อุปกรณ์เปลี่ยนจอภาพโดยใช้เซนเซอร์สามแกน

APPARATUS FOR CHANGING THE DISPLAY USING 3-AXIS SENSOR

จิรวรรธก์ พึ่งเกษม
JIRAWAT PHUENGKASEM
เจนจิรา ภาผิวดี
JANEJIRA PAPHIWDEE

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมสารสนเทศ
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2553

APPARATUS FOR CHANGING THE DISPLAY USING 3-AXIS SENSOR

JIRAWAT PHUENGKASEM

JANEJIRA PAPHIWDEE

THIS THESIS IS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INFORMATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2010

หัวข้อปริญญานิพนธ์	อุปกรณ์เปลี่ยนจอภาพโดยใช้เซนเซอร์สามแกน	
รายชื่อนักศึกษา	นายจิรวรรธก์ พึ่งเกษม	รหัสนักศึกษา 50010234
	นางสาวเจนจิรา ภาผิวดี	รหัสนักศึกษา 50010266
ปริญญา	วิศวกรรมศาสตรบัณฑิต	
สาขาวิชา	วิศวกรรมสารสนเทศ	
พ.ศ.	2553	
อาจารย์ที่ปรึกษาปริญญานิพนธ์	ผศ.บุณย์ชนะ ภู่ระหงษ์	

ปริญญานิพนธ์ฉบับนี้ ได้รับการอนุมัติให้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรวิศวกรรมศาสตรบัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง

.....

(ผศ.บุณย์ชนะ ภู่ระหงษ์)

อาจารย์ผู้ควบคุมปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	อุปกรณ์เปลี่ยนจอภาพโดยใช้เซนเซอร์สามแกน	
รายชื่อนักศึกษา	นายจิรวรรธก์ พึ่งเกษม	รหัสนักศึกษา 50010234
	นางสาวเจนจิรา ภาผิวดี	รหัสนักศึกษา 50010266
ปริญญา	วิศวกรรมศาสตรบัณฑิต	
สาขาวิชา	วิศวกรรมสารสนเทศ	
พ.ศ.	2553	
อาจารย์ที่ปรึกษาปริญญานิพนธ์	ผศ.บุณย์ชนะ ภู่ระหงษ์	

บทคัดย่อ

โครงงานนี้ศึกษาและนำกระบวนการประมวลผลข้อมูลจากการเคลื่อนที่ของร่างกายผ่านทาง อุปกรณ์ตัวจับความเร่ง เพื่อควบคุมการใช้งานโปรแกรมบนคอมพิวเตอร์ โดย ระบุความเร่งในการเคลื่อนที่ของมือให้อยู่ในรูปแบบความเร่งในแนวแกน x,y,z จากนั้นนำข้อมูลที่ได้จากการตรวจจับความเร่งในการเคลื่อนที่ของมือในแนวแกน x,y,z เพื่อบอกทิศทางการเคลื่อนที่และอัตราเร่งของมือได้อย่างถูกต้อง จากนั้นนำทิศทางและอัตราเร่งของแนวแกนเหล่านี้มาประมวลผลเป็นพิกัดบนหน้าจอคอมพิวเตอร์ และนำพิกัดที่ได้มาควบคุมการใช้งานโปรแกรมบนคอมพิวเตอร์ในเบื้องต้น

Thesis Title	Apparatus for changing the display using 3-axis sensor		
Student	Mr. Jirawat	Phuengkasem	Student ID. 50010234
	Miss. Janejira	Paphiwdee	Student ID. 50010266
Degree	Bachelor of Engineering		
Program	Information Engineering		
Year	2010		
Thesis Advisor	Assist. Prof. Boonchana Purahong		

ABSTRACT

This project is to study and processing data from the motion of hand via accelerationsensor. To control your computer by specifying the acceleration in the movement of the hand in the form acceleration along the axis x,y,z the data obtained by detecting the acceleration in the movement of the hand in the axis x,y,z to indicate the direction of motion and acceleration of the hand correctly. Then the direction and acceleration of the axis of these outputs are the coordinates on a computer screen. And the coordinates are used to control programs on the computer initially.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จได้ด้วยความช่วยเหลือและสนับสนุนจากบุคคลหลายฝ่าย ผู้จัดทำปริญญานิพนธ์รู้สึกซาบซึ้งเป็นอย่างยิ่งและขอกราบขอบพระคุณมา ณ โอกาสนี้ด้วย

ขอขอบพระคุณ ผศ.บุณย์ชนะ ภู่ระหงษ์ อาจารย์ที่ปรึกษา ที่กรุณาให้ความรู้ คำปรึกษา ข้อเสนอแนะทางวิชาการ และสนับสนุนในด้านต่างๆ

ขอบพระคุณ คณาจารย์ ในหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมสารสนเทศทุกท่าน ที่ประสิทธิ์ประสาทวิชาให้สามารถมีความรู้ในการศึกษาและทำปริญญานิพนธ์

ขอขอบพระคุณ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง ที่ได้ให้เงินทุนสำหรับสนับสนุนบางส่วนในการทำปริญญานิพนธ์นี้

ขอขอบคุณ เพื่อนๆ พี่ๆ น้องๆ ที่ได้ให้ความช่วยเหลือในการทำปริญญานิพนธ์

ท้ายที่สุด ขอกราบขอบพระคุณ คุณพ่อและคุณแม่ ผู้เป็นที่รัก ผู้ให้กำลังใจและให้โอกาสการศึกษาอันมีค่ายิ่ง

จิรวรรธก์ พึ่งเกษม

เจนจิรา ภาผิวดี

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันการศึกษาการเคลื่อนไหวในอิริยาบถต่างๆของมนุษย์ ในลักษณะต่างๆ เริ่มมีมากขึ้นเรื่อยๆ และถูกนำไปใช้กับอุปกรณ์หลายอย่าง จึงได้ศึกษาการเคลื่อนไหวของมนุษย์ให้สอดคล้องกับการใช้งานบนหน้าจอคอมพิวเตอร์โดยใช้เซนเซอร์จับการเคลื่อนไหวแบบ 3 แกน (Triple Axis Sensor) ซึ่งมีความสะดวกในการใช้ศึกษาการเคลื่อนไหวของมนุษย์ แต่มีข้อเสียคือต้องทำการติดตั้งตัวอุปกรณ์ ไว้กับส่วนของอวัยวะที่ต้องการศึกษา

ในการเคลื่อนไหวของมนุษย์นั้นสามารถแบ่งได้เป็น 2 วิธีหลัก คือ การใช้กล้องจับความเคลื่อนไหว (Image detection) จะเป็นการตรวจจับคุณลักษณะของอวัยวะที่ต้องการศึกษาเป็นส่วนๆ โดยใช้กล้องเป็นตัวตรวจจับ ซึ่งวิธีนี้จะต้องมีการเก็บข้อมูลคุณลักษณะของอวัยวะส่วนที่จะศึกษานั่นมีหลายลักษณะ และ อีกวิธีหนึ่งคือ การใช้เซนเซอร์จับการเคลื่อนไหวแบบ 3 แกน (Triple Axis Sensor) ซึ่งเป็นการติดอุปกรณ์ไปกับอวัยวะส่วนที่จะศึกษาแล้วสามารถ ดูลักษณะการเคลื่อนไหวของอวัยวะส่วนนั้นๆได้ ซึ่งผู้ที่ใช้จะมีได้หลากหลายกว่าวิธีแรกรวมถึงการประมวลผลข้อมูลของอุปกรณ์จะมีประสิทธิภาพมากกว่าวิธีแรก

ทางผู้เสนอโครงงานได้เล็งเห็นว่าอุปกรณ์ Triple Axis Sensor นี้เพียงพอและสามารถรองรับฟังก์ชันพอที่จะนำไปใช้ในการทดลองนี้ได้จึงสนใจที่จะนำอุปกรณ์ตัวนี้ไปติดไว้กับมือ ซึ่งเป็นอวัยวะที่จะควบคุมการทำงานของหน้าจอคอมพิวเตอร์ในลักษณะต่างๆได้สะดวก และสร้างระบบเงื่อนไขในการเคลื่อนไหวของมือ ซึ่งมีผลต่อการควบคุมคอมพิวเตอร์ในลักษณะต่างๆได้อย่างถูกต้องและมีประสิทธิภาพ พร้อมทั้งง่ายต่อการพัฒนาและมีต้นทุนต่ำ นำไปใช้งานในเชิงพาณิชย์ อันจะนำไปสู่การพัฒนาประยุกต์ใช้ในงานที่หลากหลาย มีการผลิตซึ่งมีต้นทุนต่ำลงและมีประสิทธิภาพมากขึ้น

1.2 จุดประสงค์

- เพื่อศึกษาการและทำการทดลองการเปลี่ยนหน้าจอคอมพิวเตอร์จากการเคลื่อนไหวมือของมนุษย์
- เพื่อพัฒนาโปรแกรมที่สามารถนำมาใช้ได้กับการทดลองตามวัตถุประสงค์
- สามารถติดตั้งและใช้งานได้จริง

1.3 ขอบเขตของโครงงาน

- สามารถสื่อสารการกันระหว่างการเคลื่อนไหวของมือกับคอมพิวเตอร์
- สามารถดึงการเคลื่อนที่ในแนวสามแกนให้มาสัมพันธ์กับคอมพิวเตอร์
- พัฒนาโปรแกรมให้สามารถใช้อุปกรณ์ได้กับคอมพิวเตอร์

1.4 อุปกรณ์ที่ต้องใช้

1.4.1 ฮาร์ดแวร์

- Triple Axis Accelerometer Breakout - ADXL345
- XBee 1mW Chip Antenna
- ATmega328 Dev Board - Arduino Duemilanove Compatible

1.4.2 ซอฟท์แวร์

- โปรแกรม Arduino IDE
- โปรแกรม Microsoft Visual Studio
- โปรแกรม X-CTU

1.5 ผลที่คาดว่าจะได้รับ

- มีความรู้ความเข้าใจเกี่ยวกับการใช้เทคโนโลยีมาใช้กับการเคลื่อนที่ของมนุษย์
- เพิ่มความสะดวกสบายในการใช้เทคโนโลยีเข้ามาช่วยในชีวิตประจำวัน
- สามารถออกแบบและพัฒนาเพื่อนำไปใช้ประโยชน์ได้จริง
- สามารถนำไปใช้กับคนพิการที่ไม่สามารถใช้เมาส์ในการใช้คอมพิวเตอร์ได้

1.6 ขั้นตอนการดำเนินงาน

ตารางที่ 1.1 แสดงระยะเวลาในการดำเนินงาน

ที่	รายละเอียด	พ.ศ.2553							พ.ศ.2554	
		มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.
1	วางขอบเขตและจุดประสงค์ของโครงงาน	↔								
2	กระบวนศึกษาข้อมูล	↔	↔	↔	↔					
3	● ศึกษาหลักการทำงานของอุปกรณ์		↔	↔						
4	● ศึกษาข้อมูลเกี่ยวกับโปรแกรมที่ใช้		↔	↔	↔					
5	กระบวนการออกแบบ			↔	↔	↔				
6	● ออกแบบวงจรและตรวจสอบ			↔	↔					
7	● ออกแบบโปรแกรมและตรวจสอบ				↔	↔				
8	กระบวนการสร้างและพัฒนา					↔	↔	↔	↔	
9	● จัดเตรียมอุปกรณ์					↔	↔			
10	● สร้างฮาร์ดแวร์และทดสอบ						↔	↔		
11	● สร้างโปรแกรมเพื่อรวมระบบระหว่างซอฟแวร์และฮาร์ดแวร์						↔	↔	↔	
12	ทดสอบใช้งานและปรับปรุงแก้ไข						↔	↔	↔	↔
13	จัดทำเอกสารปริญญานิพนธ์				↔	↔	↔	↔	↔	↔

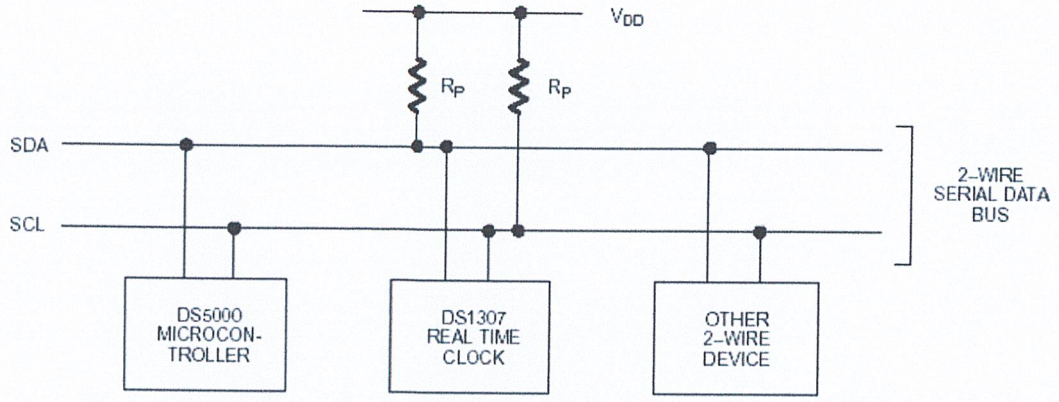
บทที่ 2
ทฤษฎีพื้นฐานที่ใช้

2.1 ระบบการเชื่อมต่ออุปกรณ์แบบ I^2C-Bus

I^2C Bus ย่อมาจาก Inter Integrate Circuit Bus (IIC Bus) นิยมเรียกกันสั้นๆว่า "I^2C Bus" (ไอ-แสคว-ซี-บัส) ซึ่งชื่อของวิธีการสื่อสารอนุกรมแบบหนึ่ง ซึ่งถูกคิดค้นและพัฒนาขึ้นโดย "PHILIPS SEMICONDUCTOR" เมื่อหลายปีก่อน แต่เพิ่งมาได้รับความนิยมอย่างแพร่หลายในระยะหลังๆมานี้เอง ซึ่งในยุคแรกๆนั้นอุปกรณ์จำพวกที่ใช้วิธีการเชื่อมต่อแบบ I^2C Bus นี้มีเพียง "PHILIPS SEMICONDUCTOR" เท่านั้นที่ทำการผลิตออกใช้งานปัจจุบันเริ่มมีผู้ผลิตรายอื่นๆ หันมาให้ความสนใจและผลิตอุปกรณ์ต่างๆที่ใช้วิธีการเชื่อมต่อแบบ I^2C Bus นี้ กันมากขึ้น เช่น บริษัท ATEL บริษัท MICROCHIPS และบริษัท DALLAS เป็นต้น เนื่องจากรูปแบบในการเชื่อมต่ออุปกรณ์ด้วยระบบบัสนี้ จะมีข้อดี คือ ใช้สัญญาณในการเชื่อมต่อเพียงสองเส้น (Serial Clock : SCL และ Serial Data : SDA) แต่สามารถเชื่อมต่ออุปกรณ์จำนวนหลายๆตัวร่วมฝนบัสเดียวกันได้ ซึ่งในปัจจุบันถือได้ว่าเป็นยุคสมัยของไมโครคอนโทรลเลอร์ขนาดเล็ก เนื่องจากระบบการทำงานของวงจรต่างๆจะมุ่งเน้นออกแบบให้มีขนาดเล็กกะทัดรัดและสามารถใช้งานได้หลากหลาย ดังนั้นอุปกรณ์จำพวก Chips Support ต่างๆ ไม่ว่าจะเป็น ไอซีหน่วยความจำ ไอซี ADC ไอซีฐานเวลา(RTC) หรือจำพวก Port I/O ต่างๆ ก็เริ่มมีการออกแบบให้ใช้การเชื่อมต่อกับ CPU เป็นบัสแบบ I^2C Bus กันมากยิ่งขึ้น ซึ่งข้อกำหนดของการเชื่อมต่อบัสแบบนี้จะมีรูปแบบที่เป็นมาตรฐานเหมือนกัน แต่อาจมีความแตกต่างกันบ้างในบางจุด เช่น จำนวนของไบท์ของข้อมูลที่ใช้ในการสื่อสารของอุปกรณ์แต่ละประเภท อาจใช้จำนวนไบท์มากน้อยไม่เท่ากันแต่รูปแบบโดยรวมจะมีความเหมือนกัน

2.1.1 การเชื่อมต่อบัสแบบ I^2C Bus

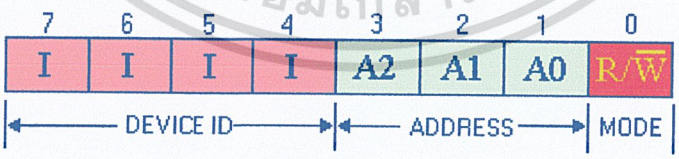
ในการเชื่อมต่ออุปกรณ์ โดยใช้บัสแบบ I^2C Bus นี้จะใช้สัญญาณทั้งหมด 2 เส้น คือ SCL และ SDA โดยการติดต่อระหว่างอุปกรณ์จะเป็นแบบ 2 ทิศทาง โดยสัญญาณทั้งสองเส้นจะต้องต่อกับตัวต้านทาน Pull-Up ไว้ เพื่อให้สถานะของบัสในขณะไม่ถูกใช้งานจะมีสภาวะเป็นบัสว่างหรือ "1" ทั้งคู่ โดยอุปกรณ์ต่างๆที่ถูกออกแบบมาเชื่อมต่อกับระบบบัสแบบนี้ จะต้องสร้างวงจรภาคเอาต์พุท ให้เป็นแบบ Open Drain หรือ Open Collector เสมอ เพื่อให้สามารถต่ออุปกรณ์ร่วมกันในระบบบัสเดียวได้มากกว่าหนึ่งอุปกรณ์



รูปที่ 2.1 แสดงลักษณะโครงสร้างการต่อบัสแบบ I^2C

2.1.2 การรับส่งข้อมูลของ I^2C Bus

การรับส่งข้อมูลของอุปกรณ์ I^2C Bus จะเริ่มต้นด้วยการที่ตัวแม่สร้างสภาวะเริ่มต้น (Start Condition) เพื่อขอใช้บัสจากนั้นจึงเริ่มการส่งรหัสควบคุม (Control Byte) เพื่อใช้ระบุตำแหน่งแอดเดรสของตัวลูกที่ต้องการจะติดต่อด้วยในระบบบัส โดยค่าตำแหน่งแอดเดรสนี้ อุปกรณ์แต่ละตัวจะมีรหัสแอดเดรสเฉพาะตัวที่แตกต่างกันออกไป ไม่มีการซ้ำกันในระบบบัสเดียวกัน โดยรหัส Control Byte นี้จะมีขนาด 8 บิต ซึ่ง 7 บิตแรก (เริ่มจาก MSB) จะเป็นค่าตำแหน่งแอดเดรสของตัวลูก ส่วนบิตที่ 8 (LSB) จะเป็นบิตสุดท้ายของไบท์ใช้สำหรับระบุทิศทางของข้อมูลในการรับส่ง (R/W) โดยถ้าบิต LSB มีค่าเป็น "0" จะหมายถึงตัวแม่(CPU) เขียนข้อมูลไปให้ตัวลูก(อุปกรณ์) แต่ถ้าบิต LSB มีค่าเป็น "1" จะหมายถึง ตัวแม่ (CPU) ต้องการอ่านข้อมูลจากตัวลูก(อุปกรณ์) โดยข้อมูลจะทำการรับส่งกันครั้งละหนึ่งไบท์ (8 บิต) และปิดท่นข้อมูลของแต่ละไบท์ด้วบบิตแสดงการตอบรับ (Acknowledge Bit) โดยลักษณะโครงสร้างของ Control Byte ของอุปกรณ์แบบ I^2C มีดังนี้



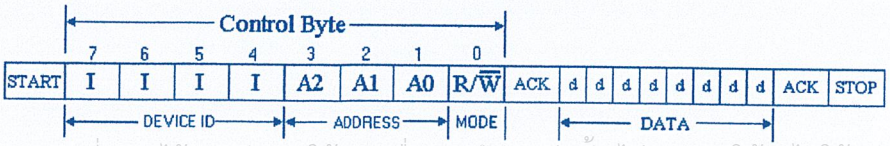
รูปที่ 2.2 แสดงลักษณะของ Control Byte ของ I^2C Bus

ซึ่งจะเห็นได้ว่ารหัส Control Byte ของอุปกรณ์ I^2C นั้น จะมีขนาด 8 บิต โดยที่ บิต7-บิต4 จะเป็นรหัสประจำตัวของอุปกรณ์แต่ละตัวที่ถูกกำหนดไว้ตายตัวจากโรงงาน ซึ่งผู้ใช้งานต้องศึกษาจากคู่มือ Data Sheet ของอุปกรณ์นั้นๆเองว่าอุปกรณ์ที่จะนำมาใช้งานมีรหัสประจำตัวเป็นเท่าใด ส่วน บิต3-บิต1 นั้น จะมีไว้สำหรับเลือกเบอร์อุปกรณ์ที่ต่ออยู่ในบัส โดยค่าของทั้ง 3 บิตนี้

จะต้องมีค่าตรงกับการกำหนดสถานะทางลอจิกให้กับขาสัญญาณ A2, A1 และ A0 ของอุปกรณ์ด้วย ตัวอย่างเช่น อุปกรณ์ที่มีรหัสประจำตัวเป็น “0111” อาจถูกออกแบบให้สามารถต่อร่วมกันภายในบัสเดียวกันได้จำนวน 8 ตัว โดยการกำหนดสภาวะลอจิกให้กับขาสัญญาณ A2, A1 และ A0 ของอุปกรณ์ให้มีความแตกต่างกันจากวงจรที่ต่ออยู่ ดังนั้นเมื่อตัวแม่ต้องการติดต่อกับอุปกรณ์ที่มีรหัสประจำตัว “0111” ตัวใดในระบบบัสก็จะส่งค่า Control Byte ด้วยค่าใน บิต3-บิต1 ตรงกับสถานะทางลอจิกของอุปกรณ์ตัวนั้นๆออกไป ตัวอย่างเช่น ถ้าตัวแม่ส่งรหัส Control Byte ด้วยค่าใน บิต3-บิต1 ตรงกับสถานะทางลอจิกของอุปกรณ์ตัวนั้นๆออกไป ตัวอย่างเช่น ถ้าตัวแม่ส่งรหัส Control Byte ด้วยค่า “01110000” ออกไปในบัส ก็จะหมายถึงว่า ตัวแม่ต้องการจะเขียนข้อมูลไปยังอุปกรณ์ที่มีรหัสประจำตัว “0111” ตัวที่ขาสัญญาณ A2,A1 และ A0 มีค่าเป็น “0” อยู่ ส่วนตัวอุปกรณ์ที่มีรหัส “0111” แต่มีสภาวะลอจิกของขาสัญญาณ A2,A1 และ A0 ไม่ตรงกับ “000” ก็จะไม่สนใจรหัสนั้น

อุปกรณ์ I^2C บางตัว อาจถูกออกแบบให้ต่อได้ในระบบบัสเดียวกันเพียงตัวเดียวโดยไม่มีขาสัญญาณในการเลือกตำแหน่งของอุปกรณ์อยู่ด้วย ค่าของ Control Byte ในตำแหน่ง บิต3-บิต1 ก็อาจถูกกำหนดไว้ตายตัวเป็น “000” เสมอก็เป็นไปได้ ส่วนค่า Control Byte ในตำแหน่งของบิต0 นั้นจะใช้เป็นบิตกำหนดทิศทางของข้อมูล โดยถ้าบิต0 มีค่าเป็น “0” จะหมายถึงตัวแม่ต้องการเขียนค่าไปยังอุปกรณ์ แต่ถ้าค่าในบิต 0 มีค่าเป็น “1” จะหมายถึงตัวแม่ต้องการอ่านค่าข้อมูลจากอุปกรณ์เป็นต้น

