตารางที่ 4.3 ค่าระดับน้ำที่แสดงบนส่วนต่อประสานกราฟิกกับผู้ใช้หลังจากใช้ค่าชดเชย

ระดับน้ำ	ค่าระดับน้ำที่แสดงบน GUI (cm)				ค่าความคลาดเคลื่อน	
(cm)	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ค่าเฉลี่ย	(cm)	(%)
0	1.13	0.72	0.80	0.88	0.88	-
10	11.18	9.75	9.32	10.08	0.08	0.8
20	20.16	20.41	20.76	20.44	0.44	2.2
30	30.73	31.60	29.27	30.53	0.53	1.77
40	40.81	41.57	39.41	40.62	0.62	1.55
50	51.72	52.39	49.68	51.26	1.26	2.52
60	60.64	61.26	63.10	61.67	1.67	2.78
70	71.09	71.72	72.79	71.87	1.87	2.67
80	82.91	82.42	82.91	82.75	2.75	3.44
90	91.04	92.82	94.45	92.77	2.77	3.07
100	102.49	102.60	103.58	102.89	2.89	2.89

จากตารางที่ 4.3 การใช้ค่าชดเชยตามสมการที่ (4.2) สามารถลดความคลาดเคลื่อนในการแสดงผลมีลงได้อย่างมาก ส่งผลให้ระดับน้ำที่แสดงผลบนส่วนต่อประสานกราฟิกกับผู้ใช้มีค่าใกล้เคียงกับระดับน้ำจริง

4.3 การทดสอบระยะการส่งสูงสุดของ XBee-PRO

เราสามารถทดสอบหาระยะการส่งสูงสุดของ XBee-PRO โดยทดลองภายในบริเวณมหาวิทยาลัยนเรศวรโดยส่งข้อมูลจากคอมพิวเตอร์ผ่าน XBee-PRO ไปยังไมโครคอนโทรลเลอร์ในระยะทางต่างๆเพื่อสั่งให้รีเลย์ทำงานโดยเปลี่ยนหน้าสัมผัส ในที่นี้ได้ทดสอบใน 2 กรณีดังนี้

1) การทดสอบหาระยะการส่งที่สูงสุดในร่ม โดยทดสอบวัดระยะการส่งไกลที่สุดในแนวระดับระหว่างบริเวณชั้นล่างของอาคารภาควิชาวิศวกรรมอุตสาหการกับชั้นล่างของอาคารวิศวกรรมโยธา และได้ระยะการส่งสูงสุดประมาณ 80 m ในขณะที่การทดสอบวัดระยะการส่งสูงสุดในแนวดิ่งเพื่อทำการส่งข้อมูลจากบริเวณชั้นล่างของอาคารภาควิศวกรรมไฟฟ้าและคอมพิวเตอร์ พบว่าสามารถส่งข้อมูลได้สูงสุดไปถึงชั้นที่ 3 ของอาคารเดียวกันซึ่งคิดเป็นระยะทางประมาณ 6.50 m อย่างไรก็ตามข้อมูลจากผู้ผลิตระบุว่าระยะการส่งสูงสุดในร่มของ XBee-PRO ได้ประมาณ 100 m ในขณะที่จากการทดสอบได้ค่าต่ำกว่าค่าที่ระบุอันมีสาเหตุจากสิ่งกีดขวางอยู่ระหว่างบริเวณที่ส่งและรับสัญญาณจึงทำให้ได้ระยะน้อยลง

2) การทดสอบหาระยะการส่งสูงสุดกลางแจ้ง โดยทดสอบวัดระยะทางการส่ง ซึ่งได้ระยะในการส่งระหว่างชั้นล่างของอาคารภาควิชาคณิตศาสตร์และอาคารเภสัชศาสตร์ ซึ่งได้ระยะในการส่งสูงสดประมาณ 600 m ซึ่งน้อยกว่าที่ระบุโดยผู้ผลิต (1,500 m) อันเนื่องจากระยะ 600 m ที่วัดได้นั้นเป็นระยะทางสูงสุดกลางแจ้งที่สามารถทดสอบได้โดยไม่มีสิ่งกีดขวางในพื้นที่มหาวิทยาลัยนเรศวร

บทที่ 5
สรุปผลและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

ในโครงงานนี้ได้พัฒนาระบบควบคุมการเปิดและปิดมอเตอร์ปั๊มน้ำโดยใช้วิธีการควบคุมทางไกลแบบไร้สาย โดยเลือกใช้โมดูลสื่อสารไร้สายย่าน 2.4 GHz รุ่น XBee-PRO เพื่อรับและส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์กับไมโครคอนโทรลเลอร์รุ่น PIC30F4011 โดยการเขียนชุดคำสั่งด้วยโปรแกรมวิชวลเบสิก 2010 เพื่อควบคุมการรับและส่งข้อมูลรวมทั้งสร้างส่วนต่อประสานกราฟิกกับผู้ใช้ ในส่วนของการประมวลผลด้วยไมโครคอนโทรลเลอร์มีการเขียนโปรแกรมด้วยภาษาซีเพื่อควบคุมการทำงานของรีเลย์ซึ่งเป็นตัวเชื่อมต่อระหว่างแหล่งจ่ายไฟกระแสสลับขนาด 220 V ความถี่ 50 Hz กับมอเตอร์ปั๊มน้ำ ในโครงงานได้สร้างแบบจำลองปั๊มน้ำขึ้นโดยใช้มอเตอร์ปั๊มน้ำขนาด 2,500 W เพื่อปั๊มเข้าไปเก็บในถังน้ำสูง 110 cm และใช้ตัวรับรู้ความดันส่วนต่างรุ่น MPX5010DP เพื่อตรวจวัดระดับน้ำในถัง โดยตัวรับรู้ความดันส่วนต่างจะให้ค่าแรงดันเอาท์พุตแปรผันตรงกับค่าระดับความสูงของน้ำที่ต้องการวัด ไมโครคอนโทรลเลอร์รับค่าแรงดันจากตัวรับรู้ความดันส่วนต่างเข้ามาประมวลผลตลอดเวลา หลังจากที่เชื่อมต่อระบบควบคุมเรียบร้อยแล้วการเปิดและปิดมอเตอร์ปั๊มน้ำจะเป็นไปอย่างอัตโนมัติโดยขึ้นอยู่กับค่าระดับน้ำในถังซึ่งวัดได้จากตัวรับรู้เทียบกับค่าขีดจำกัดล่างและขีดจำกัดบนซึ่งผู้ใช้สามารถกำหนดค่าเองได้ อย่างไรก็ตามผู้ใช้สามารถหยุดการทำงานของมอเตอร์ปั๊มน้ำได้ทุกขณะที่ต้องการ

5.2 ปัญหาและแนวทางการแก้ไข

1) ค่าความคลาดเคลื่อนในการวัดระดับน้ำส่งผลให้ระดับน้ำที่แสดงผลบนหน้าต่างส่วนต่อประสานกราฟิกกับผู้ใช้มีค่าไม่ตรงกับค่าจริงของระดับน้ำในถัง เนื่องจากมีความคลาดเคลื่อนภายในตัวรับรู้ความดันส่วนต่าง ความคลาดเคลื่อนจากการติดตั้งสายยางในขณะทำการวัด และการหน่วงเวลาในการรับและส่งสัญญาณข้อมูล แนวทางการลดค่าความคลาดเคลื่อนในการวัดดังกล่าว ได้แก่ การติดตั้งสายยางที่ต่อกับจุกตัวรับรู้ความดันส่วนต่างให้ตรง และปรับตั้งค่าการการหน่วงเวลาในการรับและส่งสัญญาณข้อมูลให้เหมาะสม
2) ความคลาดเคลื่อนในการวัดค่าของตัวรับรู้ความดันส่วนต่างส่งผลต่อการเปรียบเทียบค่ากับขีดจำกัดบนและขีดจำกัดล่างที่ผู้ใช้กำหนด ทำให้มีการสั่งการปิดและเปิด

มอเตอร์ปั๊มน้ำช้ากว่าความเป็นจริง แนวทางการแก้ไขปัญหาได้แก่การใส่ค่าชดเชยความผิดพลาดของระดับน้ำเข้าไปในโปรแกรมเพื่อให้ตัวเลขระดับน้ำมีความใกล้เคียงกับความเป็นจริง

3) ความเสียหายของอุปกรณ์อันเนื่องมาจากการใช้อุปกรณ์ไม่ถูกวิธี แนวทางการแก้ไขปัญหาคือการศึกษาวิธีใช้งานตลอดจนข้อห้ามในการใช้งานอุปกรณ์อย่างละเอียด รวมทั้งการใช้งานอย่างระมัดระวัง

5.3 แนวทางในการพัฒนาต่อไป

จากหลักการและระบบควบคุมที่พัฒนาขึ้นในโครงงานนี้ เราสามารถเพิ่มจำนวนปั๊มน้ำที่ต้องการควบคุม รวมถึงใช้ควบคุมในรูปแบบอื่นนอกเหนือจากการเปิดและปิด ยังสามารถนำหลักการไปประยุกต์ใช้กับการควบคุมการทำงานของอุปกรณ์ไฟฟ้าอื่นๆที่อยู่ไกลจากห้องควบคุม ได้แก่ เครื่องจักรกลไฟฟ้าภายในโรงงานเพื่อเป็นการอำนวยความสะดวกแก่ผู้ใช้ นอกจากนี้ยังสามารถเปลี่ยนโมดูลสื่อสารข้อมูลจากรุ่นที่ใช้ย่านความถี่ 2.4 GHz เป็นโมดูลสื่อสารข้อมูลที่สามารถเชื่อมต่อกับอินเตอร์เน็ต ได้แก่ BLUE STICK และ TTL UART to WiFi เพื่อช่วยลดปัญหาจากสิ่งกีดขวางในการรับและส่งข้อมูล

เอกสารอ้างอิง

[1] สรกฤช สิริปรีดากุล และพิมลลักษณ์ จิรกุลกนก. (2551). "การพัฒนาระบบควบคุมไฟฟ้าโดยใช้ Zigbee ประเภทหัวข้อพิเศษ Mobile application". ใน **โครงการวิจัย พัฒนา และวิศวกรรม** (หน้า 5-7). กรุงเทพฯ: ภาควิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยธรรมศาสตร์.

[2] บริษัท วีนัส ซัพพลาย จำกัด. "โมดูลสื่อสารข้อมูลอนุกรมแบบไร้สาย". สืบค้นเมื่อ 18 สิงหาคม 2555, จาก http://www.thaieasyelec.com.

[3] บริษัท บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด. "เอกสารประกอบการใช้งานโมดูลสื่อสารไร้สาย 2.4 GHz " สืบค้นเมื่อ 20 กรกฎาคม 2555, จาก http://www.inex.co.th

[4] บริษัท อีทีที จำกัด. "เอกสารประกอบการใช้งานไมโครคอนโทรลเลอร์" สืบค้นเมื่อ 7 มกราคม 2556, จาก http://www.etteam.com.

[5] บริษัท เอส.ที.คอนโทรล จำกัด. "ทรานสดิวเซอร์ (Transducer)" สืบค้นเมื่อ 21 พฤศจิกายน 2555, จาก http://www.stcontrol.com.

[6] บริษัท อีลีเมนต์ โฟร์ทีน จำกัด. "เอกสารประกอบการใช้งานตัวรับรู้ความดันส่วนต่าง" สืบค้นเมื่อ 20 ธันวาคม 2555, จาก http://th.element14.com

ภาคผนวก ก

คู่มือการติดตั้งและใช้งานโปรแกรม X-CTU ของ XBee-PRO

5.2 ติดตั้งไดรเวอร์ USB สำหรับบอร์ด ZX-XBeeU

(5.21) ดับเบิลคลิกที่ไฟล์ *USBDriverInstallerV2.04.06.exe* (ตัวเลขเวอร์ชันอาจเปลี่ยนแปลงได้ตามการปรับปรุงล่าสุด) เพื่อเริ่มต้นการติดตั้งไดรเวอร์

จะปรากฏไดอะล็อกบ็อกแจ้งการติดตั้งเสร็จเรียบร้อย

(5.2.2) จากนั้นเสียบ ZX-XBeeU เข้ากับพอร์ต USB ของคอมพิวเตอร์ รอจนกระทั่งไฟสีน้ำเงินที่ตำแหน่ง USB บนแผงวงจรควบคุมติดสว่าง

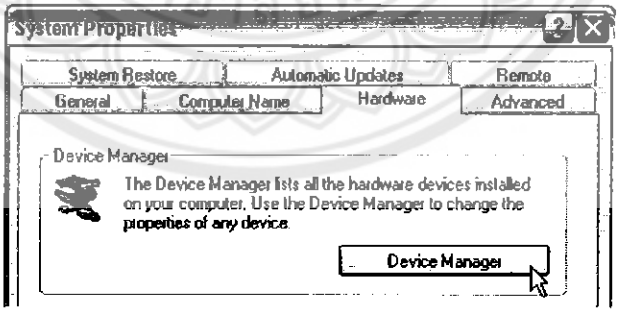
5.3 การตรวจสอบตำแหน่งของพอร์ตอนุกรมเสมือน หรือ USB Serial port สำหรับ ZX-XBeeU

หลังจากติดตั้งไดรเวอร์ของ ZX-BTU แล้ว ขั้นตอนต่อไปคือ ตรวจสอบตำแหน่งพอร์ตอนุกรมเสมือนที่เกิดขึ้นหลังจากเชื่อมต่อ ZX-XBeeU เข้ากับพอร์ต USB ของคอมพิวเตอร์ มีขั้นตอนดังนี้

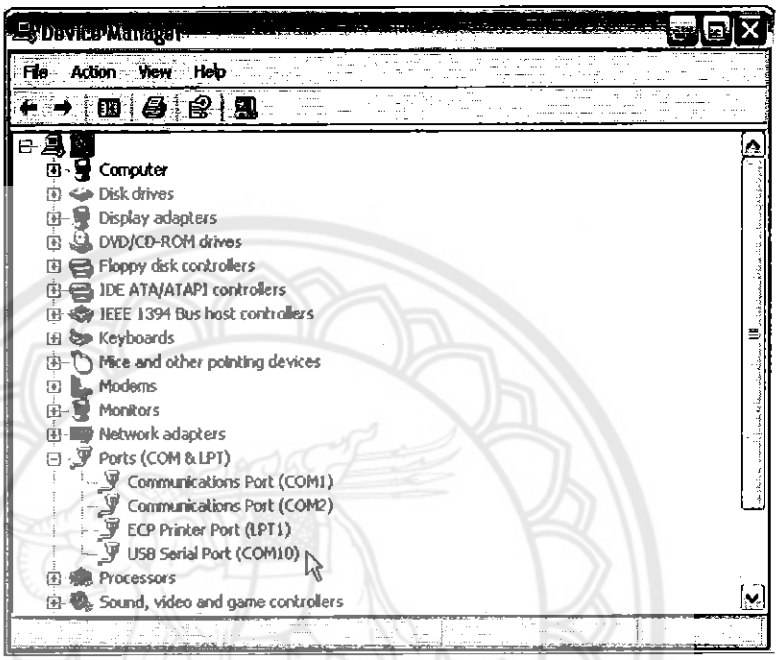
(5.3.1) คลิกที่ปุ่ม Start แล้วเลือกไปที่ Control Panel

(5.3.2) จากนั้นดับเบิลคลิกเลือกที่ System

(5.3.3) เลือกไปที่แท็ป Hardware แล้วคลิกที่ Device Manager



(5.3.4) ตรวจสอบรายการฮาร์ดแวร์ที่หัวข้อ Port จะพบ USB Serial port ให้ดูว่ามีการเลือกตำแหน่งของ USB Serial port ไว้ที่ตำแหน่งใด *ปกติจะเป็น COM3 ขึ้นไป* ให้ใช้ค่าของตำแหน่งของพอร์ตนี้ในการกำหนดการเชื่อมต่อกับโปรแกรมต่อไป ตามรูปตัวอย่างจะเป็น COM10



5.4 ขั้นตอนการตั้งค่า

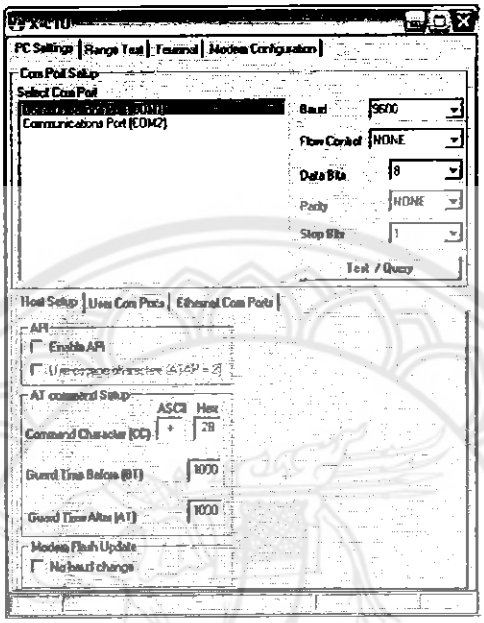
(5.4.1) ติดตั้งโปรแกรม X-CTU เวอร์ชั่นล่าสุด ดาวน์โหลดได้ฟรีจากเวบไซต์ของผู้ผลิตที่ www.digi.com

(5.4.2) ติดตั้งโมดูล XBee-PRO ลงบนบอร์ด ZX-XBeeU ต้องระวังเรื่องตำแหน่งขาและทิศทางของโมดูลต้องติดตั้งให้ถูกต้อง และไม่เกิดการเหลื่อมกันเด็ดขาด เพราะหากติดตั้งผิดแล้ว เมื่อจ่ายไฟ จะทำให้โมดูลเสียหายทันที

(5.4.3) ต่อสายเชื่อมต่อพอร์ต USB ระหว่างบอร์ด ZX-XBeeU กับพอร์ต USB ของคอมพิวเตอร์

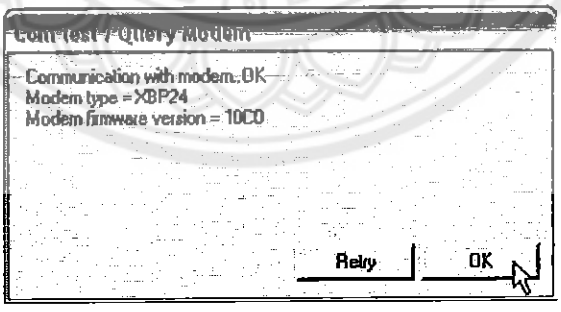
(5.4.4) จ่ายไฟให้แก่บอร์ดซึ่งใช้ไฟเลี้ยง +5V จากพอร์ต USB ของคอมพิวเตอร์ LED ในตำแหน่ง POWER และ ON ติด และ LED ในตำแหน่ง ASS. กะพริบ หากไม่เป็นไปตามนี้ ให้รีบปิดสวิตช์ ปลดไฟเลี้ยง แล้วตรวจสอบการติดตั้งโมดูล XBee-PRO ทันที รวมทั้งตรวจสอบไฟเลี้ยงที่ขา Vcc ของ XBee-PRO ว่าต้องอยู่ในช่วง +2.8 ถึง +3.3V โดยในการตรวจสอบนั้นต้องถอดโมดูลXBee-PRO ออกมาก่อน แล้ววัดแรงดันที่คอนเน็กเตอร์ตัวเมียที่ใช้สำหรับติดตั้งโมดูล XBee-PRO