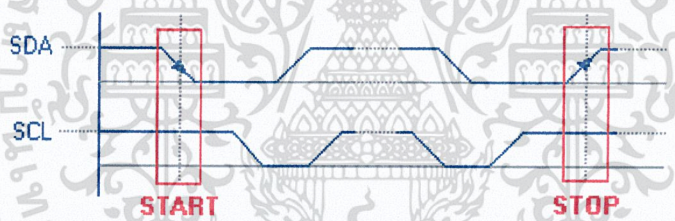
สำหรับจำนวนไบท์ข้อมูลในการรับส่งกันนั้น ไม่มีข้อกำหนดตายตัวว่าจะต้องส่งกันครั้งละกี่ไบท์ ขึ้นอยู่กับข้อตกลงระหว่างอุปกรณ์แต่ละชนิดจะกำหนดขึ้น โดยในการรับส่งแต่ละครั้งนั้น ตัวแม่จะเป็นตัวควบคุมการรับส่งเองทั้งหมด ซึ่งในกรณีที่ตัวแม่ต้องการติดต่อกับอุปกรณ์หลายๆตัวนั้น หลังจากตัวแม่สร้างสภาวะเริ่มต้น (Start Condition) ขึ้นมาและทำการรับส่งข้อมูลกับอุปกรณ์ตัวหนึ่งเสร็จแล้ว ไม่จำเป็นต้องสร้างสภาวะสุด (Stop Conditin) เพื่อกลับไปเริ่มต้นรับส่งข้อมูลกับอุปกรณ์ตัวต่อไปอีกก็ได้ แต่ตัวแม่สามารถสร้างสภาวะเริ่มต้น (Start Condition) ขึ้นมาซ้ำใหม่พร้อมกับส่งค่า Control Byte ซึ่งระบุตำแหน่งแอดเดรสของอุปกรณ์ตัวต่อไปที่ต้องการติดต่อด้วยได้ทันที แต่เมื่อทำการติดต่อรับส่งข้อมูลกับอุปกรณ์ทุกตัวในบัสเสร็จเรียบร้อยแล้วจึงสร้างสภาวะสิ้นสุด (Stop Condition) เพื่อเป็นการเลิกใช้บัสและทำให้บัสอยู่ในสภาวะว่างในภายหลังก็ได้เช่นกัน



รูปที่ 2.3 รูปแบบการเขียน/อ่านข้อมูลแบบ I^2C Bus

2.1.3 ข้อกำหนดในการเริ่มต้น (Start) และสิ้นสุด (Stop)

การกำหนดสภาวะเริ่มต้น (Start Condition) และสภาวะสิ้นสุด (Stop Condition) จะถูกกำหนดโดยตัวแม่ (CPU) โดยสภาวะปรกติของบัส หรือสภาวะบัสว่างนั้น สัญญาณ SCL และ SDA จะมีสถานะเป็น "1" ทั้งคู่ แต่เมื่อต้องการเริ่มต้นการรับส่งข้อมูลในบัส ตัวแม่จะต้องสร้างสภาวะเริ่มต้น (Start Condition) โดยการเปลี่ยนสถานะของสัญญาณ SDA จาก "1" มาเป็น "0" ในขณะที่สัญญาณ SCL ยังมีค่าเป็น "1" อยู่ ซึ่งอุปกรณ์ ที่ทำหน้าที่เป็นตัวลูกจะรอคอยตรวจสอบสภาวะการเปลี่ยนแปลงของบัสอยู่เสมอ เมื่อตรวจสอบพบสภาวะเริ่มต้น จึงมีการเริ่มต้นรับส่งข้อมูลกัน โดยตัวแม่จะต้องส่งข้อมูลของ Control Byte เป็นไบท์เริ่มต้น เพื่อกำหนดตำแหน่งแอดเดรสของอุปกรณ์ปลายทางที่ต้องการจะติดต่อด้วย โดยตัวแม่จะเป็นตัวสร้างสัญญาณ SCL เพื่อควบคุมการรับส่งข้อมูลในบัสตลอดการรับส่ง โดยสภาวะของสัญญาณข้อมูล SDA จะถูกเปลี่ยนแปลงในขณะที่สัญญาณนาฬิกา SCL มีค่าเป็น "0" โดยข้อมูลจะถูกรับส่งในขณะที่สัญญาณนาฬิกา SCL มีค่าเป็น "1" และเมื่อต้องการสิ้นสุดการใช้บัส ตัวแม่ก็จะสร้างสภาวะสิ้นสุด (Stop Condition) โดยการเปลี่ยนสภาวะของสัญญาณ SDA จาก "0" กลับไปเป็น "1" ในขณะที่สัญญาณ SCL ยังมีค่าเป็น "1" อยู่



รูปที่ 2.4 I^2C Bus START and STOP Conditions

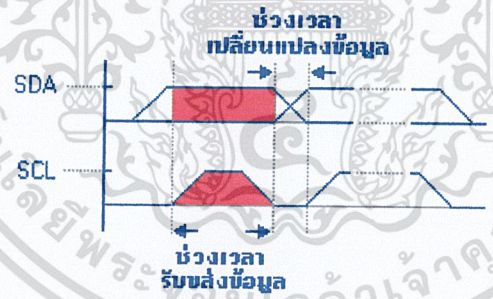
2.1.4 การแจ้งสภาวะรับทรายในบัส (Acknowledge)

ข้อบังคับอีกประการหนึ่งในการรับส่งข้อมูลในระบบบัสแบบ I^2C Bus คือ จะต้องมีการแจ้งรับทราบ (Acknowledge) เพื่อแจ้งให้ทราบว่า ข้อมูลที่ถูกส่งออกไปนั้นได้รับแล้วและมีความถูกต้องสมบูรณ์ ซึ่งตัวแม่จะใช้สัญญาณนาฬิกา SCL ในการควบคุมการรับส่งข้อมูลทางขา SDA เมื่อข้อมูลถูกส่งครบ 8 บิต ตัวลูกจะต้องมีการตอบรับให้ตัวแม่รับรู้ โดยเมื่อตัวแม่ส่งข้อมูลครบ 8 บิตแล้ว ตัวแม่จะปล่อยสัญญาณข้อมูล SDA ให้อยู่ในสภาวะว่าง ("1") และสร้างสัญญาณนาฬิกา SCL ออกมาในบัสอีกจำนวน 1 ลูกคลื่น ซึ่งในสภาวะนี้ตัวลูกจะต้องส่งสัญญาณ "0" ออกมาทางขา SDA ในช่วงที่สัญญาณนาฬิกา SCL มีค่าเป็น "0" เป็นเวลา 1 ลูกคลื่นของสัญญาณนาฬิกามีค่าเป็น "1" เสมอ

ในกรณีที่ตัวลูกไม่ยอมส่งสัญญาณการรับรู้มาให้ ตัวแม่อาจต้องเริ่มต้นกระบวนการส่งข้อมูลใหม่ทั้งหมด หรืออาจยกเลิกหรือหยุดการติดต่อการรับส่งข้อมูลใครครั้งนั้นก็ได้

สำหรับรายละเอียดที่กล่าวอธิบายแล้วนี้จะเป็นลักษณะข้อกำหนดโดยรวมของวิธีการสื่อสารอนุกรมแบบ I^2C Bus เท่านั้น ซึ่งตามปรกติแล้วอุปกรณ์ทุกตัวที่ออกแบบให้ใช้การติดต่อสื่อสารกับแบบอนุกรม I^2C Bus นั้น มักจะใช้ข้อกำหนดต่างๆเหล่านี้เป็นมาตรฐานเดียวกันแทบทั้งสิ้น ไม่ว่าจะเป็นลักษณะของสัญญาณในการสร้างสภาวะเริ่มต้น (Start Condition) สภาวะสิ้นสุด (Stop Condition) วิธีการแจ้งสภาวะรับทราบในบัส (Acknowledge) ช่วงเวลาของการรับข้อมูลและ ช่วงเวลาของการเปลี่ยนแปลงสัญญาณข้อมูล ช่วงเวลาของการส่งข้อมูล เป็นต้น

เมื่อต้องการใช้งานอุปกรณ์ I^2C ตัวใด ผู้ใช้จำเป็นต้องศึกษาคู่มือ Data Sheet หรือรายละเอียดเพิ่มเติมเฉพาะของอุปกรณ์ตัวนั้นๆ ประกอบด้วย เช่น ข้อกำหนดในการรับส่งข้อมูลของแต่ละอุปกรณ์ เช่นในการเขียนข้อมูลให้กับอุปกรณ์แบบ I^2C นั้น บางตัวอาจใช้วิธีการส่งรหัส Control Byte, ตำแหน่งแอดเดรส และ ไบท์ที่เป็นข้อมูลอย่างละ 1 ไบท์ แต่อุปกรณ์บางตัวอาจต้องส่งจำนวนไบท์สำหรับระบุตำแหน่งแอดเดรสเพิ่มเป็น 2 ไบท์ เป็นต้น หรืออุปกรณ์บางตัวอาจมีความสามารถในการรับส่งข้อมูลหรือรับสัญญาณนาฬิกาจากไมโครคอนโทรลเลอร์ที่มีความเร็วได้ไม่เท่ากัน บางตัวสามารถทำงานกับสัญญาณที่มีความเร็วสูงๆ ได้แต่บางตัวอาจมีข้อจำกัดเรื่องความเร็วในการทำงานอยู่บ้าง ดังนั้นจึงต้องพิจารณาถึงคุณสมบัติต่างๆเหล่านี้ประกอบในการใช้งานเสมอ



รูปที่ 2.5 การรับส่งบิตข้อมูลของ I^2C Bus

2.2 Zigbee and Xbee BASIC

ระบบสื่อสารไร้สาย (Wireless Telecommunication) มีมากมายหลายรูปแบบ เช่น GSM, CDMA, วิทยุย่าน 27 Mhz และ 433 Mhz, wireless lan, Wifi, WiMax ฯลฯ สำหรับในสมัยก่อนการจะทำเครื่องส่งเครื่องรับ ต้องมีความรู้ทางด้าน RF Engineer ซึ่งจะสามารถออกแบบวงจรเครื่องส่ง เครื่องรับ, มีหน้าที่ทำการ Matching สายอากาศ, การออกแบบสายอากาศสำหรับย่านความถี่ต่าง ๆ ฯลฯ

แต่สมัยนี้ มี IC RF ที่ Integrated รวมเอาภาคการออกแบบของ RF Engineer เข้าไปหลายส่วน ทำให้ง่ายในระดับที่ไม่ต้องมีความรู้ทางด้าน RF Engineer ก็สามารถสร้างวงจรส่งและรับได้แล้ว

งานทางด้านไมโครคอนโทรลเลอร์ จะมีความเกี่ยวข้องกับการสื่อสารทั้งสิ้น เช่น การสร้างเครือข่ายของระบบหนึ่งๆ, การติดต่อสื่อสารใช้งานอุปกรณ์ RF Module และการสื่อสารเพื่อใช้งานติดต่อกับอุปกรณ์อื่นๆ ผ่าน Interface ต่างๆ เช่น RS232(UART), SPI, I2C, CAN, RS485, Ethernet, LAN, TCP/IP, USB ฯลฯ ผู้ที่เคยเขียน software ที่เกี่ยวข้องกับการสื่อสารต่างๆ จะมีความเข้าใจเกี่ยวกับโปรโตคอลสื่อสารสามารถเรียนรู้การสื่อสารแบบอื่นๆได้ไม่ยาก การเขียน software ลักษณะการรับ stream data เพื่อมาเก็บใน buffer แล้วทำการ encapsulate, de-capsulate ข้อมูล (เช่น การเขียนโปรแกรมทางด้าน network security, UART, I2C ฯลฯ) แล้วนำข้อมูลไปใช้งานเป็นสิ่งจำเป็นสำหรับงานทางด้านการติดต่อสื่อสารเกือบทุกรูปแบบ

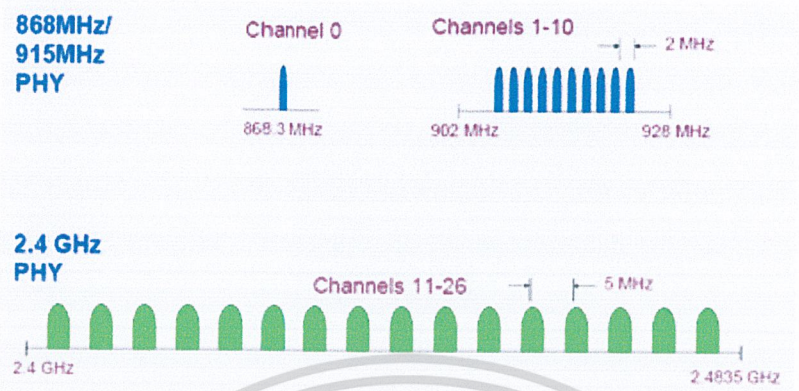
สำหรับการอธิบายในเชิงทฤษฎีนั้น การสื่อสารแต่ละแบบควรที่จะอธิบายอ้างอิงกับ OSI Layer ได้ยกตัวอย่าง ระบบเครือข่าย LAN จะใช้โปรโตคอล TCP/IP ซึ่งสามารถที่จะแยกได้ว่าขั้นตอนไหนจัดอยู่ในลำดับชั้น OSI Layer อะไร สำหรับ การสื่อสารไร้สายก็จะสามารถอ้างอิงได้ว่าเป็นภาคทางด้าน Physical Layer หรือชั้นที่สูงขึ้น แยกแยะได้ตาม OSI Layer เช่นกัน แต่ในทางการปฏิบัติใช้งานจริง นักพัฒนาเพียงทราบทฤษฎีเพียงเล็กน้อย ก็พอที่จะสามารถนำไปพัฒนางานได้แล้ว

2.2.1 Zigbee

ZigBee มาตรฐานสากล กำหนดโดย ZigBee Alliance เป็นการสื่อสารแบบไร้สายที่มีอัตราการรับส่งข้อมูลต่ำ ใช้พลังงานต่ำ ราคาถูก จุดประสงค์ก็เพื่อให้สามารถสร้างระบบที่เรียกว่า Wireless Sensor Network ได้ ซึ่งระบบนี้จะสามารถทำงาน ในร่มกลางแจ้ง ทนแดด ทนฝน และอยู่ได้ด้วยแบตเตอรี่ก้อนเล็ก (เช่นถ่าน AA 2 ก้อน) นานเป็นเดือน เป็นปี เหมาะสมใช้งานกับพวก Monitoring ต่างๆ

Zigbee กำหนดย่านความถี่ใช้งานตามมาตรฐานไว้ 3 ย่านความถี่คือ ย่าน 2.4 Ghz, ย่าน 915 Mhz และย่าน 868 Mhz โดยแต่ละย่านจะมีช่องสัญญาณ 16 ช่อง, 10 ช่อง

และ 1 ช่อง ตามลำดับ ส่วนอัตรารับส่งข้อมูล (ทางอากาศ) จะอยู่ที่ 250 Kbps, 40 Kbps, 20 Kbps ตามลำดับเช่นกัน

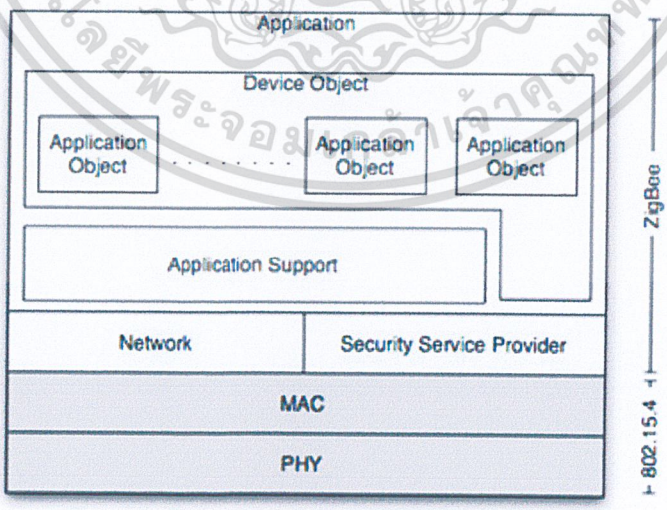


รูปที่ 2.6 ย่านความถี่ใช้งานตามมาตรฐาน

สรุป

- ย่านความถี่ 2.4 Ghz มี 16 ช่องสัญญาณ อัตรารับส่งข้อมูล 250 Kbps
- ย่านความถี่ 915 Ghz มี 10 ช่องสัญญาณ อัตรารับส่งข้อมูล 40 Kbps
- ย่านความถี่ 868 Ghz มี 1 ช่องสัญญาณ อัตรารับส่งข้อมูล 20 Kbps

ZigBee นำ Physical Layer และ MAC Layer ของ IEEE 802.15.4 ซึ่งเป็นมาตรฐานการกำหนดการสื่อสารไร้สายแบบ WPAN (Wireless Personal Area Network) มาทำงานใน Layer ที่ต่ำกว่า เช่น เรื่องของ ระดับกำลังสัญญาณ, Link Quality, Access control, Security ฯลฯ



รูป Zigbee Stack

รูปที่ 2.7 Zigbee Stack

ZigBee สามารถสร้างเป็นเครือข่ายได้ ทั้งนี้ ZigBee ได้อ้างอิงมาตรฐานตาม IEEE 802.15.4 โดย IEEE 802.15.4 แบ่งชนิดอุปกรณ์ในเครือข่ายออกเป็น 2 ประเภท คือ FFD (Full Function Device) ซึ่งหมายถึงอุปกรณ์ที่สามารถทำงานได้ทุกอย่างในเครือข่าย และ RFD (Reduce Function Device) ซึ่งหมายถึงอุปกรณ์ที่ถูกลดความสามารถการทำงานในเครือข่าย

ZigBee ได้แบ่งตามลักษณะการทำงาน 3 แบบ คือ

- Coordinator มีหน้าที่สร้างการสื่อสาร เชื่อมโยงเครือข่าย ระหว่าง End Device กับ Router หรือ Coordinator กับ Coordinator ด้วยกัน หรือ Coordinator กับ Router กำหนด address ให้กับ device ที่อยู่ในวงเครือข่าย ไม่ให้ซ้ำกัน ดูแลจัดการเรื่องการ Routing เส้นทาง ซึ่งเทียบได้กับ FFD
- End Device เป็นอุปกรณ์ปลายทางสุด ซึ่งจะใช้รับสัญญาณจาก Sensor ที่ปลายทาง โดยที่ใช้พลังงานต่ำในการทำงาน เทียบได้กับ RFD หรือ FFD บางกรณี ขึ้นอยู่กับ sensor ที่ใช้
- Router มีหน้าที่ รับส่งข้อมูล ในเส้นทางต่าง ๆ ของเครือข่าย ซึ่งเทียบได้กับ FFD

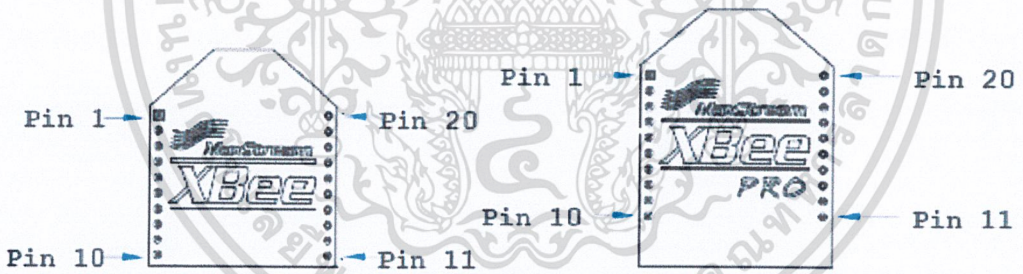
2.2.2 Xbee

Xbee เป็นอุปกรณ์ที่มี Microcontroller และ RF IC อยู่ภายใน ทำหน้าที่เป็นอุปกรณ์ transceiver (อุปกรณ์รับ-ส่งสัญญาณ) แบบ แบบ Half Duplex ย่านความถี่ 2.4 Ghz มีการจัดการโดยใช้พลังงานต่ำ ใช้งานง่าย มี interface ที่ใช้รับและส่งข้อมูลกับ Xbee เป็น UART (TTL) ซึ่งสำหรับทางด้านไมโครคอนโทรลเลอร์ นำขาที่ใช้ติดต่อสื่อสาร UART ของ Xbee ต่อเข้ากับ UART ของไมโครคอนโทรลเลอร์ได้เลย

Xbee สามารถใช้งานตามมาตรฐาน Zigbee ได้ โดยที่ไม่ต้องเขียนโปรแกรมสร้างเครือข่าย Zigbee เลย เพราะว่าทางผู้ผลิตได้จัดทำ firmware ที่จะโหลดเข้าไปในตัว Xbee ให้สามารถ set parameter ผ่าน software interface (X-CTU หรือโปรแกรมที่เขียนขึ้นเอง) , ผ่านทาง At command (เหมือนกับการควบคุม GSM Module) โดยใช้ Hyper terminal หรือ ผ่านทางการรับส่งข้อมูลด้วยไมโครคอนโทรลเลอร์ ได้อย่างง่ายดาย โดยเมื่อ set Xbee ให้ทำงานเป็นอุปกรณ์ในเครือข่าย Zigbee แล้ว เราจะเรียก Xbee แต่ละตัวว่าเป็น Node

Feature Summary ของ Xbee โดยรวมที่เหมือนกัน

- Operating Frequency ISM Band 2.4 Ghz (ISM Band หมายถึง ย่านความถี่ใช้งานเพื่อการวิจัย ซึ่งจะอนุญาตให้ใช้กับ อุตสาหกรรม (Industrial) วิทยาศาสตร์ (Scientific) และ ทางการแพทย์ (Medical) รวมเป็น ISM)
- มีสายอากาศให้เลือกใช้หลายแบบ คือ แบบ Chip Ant, Whip Ant, UFL con, RPSMA con โดย 2 แบบหลัง เราต้องไปหาเสาอากาศย่าน 2.4 Ghz ที่เป็น connector แบบ UFL หรือ SMA ครับ
- Supply Voltage อยู่ที่ 2.8-3.4 V
- Power Down Current < 10uA
- มี RF data rate อยู่ที่ 250 Kbps (เป็นส่วนของ สัญญาณที่ส่งผ่านอากาศ
- มี Serial interface data rate อยู่ระหว่าง 1200 – 115200 Bps (เป็นส่วนที่ติดต่อสื่อสารกับไมโครคอนโทรเลอร์)
- เป็น Spread Spectrum ชนิด DSSS (Direct Sequence)
- การกำหนด addressing มีลำดับลักษณะคือ กำหนด PAN ID สำหรับเครือข่ายหนึ่งๆ , กำหนด Channel และ กำหนด address ของแต่ละตัว