(5.4.5) เปิดโปรแกรม X-CTU โดยดับเบิลคลิกที่ไอคอน  บน Desktop ของคอมพิวเตอร์ หรือคลิกที่ **Start → All Programs → Digi-Maxstream → X-CTU** หน้าต่างกำหนดการเชื่อมต่อจะปรากฏขึ้นมาดังรูป



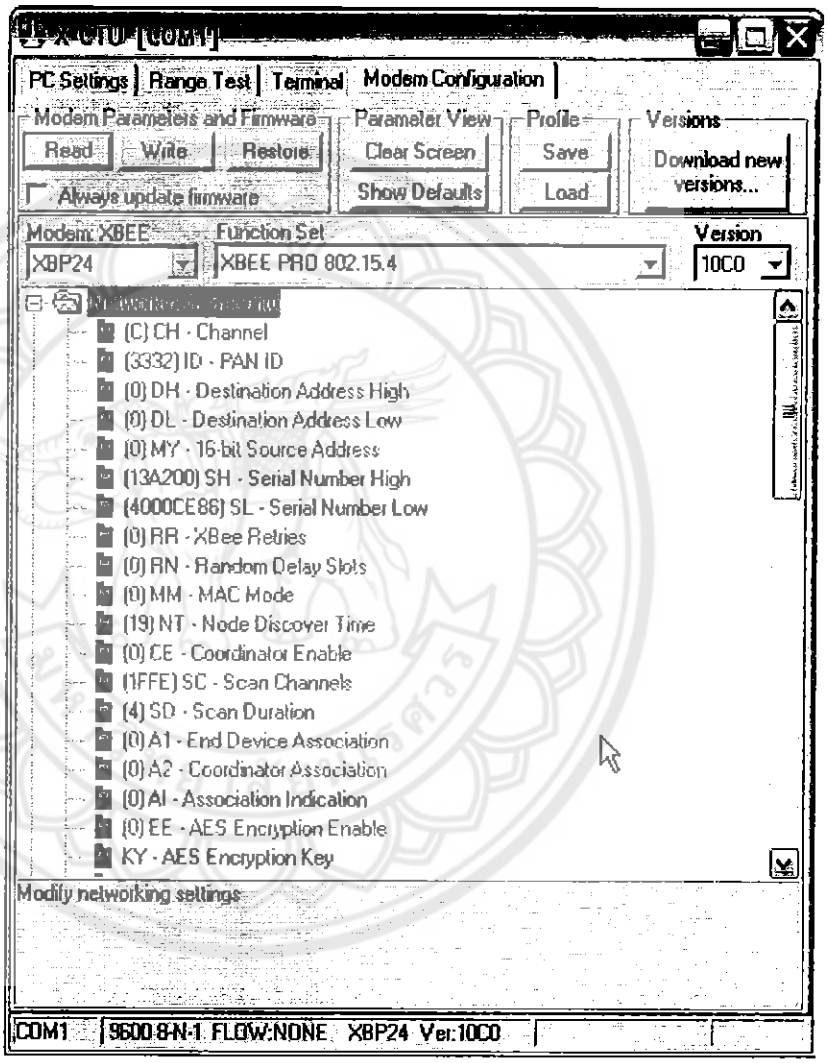
ให้ทำการเลือกพอร์ตที่ทำการเชื่อมต่อ เลือกอัตราบอด (Baurate) เป็น 9600, Data 8 บิต, Parity ไม่มีการตรวจสอบ และ Stop เป็น 1 บิต

(5.4.6) กดปุ่ม Test เพื่อทดสอบการติดต่อระหว่าง XBee-PRO กับโปรแกรม X-CTU หากติดต่อกันได้ จะปรากฏหน้าต่างแจ้งผลการติดต่อและข้อมูลทางฮาร์ดแวร์เบื้องต้นของโมดูล XBee-PRO ดังรูป



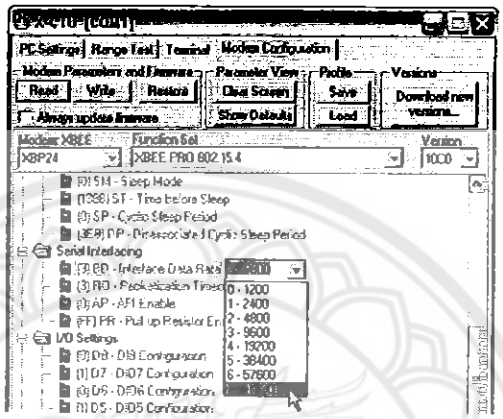
หากมีการแจ้งความผิดพลาดใดๆ เกิดขึ้นในขั้นตอนนี้ให้รีบปิดสวิตช์ ปลดไฟเลี้ยง แล้วตรวจสอบการติดตั้งโมดูล XBEe-PRO และการเชื่อมต่ออีกครั้ง รวมทั้งตำแหน่งขาพอร์ตด้วย และถ้าจำเป็นอาจต้องทดลองเลือกอัตราบอดใหม่

(5.4.7) หลังจากนั้นคลิกไปที่แท็ป Modem Configuration แล้วกดปุ่ม Read ในกรอบ Modem Parameters and Firmware จะปรากฏข้อมูลชื่อรุ่นของโมดูล XBee-PRO ชื่อฟังก์ชั่น หมายเลขเวอร์ชันของเฟิร์มแวร์ และค่าพารามิเตอร์ต่างๆ ดังรูป



(5.4.8) เมื่อมาถึงขั้นตอนนี้ ผู้ใช้งานสามารถเปลี่ยนค่าคอนฟิกูเรชั่นได้ตามต้องการ ไม่ว่าจะเป็นการเลือกอัตราบอดใหม่ กำหนดรูปแบบการทำงานของขาพอร์ตของ XBee-PRO

(5.4.8.1) หากต้องการเปลี่ยนอัตราบอดให้ไปที่หัวข้อ Serial Interfacing คลิกเลือกที่บรรทัด BD - Interface Data Rate จะปรากฏเมนูให้เลือกค่าอัตราบอด 8 ค่า ตั้งแต่ 1,200 ถึง 115,200 บิตต่อวินาที



(5.4.8.2) หากต้องการเปลี่ยนการทำงานของขาพอร์ต ให้เลือกไปที่หัวข้อ I/O Setting แล้วเลือกไปที่ขาพอร์ตที่ต้องการเปลี่ยนค่า จะปรากฏช่องให้เลือกฟังก์ชั่นการทำงาน ซึ่งมีด้วยกัน 5 แบบคือ

0 - DISABLE หมายถึง ปิดการทำงาน (ปกติเป็นค่าตั้งต้น)

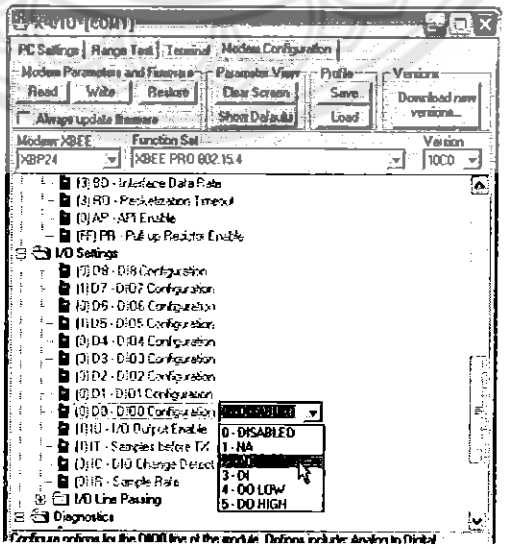
1 - NA หมายถึง ยังไม่มีการกำหนดฟังก์ชั่น หรือสำรองไว้

2 - ADC หมายถึง เลือกเป็นอินพุตอะนาลอก

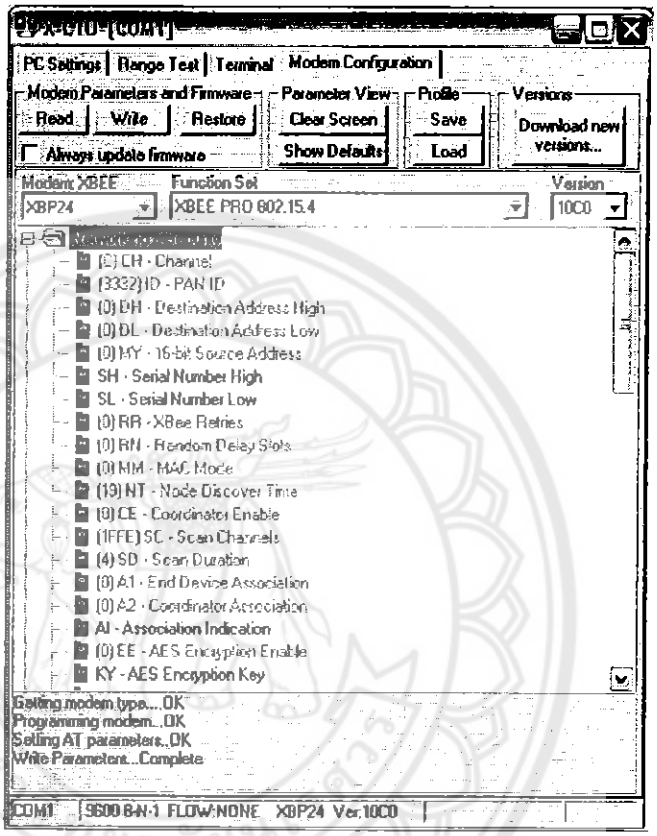
3 - DI หมายถึง เลือกเป็นอินพุตดิจิตอล

4 - DO LOW หมายถึง เลือกเป็นเอาต์พุตดิจิตอลลอจิกต่ำ

5 - DO HIGH หมายถึง เลือกเป็นเอาต์พุตดิจิตอลลอจิกสูง



(5.4.8.3) จากนั้นกดปุ่ม Write รอสักครู่ สังเกตที่ด้านล่างของหน้าต่าง Modem Configuration จะแสดงข้อความเพื่อแจ้งสถานะการทำงาน หากเป็นดังรูปแสดงว่า การกำหนดค่าสร็จสมบูรณ์



6. รีจิสเตอร์ที่ควรทราบของโมดูล Xbee-PRO

การส่งข้อมูลและควบคุมระหว่างไมโครคอนโทรเลอร์หรือคอมพิวเตอร์กับโมดูล XBee-PRO นั้น จะใช้การสื่อสารแบบอนุกรม UART ซึ่งโมดูล XBee-PRO สามารถใช้ความเร็วในการส่งข้อมูล (Baud rate) ได้ตั้งแต่ 1,200 จนถึง 115,200 บิตต่อวินาที (bps : bit per second) โดยค่าที่กำหนดเป็นค่าเริ่มต้นคือ 9,600 บิตต่อวินาที และสามารถเปลี่ยนความเร็วในการส่งข้อมูลได้ที่รีจิสเตอร์ BD

ในการติดต่อสื่อสารระหว่างโมดูล XBee-PRO สามารถจัดเครือข่ายได้หลายรูปแบบ โดยการแยกช่องสัญญาณและเครือข่าย รีจิสเตอร์ที่ใช้สำหรับจัดการเกี่ยวกับเครือข่าย มีดังนี้

1. รีจิสเตอร์ CH (Channel) ใช้กำหนดช่องสัญญาณ เลือกได้ตั้งแต่ช่องที่ 0x0C ถึง 0x17 แต่ละช่องไม่สามารถส่งข้อมูลข้ามช่องสัญญาณกันได้

2. รีจิสเตอร์ ID (PAN ID / Personal Area Network ID) ใช้กำหนดหมายเลขเครือข่าย เลือกค่าได้ตั้งแต่ 0x0000 จนถึง 0xFFFE โดยแต่ละเครือข่ายจะไม่สามารถส่งข้อมูลข้ามเครือข่ายได้ ยกเว้นกำหนดด้วยค่า 0xFFFF จะสามารถส่งข้อมูลไปทุกเครือข่ายได้ แต่จะไม่สามารถรับข้อมูลจากเครือข่ายอื่นได้

3. รีจิสเตอร์ MY (16-bit Source Address) ใช้กำหนดแอดเดรส 16 บิตของแต่ละโมดูล เลือกค่าได้ตั้งแต่ 0x0000 ถึง 0xFFFD และสามารถยกเลิกแอดเดรส 16 บิตนี้เพื่อไปใช้แอดเดรสขนาด 64 บิตที่รีจิสเตอร์ SH และ SL แทนได้ เพื่อขยายให้มีจำนวนโมดูลลูกข่ายได้มากขึ้น โดยกำหนด MY เป็น 0xFFFE และ 0xFFFF

4. รีจิสเตอร์ SH และ SL (Serial Number High / Low) เป็นรีจิสเตอร์เก็บค่าหมายเลขเฉพาะหรือ Serial number ของแต่ละโมดูล สามารถใช้เป็นแอดเดรส 64 บิต (SH รวมกับ SL) โดยต้องยกเลิกแอดเดรส 16 บิตที่รีจิสเตอร์ MY ก่อน ค่าในรีจิสเตอร์ SH และ SL ไม่สามารถเปลี่ยนแปลงได้

5. รีจิสเตอร์ DH และ DL (Destination Address High/Low) ใช้กำหนดแอดเดรสของโมดูลตัวรับ

5.1 ถ้าโมดูลตัวรับใช้รีจิสเตอร์ MY (แอดเดรส 16 บิต) ให้กำหนดค่าของรีจิสเตอร์ DH เป็น 0x0000 และ DL เป็นค่า MY ของโมดูลตัวรับ

5.2 ถ้าโมดูลตัวรับใช้รีจิสเตอร์ SH ร่วมกับ SL (แอดเดรส 64 บิต) ให้กำหนดค่าของรีจิสเตอร์ DH เป็นค่าของ SH และค่าของรีจิสเตอร์ DL เป็นค่าของ SL ของโมดูลตัวรับ

การตั้งค่าของโมดูล XBee-PRO ทำได้ 2 ทางคือ ใช้โปรแกรม X-CTU กับบอร์ด ZX-XBee ต่อกับคอมพิวเตอร์ทางพอร์ตอนุกรม อีกทางหนึ่งคือใช้ AT Command ซึ่งสามารถดูคำสั่งต่างๆ เพิ่มเติมได้จากไฟล์ในแผ่นซีดีรอม

ก่อนการใช้โมดูล XBee PRO ควรจะทำการตรวจสอบ ตั้งค่าแอดเดรสต่างๆ และรูปแบบของ การส่งข้อมูลก่อนนำไปติดตั้งกับบอร์ดไมโครคอนโทรลเลอร์

ภาคผนวก ข

รหัสต้นฉบับเพื่อสร้างส่วนต่อประสานกราฟิกกับผู้ใช้โดยใช้วิชวลเบสิก 2010

```
Imports System 'นำเข้า System
Imports System.Threading 'นำเข้า System.Threading
Imports System.IO.Ports 'นำเข้า System.IO.Ports
Imports System.ComponentModel 'นำเข้า System.ComponentModel
Imports System.Collections.Generic 'นำเข้า System.Collections.Generic
Imports System.Data 'นำเข้า System.Data
Imports System.Drawing 'นำเข้า System.Drawing
Imports System.Text 'นำเข้า System.Text
Imports System.Windows.Forms 'นำเข้า System.Windows.Forms

Public Class Form1
    'ประกาศตัวแปร และชนิดตัวแปร
    Dim myPort As Array
    Dim cntRx As Integer = 0
    Dim cntTx As Integer = 0
    Dim Rx_Buff(10) As Byte
    Dim Flag_Read As Byte
    Dim Receive_Data As String
    Dim Tx_Data(10) As Byte
    Dim command(10) As Byte
    Dim Buffer(5) As String
    Dim level(5) As Integer
    Dim Datacm As Double
    Dim Datacm2 As Double
    Dim Datam As Double
    Dim Flag_level As Integer = 0
    Dim Datab As Integer
    Dim Datab1 As Integer
    Dim Datab2 As Integer
    Dim Datac As Integer
    Dim Datacm3(4) As String
    Dim SelectData As String
```

```
Dim mincm As Byte
Dim minm As Byte
Dim SelectData2 As String
Dim SelectData3 As String
Dim SelectData4 As String
Dim maxcm As Integer
Dim maxm As Double
Dim sensorcm1 As String
Dim sensorcm2 As Byte
Dim sensorm1 As String
Dim sensorm2 As Byte

'เมื่อโหลดแบบฟอร์ม 1 ขึ้นมา
Private Sub Form1_Load_1(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    TextBox5.Visible = False 'TextBox5 เป็น False
    TextBox6.Visible = False 'TextBox6 เป็น False
    TextBox3.Visible = False 'TextBox3 เป็น False
    TextBox4.Visible = False 'TextBox4 เป็น False
    Chart2.Visible = True 'Chart2 จะเป็น True
    Chart1.Visible = True 'Chart1 จะเป็น True
    ComboBox1.Text = "cm." 'Combobox1 ขึ้นข้อความ cm.

    If ComboBox1.Text = "cm." Then 'ถ้า Combobox1 ขึ้นข้อความ cm.
       DomainUpDown1.Text = "0"  'DomainUpDown1 ขึ้นข้อความ 0
       DomainUpDown2.Text = "0"  'DomainUpDown2 ขึ้นข้อความ 0
    End If 'จบการทำงาน IF
    If ComboBox1.Text = "m." Then 'ถ้า Combobox1 ขึ้นข้อความ cm.
       DomainUpDown3.Text = "0" 'DomainUpDown1 ขึ้นข้อความ 0
       DomainUpDown4.Text = "0" 'DomainUpDown1 ขึ้นข้อความ 0
    End If 'จบการทำงาน IF
```

```
ComboBox3.Text = "1" 'ComboBox3 ขึ้นข้อความ 1
ComboBox2.Text = "100" 'ComboBox2 ขึ้นข้อความ 100
DomainUpDown1.Text = "0" 'DomainUpDown1 ขึ้นข้อความ 0
DomainUpDown2.Text = "0"
Chart2.Series("Water").Points.Clear() 'chart2 เคลียร์ค่าในกราฟแท่ง
Chart1.Series("Water").Points.Clear() 'chart1 เคลียร์ค่าในกราฟแท่ง
'ComboBox2 เพิ่มข้อมูล 60 และ 100
ComboBox2.Items.Add(100)
ComboBox2.Items.Add(60)
'ComboBox3 เพิ่มข้อมูล 5 และ 10
ComboBox3.Items.Add(5)
ComboBox3.Items.Add(10)
'DomainUpDown1 เพิ่ข้อมูล 10 20 30 40 และ 50
DomainUpDown1.Items.Add(10)
DomainUpDown1.Items.Add(20)
DomainUpDown1.Items.Add(30)
DomainUpDown1.Items.Add(40)
DomainUpDown1.Items.Add(50)
'DomainUpDown- เพิ่ข้อมูล 1 2 3 4 และ 5
DomainUpDown3.Items.Add(1)
DomainUpDown3.Items.Add(2)
DomainUpDown3.Items.Add(3)
DomainUpDown3.Items.Add(4)
DomainUpDown3.Items.Add(5)
'DomainUpDown1 เพิ่ข้อมูล 60 70 80 และ 90
DomainUpDown2.Items.Add(60)
DomainUpDown2.Items.Add(70)
DomainUpDown2.Items.Add(80)
DomainUpDown2.Items.Add(90)
'DomainUpDown4 เพิ่ข้อมูล 6 7 8 และ 9
DomainUpDown4.Items.Add(6)
DomainUpDown4.Items.Add(7)
```

```
DomainUpDown4.Items.Add(8)
DomainUpDown4.Items.Add(9)
'ComboBox1 เพิ่มข้อมูลเป็น "cm." และ"m."
ComboBox1.Items.Add("cm.")
ComboBox1.Items.Add("m.")
ComboBox1.Visible = True 'ComboBox1 เป็นจริง
ComboBox1.Items.Add(" ") 'ComboBox1 เพิ่มข้อมูล " '
myPort = IO.Ports.SerialPort.GetPortNames() 'myPort= ทุกComPort ที่หาเจอ
ComPortcbb.Items.AddRange(myPort) 'ComPortcbb ของ myPort
myPort = IO.Ports.SerialPort.GetPortNames()
'BaudRatecbb เพิ่มข้อมูลBaudRate 9600 19200 38400 57600 และ115200
BaudRatecbb.Items.Add(9600)
BaudRatecbb.Items.Add(19200)
BaudRatecbb.Items.Add(38400)
BaudRatecbb.Items.Add(57600)
BaudRatecbb.Items.Add(115200)
'FlowControlcbb เพิ่มข้อมูล NONE XOnXoOff และHARDWARE
FlowControlcbb.Text = "NONE"
FlowControlcbb.Items.Add("NONE")
FlowControlcbb.Items.Add("XOnXoOff")
FlowControlcbb.Items.Add("HARDWARE")
'DataBitscbb เพิ่มข้อมูล 4 5 6 7 และ8
DataBitscbb.Items.Add(4)
DataBitscbb.Items.Add(5)
DataBitscbb.Items.Add(6)
DataBitscbb.Items.Add(7)
DataBitscbb.Items.Add(8)
'Paritycbb พิ่มข้อมูล NONE ODD EVEN MARK และ SPACE
Paritycbb.Text = "NONE"
Paritycbb.Items.Add("NONE")
Paritycbb.Items.Add("ODD")
Paritycbb.Items.Add("EVEN")
```

```
        Paritycbb.Items.Add("MARK")
        Paritycbb.Items.Add("SPACE")
        'StopBitscbb เพิ่มขอ้มูล 1 1.5 และ 2
        StopBitscbb.Items.Add(1)
        StopBitscbb.Items.Add(1.5)
        StopBitscbb.Items.Add(2)
        ' ให้เริ่มต้นที่ i=0 โดย ComPort BaudRate DataBits และStopBitscbb จะเริ่มต้นจากค่าที่น้อย
ที่สุด

        For i = 0 To UBound(myPort)
            ComPortcbb.Items.Add(myPort(i))
        Next
        ComPortcbb.Text = ComPortcbb.Items.Item(0)
        BaudRatecbb.Text = BaudRatecbb.Items.Item(0)
        DataBitscbb.Text = DataBitscbb.Items.Item(0)
        StopBitscbb.Text = StopBitscbb.Items.Item(0)
        If ComboBox1.Text = "cm." Then 'ถ้า ComboBox1มีข้อความของ "cm." แล้ว
            sensorcm1 = ComboBox2.Text 'sensorm1 = ข้อความของ ComboBox3
            sensorcm2 = sensorcm1 'sensorcm1 จะถูกเก็บในsensorcm2 ในรูปตัวแปร Byte
        End If ' จบการทำงาน IF
        If sensorm1 = ComboBox3.Text Then 'ถ้า sensorm1 = ข้อความของ ComboBox3 แล้ว
            sensorm2 = sensorcm1 'sensorcm1 จะถูกเก็บในsensorm2 ในรูปตัวแปร Byte
        End If ' จบการทำงาน IF
        Chart2.Series("Water").Points.AddXY("  ", sensorcm2)
        'เริ่มต้นการฟแท่ง 1 Addข้อมูลแกนy=sensorcm2
        Chart1.Series("Water").Points.AddXY("  ", sensorm2)
        'เริ่มต้นการฟแท่ง 2 Addข้อมูลแกนy=sensorm2
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Button1.Text = "Connect" 'บน Button1 จะขึ้นข้อความ Connect
```

```
Dim btn As String = Button1.Text
' ประกาศ btn ตัวแปรชนิด string และ btn= ข้อความของ Button1
Select Case btn ' เลือก Case ของ btn
  Case "Connect" 'ถ้า btn ="Connect"
    TextBox2.Text = "OFF" 'TextBox2 จะขึ้นข้อความ "OFF"
    OvalShape1.BackgroundImage = PROJECT.My.Resources.Resources._1
    ' แสดงภาพจากไฟล์ Project_EE.My.Resources.Resources._1
    OvalShape2.BackgroundImage = PROJECT.My.Resources.Resources._2
    Timer1.Enabled = True 'Timer1 ทำงาน
    Timer1.Interval = 800 'Timer1 ทำงานทุก 800 mS
    Timer2.Enabled = True 'Timer2 ทำงาน
    Timer2.Interval = 1   'Timer2 ทำงานทุก 1 mS
    Timer3.Enabled = True 'Timer3 ทำงาน
    Timer3.Interval = 1800 'Timer3 ทำงานทุก 1800 mS
    Timer4.Enabled = False  'Timer4 ทำงาน
    Timer4.Interval = 2800 'Timer4 ทำงานทุก 2800 mS
    ComboBox1.Text = "cm." 'ComboBox1 ขึ้นข้อความเป็น"cm."
    Button1.Text = "Disconnect" 'Button1 ขึ้นข้อความ "Disconnect"
    SerialPort1.PortName = ComPortcbb.Text
    'SerialPort1.PortName = ComPort ที่ผู้ใช้เลือก
    SerialPort1.BaudRate = BaudRatecbb.Text
     'SerialPort1.BaudRate = BaudRate ที่ผู้ใช้เลือก
    SerialPort1.Handshake = IO.Ports.Handshake.None 'SerialPort1.Handshake=None
    SerialPort1.DataBits = DataBitscbb.Text 'SerialPort1.DataBits = DataBit ที่ผู้ใช้เลือก
    SerialPort1.Parity = IO.Ports.Parity.None ' SerialPort1.Parity = None
    SerialPort1.StopBits = StopBitscbb.Text 'SerialPort1.StopBits = StopBits ที่ผู้ใช้เลือก
    SerialPort1.Open() ' เปิด  SerialPort1

  Case "Disconnect" ' ถ้า cbb="Disconnect"
    Button1.Text = "Connect" 'Button1ขึ้นข้อความ"Connect"
    SerialPort1.Close() ' ปิด  SerialPort1
    OvalShape2.BackgroundImage = PROJECT.My.Resources.Resources._1
```

```
    End Select ' จบการทำงาน Select
End Sub

Sub SendSerialData(ByVal data As String)
    Timer1.Enabled = True
    SerialPort1.Write(data)
End Sub

Private Sub Startbtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Startbtn.Click
    Timer1.Enabled = False  'Timer 1 หยุดทำงาน
    Timer4.Enabled = True
    ComboBox1.Text = "cm."   'ComboBox1ขึ้นข้อความ cm.
    Stopbtn.Enabled = True 'ปุ่ม OFF เป็น True
    Tx_Data(0) = 80  'ข้อมูลตัวที่1= "P"
    Tx_Data(1) = 66  'ข้อมูลตัวที่2= "B"
    Tx_Data(2) = 48  'ข้อมูลตัวที่3= "0"
    Tx_Data(3) = 48  'ข้อมูลตัวที่4= "0"
    Tx_Data(4) = 48  'ข้อมูลตัวที่5= "0"
    Tx_Data(5) = 48  'ข้อมูลตัวที่6= "0"
    Tx_Data(6) = 48  'ข้อมูลตัวที่7= "0"
    Tx_Data(7) = 48  'ข้อมูลตัวที่8= "0"
    Tx_Data(8) = 48  'ข้อมูลตัวที่9= "0"
    Tx_Data(9) = 68  'ข้อมูลตัวที่10= "D"
    Tx_Data(9) = 68
    SerialPort1.Write(Tx_Data, 0, 10) ' ส่งข้อมูลทั้ง 10 ตัว ออกSerialPort1
    Startbtn.Enabled = False 'ปุ่ม ON เป็นFalse
    If ComboBox1.Text = "cm." Then        'ถ้า ComboBox1 มีข้อควมเป็น "cm."
      SelectData2 = DomainUpDown1.Text
       'SelectData2 จะเก็บข้อความของ DomainUpDown1
    End If
```

```
    If ComboBox1.Text = "cm." Then ' ถ้า ComboBox1 มีข้อความ "cm." แล้ว
        SelectData = DomainUpDown2.Text 'SelectData จะเก็บข้อความ DomainUpDown2
    End If ' จบการทำงาน If
End Sub

Private Sub Stopbtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Stopbtn.Click
    Timer1.Enabled = False 'Timer1 หยุดทำงาน
    Timer4.Enabled = False
    Startbtn.Enabled = True ' ปุ่ม ON เป็น True
    Tx_Data(0) = 80  'ข้อมูลตัวที่ 1= "P"
    Tx_Data(1) = 65  'ข้อมูลตัวที่ 2= "B"
    Tx_Data(2) = 53  'ข้อมูลตัวที่ 3= "5"
    Tx_Data(3) = 53  'ข้อมูลตัวที่ 4= "5"
    Tx_Data(4) = 48  'ข้อมูลตัวที่ 5= "0"
    Tx_Data(5) = 48  'ข้อมูลตัวที่ 6= "0"
    Tx_Data(6) = 48  'ข้อมูลตัวที่ 7= "0"
    Tx_Data(7) = 48  'ข้อมูลตัวที่ 8= "0"
    Tx_Data(8) = 48  'ข้อมูลตัวที่ 9= "0"
    Tx_Data(9) = 68  'ข้อมูลตัวที่ 10= "D"
    SerialPort1.Write(Tx_Data, 0, 10) ' ส่งข้อมูลทั้ง 10 ตัว ออกSerialPort1
    Stopbtn.Enabled = False  ' ปุ่ม OFF เป็น False
    If ComboBox1.Text = "cm." Then      'ถ้า ComboBox1 มีข้อควมเป็น "cm."
        SelectData2    =    DomainUpDown1.Text          'SelectData2    จะเก็บข้อความของ
DomainUpDown1
    End If
    If ComboBox1.Text = "cm." Then ' ถ้า ComboBox1 มีข้อความ "cm." แล้ว
        SelectData = DomainUpDown2.Text 'SelectData จะเก็บข้อความ DomainUpDown2
    End If ' จบการทำงาน If
End Sub
```

```
' การรับข้อมูลผ่าน PortRS232
Private Sub PortRS232_DataReceive(ByVal sender As Object, ByVal e As System.IO.Ports.SerialDataReceivedEventArgs) Handles SerialPort1.DataReceived
    'ขณะที่ข้อมูลถูกอ่านหมด
    System.Threading.Thread.Sleep(1)
    Receive_Data = SerialPort1.ReadByte 'อ่านข้อมูลที่รับเข้ามาในSerialPort1.จะถูกเก็บใน Receive_Data
    Flag_Read = 1 'ให้ Flag_Read = 1
    Rx_Buff(cntRx) = Receive_Data ' ข้อมูลในReceive_Data จะถูกเก็บใน Rx_Buff(cntRx)
    cntRx += 1 ' ให้ cntRx นับเพิ่มขึ้นทีละ 1
End Sub ' จบการทำงานย่อย

Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick
    Tx_Data(0) = 80 'ข้อมูลตัวที่ 1= "P"
    Tx_Data(1) = 65 'ข้อมูลตัวที่ 2= "A"
    Tx_Data(2) = 48 'ข้อมูลตัวที่ 3= "0"
    Tx_Data(3) = 48 'ข้อมูลตัวที่ 4= "0"
    Tx_Data(4) = 48 'ข้อมูลตัวที่ 5= "0"
    Tx_Data(5) = 48 'ข้อมูลตัวที่ 6= "0"
    Tx_Data(6) = 48 'ข้อมูลตัวที่ 7= "0"
    Tx_Data(7) = 48 'ข้อมูลตัวที่ 8= "0"
    Tx_Data(8) = 48 'ข้อมูลตัวที่ 9= "0"
    Tx_Data(9) = 68 'ข้อมูลตัวที่ 10= "D"
    SerialPort1.Write(Tx_Data, 0, 10) ' ส่งข้อมูลทั้ง 10 ตัว ออกSerialPort1
End Sub

Private Sub Timer2_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer2.Tick
    If ComboBox1.Text = "cm." Then ' ถ้า ComboBox1 ขึ้นข้อความ cm. แล้ว
        ComboBox2.Visible = True 'ComboBox2 เป็น True
        ComboBox3.Visible = False 'ComboBox3 เป็น False
```

```
        DomainUpDown1.Visible = True 'DomainUpDown1 เป็น True
        DomainUpDown2.Visible = True 'DomainUpDown2 เป็น True
        DomainUpDown3.Visible = False 'DomainUpDown3 เป็น False
        DomainUpDown4.Visible = False 'DomainUpDown4 เป็น False
    End If ' จบการทำงาน
    If ComboBox1.Text = "m." Then ' ถ้า ComboBox1 ขึ้นข้อความ m. แล้ว
        ComboBox2.Visible = False 'ComboBox2 เป็น False
        ComboBox3.Visible = True  'ComboBox3 เป็น True
        DomainUpDown1.Visible = False 'DomainUpDown1 เป็น False
        DomainUpDown2.Visible = False 'DomainUpDown2 เป็น False
        DomainUpDown3.Visible = True 'DomainUpDown3 เป็น True
        DomainUpDown4.Visible = True 'DomainUpDown4 เป็น True
    End If ' จบการทำงาน If
    If Flag_Read = 1 Then 'If Flag_Read = 1 แล้ว
        If cntRx > 9 Then cntRx = 0 'ถ้า cntRx นับจำนวน > 9 แล้ว cntRx = 0
        Flag_Read = 0 ' Flag_Read = 0
        '" Me.Text = Me.Text + Chr(Rx_Buff(cntRx))
    End If 'จบการทำงาน IF
    If Rx_Buff(1) = 66 Then 'ถ้า ข้อมูลตัวที่ 2 ="B" แล้ว
        Timer1.Enabled = True ' Timer1 จะทำงาน
    End If ' จบการทำงาน

    If Rx_Buff(0) = 68 And Rx_Buff(1) = 65 And Rx_Buff(2) = 48 And Rx_Buff(3) = 48 And
Rx_Buff(9) = 80 Then
        'ถ้าข้อมูลตัวที่ 1="D" และข้อมูลตัวที่2="A" และ ข้อมูลตัวที่ 3="0" และข้อมูลตัวที่4="0"
และ ข้อมูลตัวที่ 10 ="P" แล้ว
        Flag_level = 1 'Flag_level = 1
        Startbtn.Enabled = False 'Startbtn เป็น False
        Stopbtn.Enabled = True 'Stopbtn เป็น True
        DomainUpDown1.Visible = False 'DomainUpDown1 เป็น False
        DomainUpDown2.Visible = False 'DomainUpDown2 เป็น False
```

```
        TextBox3.Text = DomainUpDown1.Text 'ข้อความใน TextBox3 = ข้อความใน DomainUpDown1
        TextBox4.Text = DomainUpDown2.Text 'ข้อความใน TextBox4 = ข้อความใน DomainUpDown2
        TextBox3.Visible = True 'TextBox3 เป็น True
        TextBox4.Visible = True 'TextBox4 เป็น True
        ComboBox1.Visible = False 'ComboBox1 เป็น False
        ComboBox2.Visible = False 'ComboBox2 เป็น False
        TextBox5.Text = ComboBox1.Text 'ข้อความใน TextBox5 = ข้อความใน ComboBox1
        TextBox6.Text = ComboBox2.Text 'ข้อความใน TextBox6 = ข้อความใน ComboBox2
        TextBox5.Visible = True 'TextBox5 เป็น True
        TextBox6.Visible = True 'TextBox6 เป็น True
    End If ' จบการทำงาน if

    If Rx_Buff(0) = 68 And Rx_Buff(1) = 65 And Rx_Buff(2) = 53 And Rx_Buff(3) = 53 And Rx_Buff(9) = 80 Then
        'ถ้าข้อมูลตัวที่ 1="D" และข้อมูลตัวที่2="A" และ ข้อมูลตัวที่ 3="5" และข้อมูลตัวที่4="5" และ ข้อมูลตัวที่ 10 ="P" แล้ว
        Flag_level = 0 'Flag_level = 0
        Startbtn.Enabled = True 'Startbtn เป็น True
        Stopbtn.Enabled = False 'Stopbtn เป็น False
        DomainUpDown1.Visible = True 'DomainUpDown1 เป็น True
        DomainUpDown2.Visible = True 'DomainUpDown2 เป็น False
        TextBox3.Visible = False 'TextBox3 เป็น False
        TextBox4.Visible = False 'TextBox4 เป็น False
        ComboBox1.Visible = True 'ComboBox1 เป็น True
        ComboBox2.Visible = True 'ComboBox2 เป็น True
        TextBox5.Visible = False 'TextBox5 เป็น False
        TextBox6.Visible = False 'TextBox6 เป็น False
        '"TextBox1.Text = Chr(Rx_Buff(4)) & Chr(Rx_Buff(5)) & Chr(Rx_Buff(6)) & Chr(Rx_Buff(7)) & Chr(Rx_Buff(8))
    End If ' จบการทำงาน if
```

```
If Rx_Buff(0) = 68 And Rx_Buff(1) = 65 And Rx_Buff(9) = 80 And ComboBox1.Text =
"cm." Then
    ' 'ถ้าข้อมูลตัวที่ 1="D" และข้อมูลตัวที่2="A" และ ข้อมูลตัวที่ 10 ="P" และ ComboBox1 มี
ข้อความเป็น "cm." แล้ว
    sensorcm1 = ComboBox2.Text 'sensorcm1 จะเก็บข้อมูลตัวอักษรของ ComboBox2
    sensorcm2 = sensorcm1 'sensorcm2 จะเก็บข้อมูลของ sensorcm1 ในรูปแบบตัวแปร Byte
    'แปลงข้อมูลที่รับมาจากตัวแปร byte เป็น integer
    Buffer(0) = Rx_Buff(5)
    level(0) = (Buffer(0) - 48) * 1000
    Buffer(1) = Rx_Buff(6)
    level(1) = (Buffer(1) - 48) * 100
    Buffer(2) = Rx_Buff(7)
    level(2) = (Buffer(2) - 48) * 10
    Buffer(3) = Rx_Buff(8)
    level(3) = (Buffer(3) - 48) * 1
    'แปลงข้อมูลจากแรงดันไฟฟ้าที่รับมาเป็นเลขฐานสิบ ให้เป็นระดับน้ำ
    Datacm = (((level(0) + level(1) + level(2) + level(3)) * sensorcm2) / 1024)
    'บวกค่าความผิดพลาดของระดับน้ำ
    Datacm2 = Datacm + (Datacm * 0.2) - 2
    'แสดงผลระดับน้ำบน TextBox1 อยู่ในรูปทศนิยม 3 ตำแหน่ง
    TextBox1.Text = Format(Datacm2, "0.00")
    'คิดค่าระดับน้ำเก็บในรูปตัวแปร integer
    Datab = ((((level(0) + level(1) + level(2) + level(3)) * sensorcm2) / 1024) * 1.2) - 2
    'Datab1 เก็บข้อมูลในรูปทศนิยม 2 ตำแหน่ง
    Datab1 = Format(Datacm, "0.00")
End If 'จบการทำงาน if

If Rx_Buff(0) = 68 And Rx_Buff(1) = 65 And Rx_Buff(9) = 80 And ComboBox1.Text =
"m." Then
    'ถ้าข้อมูลตัวที่ 1="D" และข้อมูลตัวที่ 2= "A" และข้อมูลตัวที่ 10 ="P" และ ComboBox1มี
ข้อความเป็น "m." แล้ว
    'แปลงข้อมูลที่รับมาจากตัวแปร byte เป็น integer
```

```
        sensorm1 = ComboBox3.Text * 100
        sensorm2 = sensorm1
        Buffer(0) = Rx_Buff(5)
        level(0) = (Buffer(0) - 48) * 1000
        Buffer(1) = Rx_Buff(6)
        level(1) = (Buffer(1) - 48) * 100
        Buffer(2) = Rx_Buff(7)
        level(2) = (Buffer(2) - 48) * 10
        Buffer(3) = Rx_Buff(8)
        level(3) = (Buffer(3) - 48) * 1
        'แปลงข้อมูลจากแรงดันไฟฟ้าที่รับมาเป็นเลขฐานสิบ ให้เป็นระดับน้ำ
        Datam = (level(0) + level(1) + level(2) + level(3)) * sensorm2 / 1024
        'แสดงผลระดับน้ำบน TextBox1 อยู่ในรูปทศนิยม 3 ตำแหน่ง
        TextBox1.Text = Format(Datam, "0.00")
        'คิดค่าระดับน้ำเก็บในรูปตัวแปร integer
        Datab = (level(0) + level(1) + level(2) + level(3)) * sensorm2 / 1024
        'Datab2 เก็บข้อมูลในรูปทศนิยม 2 ตำแหน่ง
        Datab2 = Format(Datam, "0.00")
    End If 'จบการทำงาน if
End Sub

Private Sub Timer3_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Timer3.Tick
    Select Case Flag_level 'เลือก Case ของ Flag_level
        Case 1 'ถ้า Flag_level =1
            OvalShape1.BackgroundImage = PROJECT.My.Resources.Resources._2
            'จะแสดงภาพที่เก็บใน Project_EE.My.Resources.Resources._2
            TextBox2.Text = "ON"
            'TextBox2 แสดงข้อความ "ON"
        Case 0 'ถ้า Flag_level =0
            OvalShape1.BackgroundImage = PROJECT.My.Resources.Resources._1
            'จะแสดงภาพที่เก็บใน Project_EE.My.Resources.Resources._1
```

```
        TextBox2.Text = "OFF"
        'TextBox2 แสดงข้อความ "OFF"
    End Select 'จบการทำงาน Select

    Dim cbb As String = ComboBox1.Text 'ประกาศตัวแปร cbb ชนิด String และcbb= ข้อความ
ใน ComboBox1
    Select Case cbb 'เลือก Case ของ cbb
      Case "cm." ' ถ้า cbb="cm."
        Label5.Text = "cm."  'ข้อความใน Label5 = "cm."
        Chart2.Visible = True 'Chart2 เป็น True
        Chart1.Visible = False 'Chart1 เป็น False
        Chart2.Series("Water").Points(0).SetValueXY(" ", Datacm2) 'แทนที่ข้อมูล Datacm2 ลง
ไปในแกน y ของกราฟแท่ง
      Case "m." ' ถ้า cbb="m."
        Label5.Text = "m." 'ข้อความใน Label5 = "m."
        Chart2.Visible = False 'Chart2 เป็น False
        Chart1.Visible = True 'Chart1 เป็น True
        Chart1.Series("Water").Points(0).SetValueXY(" ", Datam) 'แทนที่ข้อมูล Datam ลงไป
ในแกน y ของกราฟแท่ง
    End Select 'จบการทำงาน Select

    If Startbtn.Enabled = False And Stopbtn.Enabled = True And TextBox2.Text = "OFF" Then
      'ถ้าปุ่ม ON เป็น False และ ปุ่ม OFF เป็น True และ TextBox2 ขึ้นข้อความ "OFF" แล้ว
      Timer1.Enabled = True 'Timer1 ทำงาน
      Tx_Data(0) = 80 ' ข้อมูลตัวที่ 1 ="P"
      Tx_Data(1) = 66 ' ข้อมูลตัวที่ 2 ="B"
      Tx_Data(2) = 48 ' ข้อมูลตัวที่ 3 ="0"
      Tx_Data(3) = 48 ' ข้อมูลตัวที่ 4 ="0"
      Tx_Data(4) = 48 ' ข้อมูลตัวที่ 5 ="0"
      Tx_Data(5) = 48 ' ข้อมูลตัวที่ 6 ="0"
      Tx_Data(6) = 48 ' ข้อมูลตัวที่ 7 ="0"
      Tx_Data(7) = 48 ' ข้อมูลตัวที่ 8 ="0"
```

```
        Tx_Data(8) = 48 ' ข้อมูตัวที่ 9 ="0"
        Tx_Data(9) = 68 ' ข้อมูตัวที่ 10 ="D"
        SerialPort1.Write(Tx_Data, 0, 10) ' ส่งข้อมูลทั้ง 10 ตัว ออกSerialPort1
      End If

      If Startbtn.Enabled = True And Stopbtn.Enabled = False And TextBox2.Text = "ON" Then
        ' ปุ่ม ON เป็น True และ ปุ่ม OFF เป็น False และ TextBox2 ขึ้นข้อความ "ON" แล้ว
        Timer1.Enabled = True 'Timer 1 ทำงาน
        Tx_Data(0) = 80 ' ข้อมูตัวที่ 1 ="P"
        Tx_Data(1) = 66 ' ข้อมูตัวที่ 2 ="B"
        Tx_Data(2) = 53 ' ข้อมูตัวที่ 3 ="5"
        Tx_Data(3) = 53 ' ข้อมูตัวที่ 4 ="5"
        Tx_Data(4) = 48 ' ข้อมูตัวที่ 5 ="0"
        Tx_Data(5) = 48 ' ข้อมูตัวที่ 6 ="0"
        Tx_Data(6) = 48 ' ข้อมูตัวที่ 7 ="0"
        Tx_Data(7) = 48 ' ข้อมูตัวที่ 8 ="0"
        Tx_Data(8) = 48 ' ข้อมูตัวที่ 9 ="0"
        Tx_Data(9) = 68 ' ข้อมูตัวที่ 10 ="D"
        SerialPort1.Write(Tx_Data, 0, 10) ' ส่งข้อมูลทั้ง 10 ตัว ออกSerialPort1
      End If 'จบการทำงาน If
    End Sub

    Private Sub Timer4_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Timer4.Tick
      'Timer 4 ทำงาน
      If ComboBox1.Text = "cm." Then      'ถ้า ComboBox1 มีข้อความเป็น "cm."
        SelectData2 = DomainUpDown1.Text      'SelectData2 จะเก็บข้อความของ
DomainUpDown1
      End If 'จบการทำงาน if
      If SelectData2 >= Datab1 Then
      'ถ้า SelectData2 >= Datab1 และแล้ว
        Timer1.Enabled = True ' Timer1 เป็น True
```

```
    Tx_Data(0) = 80 ' ข้อูลตัวที่ 1="P"
    Tx_Data(1) = 65 ' ข้อูลตัวที่ 2="A"
    Tx_Data(2) = 48 ' ข้อูลตัวที่ 3="0"
    Tx_Data(3) = 48 ' ข้อูลตัวที่ 4="0"
    Tx_Data(4) = 48 ' ข้อูลตัวที่ 5="0"
    Tx_Data(5) = 48 ' ข้อูลตัวที่ 6="0"
    Tx_Data(6) = 48 ' ข้อูลตัวที่ 7="0"
    Tx_Data(7) = 48 ' ข้อูลตัวที่ 8="0"
    Tx_Data(8) = 48 ' ข้อูลตัวที่ 9="0"
    Tx_Data(9) = 68 ' ข้อูลตัวที่ 10="D"
    SerialPort1.Write(Tx_Data, 0, 10) ' ส่งข้อมูลทั้ง 10 ตัว ออกSerialPort1
    Startbtn.Enabled = False 'ปุ่ม ON เป็น False
  End If 'จบการทำงาน If

  If ComboBox1.Text = "cm." Then ' ถ้า ComboBox1 มีข้อความ "cm." แล้ว
    SelectData = DomainUpDown2.Text 'SelectData จะเก็บข้อความ DomainUpDown2
  End If ' จบการทำงาน If
  If SelectData <= Datab1  Then
'ถ้า SelectData <= Datab1 แล้ว
    Timer1.Enabled = True ' Timer 1 ทำงาน
    Stopbtn.Enabled = False ' ปุ่ม OFF เป็น False
    Tx_Data(0) = 80 ' ข้อูลตัวที่ 1="P"
    Tx_Data(1) = 65 ' ข้อูลตัวที่ 2="A"
    Tx_Data(2) = 53 ' ข้อูลตัวที่ 3="5"
    Tx_Data(3) = 53 ' ข้อูลตัวที่ 4="5"
    Tx_Data(4) = 48 ' ข้อูลตัวที่ 5="0"
    Tx_Data(5) = 48 ' ข้อูลตัวที่ 6="0"
    Tx_Data(6) = 48 ' ข้อูลตัวที่ 7="0"
    Tx_Data(7) = 48 ' ข้อูลตัวที่ 8="0"
    Tx_Data(8) = 48 ' ข้อูลตัวที่ 9="0"
    Tx_Data(9) = 68 ' ข้อูลตัวที่ 10="D"
    SerialPort1.Write(Tx_Data, 0, 10) ' ส่งข้อมูลทั้ง 10 ตัว ออกSerialPort1
```

```
    Startbtn.Enabled = True  'ปุ่ม ON เป็น True
End If 'จบการทำงาน If

If SelectData3 >= Datab2 Then
'ถ้า SelectData3 >= Datab2 และ แล้ว
    Timer1.Enabled = True 'Timer 1 ทำงาน
    Stopbtn.Enabled = True 'ปุ่ม OFF เป็น True
    Tx_Data(0) = 80 ' ข้อมูลตัวที่ 1 ="P"
    Tx_Data(1) = 65 ' ข้อมูลตัวที่ 2 ="A"
    Tx_Data(2) = 48 ' ข้อมูลตัวที่ 3 ="0"
    Tx_Data(3) = 48 ' ข้อมูลตัวที่ 4 ="0"
    Tx_Data(4) = 48 ' ข้อมูลตัวที่ 5 ="0"
    Tx_Data(5) = 48 ' ข้อมูลตัวที่ 6 ="0"
    Tx_Data(6) = 48 ' ข้อมูลตัวที่ 7 ="0"
    Tx_Data(7) = 48 ' ข้อมูลตัวที่ 8 ="0"
    Tx_Data(8) = 48 ' ข้อมูลตัวที่ 9 ="0"
    Tx_Data(9) = 68 ' ข้อมูลตัวที่ 10 ="D"
    SerialPort1.Write(Tx_Data, 0, 10) ' ส่งข้อมูลทั้ง 10 ตัว ออกSerialPort1
    Startbtn.Enabled = False   'ปุ่ม ON เป็น False
End If 'จบการทำงาน If

If SelectData4 <= Datab2 Then
' ถ้า SelectData4 <= Datab2 แล้ว
    Timer1.Enabled = True 'Timer1 ทำงาน
    Stopbtn.Enabled = False ' ปุ่ม OFF เป็น False
    Tx_Data(0) = 80 ' ข้อมูลตัวที่ 1 ="P"
    Tx_Data(1) = 65 ' ข้อมูลตัวที่ 2 ="A"
    Tx_Data(2) = 53 ' ข้อมูลตัวที่ 3 ="5"
    Tx_Data(3) = 53 ' ข้อมูลตัวที่ 4 ="5"
    Tx_Data(4) = 48 ' ข้อมูลตัวที่ 5 ="0"
    Tx_Data(5) = 48 ' ข้อมูลตัวที่ 6 ="0"
    Tx_Data(6) = 48 ' ข้อมูลตัวที่ 7 ="0"
```

```
            Tx_Data(7) = 48 ' ข้อมูลตัวที่ 8 ="0"
            Tx_Data(8) = 48 ' ข้อมูลตัวที่ 9 ="0"
            Tx_Data(9) = 68 ' ข้อมูลตัวที่ 10 ="D"
            SerialPort1.Write(Tx_Data, 0, 10) ' ส่งข้อมูลทั้ง 10 ตัว ออกSerialPort1
            Startbtn.Enabled = True  'ปุ่ม ON เป็น True
        End If 'จบการทำงาน If
    End Sub ' จบการทำงานย่อย
End Class
```