รูปที่ 2.8 Datasheet ของ Xbee

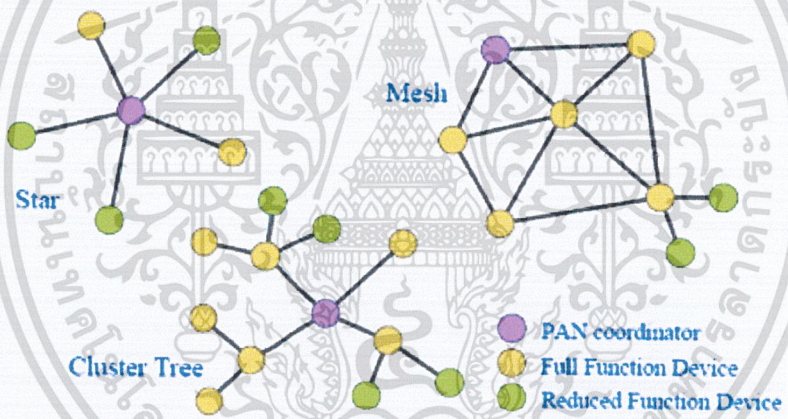
Xbee จะมีอยู่ 2 รุ่นคือ รุ่น series1 (รุ่น IEEE802.15.4) และรุ่น series 2 (รุ่น ZNET2.5) และยังมีขนาด power ให้เลือกอีก 2 แบบ คือ แบบธรรมดา (1 mw – 2 mw) และ แบบ PRO (50mw- 60 mw) ซึ่งจะมีผลเรื่องระยะทางการรับส่งข้อมูล โดยแต่ละ series นั้น สามารถสร้างเครือข่ายได้หลายแบบ แต่จะมีเพียง series 2 เท่านั้นที่จะทำเครือข่ายแบบ mesh ได้ ซึ่งยังมีรายละเอียดปลีกย่อยในเรื่องของความแตกต่างในแต่ละ series

Xbee ทั้ง 2 Series นี้สามารถสร้าง Topology ได้ดังนี้

- series1 (รุ่น IEEE802.15.4) Peer-to-peer, point-to-point, point-to-multipoint (Broadcast)
- series 2 (รุ่น ZNET2.5 / รุ่น ZB)Mesh, Peer-to-peer, point-to-point, point-to-multipoint (Broadcast)

ที่สำคัญ สำหรับ Xbee series2 ที่ทำ mesh ได้ จะมี parameter มากกว่า series 1 หากใช้ Series2 ส่งแบบ point-to-point จะวุ่นวายกว่า Series1

หมายเหตุ peer-to-peer network หมายถึง เครือข่ายที่อยู่ในระดับชั้นเดียวกัน ยกตัวอย่าง ใน OSI Layer เช่น ระดับ Transport Network Layer กับ Transport Network Layer นั่นก็คือ TCP Protocol ระหว่างคอมพิวเตอร์ 2 เครื่อง แต่สำหรับ Xbee คำว่า peer-to-peer network หรือ Non Beacon Network คือ การที่ set node เป็น End Device หมดทุกตัว ไม่มีการกำหนดตายตัวว่า ตัวใดจะเป็น Master ตัวใดจะเป็น Slave แต่จะให้ระบบจัดการกันเอง โดยในเครือข่าย จะต้องกำหนด parameter ID (PAN ID) และ CH (Channel)



รูปที่ 2.9 เครือข่าย Zigbee แบบ Star, Cluster, Mesh

2.2.2.1 การศึกษาการใช้งาน Xbee ในเบื้องต้น

สามารถทดสอบด้วยการปรับ parameter ที่สำคัญต่างๆ ผ่าน software user interface ได้ สามารถ download software user interface ที่ใช้ร่วมกันกับ Xbee ชื่อ X-CTU มาได้จาก Digi การใช้งานสามารถอ่านจากคู่มือ X-CTU Configuration & Test Utility Software User Guide

นอกจากใช้ software แล้ว ต้องมีอุปกรณ์ที่จะเชื่อมต่อ Xbee เข้ากับคอมพิวเตอร์เพื่อทำการติดต่อสื่อสารกับ X-CTU ด้วย อุปกรณ์ที่ว่านี้ คือตัวที่จะนำขาบางขาของ Xbee มาต่อเข้ากับ max232 เพื่อเปลี่ยนระดับสัญญาณ TTL ให้สามารถติดต่อสื่อสารกับ

คอมพิวเตอร์ผ่าน RS232 (DB9) ได้ หรือจะใช้ FT232RL สำหรับแปลง serial เป็น USB ในกรณีที่ computer ไม่มีพอร์ต DB9 แล้วก็ได้

ตัวอย่าง อุปกรณ์ที่ใช้เชื่อมต่อกับคอมพิวเตอร์เพื่อ Update, Config Parameter โปรแกรม firmware ใหม่ และสำหรับใช้ทำการทดสอบเบื้องต้น เช่น Xbee Socket, Xbee USB Dongle, Xbee Breadboard, X-bee RS232 (DB9) Dongle อุปกรณ์เหล่านี้ถือว่าเป็นอุปกรณ์เสริมที่ลดภาระงานทางด้าน Hardware ได้ส่วนหนึ่ง ในระยะเริ่มต้นก็จะสะดวกดี แต่หากต้องการจะทำ Hardware ขึ้นเอง ก็สามารถดูวงจร Schematic จาก Digi

2.2.2.2 กำลังส่งสายอากาศ และสัญญาณรบกวนของ Xbee

Xbee ใช้ย่านความถี่ 2.4 Ghz ซึ่งเป็นย่านเดียวกันกับ Bluetooth หรือ Wireless Lan มาถึงเรื่องของสายอากาศ ตัว Xbee มีให้เลือกที่เป็นแบบสำเร็จรูปพร้อมใช้ ไม่ต้องหาสายอากาศมาต่อเพิ่มคือ สายอากาศแบบ chip ant และ whip ant ซึ่ง Pattern การแพร่กระจายคลื่นจะบ่งบอกอย่างหนึ่ง ได้ว่า สายอากาศนี้ จะมีอัตราการขยายดีอย่างไร

Chip ant นั้น ก็มีข้อดีตรงที่มันทำให้ขนาด Dimension รวมมันเล็กลง แต่ Gain น้อยกว่าแบบ Whip ant Chip ant จึงมีระยะรับส่งข้อมูลที่ลดลงจาก spec ใน datasheet ยกตัวอย่างเช่น รุ่น Pro ที่บอกว่าสามารถส่งได้ไกลสูงสุด 1.5 km แบบ line of sight แต่ถ้าเราเลือก chip ant แล้ว จะได้ระยะสูงสุดอยู่ที่ 500 กว่าเมตร

Module	Antenna Type	Outdoor Distance (Visual Line-of-Sight)	Indoor Distance (Office Building)	Indoor Distance (Warehouse)
XBee	Chip	470 ft. (143 m)	80 ft. (24 m)	-
	Whip	845 ft. (258 m)	80 ft. (24 m)	84 ft. (26 m)
XBee-PRO	Chip	1690 ft. (515 m)	140 ft. (43 m)	-
	Whip	4382 ft. (1335 m)	140 ft. (43 m)	355 ft. (108 m)

รูปที่ 2.10 กำลังส่ง สายอากาศ และ สัญญาณ รบกวน ของ Xbee

2.2.2.3 Xbee Association

ในเครือข่าย Zigbee ต้องมีการทำงานในโหมดประหยัดพลังงาน ในช่วงเวลาที่ไม่มีการทำงาน รับส่งข้อมูล ดังนั้นตัว Xbee จึงมี Parameter ที่จะกำหนดการทำงานสำหรับ Sleep mode อยู่ (Parameter A1,A2,SP,ST)

2.2.2.4 Xbee Addressing

ตัว Xbee จะสามารถกำหนดค่าประจำตัวอ้างอิงของมัน (Address) 2 แบบ คือ แบบ 16 bit address และ 64 bit address ปกติแล้ว Xbee ทุกตัวจะถูกกำหนดค่ามาจากโรงงานเป็น Address 64 bit อยู่แล้ว ซึ่วจะสามารถอ่านค่าได้จาก parameter SH+SL การใช้งาน Address 64 bit สามารถทำได้โดยกำหนด parameter MY ให้มีค่า 0xFFFF หรือ 0xFFFE ส่วน การกำหนด 16 bit addressนั้นทำได้โดย กำหนด parameter MY ให้มีค่าน้อยกว่า 0xFFFE โดยจะเรียกเป็น mode การทำงาน 2 ประเภทคือ

- Unicast Mode คือ การรับส่งข้อมูล โดยอาศัยหลักการ Acknowledgement คือหากทางด้านส่งนั้น ส่งข้อมูลไป แต่ไม่ได้รับ Ack ตอบกลับจากตัวรับ ก็จะทำการส่งข้อมูลใหม่
- Broadcast Mode คือการส่งข้อมูลไปยังปลายทางให้ได้รับข้อมูลทุกตัว

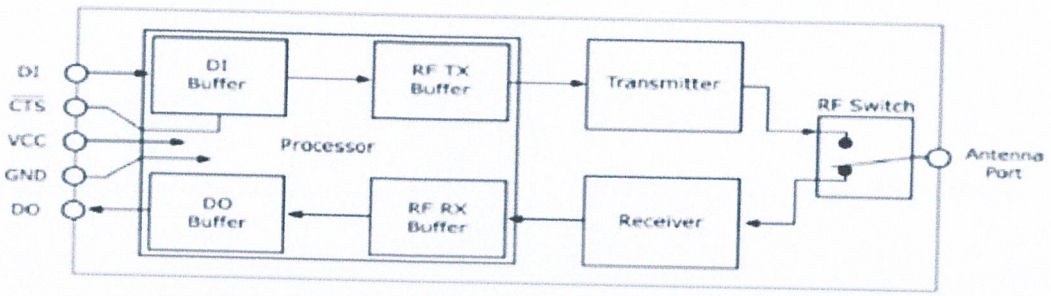
2.2.2.5 Xbee Operation Mode

Xbee จะสามารถแบ่งช่วงการทำงานได้เป็น 5 แบบ คือ

- Idle Mode โหมดนี้ จะเป็นโหมดที่ไม่ได้รับส่งข้อมูล ตัว Xbee เตรียมที่จะทำงานในโหมดอื่น ๆ ต่อไปทันที หากมีเงื่อนไขบางอย่าง
- Transmit / Receive Mode (พูดรวม 2 Mode ครับ) คือช่วงที่ Xbee มีการรับ หรือ ส่งข้อมูล โดยจะแบ่งลักษณะการทำงานย่อยออกเป็น Direct กับแบบ Indirect , การกำหนด Address ต้นทางและปลายทาง, Clear Channel Assessment และ การตอบรับ Acknowledgement
- Sleep Mode คือ ช่วงที่ Xbee อยู่ในสถานการณ์ทำงานพลังงานต่ำที่สุด เมื่อไม่มีการใช้งาน
- Command Mode คือ เป็นส่วนการปรับ parameter ของ Xbee ซึ่งจะมีการกำหนด 2 แบบคือ แบบ AT command กับแบบ API Command

2.2.2.6 Data Throughput ของ Xbee

Figure 2-03. Internal Data Flow Diagram



รูปที่ 2.11 Internal Data Flow Diagram

โดยทั่วไปการใช้งาน RF Module ควรจะกำหนดให้มี Buffer ด้วย เพื่อการปรับอัตรารับส่งข้อมูลระหว่างตอนที่รับส่งทางอากาศ กับตอนที่รับส่งไปยังไมโครคอนโทรลเลอร์ หรือ อุปกรณ์อื่น ๆ ได้อย่างเหมาะสม

Data ที่รับส่งระหว่าง MCU กับ Xbee จะมีข้อจำกัด เรื่อง Packet อาจถูก Drop ได้ เนื่องมากจาก Data Over Flow โดยสำหรับด้านการส่งข้อมูลไปที่ Xbee เพื่อออกอากาศนั้น ที่ ขา DI จะมี Buffer อยู่ประมาณ 202 Bytes หากส่งเกิน Buffer จะเกิดการ Drop packet ทิ้ง ซึ่งทางฝั่งรับข้อมูล ที่ขา DO ก็มี Buffer อยู่เช่นกัน โดยจะมี Parameter ที่เกี่ยวข้องกับ Data Throughput คือ RO และ BD

ค่า RO คือค่า Packetization Timeout ซึ่งเป็น delay ของข้อมูลที่อยู่ใน DI Buffer ก่อนที่จะถูก encapsulate ไปที่ส่วน RF transmission เพื่อส่งข้อมูลออกอากาศ หากตั้ง RO = 0 Data ที่รับเข้ามาจาก MCU จะถูก Xbee Encapsulate Packet ส่ง ออกอากาศทันที ดังนั้นจะมี Parameter RO และ BD ที่จะช่วยในการปรับ Data รับส่งให้สามารถรับส่งกันได้ทัน ไม่ให้มีการ Drop Packet ได้ ในกณรีที่ส่งข้อมูลเกิน 200 Bytes

นอกจากนี้ยังมี PIN CTS(ขา12) และ RTS(ขา16) ช่วยเตือนเวลาที่ Buffer ภายในใกล้จะเต็มด้วย โดยในฝั่งส่ง DI Buffer จะส่ง Signal มาทาง CTS เมื่อ DI Buffer เหลือพื้นที่จัดเก็บอยู่อีก 17 Bytes และส่ง Clear Signal ที่ CTS เมื่อ DI Buffer เหลือพื้นที่จัดเก็บมากกว่า 34 Bytes

ดังนั้น สำหรับการเขียนโปรแกรม รับส่งข้อมูลกับ Xbee ต้องคำนึงเรื่อง Buffer ด้วย แต่ในทางปฏิบัติ สำหรับงาน Sensor Network ก็ไม่ได้รับส่งข้อมูล Stream Data ยาวๆ

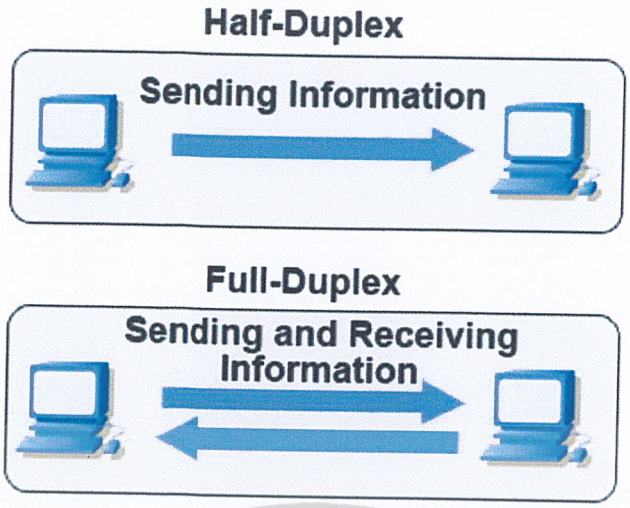
2.3 มาตรฐาน RS-232

RS-232 ย่อมาจาก Recommended Standard-232 (มาตรฐานแนะนำรุ่น 232) เป็นมาตรฐานการเชื่อมต่อข้อมูล แบบอนุกรม (Serial Port) กำหนดโดย EIA (Electronics Industry Association) หรือ สมาคมผู้ประกอบการอุตสาหกรรมอิเล็กทรอนิกส์ของอเมริกา ใช้กับการสื่อสารแบบจุดต่อจุด โดยใช้สายเชื่อมต่อ D-Type (DB) แบบ 25 และ 9 เข็ม ที่ไม่ประสานจังหวะระหว่างคอมพิวเตอร์กับ อุปกรณ์ต่อพ่วง มีการทำงานแบบสองทางพร้อมกัน (Full-duplex) โดยอาจใช้สายสัญญาณอื่นร่วมด้วย เพื่อทำแฮนด์เชค (Hand-shake) หรือไม่ก็ได้ทั้งนี้มาตรฐาน RS-232 จำกัดความยาวสายไว้ที่ 50 ฟุต (ประมาณ 15 เมตร) สำหรับการส่งสัญญาณที่ความเร็ว 19,200 บิตต่อวินาที โดยที่ความยาวสายจะต้องสั้นลงถ้าต้องการสื่อสารที่ความเร็วสูงขึ้น RS-232 มีจุดเริ่มต้นจากความต้องการที่จะกำหนดมาตรฐานการเชื่อมต่อระหว่างคอมพิวเตอร์กับโมเด็มในสมัยนั้น ตัวมาตรฐานจะกำหนดสิ่งที่เกี่ยวข้องกับการเชื่อมต่อนี้ด้วยกันทั้งหมด 4 หัวข้อหลักๆ ด้วยกันคือ

- คุณสมบัติทางไฟฟ้าของสัญญาณ
- คุณสมบัติทางกลของการเชื่อมต่อซึ่งหมายถึงตัวคอนเน็กเตอร์นั่นเอง
- หน้าที่การทำงานของวงจรสำหรับแลกเปลี่ยนข้อมูล
- มาตรฐานการเชื่อมต่อสำหรับระบบสื่อสารเฉพาะอย่าง

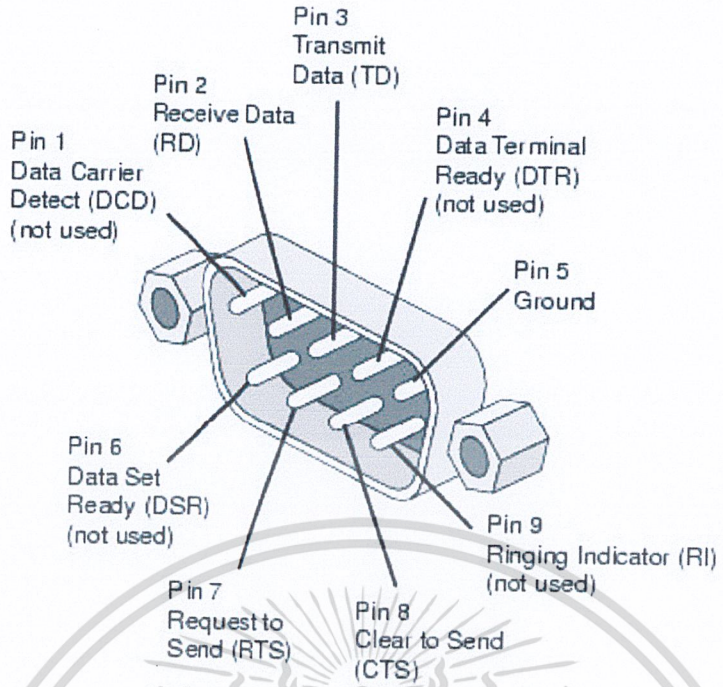
รูปที่ 2.12 หัวต่อ DB9 male และ female

มาตรฐาน RS-232-C เป็นมาตรฐาน RS-232 ที่มีการปรับปรุงแก้ไขจากมาตรฐานเดิม ซึ่งเราอาจคุ้นเคยกับชื่อนี้มากกว่า RS-232-A หรือ RS-232-B อันที่จริงแล้วยังมีมาตรฐาน RS-232-D ที่ใหม่กว่า RS-232-C โดยที่มีการเพิ่มข้อกำหนดของคอนเน็กเตอร์แบบ DB เข้าไปด้วย เช่น DB-25 ซึ่งในขณะนั้นสิทธิบัตรของตัวคอนเน็กเตอร์แบบนี้ได้หมดอายุลงพอดี จึงสามารถรวมข้อกำหนดเข้าไว้ได้



รูปที่ 2.13 แสดงการสื่อสารแบบ Half-Duplex กับ Full-Duplex

ลักษณะโดยทั่วไปของการเชื่อมต่อข้อมูลแบบอนุกรมตามมาตรฐาน RS-232 คือเป็นการสื่อสารข้อมูลแบบจุดต่อจุด ซึ่งเดิมทีเป็นการสื่อสารข้อมูลระหว่างคอมพิวเตอร์กับโมเด็ม ซึ่งจริงๆ แล้วทั้งสองฝั่งจะเป็นอะไรก็ได้ การสื่อสารเป็นแบบสองทางพร้อมกัน (Full-duplex) โดยอาจใช้สายสัญญาณอื่นร่วมเพื่อทำ แฮนด์เชค (Hand-shake) หรือไม่ก็ได้ มาตรฐาน RS-232 จำกัดความยาวสายไว้ที่ 50 ฟุต (ประมาณ 15 เมตร) สำหรับการส่งสัญญาณที่ความเร็ว 19,200 บิตต่อวินาที โดยที่ความยาวสายจะต้องสั้นลงถ้าต้องการสื่อสารที่ความเร็วสูงขึ้น และถ้ามีสัญญาณรบกวนมากๆ เช่นในโรงงาน หรือบริเวณใกล้เครื่องจักรที่เป็นแบบมีการสวิทช์สัญญาณไฟฟ้าที่กระแสสูงๆ ก็จะทำให้ต้องมีการลดความเร็วในการส่งสัญญาณลงหรือใช้สายที่สั้นลง



รูปที่ 2.14 แสดงตำแหน่งขา DB9 male

ตารางที่ 2.1 แสดงรายละเอียดของขาต่างๆ ของคอนเน็กเตอร์แบบ 9 ขา

D-Type 9 Pin	สัญลักษณ์	ชื่อสัญญาณ
Pin 1	CD	Carrier Detect
Pin 2	RD	Receive Data
Pin 3	TD	Transmit Data
Pin 4	DTR	Data Terminal Ready
Pin 5	SG	Signal Ground
Pin 6	DSR	Data Set Ready
Pin 7	RTS	Request To Send
Pin 8	CTS	Clear To Send
Pin 9	RI	Ring Indication

รายละเอียดของสายสัญญาณประกอบด้วย

Carrier Detect (CD) : ขานี้จะทำงานเมื่อมีการส่งสัญญาณ Carrier จากโมเด็ม

Receive Data (RD) : ใช้สำหรับรับข้อมูลอนุกรมเข้ามายังคอมพิวเตอร์

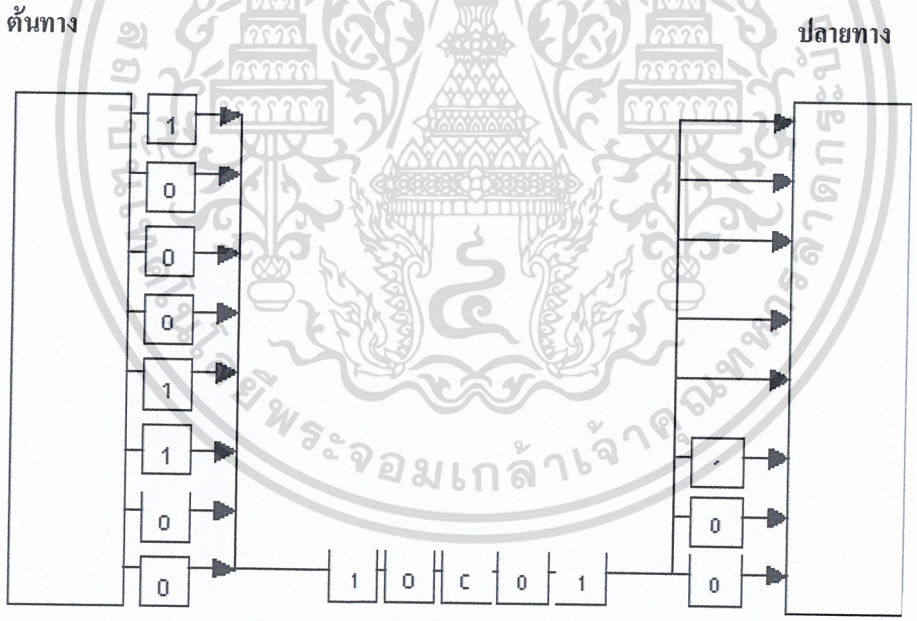
Transmit Data (TD) : ใช้สำหรับส่งข้อมูลอนุกรมออกจากคอมพิวเตอร์