ภาคผนวก ค

รายละเอียดข้อมูลของไมโครคอนโทรลเลอร์รุ่น PIC30F4011

MICROCHIP

dsPIC30F4011/4012

dsPIC30F4011/4012 Enhanced Flash 16-bit Digital Signal Controller

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *dsPIC30F Programmer's Reference Manual* (DS70030).

High Performance Modified RISC CPU:

- Modified Harvard architecture
- C compiler optimized instruction set architecture with flexible addressing modes
- 84 base instructions
- 24-bit wide instructions, 16-bit wide data path
- 48 Kbytes on-chip Flash program space (16K instruction words)
- 2 Kbytes of on-chip data RAM
- 1 Kbytes of non-volatile data EEPROM
- Up to 30 MIPs operation:
  - DC to 40 MHz external clock input
  - 4 MHz-10 MHz oscillator input with PLL active (4x, 8x, 16x)
- 30 interrupt sources
  - 3 external interrupt sources
  - 8 user selectable priority levels for each interrupt source
  - 4 processor trap sources
- 16 x 16-bit working register array

DSP Engine Features:

- Dual data fetch
- Accumulator write back for DSP operations
- Modulo and Bit-Reversed Addressing modes
- Two, 40-bit wide accumulators with optional saturation logic
- 17-bit x 17-bit single cycle hardware fractional/ integer multiplier
- All DSP instructions single cycle
- ± 16-bit single cycle shift

Peripheral Features:

- High current sink/source I/O pins: 25 mA/25 mA
- Timer module with programmable prescaler:
  - Five 16-bit timers/counters; optionally pair 16-bit timers into 32-bit timer modules
- 16-bit Capture input functions
- 16-bit Compare/PWM output functions
- 3-wire SPI™ modules (supports 4 Frame modes)
- I^2C™ module supports Multi-Master/Slave mode and 7-bit/10-bit addressing
- 2 UART modules with FIFO Buffers
- 1 CAN modules, 2.0B compliant

Motor Control PWM Module Features:

- 6 PWM output channels
  - Complementary or Independent Output modes
  - Edge and Center Aligned modes
- 3 duty cycle generators
- Dedicated time base
- Programmable output polarity
- Dead-time control for Complementary mode
- Manual output control
- Trigger for A/D conversions

Quadrature Encoder Interface Module Features:

- Phase A, Phase B and Index Pulse input
- 16-bit up/down position counter
- Count direction status
- Position Measurement (x2 and x4) mode
- Programmable digital noise filters on inputs
- Alternate 16-bit Timer/Counter mode
- Interrupt on position counter rollover/underflow

dsPIC30F4011/4012

Analog Features:

- 10-bit Analog-to-Digital Converter (A/D) with 4 S/H inputs:
  - 500 Ksps conversion rate
  - 9 input channels
  - Conversion available during Sleep and Idle
- Programmable Brown-out Detection and Reset generation

Special Microcontroller Features:

- Enhanced Flash program memory:
  - 10,000 erase/write cycle (min.) for industrial temperature range, 100K (typical)
- Data EEPROM memory:
  - 100,000 erase/write cycle (min.) for industrial temperature range, 1M (typical)
- Self-reprogrammable under software control
- Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Flexible Watchdog Timer (WDT) with on-chip low power RC oscillator for reliable operation
- Fail-Safe clock monitor operation detects clock failure and switches to on-chip low power RC oscillator
- Programmable code protection
- In-Circuit Serial Programming™ (ICSP™)
- Selectable Power Management modes
  - Sleep, Idle and Alternate Clock modes

CMOS Technology:

- Low power, high speed Flash technology
- Wide operating voltage range (2.5V to 5.5V)
- Industrial and Extended temperature ranges
- Low power consumption

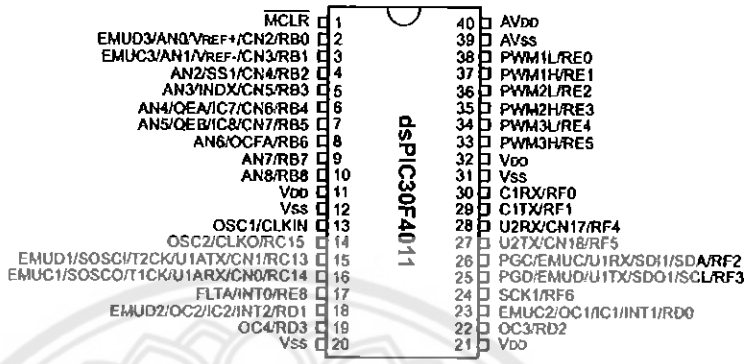
dsPIC30F Motor Control and Power Conversion Family*

Device	Pins	Program Mem. Bytes/ Instructions	SRAM Bytes	EEPROM Bytes	Timer 16-bit	Input Cap	Output Comp/Std PWM	Moto Control PWM	A/D 10-bit 500 Ksps	Quad Enc	UART	SPI™	I²C™	CAN
dsPIC30F2010	28	12K/4K	512	1024	3	4	2	6 ch	6 ch	Yes	1	1	1	-
dsPIC30F3010	28	24K/8K	1024	1024	5	4	2	6 ch	6 ch	Yes	1	1	1	-
dsPIC30F4012	28	48K/16K	2048	1024	5	4	2	6 ch	6 ch	Yes	1	1	1	1
dsPIC30F3011	40/44	24K/8K	1024	1024	5	4	4	6 ch	9 ch	Yes	2	1	1	-
dsPIC30F4011	40/44	48K/16K	2048	1024	5	4	4	6 ch	9 ch	Yes	2	1	1	1
dsPIC30F5015	64	66K/22K	2048	1024	5	4	4	8 ch	16 ch	Yes	1	2	1	1
dsPIC30F6010	80	144K/48K	8192	4096	5	8	8	8 ch	16 ch	Yes	2	2	1	2

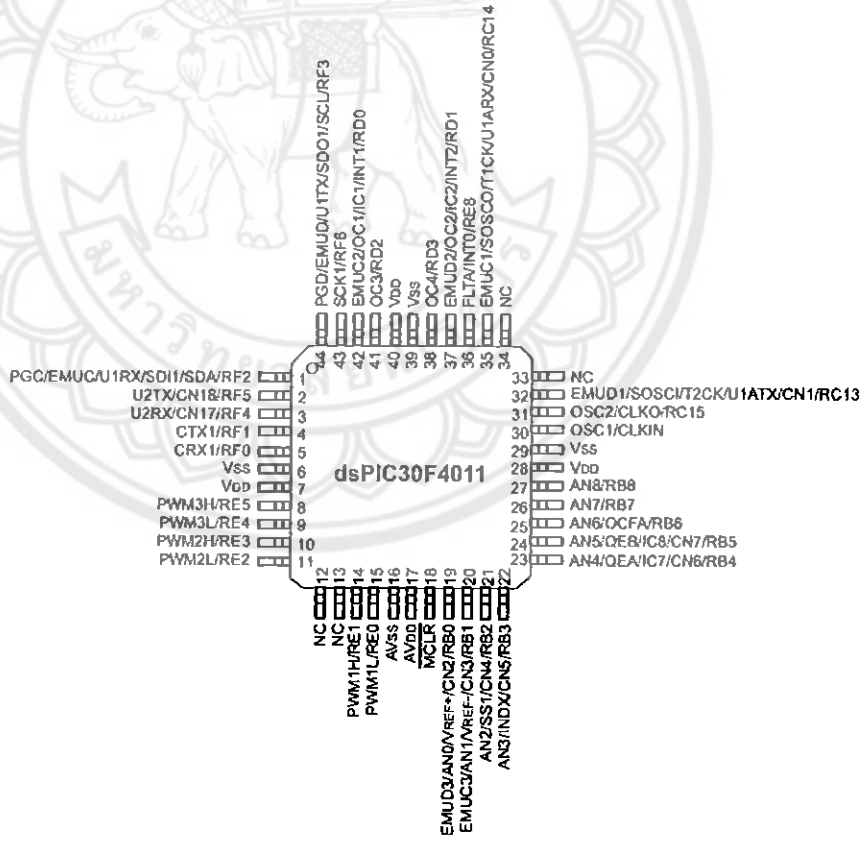
* This table provides a summary of the dsPIC30F6010 peripheral features. Other available devices in the dsPIC30F Motor Control and Power Conversion Family are shown for feature comparison.

Pin Diagrams

40-Pin PDIP



44-Pin TQFP



Pin Diagrams (Continued)

44-Pin QFN

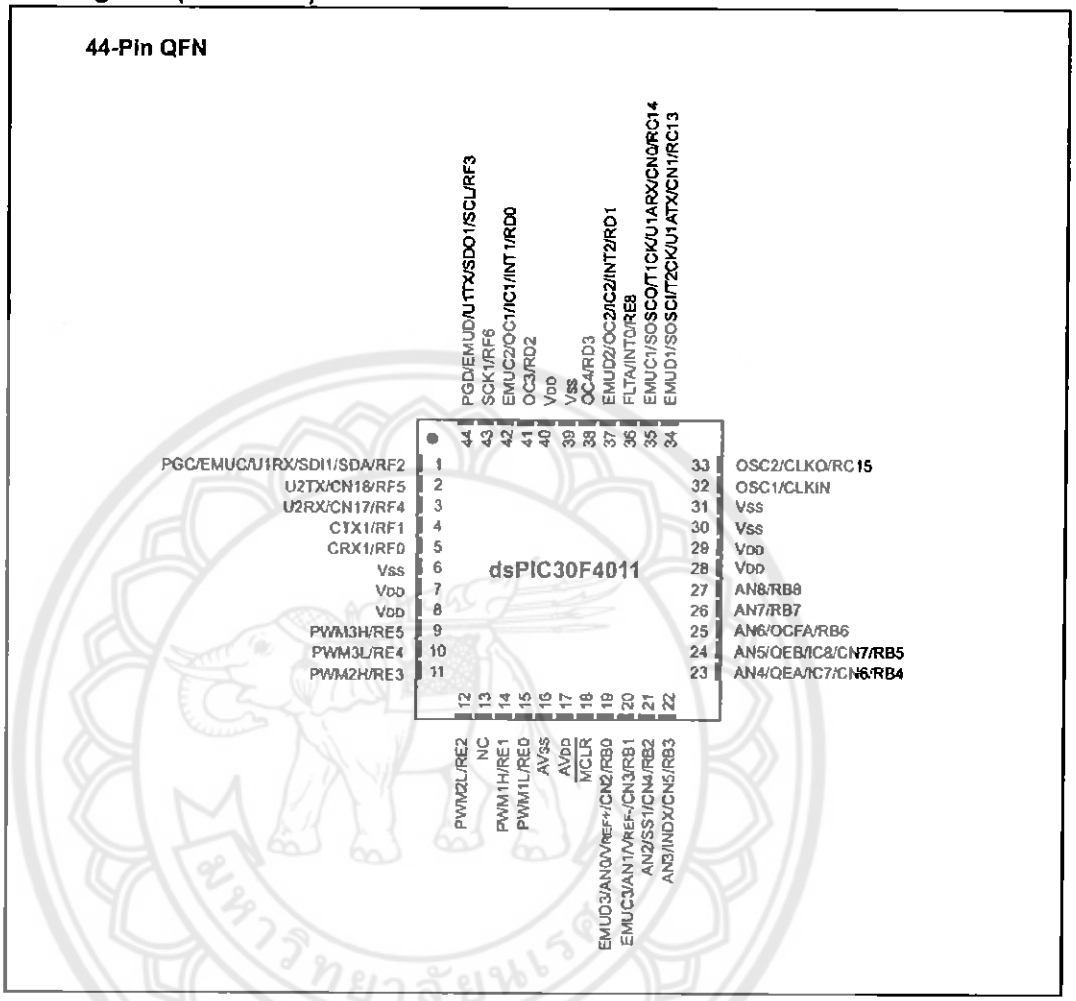


FIGURE 1-1: dsPIC30F4011 BLOCK DIAGRAM

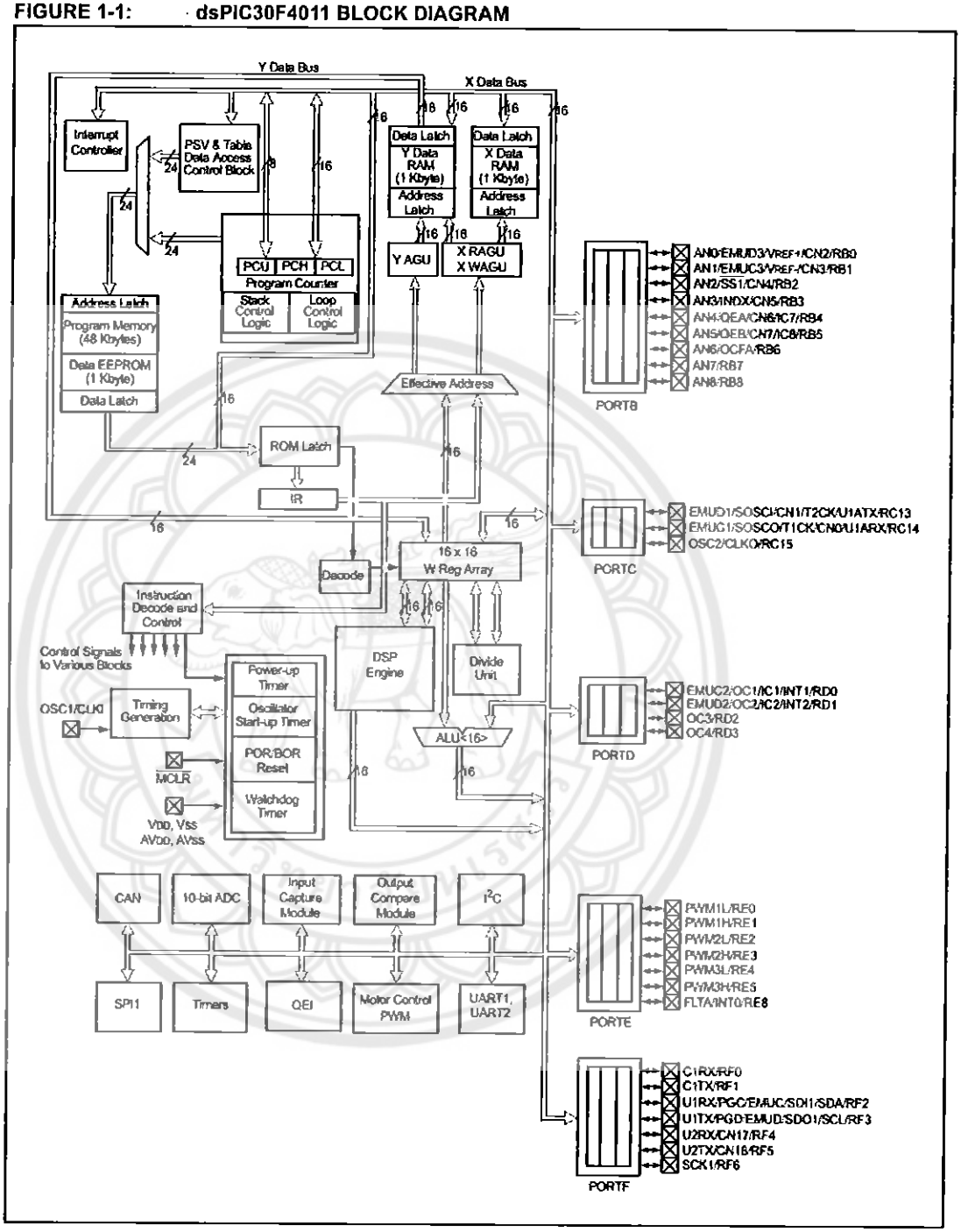


Table 1-1 provides a brief description of the device I/O pinout and the functions that are multiplexed to a port pin. Multiple functions may exist on one port pin. When multiplexing occurs, the peripheral module's functional requirements may force an override of the data direction of the port pin.

TABLE 1-1: dsPIC30F4011 I/O PIN DESCRIPTIONS

Pin Name	Pin Type	Buffer Type	Description
AN0-AN8	I	Analog	Analog input channels. AN0 and AN1 are also used for device programming data and clock inputs, respectively.
AVDD	P	P	Positive supply for analog module.
AVSS	P	P	Ground reference for analog module.
CLKI CLKO	I O	ST/CMOS —	External clock source input. Always associated with OSC1 pin function. Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. Optionally functions as CLKO in RC and EC modes. Always associated with OSC2 pin function.
CN0-CN7 CN17-CN18	I	ST	Input change notification inputs. Can be software programmed for internal weak pull-ups on all inputs.
C1RX C1TX	I O	ST —	CAN1 bus receive pin. CAN1 bus transmit pin.
EMUD EMUC EMUD1 EMUC1 EMUD2 EMUC2 EMUD3 EMUC3	I/O I/O I/O I/O I/O I/O I/O I/O	ST ST ST ST ST ST ST ST	ICD Primary Communication Channel data input/output pin. ICD Primary Communication Channel clock input/output pin. ICD Secondary Communication Channel data input/output pin. ICD Secondary Communication Channel clock input/output pin. ICD Tertiary Communication Channel data input/output pin. ICD Tertiary Communication Channel clock input/output pin. ICD Quaternary Communication Channel data input/output pin. ICD Quaternary Communication Channel clock input/output pin.
IC1, IC2, IC7, IC8	I	ST	Capture inputs 1, 2, 7 and 8.
INDX QEA QEB	I I I	ST ST ST	Quadrature Encoder Index Pulse input. Quadrature Encoder Phase A input in QEI mode. Auxiliary Timer External Clock/Gate input in Timer mode. Quadrature Encoder Phase A input in QEI mode. Auxiliary Timer External Clock/Gate input in Timer mode.
INT0 INT1 INT2	I I I	ST ST ST	External interrupt 0. External interrupt 1. External interrupt 2.
FLTA PWM1L PWM1H PWM2L PWM2H PWM3L PWM3H	I O O O O O O	ST — — — — — —	PWM Fault A input. PWM 1 Low output. PWM 1 High output. PWM 2 Low output. PWM 2 High output. PWM 3 Low output. PWM 3 High output.
$\overline{MCLR}$	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low Reset to the device.
OCFA OC1-OC4	I O	ST —	Compare Fault A input (for Compare channels 1, 2, 3 and 4). Compare outputs 1 through 4.

Legend: CMOS = CMOS compatible input or output Analog = Analog input
ST = Schmitt Trigger input with CMOS levels O = Output
I = Input P = Power

TABLE 1-1: dsPIC30F4011 I/O PIN DESCRIPTIONS (CONTINUED)

Pin Name	Pin Type	Buffer Type	Description
OSC1 OSC2	I I/O	ST/CMOS —	Oscillator crystal input. ST buffer when configured in RC mode; CMOS otherwise. Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. Optionally functions as CLKO in RC and EC modes.
PGD PGC	I/O I	ST ST	In-Circuit Serial Programming data input/output pin. In-Circuit Serial Programming clock input pin.
RB0-RB8	I/O	ST	PORTB is a bidirectional I/O port.
8RC13-RC15	8I/O	8ST	PORTC is a bidirectional I/O port.
RD0-RD3	I/O	ST	PORTD is a bidirectional I/O port.
RE0-RE5, RE8	I/O	ST	PORTE is a bidirectional I/O port.
RF0-RF6	I/O	ST	PORTF is a bidirectional I/O port.
SCK1 SDI1 SDO1 SS1	I/O I O I	ST ST — ST	Synchronous serial clock input/output for SPI™ 1. SPI 1 Data In. SPI 1 Data Out. SPI 1 Slave Synchronization.
SCL SDA	I/O I/O	ST ST	Synchronous serial clock input/output for I^2C. Synchronous serial data input/output for I^2C.
SOSCO SOSCI	O I	— ST/CMOS	32 kHz low power oscillator crystal output. 32 kHz low power oscillator crystal input. ST buffer when configured in RC mode; CMOS otherwise.
T1CK T2CK	I I	ST ST	Timer1 external clock input. Timer2 external clock input.
U1RX U1TX U1ARX U1ATX U2RX U2TX	I O I O I O	ST — ST — ST —	UART1 Receive. UART1 Transmit. UART1 Alternate Receive. UART1 Alternate Transmit. UART2 Receive. UART2 Transmit.
VDD	P	—	Positive supply for logic and I/O pins.
VSS	P	—	Ground reference for logic and I/O pins.
VREF+	I	Analog	Analog Voltage Reference (High) input.
VREF-	I	Analog	Analog Voltage Reference (Low) input.

Legend: CMOS = CMOS compatible input or output Analog = Analog input
ST = Schmitt Trigger input with CMOS levels O = Output
I = Input P = Power

18.0 UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER (UART) MODULE

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046).

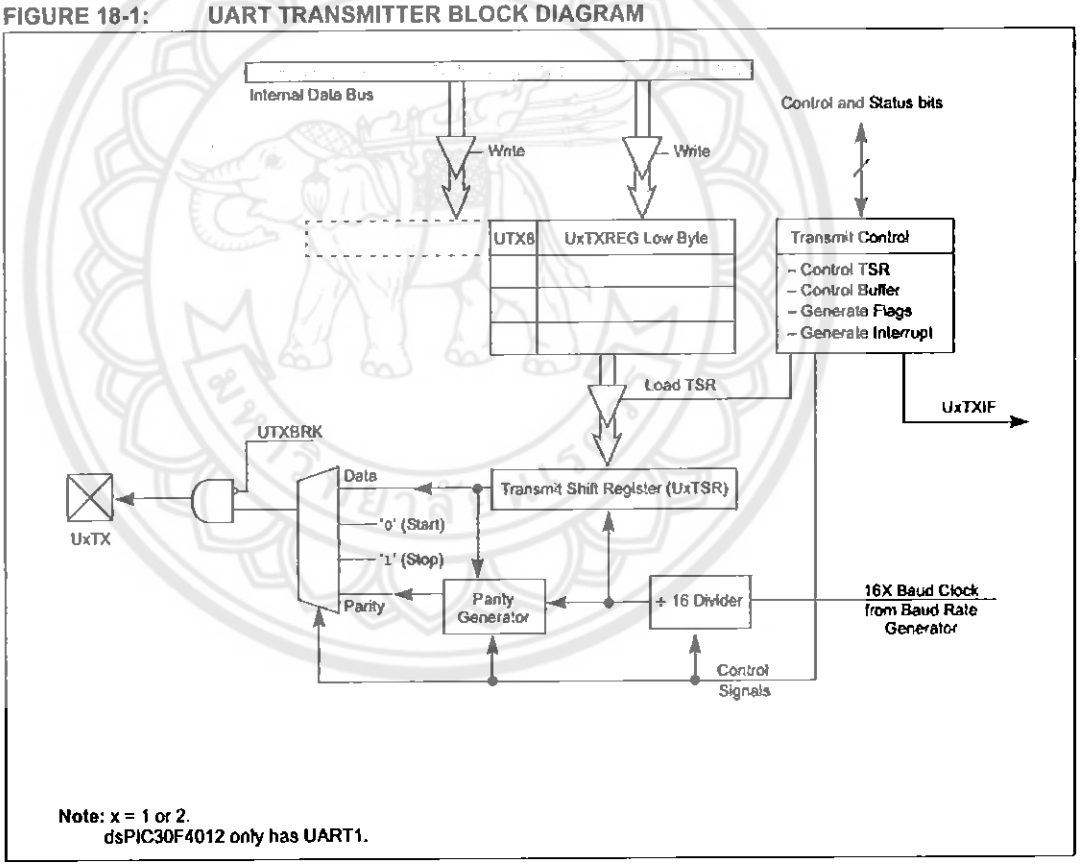
This section describes the Universal Asynchronous Receiver/Transmitter Communications module.

18.1 UART Module Overview

The key features of the UART module are:

- Full-duplex, 8 or 9-bit data communication
- Even, Odd or No Parity options (for 8-bit data)
- One or two Stop bits
- Fully integrated Baud Rate Generator with 16-bit prescaler
- Baud rates range from 38 bps to 1.875 Mbps at a 30 MHz instruction rate
- 4-word deep transmit data buffer
- 4-word deep receive data buffer
- Parity, Framing and Buffer Overrun error detection
- Support for Interrupt only on Address Detect (9th bit = 1)
- Separate Transmit and Receive Interrupts
- Loopback mode for diagnostic support

FIGURE 18-1: UART TRANSMITTER BLOCK DIAGRAM



Note: x = 1 or 2.
dsPIC30F4012 only has UART1.

18.2 Enabling and Setting Up UART

18.2.1 ENABLING THE UART

The UART module is enabled by setting the UARTEN bit in the UxMODE register (where x = 1 or 2). Once enabled, the UxTX and UxRX pins are configured as an output and an input respectively, overriding the TRIS and LATCH Register bit settings for the corresponding I/O port pins. The UxTX pin is at logic '1' when no transmission is taking place.

18.2.2 DISABLING THE UART

The UART module is disabled by clearing the UARTEN bit in the UxMODE register. This is the default state after any Reset. If the UART is disabled, all I/O pins operate as port pins under the control of the latch and TRIS bits of the corresponding port pins.

Disabling the UART module resets the buffers to empty states. Any data characters in the buffers are lost, and the baud rate counter is reset.

All error and status flags associated with the UART module are reset when the module is disabled. The URXDA, OERR, FERR, PERR, UTXEN, UTXBRK and UTXBF bits are cleared, whereas RIDLE and TRMT are set. Other control bits, including ADDEN, URXISEL<1:0>, UTXISEL, as well as the UxMODE and UxBRG registers, are not affected.