Data Terminal Ready (DTR) : ใช้บอกให้อุปกรณ์ปลายทางรับรู้ว่าต้องการติดต่อด้วยโดย ขา DTR นี้ ต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทาง

Signal Ground (SG)	: เป็นกราวด์ของระบบ
Data Set Ready (DSR)	: ใช้ตรวจสอบการเชื่อมต่อระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง จะใช้คู่กับขา DTR
Request To Send (RTS)	: ใช้สำหรับส่งข้อมูลไปยังอุปกรณ์ปลายทางเพื่อร้องขอให้อุปกรณ์ปลายทางส่งข้อมูลกลับมา
Clear To Send (CTS)	: ใช้ตรวจสอบว่าอุปกรณ์ที่เชื่อมต่อด้วยพร้อมจะรับข้อมูลหรือไม่ โดยจะคอยรับสัญญาณ RTS เมื่อทุกอย่างพร้อมก็จะทำการส่งข้อมูลออกทางขา TD
Ring Indication (RI)	: ขานี้จะทำงานเมื่อโมเด็มได้รับสัญญาณเรียกเข้า

2.3.1 การส่งผ่านข้อมูลแบบอนุกรม (Serial Transimission)

รูปแบบการส่งผ่านข้อมูลในลักษณะนี้ทุกบิตที่เข้ารหัสแทนข้อมูลหนึ่งตัวอักษรจะถูกส่งผ่านไปตามสายส่งเรียงลำดับกันไปทีละบิตในสายส่งเพียงเส้นเดียว ดังรูป



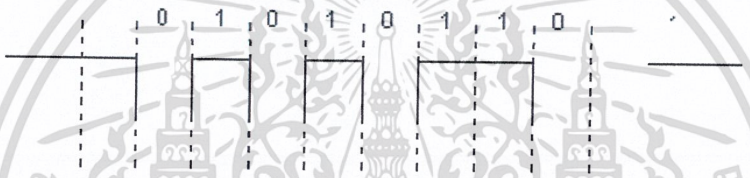
รูปที่ 2.15 การส่งข้อมูลแบบอนุกรม

จากรูป ตัวอักษรจะประกอบด้วย 8 บิต เรียงเป็นลำดับ ข้อมูลจะถูกส่งออกมาทีละบิตระหว่างต้นทาง และปลายทาง และปลายทางจะรวบรวมบิตเหล่านี้ทีละบิตจนครบ 8 บิต เป็น 1 ตัวอักษร จะเห็นว่าการส่งข้อมูลแบบนี้จะช้ากว่าแบบขนาน แต่ค่าใช้จ่ายจะถูกกว่าแบบขนาน ซึ่งเหมาะสำหรับการส่งระยะทางไกลๆ โดยทั่วไปแล้วการส่งข้อมูลนั้นจะประกอบไปด้วยกลุ่มของตัวอักษร ดังนั้นในการส่งข้อมูลแบบอนุกรมนี้จึงเกิดปัญหาขึ้นว่า แล้วต้นทางและปลายทางจะ

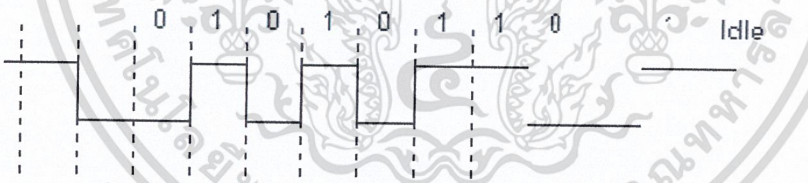
ทราบได้อย่างไรว่า จะแบ่งแต่ละตัวอักษรตรงบิตใด จึงเกิดวิธีการสื่อสารข้อมูลขึ้น 2 แบบคือ การสื่อสารแบบอะซิงโคนัส (Asynchronous Transmission) และการสื่อสารแบบซิงโคนัส (Synchronous Transmission)

2.3.1.1 การสื่อสารแบบอะซิงโคนัส (Asynchronous Transmission)

การสื่อสารแบบอะซิงโคนัส หรือเรียกอีกอย่างหนึ่งว่าเป็น การสื่อสารแบบระบุจุดเริ่มต้น และจุดสิ้นสุด (Start-Stop Transmission) ลักษณะของสัญญาณที่ใช้ในการติดต่อสื่อสารกันจะประกอบไปด้วย บิตเริ่มต้น (start bit) บิตของข้อมูลที่สื่อสาร (transmission data) จำนวน 8 บิต บิตตรวจข้อผิดพลาด (parity bit) และบิตสิ้นสุด (stop bit) สำหรับบิตตรวจสอบข้อผิดพลาดจะใช้หรือไม่ใช้ก็ได้ ดังนั้นสัญญาณจึงต้องประกอบด้วยส่วนประกอบอย่างน้อย 3 ส่วน ดังรูป



รูปที่ 2.16 การสื่อสารแบบอะซิงโคนัสที่ไม่ได้ใช้พารีตี้บิต



รูปที่ 2.17 การสื่อสารแบบอะซิงโคนัสที่ใช้พารีตี้บิต

จากรูป จะเห็นว่าขณะที่ไม่มีข้อมูลส่งออกมาสถานะของการส่งจะเป็นแบบว่าง (Idle) ซึ่งจะมีระดับของสัญญาณเป็น 1 ตลอดเวลา เพื่อความแน่ใจว่าปลายทาง หรือฝ่ายรับยังคงติดต่อกับต้นทาง หรือฝ่ายส่งอยู่ เมื่อเริ่มจะส่งข้อมูลสัญญาณของอะซิงโคนัสจะเป็น 0 หนึ่งช่วงสัญญาณนาฬิกา ซึ่งบิตนี้เราเรียกว่าบิตเริ่มต้น ตามหลังของบิตเริ่มต้นจะเป็นบิตข้อมูลสำหรับ 1 ตัวอักษร ตามหลังบิตข้อมูลก็จะเป็นบิตตรวจข้อผิดพลาด แล้วจะตามด้วยบิตสิ้นสุด ถ้าไม่ใช่บิตตรวจข้อผิดพลาด ตามหลังบิตข้อมูลก็จะเป็นบิตสิ้นสุดเลย หลังจากนั้นถ้าไม่มีข้อมูลส่งออกมาสัญญาณจะกลับไปอยู่ที่สถานะแบบว่างอีก เพื่อรอการส่งข้อมูลต่อไป จะเห็นว่าการสื่อสารแบบอะซิงโคนัสนี้ มีลักษณะเป็นไปทีละตัวอักษร และสัญญาณที่ส่งออกมา มีบางส่วนเป็นบิต

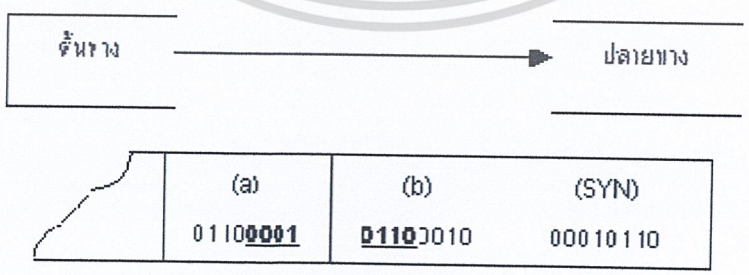
เริ่มต้น บิตสิ้นสุด และบิตตรวจข้อผิดพลาด ทำให้ความเร็วในการส่งข้อมูลต่อวินาทีน้อยลงไป เนื่องจากต้อง สูญเสียช่องทางการสื่อสารให้กับ บิตเริ่มต้น บิตสิ้นสุด และบิตตรวจข้อผิดพลาด (ถ้ามีใช้) ตลอดเวลา การสื่อสาร แบบอะซิงโคนัสนี้มักใช้ในการติดต่อระหว่างคอมพิวเตอร์กับอุปกรณ์รอบข้าง

2.3.1.2 การสื่อสารแบบซิงโคนัส (Synchronous Transmission)

การสื่อสารแบบซิงโคนัส มักใช้ในการติดต่อระหว่างคอมพิวเตอร์จะทำการจัดกลุ่มของข้อมูลเป็นกลุ่มๆ และทำการส่งข้อมูลทั้งกลุ่มไปพร้อมกันในทีเดียว เราเรียกกลุ่มของข้อมูลนี้ว่า บล็อกของข้อมูล (Block of Data) ซึ่งตัวอักษรตัวแรก และตัวถัดไปที่อยู่ในบล็อกเดียวกันจะไม่มีอะไรมาคั่นเหมือนอย่างแบบอะซิงโค นัส ที่ต้องใช้บิตเริ่มต้น และบิตสิ้นสุดคั่นทุกๆ ตัวอักษร แต่จะมีข้อมูลเริ่มต้นซึ่งเป็นลักษณะของบิตพิเศษที่ส่งมาเพื่อให้รู้ว่านั้น คือ จุดเริ่มต้นของกลุ่มตัวอักษรที่กำลังส่งเรียงกันเข้ามา เช่น อักขระซิง (SYN character) โดยที่อักขระซิงมีรูปแบบบิต คือ 00010110 ตัวอย่างของการส่งแสดงได้ดังรูป

รูปที่ 2.18 การสื่อสารแบบซิงโคนัส

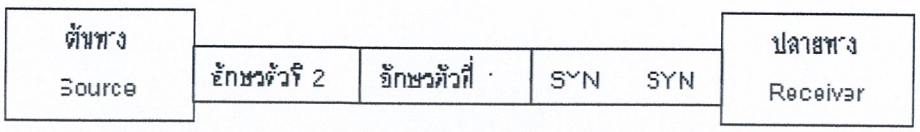
จากรูปเมื่อลายทางตรวจพบอักขระซิง หรือ 00010110 แล้วจะทราบได้ทันทีว่าบิตที่ตามมาคือบิตตัวอักษรแต่ละตัว แต่การใช้อักขระซิงเพียงตัวเดียวอาจเกิดข้อผิดพลาดได้ เช่น ถ้าเราส่งตัวอักษร b และตัวอักษร a ติดต่อกันไป ซึ่งตัวอักษร b มีรูปแบบบิตคือ 01100010 และตัวอักษร a มีรูปแบบบิตคือ 01100001 การส่งจะแสดงได้ดังรูป



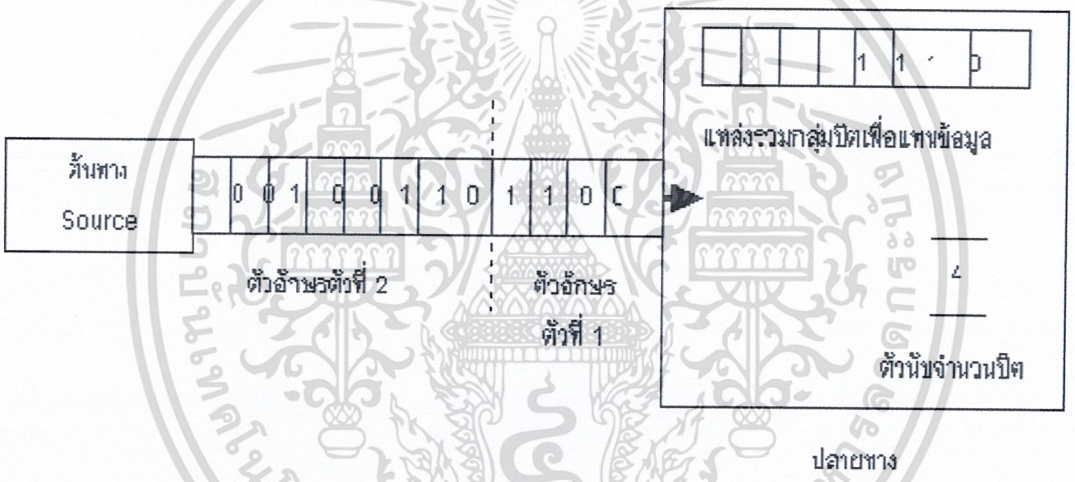
รูปที่ 2.19 การสื่อสารแบบซิงโคนัส

จะเห็นว่าเครื่องปลายทางจะตรวจพบอักขระซิงระหว่างบิตของตัวอักษร b และตัวอักษร a ทำให้เข้าใจว่าบิตต่อไปจะเป็นบิตของกลุ่มข้อมูล ซึ่งจะทำให้การรับข้อมูลนั้นเกิด

ผิดพลาดขึ้นได้ ดังนั้นจึงแก้ปัญหาด้วยการใช้อักขระซิง 2 ตัวต่อกันเป็นลักษณะของบิตพิเศษที่บอกให้ทราบว่าเป็นจุดเริ่มต้นบิตของกลุ่ม ข้อมูล ตัวอย่างของการใช้อักขระซิง 2 ตัวในการสื่อสารแบบซิงโครนัส และการตัดแถวของบิตข้อมูลออกเป็นกลุ่มทีละ 8 บิต เพื่อแทนข้อมูลแสดงได้ดังรูป

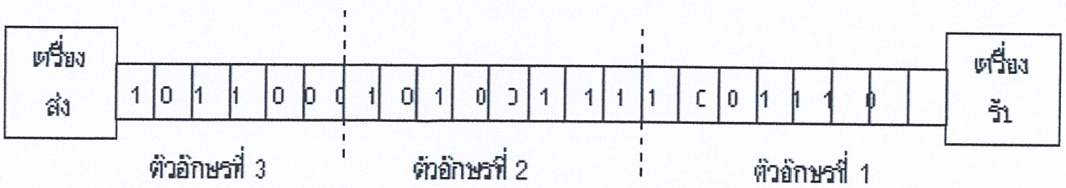


รูปที่ 2.20 ตัวอย่างการใช้อักขระซิง 2 ตัวในการสื่อสารแบบซิงโครนัส

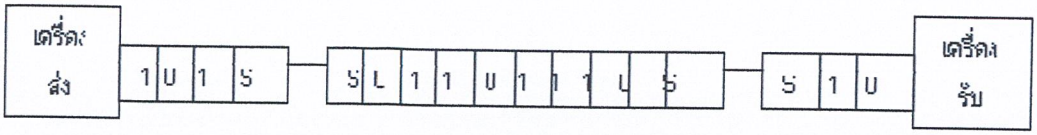


รูปที่ 2.21 แสดงการตัดแถวของบิตออกเป็นกลุ่มๆ ละ 8 บิต

2.3.1.3 ประสิทธิภาพของการส่งผ่านข้อมูลแบบอะซิงโครนัส และแบบซิงโครนัส



รูปที่ 2.22 การส่งผ่านข้อมูลแบบซิงโครนัส



รูปที่ 2.23 การส่งผ่านข้อมูลแบบอะซิงโคนัส

จากรูปที่ 2.22 แสดงให้เห็นว่าการส่งผ่านข้อมูลแบบซิงโคนัสนั้นส่วนมากแล้วตลอดทางของสายส่งจะใช้ส่งผ่านข้อมูลเต็มตลอดทั้งสาย ส่วนรูปที่ 2.23 แสดงให้เห็นว่าการส่งผ่านข้อมูลแบบอะซิงโคนัสนั้นสายส่งจะขาดความต่อเนื่อง ของสัญญาณข้อมูลที่ส่งผ่าน หรือถ้ามีสัญญาณข้อมูลที่ส่งผ่านต่อเนื่องกันเต็มตลอดทั้งสายแล้ว ก็จะสูญเสียช่องทางในการส่งไปกับการส่งบิตเริ่มต้น และบิตสิ้นสุดของแต่ละตัวอักษร ตัวอย่างเช่น กรณีที่ส่งผ่านข้อมูลที่อยู่ในรูปของรหัส ASCII ซึ่งตัวอักษรหนึ่งตัวถูกแทนด้วย 8 บิต ถ้ามีการส่งกลุ่มของข้อมูล 240 ตัวอักษร ในกรณีการส่งผ่านข้อมูลแบบซิงโคนัสมีการใช้ตัวอักขระซิง 3 ตัว และการส่งผ่าน ข้อมูลแบบอะซิงโคนัสไม่มีการใช้บิตตรวจข้อผิดพลาด ดังนั้นเราสามารถคำนวณหาอัตราส่วนระหว่างการส่งข้อมูลได้ ดังนี้

บิตทั้งหมดของตัวอักษรที่ส่งจะได้ 240 ตัวอักษร x 8 บิตต่อตัวอักษร = 1920 บิต

- แบบซิงโคนัส

  - บิตของตัวอักขระซิงที่ใช้จะได้ SYN 3 ตัว เท่ากับ 3 x 8 บิต = 24 บิต
  - ผลรวมของบิตที่ต้องส่งทั้งหมด = 1920 + 24 = 1944 บิต
  - อัตราส่วนระหว่างการส่งข้อมูลที่ต้องส่งจริง กับจำนวนบิตทั้งหมดที่จำเป็นต้องส่งคือ1920 หารด้วย 1944 จะได้ประมาณ 99 %

- แบบอะซิงโคนัส

  - บิตเริ่มต้น และบิตสิ้นสุดที่ใช้จะได้ 2 x 240 = 480 บิต
  - ผลรวมของบิตที่ต้องส่งทั้งหมด = 1920 + 480 = 2400 บิต
  - อัตราส่วนระหว่างการส่งข้อมูลที่ต้องส่งจริง กับจำนวนบิตทั้งหมดที่จำเป็นต้องส่งคือ1920 หารด้วย 2400 จะได้ประมาณ 80 %

2.3.1.4 การใช้บิตตรวจข้อผิดพลาด

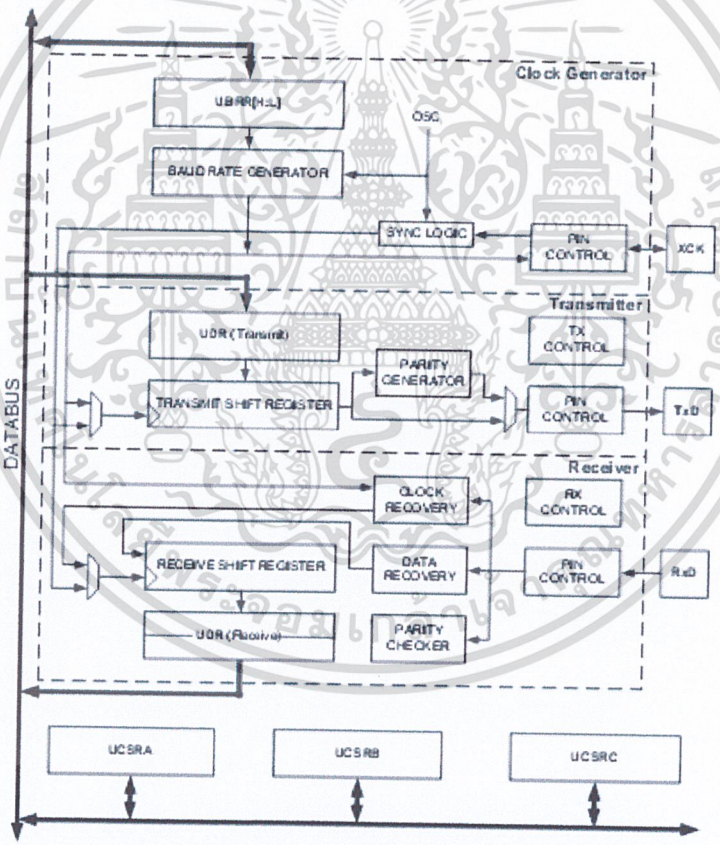
บิตตรวจข้อผิดพลาด หรือพารีตี้บิต จะเป็นบิตที่ใช้เพื่อทำหน้าที่ตรวจสอบความถูกต้องของข้อมูลที่ส่ง ซึ่งมีอยู่ 2 แบบด้วยกันคือ การตรวจสอบจำนวนคี่ (odd parity) และการตรวจสอบจำนวนคู่ (even parity)

2.3.1.4.1 การตรวจสอบจำนวนคี่ (Odd parity) หมายถึง บิตตรวจสอบจะต้องนับบิตที่มีค่าของ 1 สำหรับกลุ่มของบิตที่จะส่งและต้องการตรวจสอบอยู่เป็นจำนวนคี่ เช่น ถ้านับบิตที่มีค่าของ 1 ในกลุ่มของบิตที่จะส่ง และต้องการ ตรวจสอบได้เป็นจำนวนคู่ บิตตรวจสอบนี้จะต้องมีค่าเป็น 1 เพื่อที่จะรวมเป็นจำนวนคี่ แต่ถ้าจำนวนนับได้เป็นจำนวนคี่ บิตตรวจสอบก็จะมีค่าเป็น0 ตัวอย่าง สมมุติว่าถ้าข้อมูลที่ต้องการส่งมี 7 บิต คือ 0110011 บิตตรวจสอบจำนวนคี่จะต้องมีค่าเป็น 1 เพราะนับบิตที่มีค่าของ 1 ได้เท่ากับ 4 ตัว ซึ่งเป็นเลขคู่ เมื่อรวมกับบิตตรวจสอบจำนวนคี่ที่มีค่าเป็น 1 ก็จะนับได้เป็น 5 ตัว ซึ่งเป็นเลขคี่และการส่งข้อมูลพร้อมบิตตรวจสอบไปจะได้เป็น 10110011

2.3.1.4.2 การตรวจสอบจำนวนคู่ (Even parity) หมายถึง บิตตรวจสอบจะต้องนับบิตที่มีค่าของ 1 สำหรับกลุ่มของบิตที่จะส่งและต้องการตรวจสอบอยู่เป็นจำนวนคู่ เช่น ถ้านับบิตที่มีค่าของ 1 ในกลุ่มของบิตที่จะส่งและต้องการ ตรวจสอบได้เป็นจำนวนคู่ บิตตรวจสอบนี้จะต้องมีค่าเป็น 0 เพื่อที่จะรวมเป็นจำนวนคู่ แต่ถ้าจำนวนนับได้เป็นจำนวนคี่ บิตตรวจสอบก็จะมีค่าเป็น 1 ตัวอย่าง สมมุติว่าถ้าข้อมูลที่ต้องการส่งมี 7 บิต คือ 0110011 บิตตรวจสอบจำนวนคู่จะต้องมีค่าเป็น 0 เพราะนับบิตที่มีค่าของ 1 ได้เท่ากับ 4 ตัว ซึ่งเป็นเลขคู่ การส่งข้อมูลพร้อมบิตตรวจสอบไปจะได้เป็น 00110011 การตรวจสอบความถูกต้องทำได้โดยระหว่างต้นทางและปลายทางจะต้องตกลงกันว่าจะใช้ตัวตรวจข้อผิดพลาดชนิดใด ถ้าใช้ตัวตรวจข้อผิดพลาดแบบจำนวนคี่แล้วเมื่อปลายทางรับข้อมูลจะตรวจสอบ จำนวนบิตที่มีค่าเป็น 1 ว่าเป็นจำนวนคี่หรือไม่ ถ้าไม่เป็นจำนวนคี่แสดงว่าข้อมูลเกิดความผิดพลาดขึ้น ปลายทางจะต้องแจ้งให้ต้นทางทราบ อาจจะให้ต้นทางส่งข้อมูลมาใหม่อีกครั้ง ส่วนการใช้ตัวตรวจข้อผิดพลาดแบบจำนวนคู่ก็จะใช้หลักการคล้ายๆ กัน