Clearing the UARTEN bit while the UART is active will abort all pending transmissions and receptions and reset the module as defined above. Re-enabling the UART will restart the UART in the same configuration.

18.2.3 ALTERNATE I/O

The alternate I/O function is enabled by setting the ALTIO bit (UxMODE<10>). If ALTIO = 1, the UxATX and UxARX pins (alternate transmit and alternate receive pins, respectively) are used by the UART module instead of the UxTX and UxRX pins. If ALTIO = 0, the UxTX and UxRX pins are used by the UART module.

18.2.4 SETTING UP DATA, PARITY AND STOP BIT SELECTIONS

Control bits PDSEL<1:0> in the UxMODE register are used to select the data length and parity used in the transmission. The data length may either be 8-bits with even, odd or no parity, or 9-bits with no parity.

The STSEL bit determines whether one or two stop bits will be used during data transmission.

The default (Power-on) setting of the UART is 8 bits, no parity, 1 stop bit (typically represented as 8, N, 1).

18.3 Transmitting Data

18.3.1 TRANSMITTING IN 8-BIT DATA MODE

The following steps must be performed in order to transmit 8-bit data:

1. Set up the UART:
   First, the data length, parity and number of stop bits must be selected. Then, the Transmit and Receive Interrupt enable and priority bits are setup in the UxMODE and UxSTA registers. Also, the appropriate baud rate value must be written to the UxBRG register.
2. Enable the UART by setting the UARTEN bit (UxMODE<15>).
3. Set the UTXEN bit (UxSTA<10>), thereby enabling a transmission.
4. Write the byte to be transmitted to the lower byte of UxTXREG. The value will be transferred to the Transmit Shift register (UxTSR) immediately and the serial bit stream will start shifting out during the next rising edge of the baud clock. Alternatively, the data byte may be written while UTXEN = 0, following which, the user may set UTXEN. This will cause the serial bit stream to begin immediately because the baud clock will start from a cleared state.
5. A Transmit interrupt will be generated depending on the value of the interrupt control bit UTXISEL (UxSTA<15>).

18.3.2 TRANSMITTING IN 9-BIT DATA MODE

The sequence of steps involved in the transmission of 9-bit data is similar to 8-bit transmission, except that a 16-bit data word (of which the upper 7 bits are always clear) must be written to the UxTXREG register.

18.3.3 TRANSMIT BUFFER (UxTXB)

The transmit buffer is 9-bits wide and 4 characters deep. Including the Transmit Shift Register (UxTSR), the user effectively has a 5-deep FIFO (First In First Out) buffer. The UTXBF Status bit (UxSTA<9>) indicates whether the transmit buffer is full.

If a user attempts to write to a full buffer, the new data will not be accepted into the FIFO, and no data shift will occur within the buffer. This enables recovery from a buffer overrun condition.

The FIFO is reset during any device Reset, but is not affected when the device enters or wakes up from a Power Saving mode.

18.3.4 TRANSMIT INTERRUPT

The transmit interrupt flag (U1TXIF or U2TXIF) is located in the corresponding interrupt flag register.

The transmitter generates an edge to set the UxTXIF bit. The condition for generating the interrupt depends on UTXISEL control bit:

a) If UTXISEL = 0, an interrupt is generated when a word is transferred from the Transmit buffer to the Transmit Shift register (UxTSR). This implies that the transmit buffer has at least one empty word.

b) If UTXISEL = 1, an interrupt is generated when a word is transferred from the Transmit buffer to the Transmit Shift register (UxTSR) and the Transmit buffer is empty.

Switching between the two interrupt modes during operation is possible and sometimes offers more flexibility.

18.3.5 TRANSMIT BREAK

Setting the UTXBRK bit (UxSTA<11>) will cause the UxTX line to be driven to logic '0'. The UTXBRK bit overrides all transmission activity. Therefore, the user should generally wait for the transmitter to be idle before setting UTXBRK.

To send a break character, the UTXBRK bit must be set by software and must remain set for a minimum of 13 baud clock cycles. The UTXBRK bit is then cleared by software to generate stop bits. The user must wait for a duration of at least one or two baud clock cycles in order to ensure a valid stop bit(s) before reloading the UxTXB or starting other transmitter activity. Transmission of a break character does not generate a transmit interrupt.

18.4 Receiving Data

18.4.1 RECEIVING IN 8-BIT OR 9-BIT DATA MODE

The following steps must be performed while receiving 8-bit or 9-bit data:

1. Set up the UART (see Section 18.3.1).
2. Enable the UART (see Section 18.3.1).
3. A receive interrupt will be generated when one or more data words have been received, depending on the receive interrupt settings specified by the URXISEL bits (UxSTA<7:6>).
4. Read the OERR bit to determine if an overrun error has occurred. The OERR bit must be reset in software.
5. Read the received data from UxRXREG. The act of reading UxRXREG will move the next word to the top of the receive FIFO, and the PERR and FERR values will be updated.

18.4.2 RECEIVE BUFFER (UxRXB)

The receive buffer is 4 words deep. Including the Receive Shift register (UxRSR), the user effectively has a 5-word deep FIFO buffer.

URXDA (UxSTA<0>) = 1 indicates that the receive buffer has data available. URXDA = 0 implies that the buffer is empty. If a user attempts to read an empty buffer, the old values in the buffer will be read and no data shift will occur within the FIFO.

The FIFO is reset during any device Reset. It is not affected when the device enters or wakes up from a Power Saving mode.

18.4.3 RECEIVE INTERRUPT

The receive interrupt flag (U1RXIF or U2RXIF) can be read from the corresponding interrupt flag register. The interrupt flag is set by an edge generated by the receiver. The condition for setting the receive interrupt flag depends on the settings specified by the URXISEL<1:0> (UxSTA<7:6>) control bits.

a) If URXISEL<1:0> = 00 or 01, an interrupt is generated every time a data word is transferred from the Receive Shift Register (UxRSR) to the Receive Buffer. There may be one or more characters in the receive buffer.

b) If URXISEL<1:0> = 10, an interrupt is generated when a word is transferred from the Receive Shift Register (UxRSR) to the Receive Buffer, which, as a result of the transfer, contains 3 characters.

c) If URXISEL<1:0> = 11, an interrupt is set when a word is transferred from the Receive Shift Register (UxRSR) to the Receive Buffer, which, as a result of the transfer, contains 4 characters (i.e., becomes full).

Switching between the Interrupt modes during operation is possible, though generally not advisable during normal operation.

18.5 Reception Error Handling

18.5.1 RECEIVE BUFFER OVERRUN ERROR (OERR BIT)

The OERR bit (UxSTA<1>) is set if all of the following conditions occur:

a) The receive buffer is full.

b) The receive shift register is full, but unable to transfer the character to the receive buffer.

c) The stop bit of the character in the UxRSR is detected, indicating that the UxRSR needs to transfer the character to the buffer.

Once OERR is set, no further data is shifted in UxRSR (until the OERR bit is cleared in software or a Reset occurs). The data held in UxRSR and UxRXREG remains valid.

18.5.2 FRAMING ERROR (FERR)

The FERR bit (UxSTA<2>) is set if a '0' is detected instead of a stop bit. If two stop bits are selected, both stop bits must be '1', otherwise FERR will be set. The read only FERR bit is buffered along with the received data. It is cleared on any Reset.

18.5.3 PARITY ERROR (PERR)

The PERR bit (UxSTA<3>) is set if the parity of the received word is incorrect. This error bit is applicable only if a Parity mode (odd or even) is selected. The read only PERR bit is buffered along with the received data bytes. It is cleared on any Reset.

18.5.4 IDLE STATUS

When the receiver is active (i.e., between the initial detection of the start bit and the completion of the stop bit), the RIDLE bit (UxSTA<4>) is '0'. Between the completion of the stop bit and detection of the next start bit, the RIDLE bit is '1', indicating that the UART is Idle.

18.5.5 RECEIVE BREAK

The receiver will count and expect a certain number of bit times based on the values programmed in the PDSEL (UxMODE<2:1>) and STSEL (UxMODE<0>) bits.

If the break is longer than 13 bit times, the reception is considered complete after the number of bit times specified by PDSEL and STSEL. The URXDA bit is set, FERR is set, zeros are loaded into the receive FIFO, interrupts are generated, if appropriate and the RIDLE bit is set.

When the module receives a long break signal and the receiver has detected the start bit, the data bits and the invalid stop bit (which sets the FERR), the receiver must wait for a valid stop bit before looking for the next start bit. It cannot assume that the break condition on the line is the next start bit.

Break is regarded as a character containing all 0's, with the FERR bit set. The break character is loaded into the buffer. No further reception can occur until a stop bit is received. Note that RIDLE goes high when the stop bit has not been received yet.

18.6 Address Detect Mode

Setting the ADDEN bit (UxSTA<5>) enables this special mode, in which a 9th bit (URX8) value of '1' identifies the received word as an address rather than data. This mode is only applicable for 9-bit data communication. The URXISEL control bit does not have any impact on interrupt generation in this mode, since an interrupt (if enabled) will be generated every time the received word has the 9th bit set.

18.7 Loopback Mode

Setting the LPBACK bit enables this special mode in which the UxTX pin is internally connected to the UxRX pin. When configured for the loopback mode, the UxRX pin is disconnected from the internal UART receive logic. However, the UxTX pin still functions as in a normal operation.

To select this mode:

a) Configure UART for desired mode of operation.
b) Set LPBACK = 1 to enable Loopback mode.
c) Enable transmission as defined in Section 18.3.

18.8 Baud Rate Generator

The UART has a 16-bit baud rate generator to allow maximum flexibility in baud rate generation. The baud rate generator register (UxBRG) is readable and writable. The baud rate is computed as follows:

BRG = 16-bit value held in UxBRG register (0 through 65535)

FCY = Instruction Clock Rate (1/TCY)

The Baud Rate is given by Equation 18-1.

EQUATION 18-1: BAUD RATE

$$\text{Baud Rate} = F_{CY} / (16*(BRG+1))$$

Therefore, maximum baud rate possible is

FCY /16 (if BRG = 0),

and the minimum baud rate possible is

FCY / (16* 65536).

With a full 16-bit baud rate generator, at 30 MIPs operation, the minimum baud rate achievable is 28.5 bps.

18.9 Auto Baud Support

To allow the system to determine baud rates of received characters, the input can be optionally linked to a selected capture input (IC1 for UART1, IC2 for UART2). To enable this mode, the user must program the input capture module to detect the falling and rising edges of the start bit.

18.10 UART Operation During CPU Sleep and Idle Modes

18.10.1 UART OPERATION DURING CPU SLEEP MODE

When the device enters Sleep mode, all clock sources to the module are shutdown and stay at logic '0'. If entry into Sleep mode occurs while a transmission is in progress, then the transmission is aborted. The UxTX pin is driven to logic '1'. Similarly, if entry into Sleep mode occurs while a reception is in progress, then the reception is aborted. The UxSTA, UxMODE, transmit and receive registers and buffers, and the UxBRG register are not affected by Sleep mode.

If the Wake bit (UxMODE<7>) is set before the device enters Sleep mode, then a falling edge on the UxRX pin will generate a receive interrupt. The Receive Interrupt Select Mode bit (URXISEL) has no effect for this function. If the receive interrupt is enabled, then this will wake-up the device from Sleep. The UARTEN bit must be set in order to generate a wake-up interrupt.

18.10.2 UART OPERATION DURING CPU IDLE MODE

For the UART, the USIDL bit selects if the module will stop operation when the device enters Idle mode, or whether the module will continue on idle. If USIDL = 0, the module will continue operation during Idle mode. If USIDL = 1, the module will stop on Idle.

20.0 10-BIT HIGH SPEED ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046).

The10-bit high-speed analog-to-digital converter (A/D) allows conversion of an analog input signal to a 10-bit digital number. This module is based on a Successive Approximation Register (SAR) architecture, and provides a maximum sampling rate of 500 ksps. The A/D module has 16 analog inputs which are multiplexed into four sample and hold amplifiers. The output of the sample and hold is the input into the converter, which generates the result. The analog reference voltages are software selectable to either the device supply voltage (AVDD/AVSS) or the voltage level on the (VREF+/VREF-) pin. The A/D converter has a unique feature of being able to operate while the device is in Sleep mode.

The A/D module has six 16-bit registers:

- A/D Control Register1 (ADCON1)
- A/D Control Register2 (ADCON2)
- A/D Control Register3 (ADCON3)
- A/D Input Select Register (ADCHS)
- A/D Port Configuration Register (ADPCFG)
- A/D Input Scan Selection Register (ADCSSL)

The ADCON1, ADCON2 and ADCON3 registers control the operation of the A/D module. The ADCHS register selects the input channels to be converted. The ADPCFG register configures the port pins as analog inputs or as digital I/O. The ADCSSL register selects inputs for scanning.

Note: The SSRC<2:0>, ASAM, SIMSAM, SMPI<3:0>, BUFM and ALTS bits, as well as the ADCON3 and ADCSSL registers, must not be written to while ADON = 1. This would lead to indeterminate results.

The block diagram of the A/D module is shown in Figure 20-1.

20.1 A/D Result Buffer

The module contains a 16-word dual port read-only buffer, called ADCBUF0...ADCBUFF, to buffer the A/D results. The RAM is 10-bits wide, but is read into different format 16-bit words. The contents of the sixteen A/D conversion result buffer registers, ADCBUF0 through ADCBUFF, cannot be written by user software.

20.2 Conversion Operation

After the A/D module has been configured, the sample acquisition is started by setting the SAMP bit. Various sources, such as a programmable bit, timer time-outs and external events, will terminate acquisition and start a conversion. When the A/D conversion is complete, the result is loaded into ADCBUF0...ADCBUFF, and the A/D interrupt flag ADIF and the DONE bit are set after the number of samples specified by the SMPI bit.

The following steps should be followed for doing an A/D conversion:

1. Configure the A/D module:
   - Configure analog pins, voltage reference and digital I/O
   - Select A/D input channels
   - Select A/D conversion clock
   - Select A/D conversion trigger
   - Turn on A/D module
2. Configure A/D interrupt (if required):
   - Clear ADIF bit
   - Select A/D interrupt priority
3. Start sampling.
4. Wait the required acquisition time.
5. Trigger acquisition end, start conversion
6. Wait for A/D conversion to complete, by either:
   - Waiting for the A/D interrupt
7. Read A/D result buffer, clear ADIF if required.

20.3 Selecting the Conversion Sequence

Several groups of control bits select the sequence in which the A/D connects inputs to the sample/hold channels, converts channels, writes the buffer memory, and generates interrupts. The sequence is controlled by the sampling clocks.

The SIMSAM bit controls the acquire/convert sequence for multiple channels. If the SIMSAM bit is '0', the two or four selected channels are acquired and converted sequentially, with two or four sample clocks. If the SIMSAM bit is '1', two or four selected channels are acquired simultaneously, with one sample clock. The channels are then converted sequentially. Obviously, if there is only 1 channel selected, the SIMSAM bit is not applicable.

The CHPS bits selects how many channels are sampled. This can vary from 1, 2 or 4 channels. If CHPS selects 1 channel, the CH0 channel will be sampled at the sample clock and converted. The result is stored in the buffer. If CHPS selects 2 channels, the CH0 and CH1 channels will be sampled and converted. If CHPS selects 4 channels, the CH0, CH1, CH2 and CH3 channels will be sampled and converted.

The SMPI bits select the number of acquisition/conversion sequences that would be performed before an interrupt occurs. This can vary from 1 sample per interrupt to 16 samples per interrupt.

The user cannot program a combination of CHPS and SMPI bits that specifies more than 16 conversions per interrupt, or 8 conversions per interrupt, depending on the BUFM bit. The BUFM bit, when set, will split the 16--word results buffer (ADCBUF0...ADCBUFF) into two 8-word groups. Writing to the 8-word buffers will be alternated on each interrupt event. Use of the BUFM bit will depend on how much time is available for moving data out of the buffers after the interrupt, as determined by the application.

If the processor can quickly unload a full buffer within the time it takes to acquire and convert one channel, the BUFM bit can be '0' and up to 16 conversions may be done per interrupt. The processor will have one sample and conversion time to move the sixteen conversions.

If the processor cannot unload the buffer within the acquisition and conversion time, the BUFM bit should be '1'. For example, if SMPI<3:0> (ADCON2<5:2>) = 0111, then eight conversions will be loaded into 1/2 of the buffer, following which an interrupt occurs. The next eight conversions will be loaded into the other 1/2 of the buffer. The processor will have the entire time between interrupts to move the eight conversions.

The ALTS bit can be used to alternate the inputs selected during the sampling sequence. The input multiplexer has two sets of sample inputs: MUX A and MUX B. If the ALTS bit is '0', only the MUX A inputs are selected for sampling. If the ALTS bit is '1' and SMPI<3:0> = 0000, on the first sample/convert sequence, the MUX A inputs are selected, and on the next acquire/convert sequence, the MUX B inputs are selected.

The CSCNA bit (ADCON2<10>) will allow the CH0 channel inputs to be alternately scanned across a selected number of analog inputs for the MUX A group. The inputs are selected by the ADCSSL register. If a particular bit in the ADCSSL register is '1', the corresponding input is selected. The inputs are always scanned from lower to higher numbered inputs, starting after each interrupt. If the number of inputs selected is greater than the number of samples taken per interrupt, the higher numbered inputs are unused.

20.4 Programming the Start of Conversion Trigger

The conversion trigger will terminate acquisition and start the requested conversions.

The SSRC<2:0> bits select the source of the conversion trigger.

The SSRC bits provide for up to 5 alternate sources of conversion trigger.

When SSRC<2:0> = 000, the conversion trigger is under software control. Clearing the SAMP bit will cause the conversion trigger.

When SSRC<2:0> = 111 (Auto Start mode), the conversion trigger is under A/D clock control. The SAMC bits select the number of A/D clocks between the start of acquisition and the start of conversion. This provides the fastest conversion rates on multiple channels. SAMC must always be at least 1 clock cycle.

Other trigger sources can come from timer modules, Motor Control PWM module, or external interrupts.

Note: To operate the A/D at the maximum specified conversion speed, the Auto Convert Trigger option should be selected (SSRC = 111) and the Auto Sample Time bits shoud be set to 1 TAD (SAMC = 00001). This configuration will give a total conversion period (sample + convert) of 13 TAD.

The use of any other conversion trigger will result in additional TAD cycles to synchronize the external event to the A/D.

20.5 Aborting a Conversion

Clearing the ADON bit during a conversion will abort the current conversion and stop the sampling sequencing. The ADCBUF will not be updated with the partially completed A/D conversion sample. That is, the ADCBUF will continue to contain the value of the last completed conversion (or the last value written to the ADCBUF register).

If the clearing of the ADON bit coincides with an auto start, the clearing has a higher priority.

After the A/D conversion is aborted, a 2 TAD wait is required before the next sampling may be started by setting the SAMP bit.

If sequential sampling is specified, the A/D will continue at the next sample pulse which corresponds with the next channel converted. If simultaneous sampling is specified, the A/D will continue with the next multi-channel group conversion sequence.

20.6 Selecting the A/D Conversion Clock

The A/D conversion requires 12 TAD. The source of the A/D conversion clock is software selected using a six bit counter. There are 64 possible options for TAD.

EQUATION 20-1: A/D CONVERSION CLOCK

$$T_{AD} = T_{CY} * (0.5*(ADCS<5:0> +1))$$

$$ADCS<5:0> = 2\ \frac{T_{AD}}{T_{CY}} - 1$$

The internal RC oscillator is selected by setting the ADRC bit.

For correct A/D conversions, the A/D conversion clock (TAD) must be selected to ensure a minimum TAD time of 154 nsec (for VDD = 5V). Refer to the Electrical Specifications section for minimum TAD under other operating conditions.

Example 20-1 shows a sample calculation for the ADCS<5:0> bits, assuming a device operating speed of 30 MIPS.

EXAMPLE 20-1: A/D CONVERSION CLOCK CALCULATION

Minimum T_{AD} = 154 nsec

T_{CY} = 33 nsec (30 MIPS)

$$ADCS<5:0> = 2\ \frac{T_{AD}}{T_{CY}} - 1$$

$$= 2 * \frac{154\ nsec}{33\ nsec} - 1$$

$$= 8.33$$

Therefore,

Set ADCS<5:0> = 9

$$\text{Actual } T_{AD} = \frac{T_{CY}}{2}\ (ADCS<5:0> + 1)$$

$$= \frac{33\ nsec}{2}\ (9 + 1)$$

$$= 165\ nsec$$

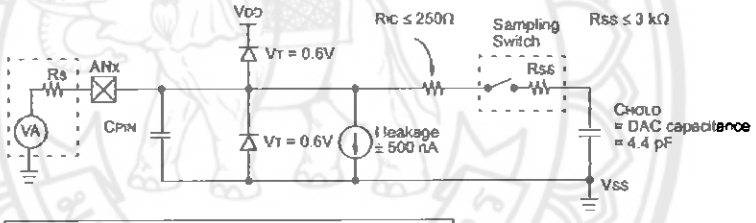
20.7 A/D Acquisition Requirements

The analog input model of the 10-bit A/D converter is shown in Figure 20-2. The total sampling time for the A/D is a function of the internal amplifier settling time, device VDD and the holding capacitor charge time.

For the A/D converter to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the voltage level on the analog input pin. The source impedance (RS), the interconnect impedance (RIC), and the internal sampling switch (RSS) impedance combine to directly affect the time required to charge the capacitor CHOLD. The combined impedance of the analog sources must therefore be small enough to fully charge the holding capacitor within the chosen sample time. To minimize the effects of pin leakage currents on the accuracy of the A/D converter, the maximum recommended source impedance, RS, is 5 kΩ. After the analog input channel is selected (changed), this sampling function must be completed prior to starting the conversion. The internal holding capacitor will be in a discharged state prior to each sample operation.

The user must allow at least 1 TAD period of sampling time, TSAMP, between conversions to allow each sample to be acquired. This sample time may be controlled manually in software by setting/clearing the SAMP bit, or it may be automatically controlled by the A/D converter. In an automatic configuration, the user must allow enough time between conversion triggers so that the minimum sample time can be satisfied. Refer to the Electrical Specifications for TAD and sample time requirements.

FIGURE 20-2: A/D CONVERTER ANALOG INPUT MODEL



Note: CPIN value depends on device package and is not tested. Effect of CPIN negligible if RS ≤ 5 kΩ

20.8 Module Power-down Modes

The module has 3 internal power modes. When the ADON bit is '1', the module is in Active mode; it is fully powered and functional. When ADON is '0', the module is in Off mode. The digital and analog portions of the circuit are disabled for maximum current savings. In order to return to the Active mode from Off mode, the user must wait for the ADC circuitry to stabilize.

20.9 A/D Operation During CPU Sleep and Idle Modes

20.9.1 A/D OPERATION DURING CPU SLEEP MODE

When the device enters Sleep mode, all clock sources to the module are shutdown and stay at logic '0'.

If Sleep occurs in the middle of a conversion, the conversion is aborted. The converter will not continue with a partially completed conversion on exit from Sleep mode.

Register contents are not affected by the device entering or leaving Sleep mode.

The A/D module can operate during Sleep mode if the A/D clock source is set to RC (ADRC = 1). When the RC clock source is selected, the A/D module waits one instruction cycle before starting the conversion. This allows the SLEEP instruction to be executed, which eliminates all digital switching noise from the conversion. When the conversion is complete, the Done bit will be set and the result loaded into the ADCBUF register.

If the A/D interrupt is enabled, the device will wake-up from Sleep. If the A/D interrupt is not enabled, the A/D module will then be turned off, although the ADON bit will remain set.

20.9.2 A/D OPERATION DURING CPU IDLE MODE

The ADSIDL bit selects if the module will stop on Idle or continue on Idle. If ADSIDL = 0, the module will continue operation on assertion of Idle mode. If ADSIDL = 1, the module will stop on Idle.

20.10 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the A/D module to be turned off, and any conversion and acquisition sequence is aborted. The values that are in the ADCBUF registers are not modified. The A/D result register will contain unknown data after a Power-on Reset.

20.11 Output Formats

The A/D result is 10-bits wide. The data buffer RAM is also 10-bits wide. The 10-bit data can be read in one of four different formats. The FORM<1:0> bits select the format. Each of the output formats translates to a 16-bit result on the data bus.

Write data will always be in right justified (Integer) format.

FIGURE 20-3: A/D OUTPUT DATA FORMATS

RAM Contents:	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00						
Read to Bus:																
Signed Fractional (1.15)	$\overline{d09}$	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0	0	0
Fractional (1.15)	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0	0	0
Signed Integer	$\overline{d09}$	$\overline{d09}$	$\overline{d09}$	$\overline{d09}$	$\overline{d09}$	$\overline{d09}$	$\overline{d09}$	d08	d07	d06	d05	d04	d03	d02	d01	d00
Integer	0	0	0	0	0	0	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00

20.12 Configuring Analog Port Pins

The use of the ADPCFG and TRIS registers control the operation of the A/D port pins. The port pins that are desired as analog inputs must have their corresponding TRIS bit set (input). If the TRIS bit is cleared (output), the digital output level (V_{OH} or V_{OL}) will be converted.

The A/D operation is independent of the state of the CH0SA<3:0>/CH0SB<3:0> bits and the TRIS bits.

When reading the PORT register, all pins configured as analog input channels will read as cleared.

Pins configured as digital inputs will not convert an analog input. Analog levels on any pin that is defined as a digital input (including the ANx pins), may cause the input buffer to consume current that exceeds the device specifications.

20.13 Connection Considerations

The analog inputs have diodes to V_{DD} and V_{SS} as ESD protection. This requires that the analog input be between V_{DD} and V_{SS}. If the input voltage exceeds this range by greater than 0.3V (either direction), one of the diodes becomes forward biased and it may damage the device if the input current specification is exceeded.

An external RC filter is sometimes added for anti-aliasing of the input signal. The R component should be selected to ensure that the sampling time requirements are satisfied. Any external components connected (via high impedance) to an analog input pin (capacitor, zener diode, etc.) should have very little leakage current at the pin.

ภาคผนวก ง
รหัสต้นฉบับเพื่อควบคุมไมโครคอนโทรลเลอร์

```
//เรียกใช้ไลบรารี่อุปกรณ์และฟังก์ชั่นต่างๆ//
#include <p30f4011.h>
#include<adc10.h>
#include<uart.h>
//กำหนดค่า การตั้งค่า ตัวชิป//
_FOSC(CSW_FSCM_OFF & XT_PLL8);
_FWDT(WDT_OFF);
_FBORPOR(PBOR_OFF & PWRT_64 & MCLR_EN);
_FGS(CODE_PROT_OFF);
//การกำหนดตัวแปรแต่ละตัว//
unsigned char Receive_Data,chk_frame,CHK_BYTE;
unsigned char  command_pump[2]={'0','0'};          //การสั่งการปั๊มน้ำ  ทำงาน=00  ไม่
ทำงาน=55
unsigned char status_pump[2]={'0','0'};            //สถานะปั๊ม ทำงาน=00 ไม่ทำงาน=55
unsigned char master_command='A';                  //'A' = อ่านคำสั่ง , 'B' = สั่งการควบคุม
ปั๊มน้ำ
unsigned char flag_master_command=0;               //1= master รอการตอบสนอง
unsigned int result[7];
//ฟังก์ชันการอินเทอรัพท์สำหรับการรับค่าจาก RS-232 ช่องที่ 2
void _ISR _U2RXInterrupt(void)
{
if(1)
{
        Receive_Data = ReadUART2();                // โหลดข้อมูลเข้ามา
        IFS1bits.U2RXIF = 0;                       // เคลียร์ค่า RX ในอินเทอร์รัพท์
        chk_frame = 0;
//การตรวจสอบข้อมูลที่รับมา
        switch(CHK_BYTE)
        {
        case 0x00:
                if(Receive_Data == 'P')
                {
```

```
                    CHK_BYTE = 0x01;
                }
else
                {
                    CHK_BYTE = 0X00;
                }
                break;
            case 0x01:
                if((Receive_Data == 'A') || (Receive_Data == 'B'))
                {
                    master_command = Receive_Data;
                    CHK_BYTE = 0x02;
                }
                else
                {
                    CHK_BYTE = 0X00;
                }
                break;
            case 0x02:
                command_pump[0] = Receive_Data;
                CHK_BYTE = 0x03;
                break;
            case 0x03:
                command_pump[1] = Receive_Data;
                CHK_BYTE = 0x04;
                break;
            case 0x04:
                CHK_BYTE = 0x05;
                break;
            case 0x05:
                CHK_BYTE = 0x06;
                break;
```

```
        case 0x06:
                CHK_BYTE = 0x07;
                break;
        case 0x07:
                CHK_BYTE = 0x08;
                break;
        case 0x08:
                CHK_BYTE = 0x09;
                break;
        case 0x09:
                if(Receive_Data == 'D')
                {
                 if(master_command == 'B')
                 {
                        if((command_pump[0]=='0') && (command_pump [1]=='0'))
                        {
                                LATDbits.LATD3 = 0;
                        }
//กำหนดค่าเริ่มต้นให้กับ โมดูลระบบสื่อสารอนุกรม RS-232 ช่องที่ 2
void uart2_init()
{
        unsigned int baudvalue;
        unsigned int U2MODEvalue;
        unsigned int U2STAvalue;

        CloseUART2();
        ConfigIntUART2( UART_RX_EN &
                        UART_RX_PR6 &
                        UART_TX_EN &
                        UART_TX_PR2);
baudvalue = 95; // Baud rate 9600 bps [PLL8]
U2MODEvalue = UART_EN &
```

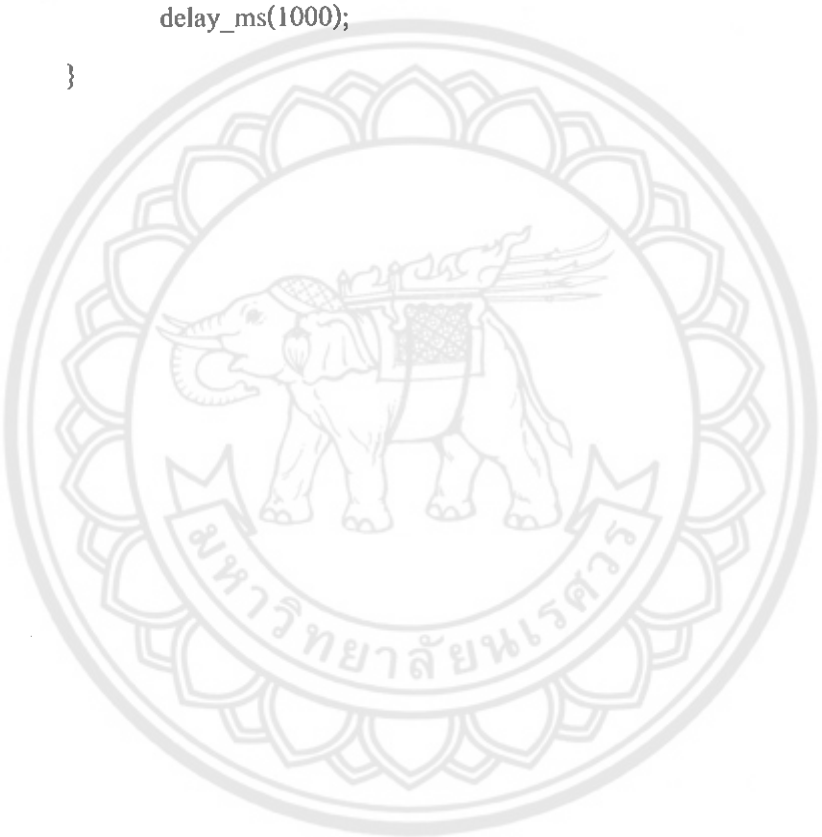
```
                    UART_IDLE_CON &
                    UART_RX_TX &
                    0xFBE7 &
                    UART_DIS_WAKE &
                    UART_DIS_LOOPBACK &
                    UART_DIS_ABAUD &
                    UART_NO_PAR_8BIT &
                    UART_1STOPBIT;
        //U2MODEbits.UARTEN = 1;
                    //U2MODEbits.USIDL = 0;
                    //U2MODEbits.ALTIO = 0;
                    //U2MODEbits.WAKE = 0;
                    //U2MODEbits.LPBACK = 0;
                    //U2MODEbits.ABAUD = 0;
                    //U2MODEbits.PDSEL = 0;
                    //U2MODEbits.STSEL = 0;
U2STAvalue = UART_INT_TX_BUF_EMPTY &
                    UART_TX_PIN_NORMAL &
                    UART_TX_ENABLE &
                    UART_INT_RX_3_4_FUL &
                    UART_ADR_DETECT_DIS &
                    UART_RX_OVERRUN_CLEAR;
        //U2STAbits.UTXISEL = 1;
                    //U2STAbits.UTXBRK = 0
                    //U2STAbits.UTXEN = 1;
                    //U2STAbits.URXISEL = 2;
                    //U2STAbits.ADDEN = 0;
                    //U2STAbits.OERR = 0;
    OpenUART2(U2MODEvalue, U2STAvalue, baudvalue);
// UART[2] Interrupt Control
IEC1bits.U2RXIE = 1;                    // เปิดการใช้งาน RX Interrupt
IEC1bits.U2TXIE = 0;                    // ปิดการใช้งาน TX Interrupt
```

```
}
//กำหนดค่าเริ่มต้นให้กับโมดูลแปลงแอนะล็อกเป็นดิจิตอลขนาด 10 บิต 1 ชองสัญญาณ
void adc_init()
{
        unsigned int Channel, PinConfig, Scanselect, Adcon3_reg, Adcon2_reg, Adcon1_reg;
ADCON1bits.ADON = 0;
Channel= ADC_CH0_POS_SAMPLEA_AN0 ;
        SetChanADC10(Channel);
        ConfigIntADC10(ADC_INT_DISABLE);
PinConfig = ENABLE_AN0_ANA ;
        Scanselect = SKIP_SCAN_AN1 &
                     SKIP_SCAN_AN2 &
                     SKIP_SCAN_AN3 &
                     SKIP_SCAN_AN4 &
                     SKIP_SCAN_AN5 &
                     SKIP_SCAN_AN6 &
                     SKIP_SCAN_AN7 &
                     SKIP_SCAN_AN8 ;
        Adcon3_reg = ADC_SAMPLE_TIME_10 &
                        ADC_CONV_CLK_INTERNAL_RC &
                        ADC_CONV_CLK_13Tcy;
Adcon2_reg =    ADC_VREF_AVDD_AVSS &
                        ADC_SCAN_ON &
                        ADC_ALT_BUF_OFF &
                        ADC_ALT_INPUT_OFF &
                        ADC_CONVERT_CH0 &
                        ADC_SAMPLES_PER_INT_16;
    Adcon1_reg = ADC_MODULE_ON &
                     ADC_IDLE_CONTINUE &
                     ADC_FORMAT_INTG &
                     ADC_CLK_MANUAL &
                     ADC_SAMPLE_SIMULTANEOUS &
```

```
                    ADC_AUTO_SAMPLING_ON;
OpenADC10(Adcon1_reg, Adcon2_reg,Adcon3_reg,PinConfig,Scanselect);
}
//ฟังก์ชันหน่วงเวลาแบบมิลลิวินาที
void delay_ms(unsigned int ms)
{
        unsigned int x, a;
        for(x=0;x<ms;x++)
        {
            for(a=0;a<1472;a++);
        }
}
int main(void)
{
    unsigned int i,j;
    char    dat;                        // Buffer ในการรับตัวอักษร
if(1)
{   unsigned int i,j;
    unsigned int dat1,dat2;
    unsigned char OLD_MIN,NEW_MIN;
    //กำหนดค่า I/O ให้กับพอร์ต
    TRISDbits.TRISD3 = 0;               // กำหนด RD3 ให้เป็นเอาท์พุต
    LATDbits.LATD3 = 1;                 //สถานะเริ่มต้นของขา RD3
//เรียกฟังก์ชันกำหนดค่าเริ่มต้นโมดูลแปลงแอนะล็อกเป็นดิจิตอล
    adc_init3();
    //เรียกค่าเริ่มต้นโมดูลระบบสื่อสารอนุกรม RS-232
    uart2_init();
while(1)
    {
        if(flag_master_call==1)         //ตรวจสอบว่าได้รับข้อความจาก
master หรือไม่
        {
```

```
                ADCON1bits.SAMP = 1;
                ADCON1bits.SAMP = 0;
                while (!ADCON1bits.DONE);
                result[0] = ADCBUF0;
                delay_ms(10);
            //สร้างข้อความตอบกลับ master
                putcUART2('D');
                delay_ms(50);
                putcUART2(master_command);          //ส่งประเภทของข้อความที่
ตอบกลับ
                delay_ms(50);
                if(LATDbits.LATD3 == 0)             //ตรวจสอบสถานะของรีเลย์
                {
                    putcUART2('0');
                    delay_ms(50);
                    putcUART2('0');
                    delay_ms(50);
                }
                else
                {
                    putcUART2('5');
                    delay_ms(50);
                    putcUART2('5');
                    delay_ms(50);
                }
            putcUART2('0');
                delay_ms(50);
                putcUART2('0'+result[0]/1000);  //แปลงค่าที่อ่านได้จากตัวรับรู้เป็น
ตัวอักษร ASCII
                delay_ms(50);
                putcUART2('0'+(result[0]%1000)/100);
                delay_ms(50);
```

```
                putcUART2('0'+((result[0]%1000)%100)/10);
                delay_ms(50);
                putcUART2('0'+((result[0]%1000)%100)%10);
                delay_ms(50);
                putcUART2('P');
                delay_ms(50);
            flag_master_call = 0;
            }
            delay_ms(1000);
        }
}
}
```



ภาคผนวก จ

รายละเอียดข้อมูลของตัวรับรู้ความดันส่วนต่าง รุ่น MPX5010DP

MOTOROLA
SEMICONDUCTOR TECHNICAL DATA

Order this document
by MPX5010/D

MPX5010 SERIES

INTEGRATED PRESSURE SENSOR 0 to 10 kPa (0 to 1.45 psi) 0.2 to 4.7 V OUTPUT

Integrated Silicon Pressure Sensor On-Chip Signal Conditioned, Temperature Compensated and Calibrated

The MPX5010 series piezoresistive transducer is a state–of–the–art monolithic silicon pressure sensor designed for a wide range of applications, but particularly those employing a microcontroller or microprocessor with A/D inputs. This patented, single element transducer combines advanced micromachining techniques, thin–film metallization, and bipolar processing to provide an accurate, high level analog output signal that is proportional to the applied pressure.