2.4 การสื่อสารข้อมูลอนุกรมผ่านโมดูล USART/UART

ไมโครคอนโทรลเลอร์ AVR สามารถที่จะสื่อสารข้อมูลแบบอนุกรมได้โดยการใช้โมดูล USART (Universal Synchronous and Asynchronous serial Receiver and Transmitter) เพื่อสื่อสารข้อมูลผ่านทางพอร์ตอนุกรมได้ทั้งแบบซิงโครนัส (ข้อมูลมีความต่อเนื่อง มีการกำหนดสัญญาณมาตรฐานที่เหมือนกันทั้งทางด้านรับและด้านส่ง เพื่อให้การรับส่งมีความสัมพันธ์กัน) และอะซิงโครนัส (ข้อมูลที่ไม่จำเป็นต้องต่อเนื่องมีบิตเริ่มต้น (Start Bit) บิตข้อมูล (Data Bit) และบิตหยุด (Stop bit) มีบิตพาริตี้ (Parity Bit) หรือไม่ก้อได้) สำหรับไมโครคอนโทรเลอร์ ATmega128 ขาพอร์ต PE0(RXD0) และ PE1(TXD0) จำนวน 1 ช่อง ส่วนที่เหลืออีก 1 ช่อง จะต่อกับสัญญาณ PD2(RXD1) และ PD3(TXD1) ใช้ในการส่งข้อมูลอนุกรม บล็อกไดอะแกรมของโมดูล USART แสดงดังรูปที่ 2.24



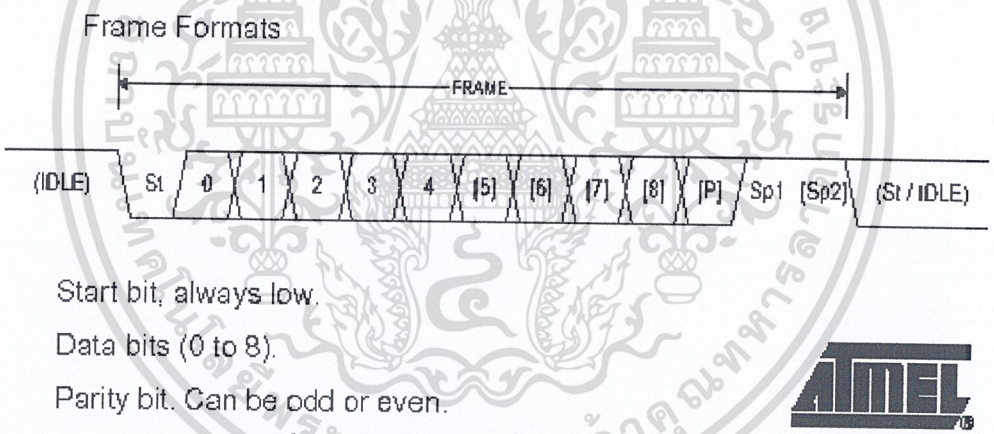
รูปที่ 2.24 บล็อกไดอะแกรมโมดูล USART

จากรูปที่ 2.24 จะเห็นว่าโมดูล USART แบ่งออกเป็น 3 ส่วนด้วยกันคือ

- ส่วนสร้างสัญญาณนาฬิกา (Clock Generator) เพื่อใช้ในการกำหนดอัตราบอดในการรับส่งข้อมูลโดยสามารถกำหนดได้ทั้งภายในและภายนอก ผ่านทางขา XCK (Transfer Clock)
- ส่วนส่งข้อมูลอนุกรม (Transmitter) โดยส่งข้อมูลออกทางขาพอร์ต TxD
- ส่วนส่งข้อมูลอนุกรม (Receiver) โดยส่งข้อมูลออกทางขาพอร์ต RxD

และมีริสจิตเตอร์ควบคุมการทำงาน 3 ตัวประกอบไปด้วย UCSRA, UCSRB และ UCSRC การส่งข้อมูลอนุกรมในรูปแบบอะซิงโครนัส จะเป็นการส่งข้อมูลเป็นเฟรม ลักษณะของเฟรมข้อมูลอนุกรมนี้ประกอบไปด้วย

- บิตเริ่มต้นข้อมูล (Strat Bit)
- บิตข้อมูล (Data Bit)
- พาริตี้บิต (Parity Bit)
- บิตหยุดข้อมูล (Stop Bit)



รูปที่ 2.25 รูปแบบเฟรมข้อมูลอนุกรม

2.4.1 คุณสมบัติที่สำคัญของโมดูล USART

- การสื่อสารข้อมูลแบบฟูลดูเพลกซ์ (Full Duplex) ตัวรับและตัวส่งแยกอิสระต่อกัน สามารถรับและส่งข้อมูลได้ในเวลาเดียวกัน (พร้อมกัน)
- ทำงานได้ทั้งในโหมดซิงโครนัสและอะซิงโครนัส
- มีคุณสมบัติของพอร์ตอนุกรมครบถ้วน เช่น การกำหนดบิตข้อมูล การกำหนดบิตหยุด และการกำหนดบิตพาริตี้ เป็นต้น
- มีส่วนตรวจสอบความผิดพลาดของเฟรมข้อมูลและข้อมูลโอเวอร์รัน (Framing Error and Data OverRun Detection)

- โหมดการสื่อสารแบบมัลติโปรเซสเซอร์
- โหมดทวีคุณความเร็วในการสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารอนุกรมโหมดอะซิงโครนัสเป็นการสื่อสารข้อมูลระหว่างไมโครคอนโทรลเลอร์กับพอร์ตอนุกรมหรือที่นิยมนำไมโครคอนโทรลเลอร์มาเชื่อมต่อกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม (RS-232) นอกจากการกำหนดจำนวนบิตข้อมูล บิตหยุดและพาริตี้บิตแล้ว จะต้องมีการกำหนดอัตราเร็วในการรับส่งข้อมูลที่เรียกว่า อัตราบอตหรือบอตเรต (baud rate) หรือการเปลี่ยนแปลงของสัญญาณใน 1 วินาทีซึ่งสามารถคำนวณหาได้ดังตารางที่ 2.2

ตารางที่ 2.2 การคำนวณหาอัตราบอตเรต

โหมดการทำงาน	**การคำนวณหาอัตราบอตเรต**	**การคำนวณหาค่า UBRR**
โหมดอะซิงโครนัสปกติ (U2X)	BAUD = fose/16(UBRR+1)	UBRR = (fosc/16BAUD)-1
โหมดอะซิงโครนัสทวีคูณ (U2X=1)	BAUD = fose/8(UBRR+1)	UBRR = (fosc/8BAUD)-1
โหมดมาสเตอร์ซิงโครนัส	BAUD = fose/2(UBRR+1)	UBRR = (fosc/2BAUD)-1

โดย BAUD : อัตราบอดตเรตในหน่วยบิตต่อวินาที

Fosc : ความถี่สัญญาณนาฬิกาหลักของระบบ

UBRR : รีจิสเตอร์ UBRRH และ UBRRL (0-4095)

การทำงานของโมดูล USART กับสัญญาณนาฬิกา หากเป็นการใช้งานสัญญาณนาฬิกาภายใน (Internal Clock) หรือสัญญาณนาฬิกาหลัก (fose) จะใช้งานกับอะซิงโครนัสโหมดและมาสเตอร์ซิงโครนัส และสัญญาณนาฬิกาภายนอก (Extermal Clock) ใช้งานในโหมดสเลฟซิงโครนัส โดยความถี่สัญญาณนาฬิกาภายนอกจะต้องน้อยกว่าความถี่สัญญาณนาฬิกาหลักหารด้วย 4(f_{XCK}<fosc/4)

2.4.2 รีจิตเตอร์ที่เกี่ยวข้องกับการทำงานในโหมด USART

- รีจิสเตอร์ UDR (USART I/O Data Register)

  รีจิสเตอร์อ่านเขียนข้อมูลขนาด 8 บิต โดยแบ่งออกเป็น 2 ตัว คือ RXB ใช้รับข้อมูลจากภายนอกเข้ามาในไมโครคอนโทรลเลอร์และ TXB ใช้สำหรับส่งข้อมูลออกจากไมโครคอนโทรลเลอร์ การอ่านเขียนข้อมูลจะทำกับรีจิสเตอร์ UDR โดยตรง

- รีจิสเตอร์ UCSRA (USART Control and Status Register A)

รีจิสเตอร์ควบคุมการทำงานและแสดงสถานการณ์ทำงานของโมดูล USART ชุด A เกี่ยวข้องกับสถานะของการสื่อสารข้อมูล

- รีจิสเตอร์ UCSRB (USART Control and Status Register B)
  รีจิสเตอร์ควบคุมการทำงานและแสดงสถานการณ์ทำงานของโมดูล USART ชุด B เกี่ยวข้องกับบิตกำหนดอินเตอร์รัปต์และการกำหนดขนาดของข้อมูลแบบ 9 บิตข้อมูล(Data Bit)
- รีจิสเตอร์ UCSRC (USART Control and Status Register C)
  รีจิสเตอร์ควบคุมการทำงานและแสดงสถานการณ์ทำงานของโมดูล USART ชุด C เกี่ยวข้องกับการกำหนดอัตราบอตในการรับส่งข้อมูล
- รีจิสเตอร์ UBRRL และ UBRRH (USART Baud Rate Register)
  รีจิสเตอร์กำหนดอัตราบอต

2.4.3 การใช้งานโมดูล USART โหมดอะซิงโครนัส

การใช้งานโมดูล USART ในโหมดอะซิงโครนัส หรือการรับส่งข้อมูลในรูปแบบมาตรฐาน RS-232 (Com port) จะต้องมีการกำหนดค่าในรีจิสเตอร์ต่อไปนี้

ตารางที่ 2.3 แสดงรีจิสเตอร์ UDR (USART I/O Data Register)

บิตที่	7	6	5	4	3	2	1	0
(Read)	RXB[7:0]							
(Write)	TXB[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
ค่าเริ่มต้น	0	0	0	0	0	0	0	0

2.4.3.1 รีจิสเตอร์ UDR (USART I/O Data Register)

รีจิสเตอร์รับส่งข้อมูล เมื่อมีการเขียนข้อมูลไปที่รีจิสเตอร์ UDR ข้อมูลจะถูกเขียนไปที่ TXB (Transmit Data Buffer Resgister) และเมื่อมีการอ่านข้อมูลจากรีจิสเตอร์ UDR จะไปอ่านข้อมูลที่เก็บอยู่ RXB (Receive Data Buffer Resgister)

ตารางที่ 2.4 แสดงรีจิสเตอร์ UCSRA (USART Control and Status Register A)

บิตที่	7	6	5	4	3	2	1	0
ชื่อบิต	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM
Read/Write	R	R/W	R	R	R	R	R/W	R/W
ค่าเริ่มต้น	0	0	0	0	0	0	0	0

2.4.3.2 รีจิสเตอร์ UCSRA (USART Control and Status Register A)

- บิตที่ 7 : บิต RXC (USART Recrive Complete)
  บิต RXC ถูกเซตเมื่อยังไม่ได้อ่านข้อมูลในบัฟเฟอร์รับข้อมูลและเคลียร์เมื่อบัฟเฟอร์ว่าง (จะไม่สามารถเก็บข้อมูลใหม่ได้หากข้อมูลยังไม่ได้อ่านออกไป)
- บิตที่ 6 : บิต TXC (USART Transmit Complete)
  บิต TXC ถูกเซตเมื่อข้อมูลในบัฟเฟอร์ส่งข้อมูลได้ส่งออกไปแล้วและไม่มีข้อมูลในบัฟเฟอร์ ส่งข้อมูล (UDR) บิต TXC จะเคลียร์โดยอัตโนมัติเมื่อการส่งข้อมูลในอินเตอร์รัปต์ส่วนส่งข้อมูลทำงานเสร็จสมบูรณ์ หรือ โดยการเขียนค่าไปที่บิตโดยตรง
- บิตที่ 5 : บิต UDRE (USART Data Register Empty)
  หากบิต IDRE ถูกเซตเป็นหนึ่งแสดงว่า รีจิสเตอร์ UDR ว่างพร้อมรับข้อมูลใหม่แล้ว (บัฟเฟอร์ข้อมูลว่าง)
- บิตที่ 4 : บิต FE (Frame Error)
  บิต FE ถูกเซตเมื่อเฟรมข้อมูลผิดพลาดหรือบิตหยุดข้อมูลเป็นศูนย์ และบิต FE จะถูกเคลียร์เมื่อบิตหยุดข้อมูลเป็นหนึ่ง
- บิตที่ 3 : บิต DOR (Data OverRun)
  บิต DOR ถูกเซตเมื่อเกิดเงื่อนไขข้อมูล Over Run หรือบัฟเฟอร์รับข้อมูลเต็มและมีข้อมูลรอเข้ามาในบัฟเฟอร์รับข้อมูล
- บิตที่ 2 : บิต PE (Parity Error)
  บิต PE ถูกเซตเมื่อเกิดข้อผิดพลาดของบิตพาริตี้
- บิตที่ 1 : บิต U2X (Double the USART Transmission Speed)
  บิต U2X แสดงอัตราการทวีคูณของการสื่อสารข้อมูล มีผลกับโหมดอะซิงโครนัส (เคลียร์บิต U2X เป็นศูนย์เมื่อใช้งานในโหมดซิงโครนัส)

- บิตที่ 0 : บิต MPCM (Multi-process Communication Mode)
  เซตบิต MPCM เมื่อต้องการใช้งานการสื่อสารข้อมูลแบบมัติโปรเซสเซอร์

ตารางที่ 2.5 แสดงรีจิสเตอร์ UCSRB (USART Control and Status Register B)

บิตที่	7	6	5	4	3	2	1	0
ชื่อบิต	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
ค่าเริ่มต้น	0	0	0	0	0	0	0	0

2.4.3.3 รีจิสเตอร์ UCSRB (USART Control and Status Register B)

- บิตที่ 7 : บิต RXCIE (RX Complete Interrupt Enable)
  เซตบิต RXCIE เป็นหนึ่งเพื่อเปิดใช้งานอินเตอร์รัปต์ที่เกี่ยวข้องกับแฟลก RCX (บิตที่ 7 ในรีจิสเตอร์ UCSRA) เมื่อมีการรับข้อมูลในอินเตอร์รัปต์เสร็จสมบูรณ์
- บิตที่ 6 : บิต TXCIE (TX Complete Interrupt Enable)
  เซตบิต TXCIE เป็นหนึ่งเพื่อเปิดใช้งานอินเตอร์รัปต์ที่เกี่ยวข้องกับแฟลก TCX (บิตที่ 6 ในรีจิสเตอร์ UCSRA) เมื่อมีการรับข้อมูลในอินเตอร์รัปต์เสร็จสมบูรณ์
- บิตที่ 5 : บิต UDRIE (USART Data Resgister Empty Interrupt Enable)
  เซตบิต UDRIE เป็นหนึ่งเพื่อเปิดใช้งานอินเตอร์รัปต์ที่เกี่ยวข้องกับแฟลก UDRE (บิตที่ 5 ในรีจิสเตอร์ UCSRA) เมื่อรีจิสเตอร์ UDR ว่าง
- บิตที่ 4 : บิต RXEN (Receiver Enable)
  เซตบิต RXEN เป็นหนึ่งเมื่อต้องการใช้งานอินเตอร์รัปต์ เนื่องจากการรับข้อมูล
- บิตที่ 3 : บิต TXEN (Transmitter Enable)
  เซตบิต TXEN เป็นหนึ่งเมื่อต้องการใช้งานอินเตอร์รัปต์ เนื่องจากการส่งข้อมูล
- บิตที่ 2 : บิต UCSZ2 (Character Size)

บิต UCSZ2 ทำงานร่วมกับบิต UCSZ1 (ในรีสจิสเตอร์ UCSRC) ในการกำหนดจำนวนข้อมูล (Data Bit) ในการรับส่ง

- บิตที่ 1 : บิต RXB8 (Register Data Bit 8)
  บิต RXB8 บิตที่เก้าของจำนวนข้อมูลรับ เมื่อทำงานในโหมดเก้าบิตข้อมูล (จะต้องอ่านก่อนที่จะอ่านบิตต่ำสุดของข้อมูลในรีจิสเตอร์ UDR)
- บิตที่ 0 : บิต TXB8 (Transmiit Data Bit 8)
  บิต TXB8 บิตที่เก้าของจำนวนข้อมูลส่ง เมื่อทำงานในโหมดเก้าบิตข้อมูล (จะต้องอ่านก่อนที่จะเขียนบิตต่ำสุดของข้อมูลในรีจิสเตอร์ UDR)

ตารางที่ 2.6 แสดงรีจิสเตอร์ UCSRC (USART Control and Status Register C)

บิตที่	7	6	5	4	3	2	1	0
ชื่อบิต	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
ค่าเริ่มต้น	1	0	0	0	0	1	1	0

2.4.3.4 รีจิสเตอร์ UCSRC (USART Control and Status Register C)

- บิตที่ 7 : บิต URSEL (Resgister Select)
  บิต URSET เลือกการเข้าถึงรีจิสเตอร์ UCSRC หรือ UBRRH เซต บิต URSEL เป็นหนึ่งเมื่อเขียนข้อมูลไปที่รีจิสเตอร์ UCSRC
- บิตที่ 6 : บิต UMSEL (USART Mode Select)
  บิต UMSEL เลือดโหมดการทำงานระหว่างอะซิงโครนัสโหมด (UMSEL=1) กับซิงโครนัสโหมด (UMSEL=0)
- บิตที่ 5:4 : บิต UPM 1:0 (Parity Mode)
  บิต UPM ใช้กำหนดพาริตี้บิต

ตารางที่ 2.7 แสดงการเซตบิต กำหนดพาริตี้บิต

UPM1	UPM0	พาริตี้บิต
0	0	ไม่ใช้งานพาริตี้บิต
0	1	สงวนไว้
1	0	พาริตี้คู่ (Even Parity)
1	1	พาริตี้คี่ (Odd Parity)

- บิตที่ 3 : บิต USBS (Stop Bit Select)

  บิต USBS บิตกำหนดจำนวนบิตหยุด (Stop Bit) โดย USBS=0 บิตหยุดข้อมูล 1 บิต, USBS=1 บิตหยุดข้อมูล 2 บิต
- บิตที่ 2:1 : บิต UCSZ1:0 (Character Size)

  บิตกำหนดจำนวนบิตข้อมูลรับส่ง

ตารางที่ 2.8 แสดงการเซตบิต กำหนดจำนวนบิตข้อมูล

UCSZ2	UCSZ1	UCSZ0	จำนวนบิตข้อมูล (บิต)
0	0	0	5
0	0	1	6
0	1	0	7
0	1	1	8
1	0	0	สงวนไว้
1	0	1	สงวนไว้
1	1	0	สงวนไว้
1	1	1	9

- บิตที่ 0 : บิต UCPOL (Clock Polarity)

  เซตบิต UVPOL เมื่อมีการใช้งานโหมดซิงโครนัส จะสัมพันธ์กับการเปลี่ยนแปลงตัวอย่างข้อมูลเอาต์พุต ข้อมูลอินพุต และสัญญาณนาฬิกาซิงโครนัส (XCX)

ตารางที่ 2.9 แสดงการเซตบิต กำหนดขอบขาสัญญาณ XCX

UCPOL	การส่งข้อมูล (ขา TXD)	การรับข้อมูล (ขา RXD)
0	ขอบขาขึ้น ของ XCX	ขอบลงขึ้น ของ XCX
1	ขอบลงขึ้น ของ XCX	ขอบขาขึ้น ของ XCX

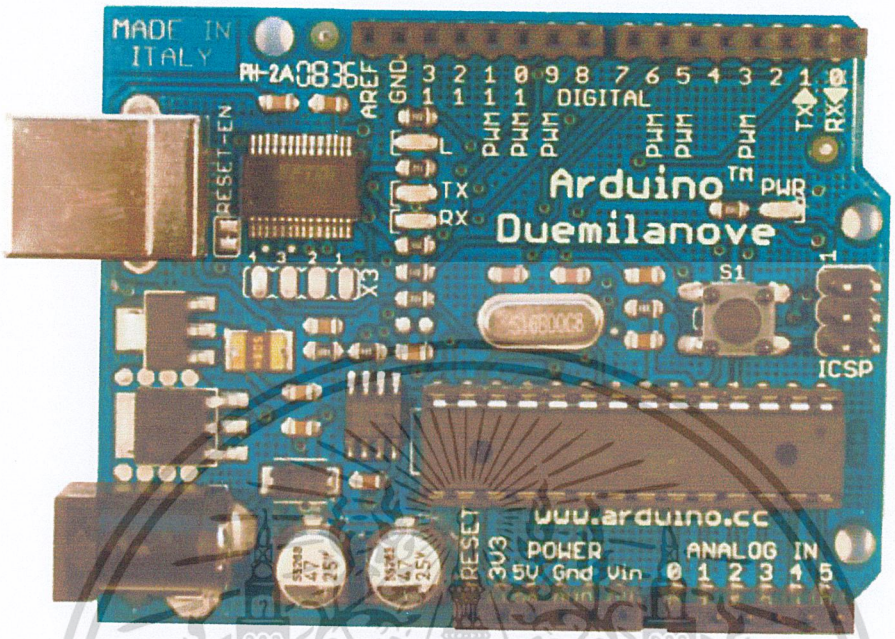
ตารางที่ 2.10 แสดงการเซตบิต กำหนดขอบขาสัญญาณ XCX

บิตที่	15	14	13	12	11	10	9	8
ชื่อบิต	URSEL	-	-	-	UBRR[11:8]			
	UBRR[7:0]							
	7	6	5	4	3	2	1	0
Read/Write	R/W R/W	R R/W	R R/W	R R/W	R/W R/W	R/W R/W	R/W R/W	R/W R/W
ค่าเริ่มต้น	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0

2.4.3.5 รีจิสเตอร์ UBRRL และUBRRH (USART Baud Rate Register)

- บิตที่ 15 : บิต URSEL (Register Select)
  บิต URSEL เลือกการเข้าถึงรีจิสเตอร์ UBRRH หรือ UCSRC เคลียร์
  บิต URSEL เป็นศูนย์เมื่อเขียนข้อมูลไปที่รีจิสเตอร์ UBRRH
- บิตที่ 14:12 : Reeserved Bit
  บิตที่ 12 ถึง 14 สงวนการใช้งานไว้
- บิตที่ 11:0 : บิต UBRRH 11:0 (USART Baud Rate Register)
  บิต UBRR ใช้ในการกำหนดอัตราบอดในการรับส่งข้อมูลอนุกรม
  คำนวณหาค่า UBRR ได้จาก ตารางที่ 2.2

2.5 ATmega328 Dev Board - Arduino Duemilanove Compatible



รูปที่ 2.26 บอร์ด Arduino รุ่น Duemilanove

Arduino เป็นภาษาอิตาลี ซึ่งใช่เป็นชื่อของโครงการพัฒนาไมโครคอนโทรลเลอร์ตระกูล AVR แบบ Open Source ที่ได้รับการปรับปรุงมาจากโครงดารพัฒนา Open Source ของ AVR อีกโครงการหนึ่งที่ชื่อว่า Wiring แต่เนื่องจากโครงการของ Wiring เลือกใช้ AVR เบอร์ ATmega128 ซึ่งเป็นไมโครคอนโทรลเลอร์ ที่มีจำนวนหน่วยความจำ และ Input/Output ค่อนข้างมาก และที่สำคัญ ATmega128 เป็นชิพที่มีตัวถังแบบ SMD จึงทำให้เป็นอุปสรรคสำหรับผู้เริ่มต้นในการสร้างบอร์ดและต่อวงจรขึ้นมาใช้งานกันเอง และบอร์ดจะมีขนาดค่อนข้างใหญ่ ซึ่งอาจดูว่าเกินความจำเป็นสำหรับผู้เริ่มต้น จึงไม่ค่อยได้รับความนิยมเท่าที่ควร แต่หลังจากที่ทางทีมงาน Arduino นำ Source Code ของ Wiring มาพัฒนาปรับปรุงใหม่ โดยให้สามารถใช้งานกับไมโครคอนโทรลเลอร์ AVR ขนาดเล็ก อย่าง Mega8 และ Mega168 ได้ จึงทำให้ระบบวงจรของบอร์ดมีขนาดเล็กลงกว่า Wiring มากและยังใช้อุปกรณ์น้อยชิ้น ทำให้ง่ายต่อการต่อวงจรใช้งานกันเอง และยังประหยัดต้นทุนในการสร้างบอร์ดไปได้มาก ด้วยเหตุผลนี้เองที่ทำให้ Arduino ได้รับความนิยมจากผู้ใช้งานทั่วโลกเป็นอย่างมาก ในระยะเวลารวดเร็ว