Features

- 5.0% Maximum Error over 0° to 85°C
- Ideally Suited for Microprocessor or Microcontroller–Based Systems
- Patented Silicon Shear Stress Strain Gauge
- Durable Epoxy Unibody Element
- Temperature Compensated over −40° to +125°C

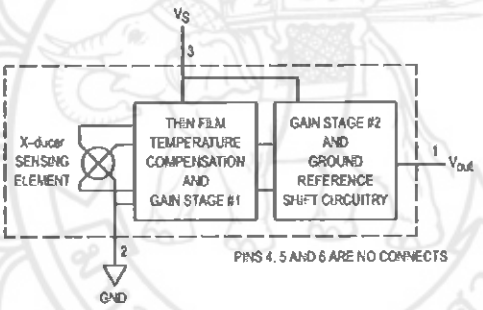
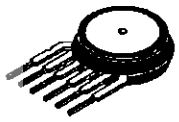
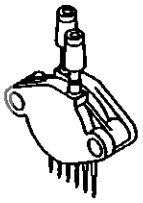


Figure 1. Fully Integrated Pressure Sensor Schematic



BASIC CHIP CARRIER ELEMENT CASE 867–08, STYLE 1



DIFFERENTIAL PORT OPTION CASE 867C–05, STYLE 1

PIN NUMBER			
1	V_{out}	4	N/C
2	Gnd	5	N/C
3	V_S	6	N/C

NOTE: Pins 4, 5, and 6 are internal device connections. Do not connect to external circuitry or ground. Pin 1 is noted by the notch in the Lead.

MAXIMUM RATINGS(1)

Parametrics	Symbol	Value	Unit
Overpressure(2) (P1 > P2)	P_{max}	75	kPa
Burst Pressure(2) (P1 > P2)	P_{burst}	100	kPa
Storage Temperature	T_{stg}	−40 to +125	°C
Operating Temperature	T_A	−40 to +125	°C

1. T_C = 25°C unless otherwise noted.
2. Exposure beyond the specified limits may cause permanent damage or degradation to the device.

MPX5010 SERIES

OPERATING CHARACTERISTICS (V_S = 5.0 Vdc, T_A = 25°C unless otherwise noted, P1 > P2)

Characteristic		Symbol	Min	Typ	Max	Unit
Pressure Range[1]		P_{OP}	0	—	10	kPa
Supply Voltage[2]		V_S	4.75	5.0	5.25	Vdc
Supply Current		I_o	—	7.0	10	mAdc
Minimum Pressure Offset[3] @ V_S = 5.0 Volts	(0 to 85°C)	V_{off}	0	0.2	0.425	Vdc
Full Scale Output[4] @ V_S = 5.0 Volts	(0 to 85°C)	V_{FSO}	4.475	4.7	4.925	Vdc
Full Scale Span[5] @ V_S = 5.0 Volts	(0 to 85°C)	V_{FSS}	—	4.5	—	Vdc
Accuracy[6]	(0 to 85°C)	—	—	—	±5.0	$\%V_{FSS}$
Sensitivity		V/P	—	450	—	mV/kPa
Response Time[7]		I_R	—	1.0	—	ms
Output Source Current at Full Scale Output		I_{O+}	—	0.1	—	mAdc
Warm–Up Time[8]		—	—	20	—	ms
Offset Stability[9]		—	—	±0.5	—	$\%V_{FSS}$

MECHANICAL CHARACTERISTICS

Characteristic	Symbol	Min	Typ	Max	Unit
Weight, Basic Element (Case 867)	—	—	4.0	—	Grams
Common Mode Line Pressure[10]	—	—	—	690	kPa

NOTES:

1. 1.0 kPa (kiloPascal) equals 0.145 psi.
2. Device is ratiometric within this specified excitation range.
3. Offset (V_{off}) is defined as the output voltage at the minimum rated pressure.
4. Full Scale Output (V_{FSO}) is defined as the output voltage at the maximum or full rated pressure.
5. Full Scale Span (V_{FSS}) is defined as the algebraic difference between the output voltage at full rated pressure and the output voltage at the minimum rated pressure.
6. Accuracy (error budget) consists of the following:
   - Linearity: Output deviation from a straight line relationship with pressure over the specified pressure range.
   - Temperature Hysteresis: Output deviation at any temperature within the operating temperature range, after the temperature is cycled to and from the minimum or maximum operating temperature points, with zero differential pressure applied.
   - Pressure Hysteresis: Output deviation at any pressure within the specified range, when this pressure is cycled to and from the minimum or maximum rated pressure, at 25°C.
   - TcSpan: Output deviation over the temperature range of 0° to 85°C, relative to 25°C.
   - TcOffset: Output deviation with minimum rated pressure applied, over the temperature range of 0° to 85°C, relative to 25°C.
   - Variation from Nominal: The variation from nominal values, for Offset or Full Scale Span, as a percent of V_{FSS}, at 25°C.
7. Response Time is defined as the time for the incremental change in the output to go from 10% to 90% of its final value when subjected to a specified step change in pressure.
8. Warm–up is defined as the time required for the product to meet the specified output voltage after the Pressure has been stabilized.
9. Offset stability is the product's output deviation when subjected to 1000 hours of Pulsed Pressure, Temperature Cycling with Bias Test.
10. Common mode pressures beyond what is specified may result in leakage at the case–to–lead interface.

ON–CHIP TEMPERATURE COMPENSATION, CALIBRATION AND SIGNAL CONDITIONING

Figure 2 illustrates the Differential/Gauge Sensing Chip in the basic chip carrier (Case 867). A fluorosilicone gel isolates the die surface and wire bonds from the environment, while allowing the pressure signal to be transmitted to the sensor diaphragm.

The MPX5010 series pressure sensor operating characteristics, and internal reliability and qualification tests are based on use of dry air as the pressure media. Media, other than dry air, may have adverse effects on sensor performance and long–term reliability. Contact the factory for information regarding media compatibility in your application.

Figure 3 shows a typical decoupling circuit for interfacing the integrated sensor to the A/D input of a microprocessor. Proper decoupling of the power supply is recommended.

Figure 4 shows the sensor output signal relative to pressure input. Typical, minimum, and maximum output curves are shown for operation over a temperature range of 0° to 85°C using the decoupling circuit below. (The output will saturate outside of the specified pressure range.)

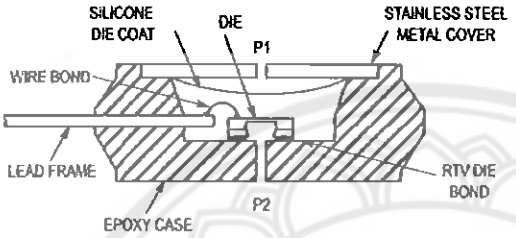


Figure 2. Cross–Sectional Diagram
(Not to Scale)

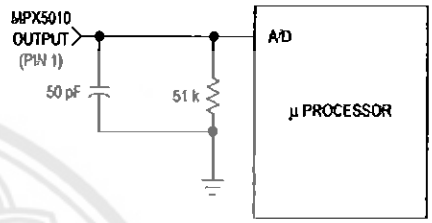


Figure 3. Typical Decoupling Filter for Sensor to Microprocessor Interface

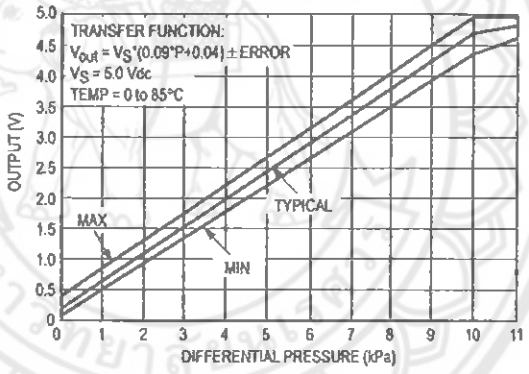


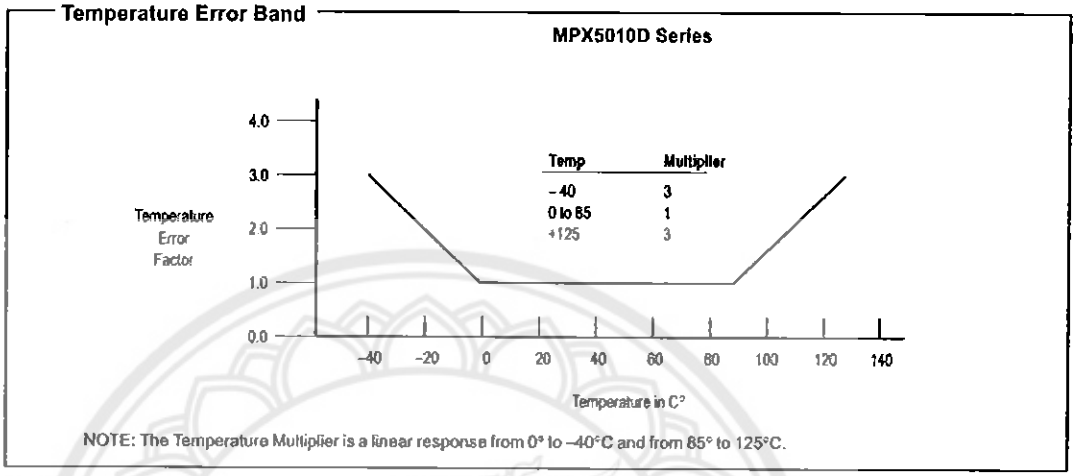
Figure 4. Output versus Pressure Differential

MPX5010 SERIES

Transfer Function (MPX5010D)

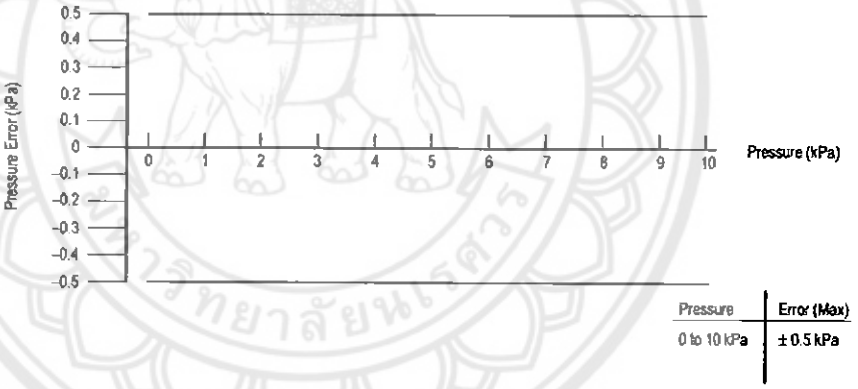
Nominal Transfer Value: $V_{out} = V_S \times (0.09 \times P + 0.04)$
$\pm$ (Pressure Error x Temp. Factor x 0.09 x V_S)
$V_S = 5.0\ V \pm 0.25\ Vdc$

Temperature Error Band



NOTE: The Temperature Multiplier is a linear response from 0° to –40°C and from 85° to 125°C.

Pressure Error Band



Pressure	Error (Max)
0 to 10 kPa	± 0.5 kPa

PRESSURE (P1)/VACUUM (P2) SIDE IDENTIFICATION TABLE

Motorola designates the two sides of the pressure sensor as the Pressure (P1) side and the Vacuum (P2) side. The Pressure (P1) side is the side containing fluoro silicone gel which protects the die from harsh media. The Motorola MPX pressure sensor is designed to operate with positive differential pressure applied, P1 > P2.

The Pressure (P1) side may be identified by using the table below:

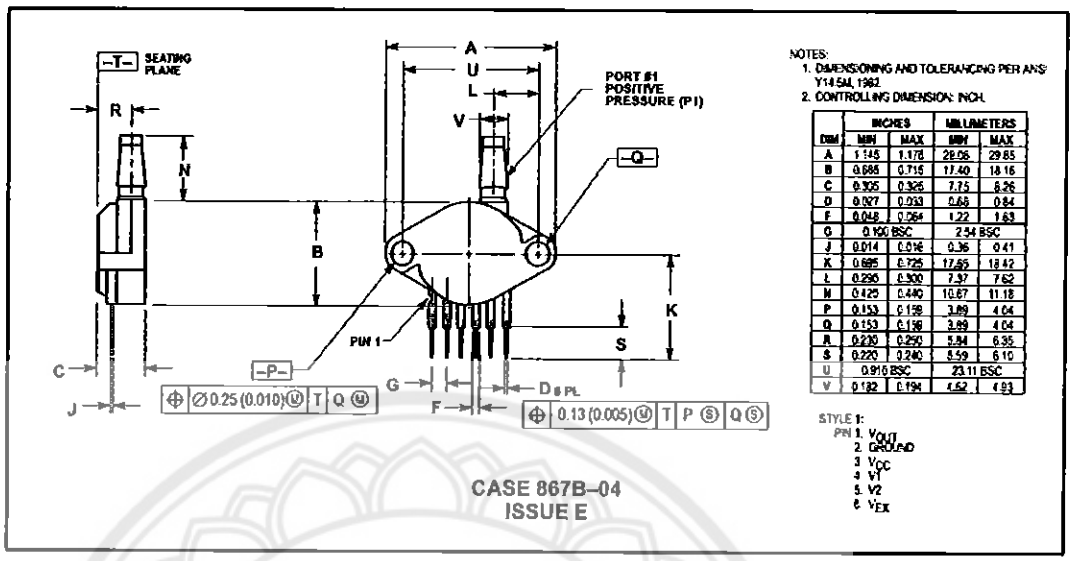
Part Number	Case Type	Pressure (P1) Side Identifier
MPX5010D	867–08	Stainless Steel Cap
MPX5010DP	867C–05	Side with Part Marking
MPX5010GP	867B–04	Side with Port Attached
MPX5010GVP	867D–04	Stainless Steel Cap
MPX5010GS	867E–03	Side with Port Attached
MPX5010GVS	867A–04	Stainless Steel Cap
MPX5010GSX	867F–03	Side with Port Attached
MPX5010GVSX	867G–03	Stainless Steel Cap

ORDERING INFORMATION

The MPX5010 pressure sensor is available in differential and gauge configurations. Devices are available in the basic element package or with pressure port fittings that provide printed circuit board mounting ease and barbed hose pressure connections.

Device Type	Options	Case Type	MPX Series	
			Order Number	Device Marking
Basic Element	Differential	867–08	MPX5010D	MPX5010D
Ported Elements	Differential Dual Ports	867C–05	MPX5010DP	MPX5010DP
	Gauge	867B–04	MPX5010GP	MPX5010GP
	Gauge Vacuum Port	867D–04	MPX5010GVP	MPX5010GVP
	Gauge, Axial	867E–03	MPX5010GS	MPX5010D
	Gauge Vacuum Axial	867A–04	MPX5010GVS	MPX5010D
	Gauge, Axial PC Mount	867F–03	MPX5010GSX	MPX5010D
	Gauge Vacuum Axial PC Mount	867G–03	MPX5010GVSX	MPX5010D

PACKAGE DIMENSIONS–CONTINUED



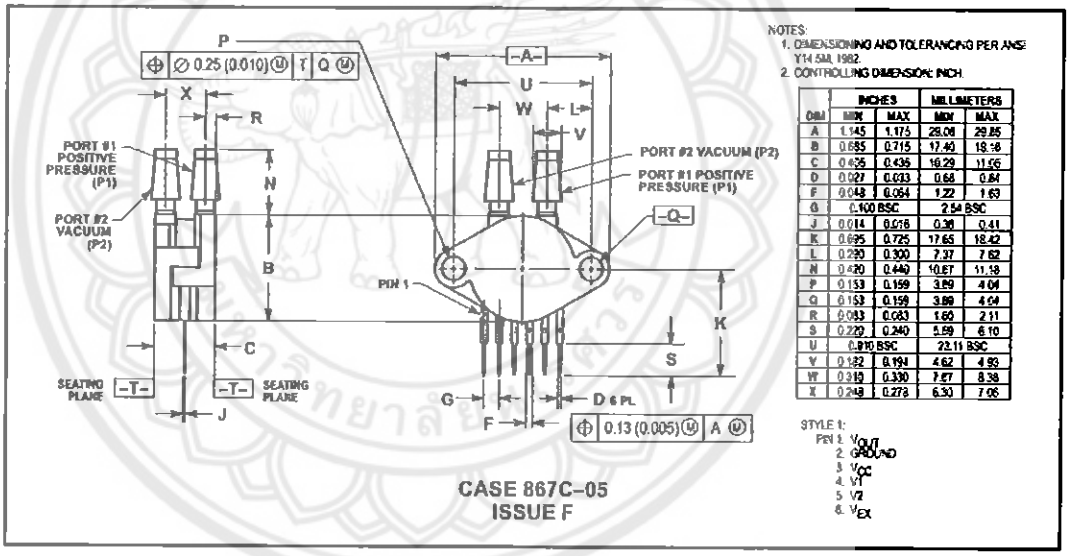
NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.

DIM	INCHES MIN	INCHES MAX	MILLIMETERS MIN	MILLIMETERS MAX
A	1.145	1.176	29.06	29.85
B	0.685	0.715	17.40	18.16
C	0.305	0.325	7.75	8.26
D	0.027	0.033	0.68	0.84
F	0.048	0.064	1.22	1.63
G	0.100 BSC		2.54 BSC	
J	0.014	0.016	0.36	0.41
K	0.695	0.725	17.65	18.42
L	0.290	0.300	7.37	7.62
N	0.425	0.440	10.67	11.18
P	0.153	0.159	3.89	4.04
Q	0.153	0.158	3.89	4.04
R	0.230	0.250	5.84	6.35
S	0.220	0.240	5.59	6.10
U	0.915 BSC		23.11 BSC	
V	0.182	0.194	4.62	4.93

STYLE 1:
PIN 1. V_{OUT}
2. GROUND
3. V_{CC}
4. V1
5. V2
6. V_{EX}

CASE 867B–04
ISSUE E

PRESSURE SIDE PORTED (GP)



NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.

DIM	INCHES MIN	INCHES MAX	MILLIMETERS MIN	MILLIMETERS MAX
A	1.145	1.175	29.08	29.85
B	0.685	0.715	17.40	18.16
C	0.425	0.435	10.29	11.05
D	0.027	0.033	0.68	0.84
F	0.048	0.064	1.22	1.63
G	0.100 BSC		2.54 BSC	
J	0.014	0.016	0.36	0.41
K	0.695	0.725	17.65	18.42
L	0.290	0.300	7.37	7.62
N	0.420	0.440	10.67	11.18
P	0.153	0.159	3.89	4.04
Q	0.153	0.159	3.89	4.04
R	0.083	0.083	1.60	2.11
S	0.220	0.240	5.59	6.10
U	0.910 BSC		23.11 BSC	
V	0.182	0.194	4.62	4.93
W	0.310	0.330	7.87	8.38
X	0.248	0.278	6.30	7.06

STYLE 1:
PIN 1. V_{OUT}
2. GROUND
3. V_{CC}
4. V1
5. V2
6. V_{EX}

CASE 867C–05
ISSUE F

PRESSURE AND VACUUM SIDES PORTED (DP)