เนื่องจาก Arduino เป็นภาษาอิตาลี ซึ่งมีสำเนียงการอ่านออกเสียงที่เป็นรูปแบบเฉพาะและยังไม่มีการกำหนดเป็นคำภาษาไทยขึ้นมาอย่างเป็นทางการ ถึงแม้ว่า Arduino จะเป็นที่รู้จักของคนไทยระยะเวลาหนึ่งแล้วตามที แต่ก็ยังไม่มีคำอ่านที่เป็นภาษาไทย อย่างเป็นทางการว่า คำๆ

นี้ ควรอ่านออกเสียงเป็นว่าอย่างไร บางคนอ่านว่า อาร์-ดู-วี-โน่ บางคนอ่านว่า อา-เดีย-โน บางคนอ่าน เอ –อา-ดู-ไอ-โน และอื่นๆอีกมากมาย

Arduino มีจุดเด่นในเรื่องของ ความง่ายในการเรียนรู้และใช้งาน เนื่องจากมีการออกแบบคำสั่งต่างๆขึ้นมาสนับสนุนการใช้งาน ด้วยรูปแบบที่ง่ายไม่ซับซ้อน ซึ่งถึงแม้ว่า Arduino เองจะมีรูปแบบการใช้งาน คล้ายๆกับไมโครคอนโทรลเลอร์อย่าง Basic Stamp ของ Parallax, BX-24 ของ Netmedias และ Handy Board ของ MIT แต่ก็มีจุดเด่นกว่ารายอื่นๆหลายอย่าง เป็นต้นว่า

- ราคาไม่แพง เนื่องจากมี Source Code และวงจร จากให้ฟรี สามารถต่อวงจรขึ้นมาใช้งานได้เอง
- โปรแกรมที่ใช้พัฒนาของ Arduino รองรับการทำงานทั้ง Windows,Linux และ Macintosh OSX
- มีรูปแบบคำสั่งที่ง่ายต่อการใช้งาน แต่สามารถนำไปใช้งานจริงๆที่มีความซับซ้อนมากๆได้ และยังสามารถสร้างคำสั่งและ Libray ใหม่ๆ ขึ้นมาใช้งานเองได้ เมื่อมีความชำนาญมากขึ้นแล้ว
- มีการเปิดเผยวงจรและ Source Code ทั้งหมดทำให้สามารถนำไปพัฒนาต่อยอดเพิ่มเติมได้ตามความต้องการทั้ง Hardware และ Software

Arduino เป็นบอร์ดไมโครคอนโทรลเลอร์ โดยใช้ AVR ขนาดเล็กเป็นตัวประพมวลผลและสั่งงานเหมาะสำหรับนำไปใช้ในการศึกษาเรียนรู้ระบบไมโครคอนโทรลเลอร์และ นำไปประยุกต์ใช้งานเกี่ยวกับการควบคุมอุปกรณ์ Input/Output ต่างๆ ได้มากมาย ทั้งในแบบที่เป็นการทำงานตัวเดียวอิสระ หรือ เชื่อมต่อสั่งงานร่วมกับอุปกรณ์อื่นๆ เช่น คอมพิวเตอร์ PC ทั้งนี้ก็เนื่องมาจากว่า Arduino สนับนุนการเชื่อมต่อกับอุปกรณ์ Input/Output ต่างๆได้มากมาย ทั้งแบบ Digital และ Analog เช่น การรับค่าจากสวิตซ์ หรือ อุปกรณ์ตรวจจับ (Sensor) แบบต่างๆ รวมไปถึง การควบคุมอุปกรณ์ Output ต่างๆตั้งแต่ LED, หลอดไฟม มอเตอร์ม รีเลย์ ฯลฯ โดยระบบฮาร์ดแวร์ของ Arduino สามารถสร้างและประกอบขึ้นใช้งานได้เอง ในกรณีที่ผู้ใช้พอมีความรู้ด้านอิเล็กทรอนิกส์อยู่บ้าง หรือ สามารถซื้อแผงวงจรสำเร็จรูปที่มีการผลิตออกจำหน่ายกันในราคาที่ไม่แพง สำหรับเรื่องของโปรแกรมที่จะใช้เป็นเครื่องมือในการพัฒนานั้น สามารถ Download มาใช้งานกันได้ฟรีโดยไม่เสียค่าใช้จ่ายใดๆ โดย Arduino จุดเด่น ในเรื่องของความง่ายต่อการพัฒนาโปรแกรมและมีเอกสารข้อมูลรวมทั้งตัวอย่างต่างๆให้ใช้เป็นแนวทางในการศึกษาเรียนรู้เป็นจำนวนมาก เนื่องจาก Arduino เป็นระบบการพัฒนาไมโครคอนโทรลเลอร์แบบ Open Source ซึ่งมีการตีพิมพ์เอกสารต่างๆที่เกี่ยวข้องออกมาเผยแพร่ให้ได้รับรู้เป็นระยะๆ รวมทั้งการเปิดเผย Source Code และตัวอย่างต่างๆ ให้ผู้ใช้นำไปใช้งาน หรือพัฒนาดัดแปลงต่อยอดได้โดยไม่เสียค่าใช้จ่าย ด้วยเหตุผลนี้จึงผู้คนทั่วไปให้ความสนใจ และนำไปศึกษาทดลองใช้งานกันมากมาย มี

การนำไปดัดแปลงและสร้างเป็นโครงงาน แบบต่างๆ กันเป็นจำนวนมาก จึงเป็นประโยชน์อย่างยิ่งสำหรับผู้เริ่มต้นที่สามารถนำเอาตัวอย่างและโครงงานต่างๆที่คนอื่นทำไว้แล้ว มาใช้อ้างอิงเป็นแนวทางในการศึกษาเรียนรู้ได้โดยงานและที่สำคัญ ฟรี ไม่เสียค่าใช้จ่าย

2.6 โหมดการติดต่อแบบ USB

USB นั้นย่อมาจาก Universal Serial Bus ซึ่งเป็นพอร์ตอเนกประสงค์ใช้รับส่งข้อมูลแบบอนุกรมมีความเร็วในการส่งข้อมูลสูง (สูงสุด 12 Mbits/sec) ลักษณะการต่อเข้ากับอุปกรณ์เป็นแบบ Hot-Swap คือสามารถถอดเปลี่ยนอุปกรณ์ได้ทันทีโดยไม่ต้องปิดเปิดเครื่องใหม่

2.6.1 ชนิดของ USB

USB ออกได้เป็น 3 ประเภท คือ USB Host, USB Hub และ USB Device

2.6.1.1 USB Host คือ ส่วนที่ทำหน้าที่จัดการและควบคุมการทำงานทั้งหมด

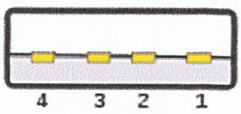
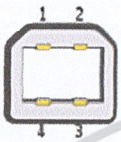
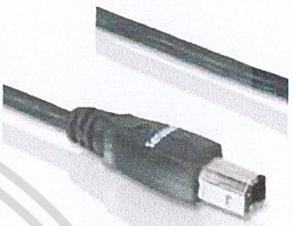
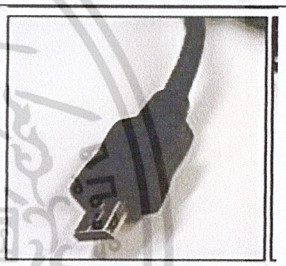
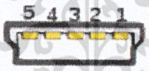
2.6.1.2 USB Hub คือ อุปกรณ์เพิ่มช่องสัญญาณ USB

2.6.1.3 USB Device คือ อุปกรณ์ USB ทั่วๆไปที่พบเห็นได้ง่ายๆ เช่น โมเด็ม, สแกนเนอร์ ในหนึ่งบัสสามารถมีอุปกรณ์ได้ถึง 127 ตัว

2.6.2 ประเภทของหัวต่อ USB

อุปกรณ์รอบข้างแบบ USB จะต้องมีการเชื่อมต่อกับช่องเสียบด้วยสัญญาณหากหัวต่อ ที่ปลายทั้งสองด้านของสาย USB เป็นแบบเดียวกัน ก็แสดงว่ามีการเชื่อมต่อสานระหว่างช่องเสียบ USB มาตรฐาน USB ซึ่งได้ออกแบบหัวต่อ เพื่อใช้งานอยู่ 4 แบบ

ตารางที่ 2.11 ประเภทหัวต่อ USB

ประเภทหัวต่อ	พอร์ตตัวเมีย	อุปกรณ์	พอร์ตตัวผู้
หัวต่อซีรี่ย์ A	 Type A	คอมพิวเตอร์ส่วนบุคคล (Personal Computer), คอมพิวเตอร์ชนิดพกพา (Notebook)	
หัวต่อซีรี่ย์ B	 Type B	ปริ้นเตอร์ (Printer), สแกนเนอร์ (Scanner)	
มินิหัวต่อซีรี่ย์ A	 Mini-A	กล่องรับส่งสัญญาณ (Set Top Box)	
มินิหัวต่อซีรี่ย์ B	 Mini-B	คอมพิวเตอร์ขนาดพกพา (Personal Digital Assistant)	

ถึงแม้จะมีการแบ่งหัวต่อออกเป็นหลายประเภท แต่ทุกประเภทมีสาย สัญญาณเหมือนกัน โดยใช้สายสองเส้นในการส่งไฟเลี้ยงให้แก่อุปกรณ์และสายอีกสองเส้นสำหรับการรับและส่งข้อมูล แต่เนื่องจากระบบ USB นั้นออกแบบมาเพื่อให้สามารถเชื่อมต่อหรือปลดอุปกรณ์ออกจากฐานระบบได้แม้อยู่ในระหว่างการใช้งาน ดังนั้นหน้าสัมผัสของหัวต่อจึงมีการออกแบบให้มีความสามารถพิเศษเล็กน้อย คือ หน้าสัมผัสของสายสองเส้นที่ใช้ส่งไฟเลี้ยงจะยื่นออกมามากกว่าหน้าสัมผัสของสายสองเส้นที่ใช่งไฟเลี้ยงจะยื่นออกมามากกว่าหน้าสัมผัสที่ใช้รับ-ส่งข้อมูล

การออกแบบหัวต่อเช่นนี้จะทำให้อุปกรณ์ได้รับไฟเลี้ยงเข้าไปก่อนที่จะได้รับสัญญาณข้อมูลเมื่อมีการเชื่อมต่ออุปกรณ์ตัวใหม่เข้าไปในระบบ ทำให้อุปกรณ์ที่เชื่อมต่อเข้าไปใหม่สามารถทำงานได้ทันทีโดยไม่ได้รับความเสียหาย (หากอุปกรณ์ได้รับสัญญาณข้อมูลก่อนที่จะได้รับไฟเลี้ยงแก่วงจรที่อยู่ภายในได้) ขาของหัวต่อแต่ละขามีหน้าที่ดังตารางที่ 2.9

ตารางที่ 2.12 หน้าที่ของขาหัวต่อ USB แต่ละขา

หมายเลขขา	ชื่อสัญญาณ	สีของสายสัญญาณ
1	ไฟเลี้ยง +5V	แดง
2	ข้อมูล – (D-)	ขาว
3	ข้อมูล + (D+)	เขียว
4	กราวด์	ดำ

2.6.3 โครงสร้างของ USB

USB เป็นมาตรฐานการรับส่งข้อมูลที่มีรูปแบบการเชื่อมต่อในระบบบัส คือ อุปกรณ์ทุกๆ ตัวจะต้องส่งสัญญาณรวมกันไปในสายสัญญาณเพียงคู่เดียว ดังนั้นอุปกรณ์ทุกๆตัวที่เชื่อมต่อกับบัสจะต้องส่งข้อมูลเรียงลำดับกันไปเพื่อไม่ให้เกิดการชนกันเนื่องจาก USB เป็นระบบบัสที่ใช้สายสัญญาณส่งเพียงคู่เดียว และส่งสัญญาณแบบผลต่าง (Differential Signaling) ทำให้ในช่วงเวลาหนึ่งๆ จะมีข้อมูลวิ่งไปได้เพียงทิศทางเดียวเท่านั้น ไม่สามารถเกิดการรับและส่งข้อมูลไปในเวลาเดียวกันได้ หรือที่เรียกกันว่าการส่งข้อมูลแบบฮาล์ฟดูเพล็กซ์ (Half Duplex)

สายสัญญาณ D+ และ D- จะมีขนาดสัญญาณเท่ากัน แต่มีเฟสต่างกัน 180 องศา ซึ่งลักษณะนี้เรียกว่า Differential Signaling โดย

- สำหรับ USB โหมดความเร็วสูง
  - ค่าของข้อมูลจะเป็น 1 เมื่อ แรงดันในสาย D+ มากกว่า D-
  - ค่าของข้อมูลจะเป็น 0 เมื่อ แรงดันในสาย D+ น้อยกว่า D-
- สำหรับ USB โหมดความเร็วต่ำ
  - ค่าของข้อมูลจะเป็น 1 เมื่อ แรงดันในสาย D+ น้อยกว่า D-
  - ค่าของข้อมูลจะเป็น 0 เมื่อ แรงดันในสาย D+ มากกว่า D-

จังหวะการรับส่งข้อมูลของระบบบัส USB ทั้งหมดถูกควบคุมโดยโฮสท์ (Host) การรับส่งข้อมูลจะถูกกำหนดเป็นเฟรมโดยทุกๆ 1 มิลลิวินาที จะเกิดการรับส่งข้อมูลขึ้นหนึ่งเฟรม ในแต่ละเฟรม จะแบ่งย่อยออกเป็นแพ็กเกต (Packet) โดยเริ่มต้นแต่เฟรมด้วยแพ็กเกต SOF (Start of Frame) แล้วตามด้วยแพ็กเกตข้อมูลที่จะรับหรือส่งของอุปกรณ์แต่ละตัวเรียงต่อกันไปเรื่อยๆ

แต่เนื่องจากแต่ละเฟรมข้อมูลจะต้องรับส่งให้เสร็จภายใน 1 มิลลิวินาที นั่นหมายความว่าข้อมูลของอุปกรณ์ทุกตัวที่เชื่อมต่อกับบัสจะต้องถูกกำหนดขนาดไม่ให้ใหญ่เกินกว่าจะสามารถส่งได้ภายใน 1 มิลลิวินาที และเล็กพอที่จะทำให้อุปกรณ์ทุกๆตัวสามารถใช้งานบัสไปพร้อมๆกันได้ ดังนั้นในระบบบัส USB จึงจำเป็นต้องอาศัยซอฟต์แสร์เข้ามาคอยจัดการจะต้องติดต่อผ่านเอ็น

พอยต์ (Endpoint) ไหน (อุปกรณ์แต่ละตัวจะมีชนิดและจำนวนเอ็นพอยต์ต่างกัน) ดังนั้นอุปกรณ์แต่ละตัวจะต้องมีไดร์เวอร์อุปกรณ์เป็นของตัวเอง

2.6.4 ไดร์เวอร์ยูเอสบี

การส่งข้อมูลต้องมีการแบ่งส่วนเวลากันอย่างพอเหมาะในเวลา 1 เฟรม เพื่อให้อุปกรณ์ทุกตัวสามารถทำงานไปได้พร้อมๆกัน ซอฟต์แวร์ที่ทำหน้าที่นี้คือ ไดรเวอร์ยูเอสบีนั่นเอง

ไดร์เวอร์อุปกรณ์ยูเอสบี ของอุปกรณ์แต่ละตัวจะส่งการร้องขอเพื่อการติดต่อมายังไดร์เวอร์ยูเอสบี รับทราบมันจะพิจารณาว่าในรอบการรับส่งข้อมูลหนึ่งๆ นั้นอุปกรณ์แต่ละตัวสามารถรับส่งข้อมูลได้มากแค่ไหน หากปริมาณข้อมูลที่ต้องการรับส่งมีขนาดใหญ่มากจะต้องถูกแบ่งออกเป็นส่วนย่อยๆแล้วเก็บไว้เพื่อรอส่งในรอบถัดไป

2.6.5 ไดร์เวอร์โฮสต์คอนโทรลเลอร์

หลังจากไดร์เวอร์ยูเอสบี พิจารณาแล้วว่าอุปกรณ์แต่ละตัวส่งข้อมูลได้เท่าใดบ้างมันจะส่งข้อมูลของอุปกรณ์แต่ละตัวที่ติดต่อในรอบการติดต่อนั้นๆ ลงมายังไดร์ฟเวอร์โฮลต์คอนโทรลเลอร์ ตัวไดร์ฟเวอร์โฮลต์คอนโทรลเลอร์จะจัดเรียงลำดับข้อมูลอุปกรณ์แต่ละชนิดลงเฟรมข้อมูลแบบ USB แล้วส่งข้อมูลทั้งหมดออกไปยังอุปกรณ์ต่างๆ

2.6.6 คุณสมบัติของ USB ในการเชื่อมต่ออุปกรณ์ทุกชนิดเข้ากับเครื่องคอมพิวเตอร์

- ความสามารถในการเชื่อมต่ออุปกรณ์หลายๆชนิดรวมเข้าสู่บัสสัญญาณข้อมูลเดียวกัน (สูงสุด 127 ตัว)
- ไม่เกิดการขัดแย้งกันสำหรับการเข้าใช้ทรัพยากรของระบบ (IRQ)
- ตรวจสอบการเชื่อมต่อและการตั้งค่าการทำงานต่างๆ โดยอัตโนมัติระหว่างที่เครื่องคอมพิวเตอร์กำลังทำงานอยู่ (Hot Attachment)
- ความเร็วในการส่งข้อมูลที่สูงถึง 12 Mbps สำหรับอุปกรณ์ความเร็วสูงและ 1.5 Mbps สำหรับอุปกรณ์ความเร็วต่ำ
- มีไฟเลี้ยงจ่ายให้ในระบบ ทำให้อุปกรณ์ที่ใช้ไฟเลี้ยงไม่มากนักไม่จำเป็นต้องอาศัยแหล่งจ่ายไฟจากภายนอก
- ค่าใช้จ่ายในการพัฒนาอุปกรณ์ต่ำเมื่อเทียบกับความเร็วที่ได้

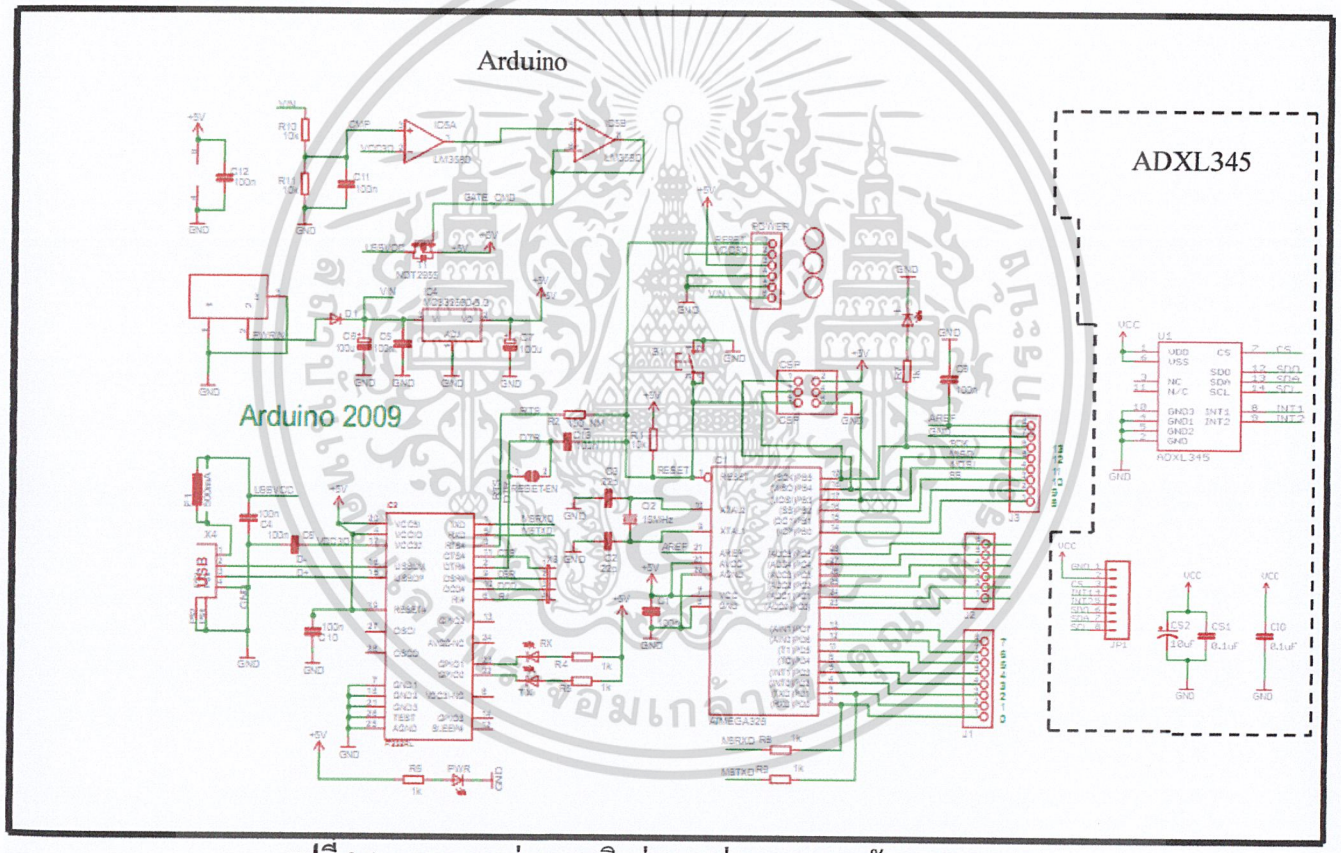
บทที่ 3

การออกแบบ

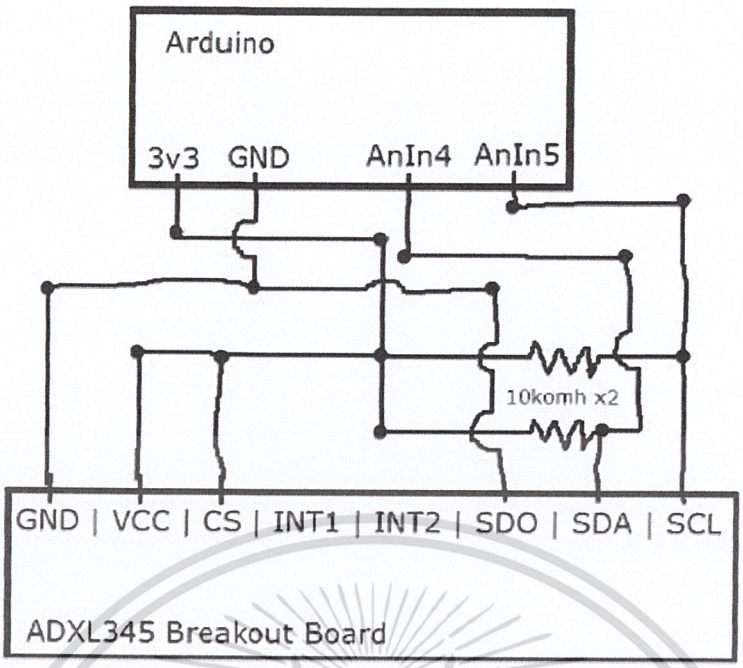
ในการออกแบบโครงงาน ผู้ออกแบบได้แบ่งออกเป็นส่วนการติดต่อ ดังนี้

3.1 การออกแบบฮาร์ดแวร์

3.1.1 ศึกษาการติดต่อระหว่างไมโครคอนโทรลเลอร์กับเซนเซอร์สามแกน



รูปที่ 3.1 แบบวงจรส่วนการติดต่อระหว่าง Arduino กับ ADXL345



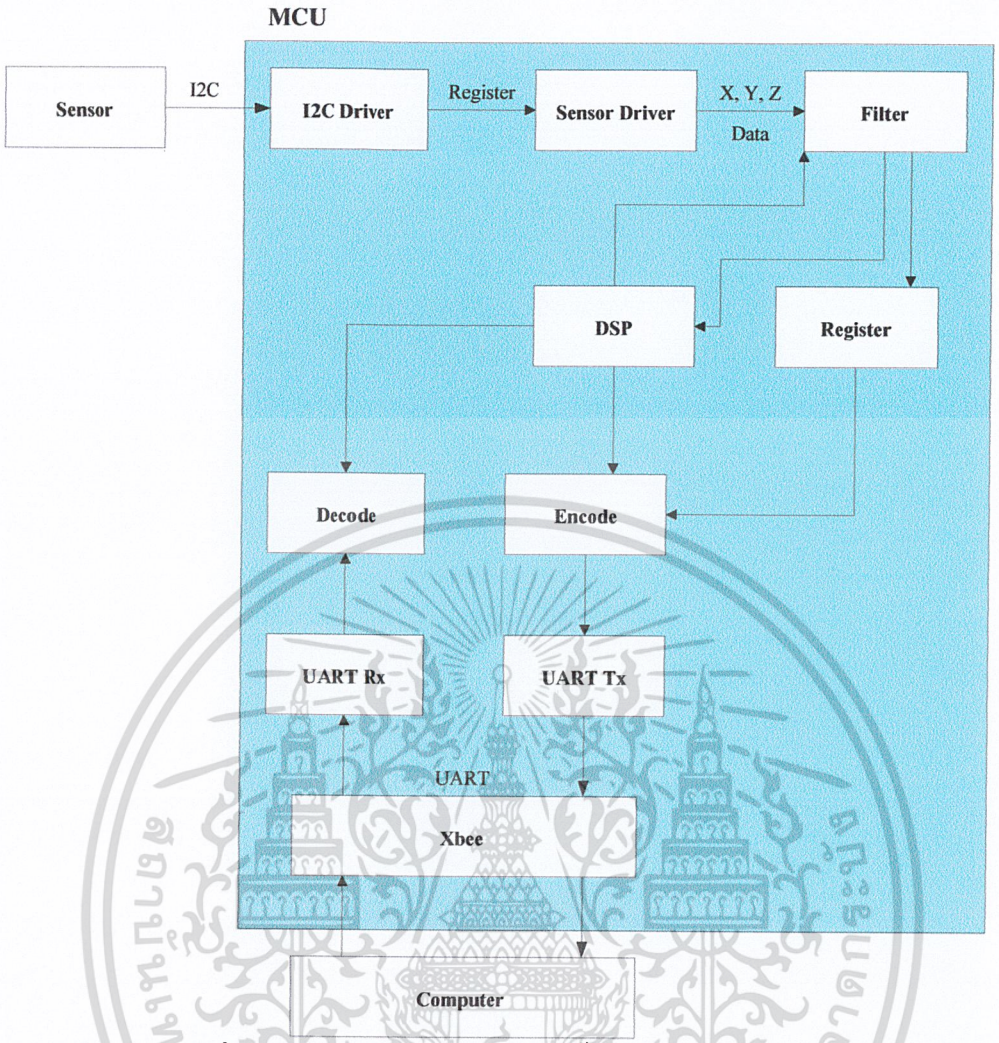
รูปที่ 3.2 การติดต่อระหว่าง Arduino กับ ADXL345

ในการติดต่อระหว่างไมโครคอนโทรลเลอร์กับเซนเซอร์ ซึ่งได้ออกแบบการส่งข้อมูลจากเซนเซอร์เข้าArduino ให้ส่งข้อมูลเป็นแบบ I^2C จึงใช้ตัวต้านทาน 10 komh มาใช้ในการติดต่อด้วยแล้วต่อเข้าขา Analog ขา4 และ ขา5 ดังรูปที่ 3.2 เพื่อส่งข้อมูลให้แสดงผล

3.1.2 ศึกษาการติดต่อระหว่างไมโครคอนโทรลเลอร์กับXbee

รูปที่ 3.3 การติดต่อระหว่างไมโครคอนโทรลเลอร์กับ Xbee

สามารถต่อ Xbee เข้ากับไมโครคอนโทรลเลอร์ได้โดยตรง แบบเบื้องต้นคือ DI(ขา3), DO(ขา2) กับ Supply Voltage ขา VCC(ขา1) และ GND(ขา10) โดยไม่ต้องใช้ CTS(ขา12) และ RTS(ขา16) ก็ได้

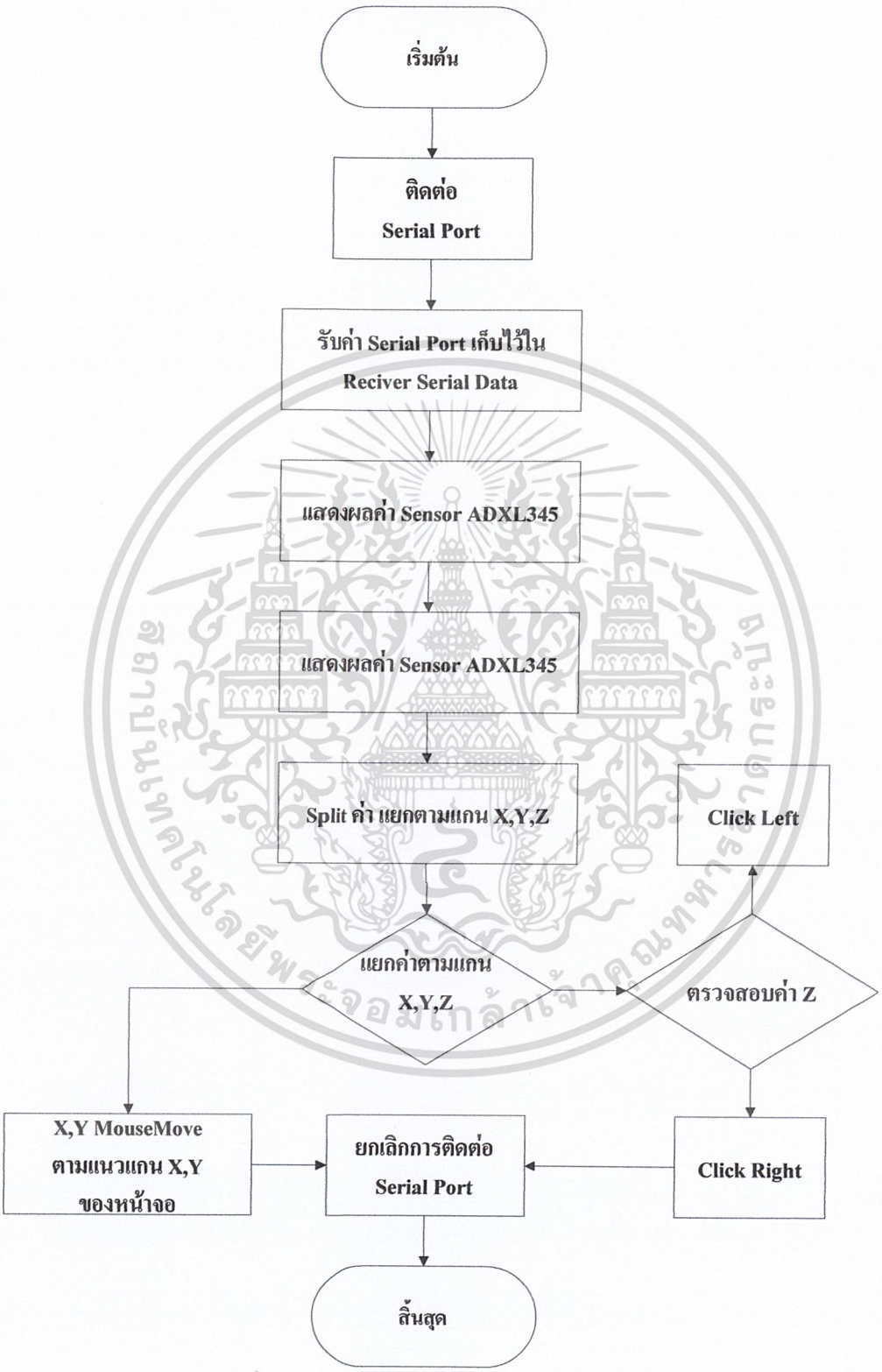


รูปที่ 3.4 การติดต่อระหว่างอุปกรณ์ทั้งหมดภายในระบบ

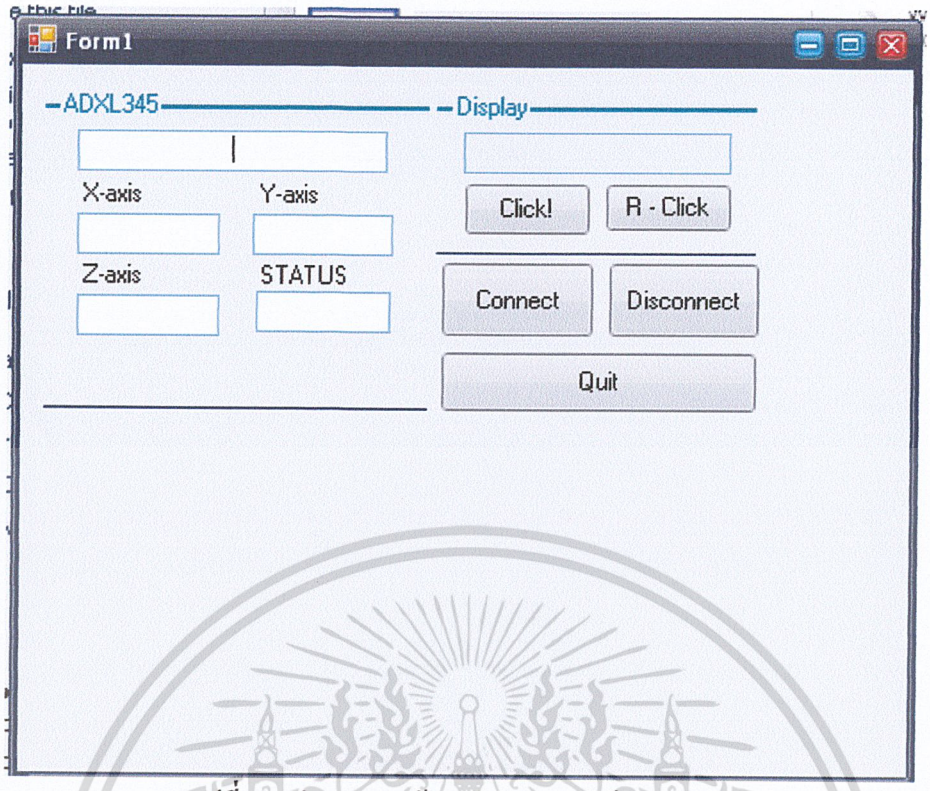
โดยในการพัฒนานั้น จะเป็นการพัฒนาทีละส่วนไปคือ จะทำการรับส่งข้อมูลผ่านทาง UART จากนั้นจะทำการติดต่อกับ Sensor เพื่อที่จะทำให้สามารถแสดงข้อมูลจากการติดต่อระหว่างอุปกรณ์ได้ถัดไปคือ ทำการ Read/Write Registers จากนั้นทำการ Decode และ Encode ข้อมูล และสุดท้ายคือ Configuration เพื่อให้ได้ค่าตามที่ต้องการ

โดยในแต่ละส่วนจะเป็นการทำงานเชื่อมโยงกันอยู่ 2 ส่วน คือ งานในส่วนล่าง ซึ่งจะเป็นการติดต่อระหว่างอุปกรณ์ทั้งหมดภายในระบบ คือ มีการติดต่อระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์ผ่านทาง UART และภายใน Hardware มีการติดต่อระหว่าง Sensor กับ MCU ผ่านทาง I^2C และงานในส่วนบนนั้น จะทำการติดต่อระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์เพื่อทำการ Read/Write Registers ผ่านการ Decode และ Encode ข้อมูล เพื่อให้ง่ายต่อการ Debug และการทดสอบระบบ และทำการติดต่อแบบไร้สายโดยการต่อ Xbee เข้าไป

3.2 การออกแบบซอฟต์แวร์



รูปที่ 3.5 Flowchart การทำงานของซอฟต์แวร์



รูปที่ 3.6 ส่วนของหน้าจอโปรแกรมให้การติดต่อ

ในส่วนของซอฟต์แวร์ได้ทำการเขียนโปรแกรมขึ้นมาให้ระบบwindowsนั้นสามารถติดต่อกับส่วนของฮาร์ดแวร์ได้ โดยทำการติดต่อทางพอร์ตอนุกรม แล้วให้แสดงค่าแกนของเซนเซอร์ทั้งสามแกน โดยค่าของแกน X, Y จะอยู่ในระนาบเดียวกับจอภาพเปรียบเสมือนการเลื่อนเมาส์ไปมาซ้าย-ขวา และ บน-ล่าง ส่วนแกน Z เปรียบการคลิกเมาส์ซ้ายและขวาตามที่กำหนด

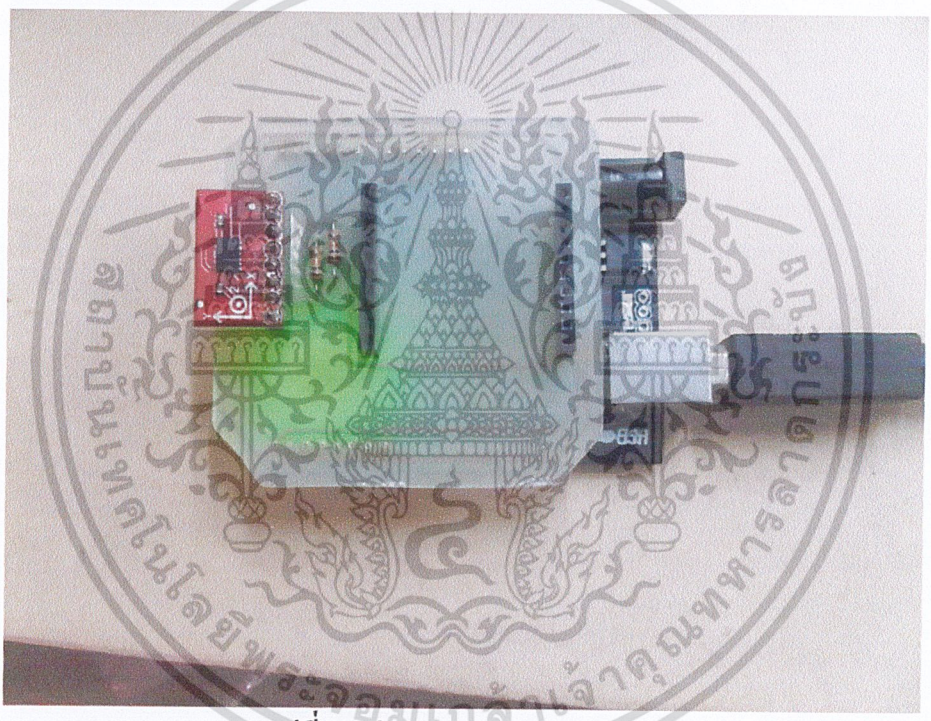
บทที่ 4

การทดลองและผลการทดลอง

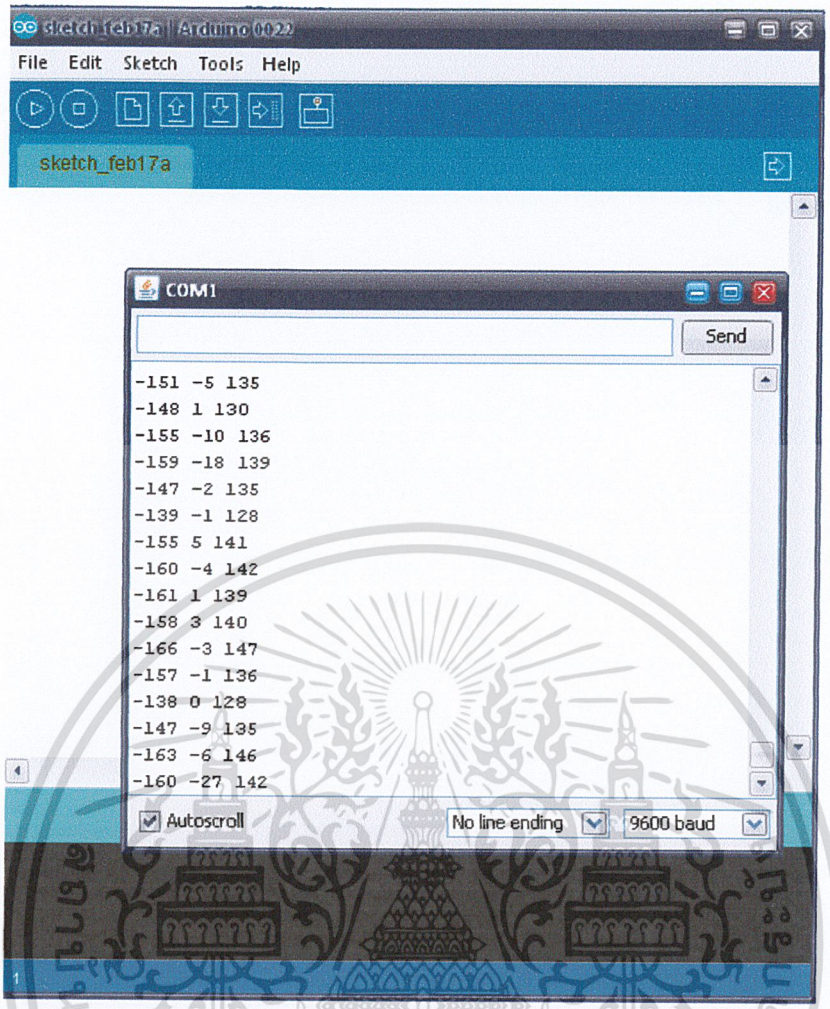
4.1 การทดลองการติดต่อSensor ADXL345 กับบอร์ด Arduino

4.1.1 ติดต่อผ่านทาง USB

ทำการเขียนโค้ดติดต่อเซนเซอร์กับบอร์ด Arduino เพื่อให้แสดงค่าแกน X, Y และZ โดยใช้โปรแกรม Arduino IDE



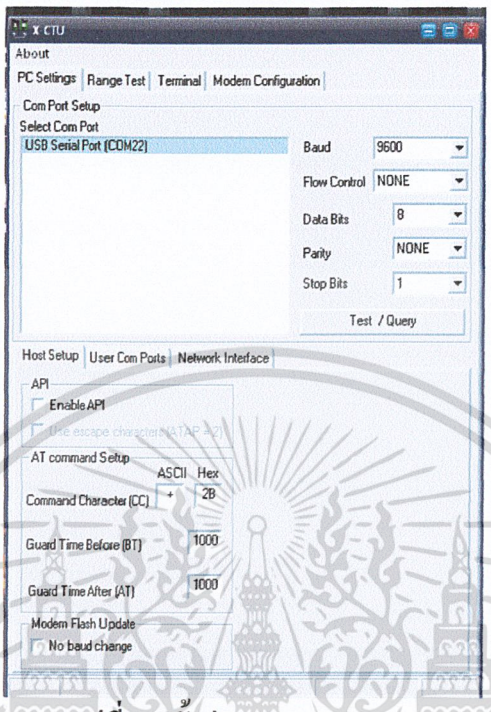
รูปที่ 4.1 ต่อSensor ADXL345



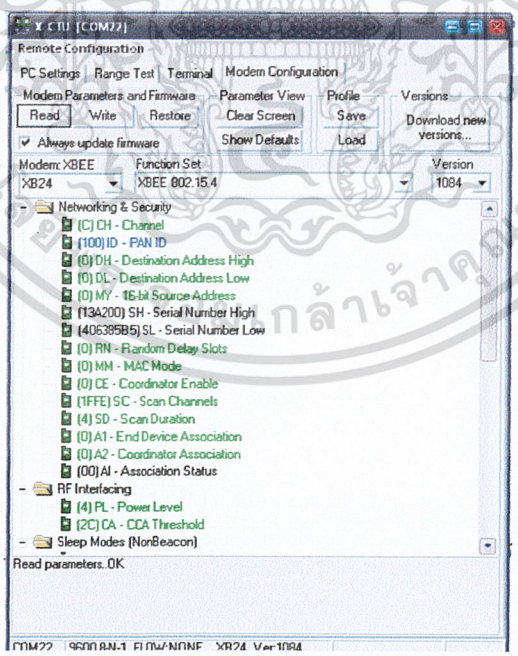
รูปที่ 4.2 หน้าจอแสดงผลค่าจาก Sensor ADXL345 ที่ต่อผ่าน USB

4.1.2 ติดต่อไร้สายผ่านทาง Xbee

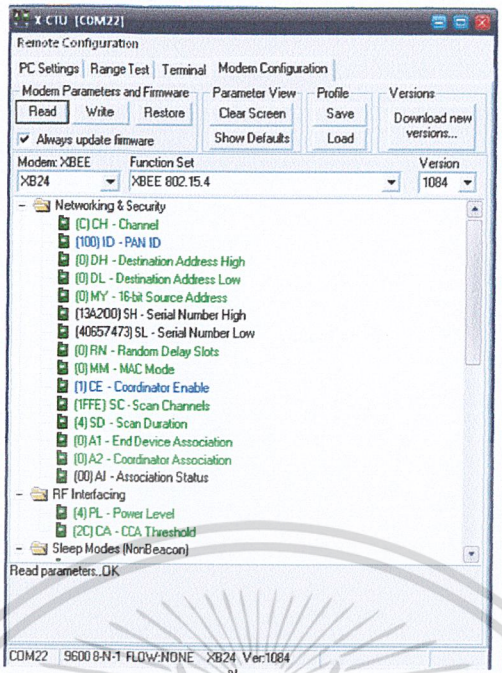
ทำการต่อ Xbee กับ Arduino เพื่อให้ติดต่อกันแบบไร้สาย



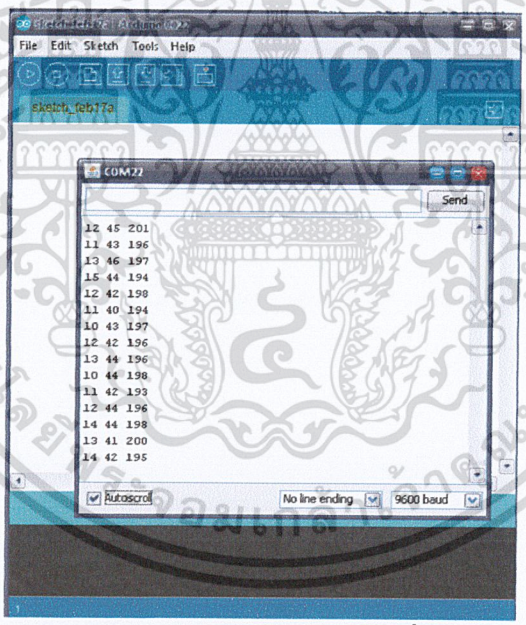
รูปที่ 4.3 ตั้งค่า Firmware Xbee



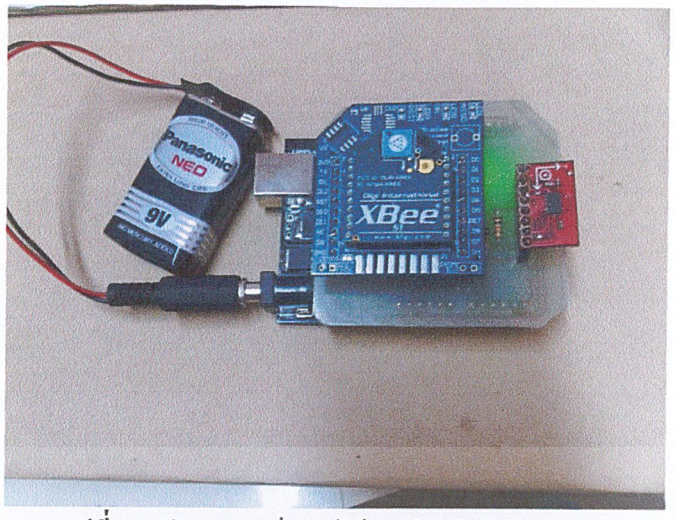
รูปที่ 4.4 ตั้งค่า ID และกำหนด Xbee ตัวนี้ให้เป็นตัวรับข้อมูลที่เชื่อมต่ออยู่กับคอมพิวเตอร์



รูปที่ 4.5 ตั้งค่า ID และกำหนด Xbee ตัวนี้ให้เป็นตัวส่งข้อมูลที่เชื่อมต่ออยู่กับArduino



รูปที่ 4.6 หน้าจอแสดงผลค่าจาก Sensor ADXL345 ที่ต่อผ่านแบบไร้สาย Xbee



รูปที่ 4.7 ต่ออุปกรณ์แบบไร้สายในส่วนของตัวส่ง

รูปที่ 4.8 ต่ออุปกรณ์แบบไร้สายส่วนของตัวรับที่เชื่อมต่อกับคอมพิวเตอร์

4.2 การทดลองใช้งานโปรแกรม Microsoft Visual Studio 2005 สำหรับการรับค่าข้อมูลจากบอร์ด ไมโครคอนโทรลเลอร์ Arduino

เนื่องจากการรับค่าผ่านพอร์ท USB-to-Serial ระบบปฏิบัติการ Windows รุ่นเก่าจะรับค่าผ่านทาง Hyper Terminal และสามารถเรียกค่าข้อมูลในโปแกรม Microsoft Visual Basic 6 ผ่านทาง Microsoft Comm Control 6.0 ดังนั้นเมื่อใช้ระบบปฏิบัติการรุ่นใหม่จึงไม่สามารถเรียกฟังก์ชั่นดังกล่าวได้ จึงต้องใช้ Microsoft Visual studio 2005 เพื่อเรียกรับค่าข้อมูลผ่านทาง Serial Port ของ Microsoft Visual Studio 2005 รวมไปถึงยังสามารถทำงานแบบ Windows Service ได้ด้วย

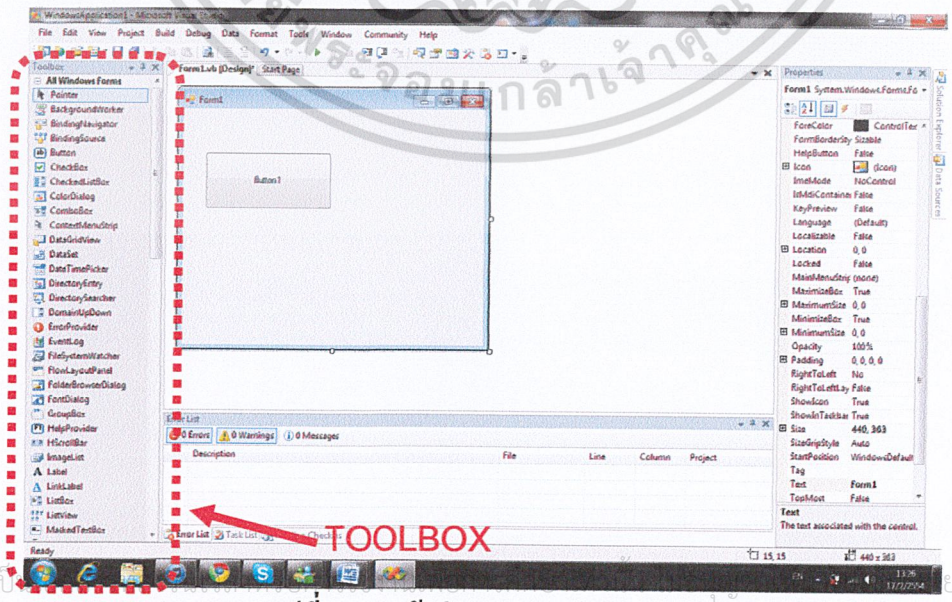
4.3 ทดลองการใช้งานโปรแกรม Microsoft Visual Studio 2005

4.3.1 การทดลองใช้โปรแกรม Microsoft Visual Studio 2005

การออกแบบและสร้างกรางโปรแกรมใน Microsoft Visual Studio 2005 ให้เป็นไปตามต้องการ ต้องอาศัยทักษะการใช้คำสั่งในการควบคุมโปรแกรมและความคิดริเริ่มในการออกแบบโปแกรมให้เหมาะสมกับการใช้งานจึงจะสามารถนำมาใช้ได้ไง

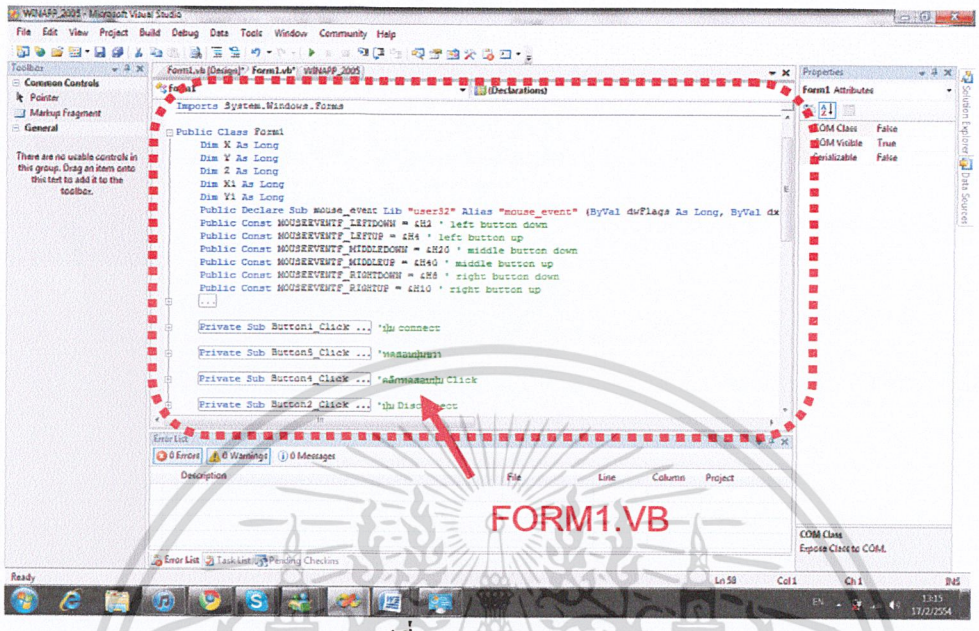
4.3.2 ตัวอย่างการออกแบบ Microsoft Visual Studio 2005

- เปิดโปรแกรม Microsoft Visual Studio 2005 เพิ่มอุปกรณ์ที่ต้องการ โดยเลือกที่ Microsoft Visual Studio 2005 >> File >> NewProject >> WindowsApplication และเลือกอุปกรร์จากทางด้านซ้าย



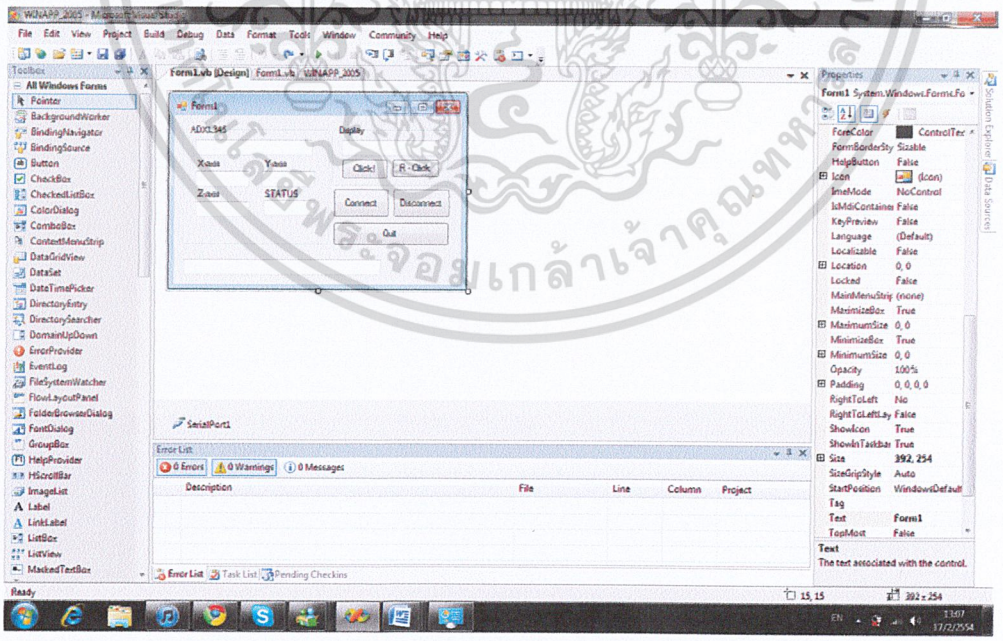
รูปที่ 4.9 หน้าต่าง Form1.vb[Design]

- เลือก Form1.vb เพื่อเลือกใช้คำสั่งในการควมคุมอุปกรณ์ที่เลือกใช้ เช่นตัวอย่างดังรูปที่ 4.10



รูปที่ 4.10 Form1.vb

- เลือกอุปกรณ์เพิ่มเติมเพื่อให้เหมาะสมกับการใช้งาน ดังเช่นตัวอย่างรูปที่ 4.11



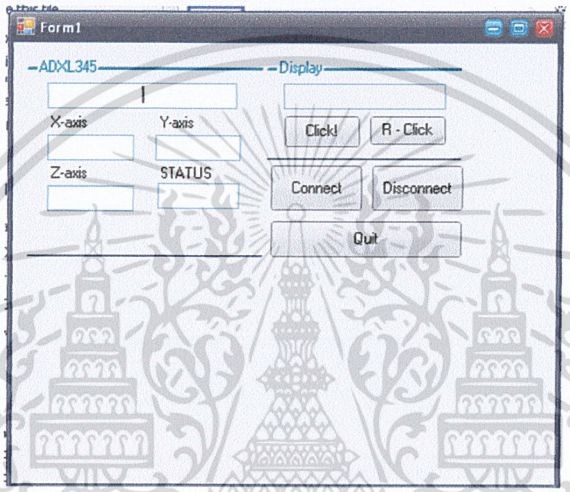
รูปที่ 4.11 ตัวอย่างโปรแกรมที่ออกแบบ

4.4 การทดลองนำค่าจาก บอร์ด Adruino ไปประมวลผลในโปรแกรม Microsoft Visual Studio 2005

ในส่วนนี้เป็นการนำค่าที่ได้จากบอร์ดในทิศทางแกนต่างๆ มาใช้งานในโปรแกรมเกมเพื่อกำหนดทิศทางและการเคลื่อนไหวในเหตุการณ์ต่างๆบนหน้าจอคอมพิวเตอร์

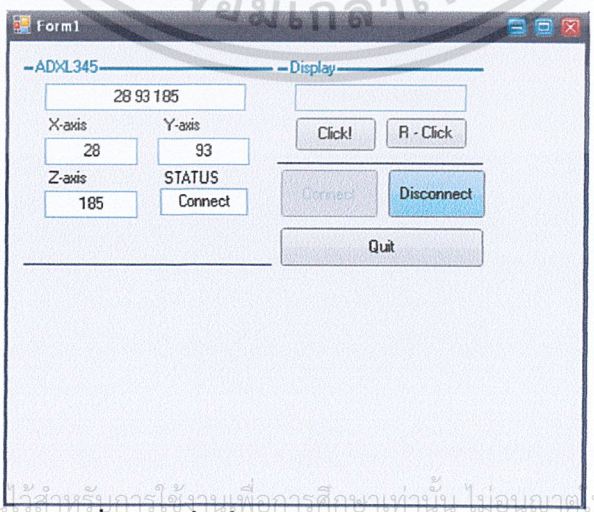
4.4.1 รับค่าจากเซนเซอร์วัดความเร่ง ADXL345

เมื่อทำงานติดตั้งและเปิดโปรแกรมจะได้โปแกรมดังภาพ

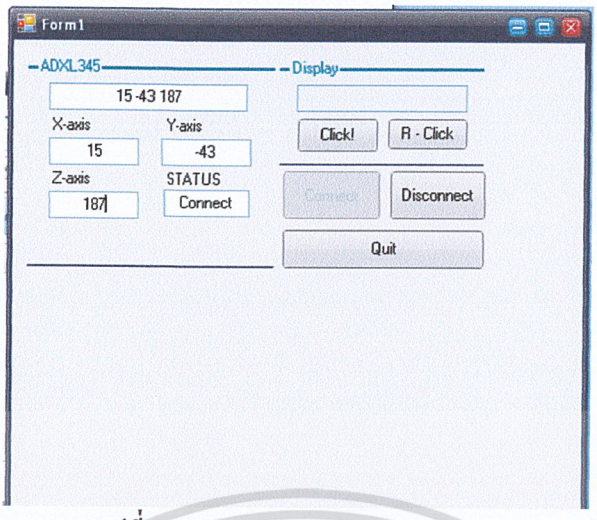


รูปที่ 4.12 โปรแกรมที่ทำการติดตั้งลงบนเครื่องแล้ว

ทำงานกดปุ่ม Connect เพื่อรับค่าจากบอร์ด Arduino ผ่านทาง Serial Portแล้วจะแสดงค่าที่ด้านซ้ายของโปแกรม และด้านขวาคือการแสดงค่าการเคลื่อนไหวบนจอภาพ



รูปที่ 4.13 เมื่อเริ่มการทำงานของโปรแกรม



รูปที่ 4.14 การทำงานของโปรแกรม

เมื่อเกิดการเคลื่อนไหวขึ้นค่าที่ได้จากเซนเซอร์จะมีการเปลี่ยนแปลงและ โปรแกรมจะไปสั่งการทำงานให้เม้าส์เกิดการเคลื่อนที่ขึ้นบนจอภาพดังรูปที่ 4.11 เมื่อ X-axis มีค่าเป็น 15 ซึ่งมีค่าเป็นบวก จะทำให้เม้าเคลื่อนที่ไปบนแกน X ไปทางด้านขวา และ Y-axis มีค่าเป็น -43 มีค่าเป็นลบ จะทำให้เม้าเคลื่อนที่ลงทางด้านล่างในแนวแกน Y

4.5 การทดลองทดสอบการควบคุมอุปกรณ์ที่ส่วนต่างๆ

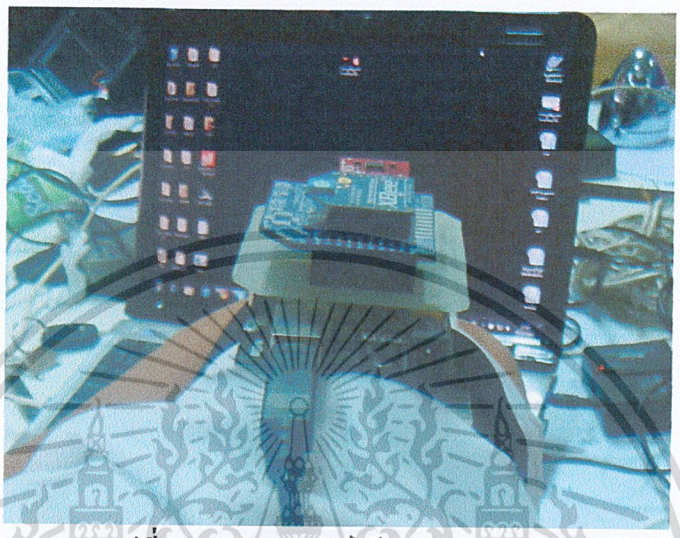
การทดลองนี้เพื่อทดสอบขอบเขตการทำงานของตัวอุปกรณ์ ด้านประสิทธิภาพการทำงานของตัวอุปกรณ์ว่าสามารถทำตามคำสั่งและการควบคุม

4.5.1 การติดอุปกรณ์ไว้กับหลังมือ

รูปที่ 4.15 ติดอุปกรณ์ไว้กับมือ

จากการทดสอบอุปกรณ์ที่ติดไว้กับมือ ประสิทธิภาพอยู่ในเกณฑ์ดี การควบคุมแม่นยำ สามารถทำตามคำสั่งได้ดี

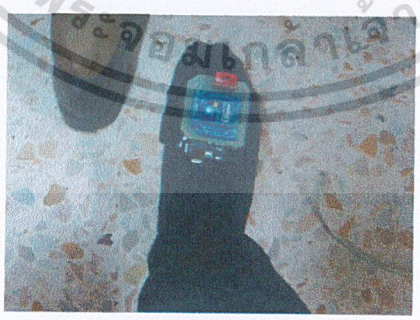
4.5.2 การติดอุปกรณ์ไว้กับข้อศอก



รูปที่ 4.16 ติดอุปกรณ์ไว้กับศอก

จากการทดสอบอุปกรณ์ที่ติดไว้กับมือ ประสิทธิภาพอยู่ในเกณฑ์พอใช้ การควบคุมไม่แม่นยำเท่าที่ควรเนื่องจากข้อศอกมีแรงของความเร่งเชิงมุมน้อยกว่ามือ สามารถทำตามคำสั่งได้พอใช้

4.5.3 การติดอุปกรณ์ไว้กับเท้า



รูปที่ 4.17 ติดอุปกรณ์ไว้กับเท้า

จากการทดสอบอุปกรณ์ที่ติดไว้กับเท้า ประสิทธิภาพอยู่ในเกณฑ์พอใช้ การควบคุมไม่แม่นยำกว่าข้อศอกมือ สามารถทำตามคำสั่งได้พอใช้การคลิ๊กทำได้ไม่ดีเท่าที่ควร

บทที่ 5
บทสรุป

5.1 บทวิจารณ์และสรุปผล

โครงงานนี้มีความสำเร็จในระดับที่พึงพอใจ โดยสามารถทำงานได้ตามขอบเขตของงานที่ได้กำหนดไว้ตอนเริ่มต้นโครงการ คือสามารถอ่านค่า 3 แกน สามารถติดต่อกับ Windows และควบคุมการทำงานของ Application ได้ และสามารถทำการสื่อสารแบบไร้สายได้ แต่ยังไม่สามารถที่จะพัฒนาอุปกรณ์ให้ใช้งานสะดวกมากขึ้นอีกระดับ และยังไม่สามารถพัฒนาซอฟต์แวร์ให้มีความน่าสนใจมากขึ้นจากเดิม รวมทั้งค่าที่ใช้ในการไปตรวจสอบการทำงานเพื่อที่จะควบคุม Application บนคอมพิวเตอร์ยังไม่แม่นยำท่าที่ควร

5.2 ปัญหาและอุปสรรคในการทำงาน

- ขาดความรู้ และประสบการณ์ในการเขียน I^2C
- ขาดประสบการณ์ในการออกแบบฮาร์ดแวร์
- ข้อมูลที่แสดงผลออกมามีความคลาดเคลื่อน
- ข้อมูลคำสั่งในโปรแกรม Microsoft Visual Studio 2005 ที่ใช้ในโปรแกรมมีคนศึกษาน้อย จึงทำให้ยากต่อการศึกษา

5.3 แนวทางการแก้ไข

- ศึกษาข้อมูลเกี่ยวกับ I^2C รวมทั้งตัวอย่างการเขียนแล้วนำมาปรับประยุกต์ใช้กับอุปกรณ์
- ทำการ Debug ข้อมูล ทั้งจาก Software และ Hardware รวมทั้งใช้ Oscilloscope จับสัญญาณ เพื่อตรวจสอบความผิดพลาดจากหลากหลายทาง
- ทำการ Simulate วงจรก่อนการลงมือจริง และในขณะต่อวงจร จะทำการใช้ Meter ในการวัดค่าต่างๆ เพื่อป้องกันความผิดพลาดที่จะเกิดขึ้น
- เพิ่มขอบเขตในการศึกษาเพิ่มเติมเกี่ยวกับโปแกรม Microsoft Visual Studio 2005

5.4 ประโยชน์ที่ได้รับ

- ทราบถึงการติดต่อสื่อสารระหว่างไมโครคอนโทรลเลอร์กับเซนเซอร์
- ทราบถึงการเขียนภาษาซีเพื่อที่จะควบคุมไมโครคอนโทรล ATmega328 (Arduino)
- ทราบถึงการทำงานของไมโครคอนโทรลในการติดต่อกับ Xbee
- ทราบถึงการติดต่อสื่อสารแบบไร้สายโดยการใช้โมดูล Xbee
- ทราบถึงการติดต่อในระบบต่างๆ เช่น UART, RS232
- ทราบถึงการเขียนโปแกรมโดยใช้ Microsoft Visual Studio 2005 และการเรียกใช้ฟังก์ชั่นต่างๆ

5.5 สรุปการทำงานและแนวทางการพัฒนา

ในการทำงานนั้นจะเริ่มต้นด้วยการติดต่อสื่อสารไมโครคอนโทรลเลอร์กับระบบคอมพิวเตอร์และได้ติดตั้ง Xbee เข้าไปเพื่อให้มีการติดต่อแบบไร้สายโดยเซนเซอร์นั้นเปรียบเสมือนเมาส์เลื่อนไปมาตามแกน X, Y และ แกน Z เปรียบเสมือนการคลิกเมาส์

เนื่องด้วยระยะเวลาในการพัฒนาชิ้นงานนี้มีเวลาจำกัด จึงยังมีข้อจำกัดและข้อควรปรับปรุงอีกหลายประการ ซึ่งมีแนวทางในการพัฒนาดังนี้

- พัฒนาอุปกรณ์ให้สามารถใช้งานได้ง่าย และสะดวกมากขึ้น
- พัฒนาซอฟต์แวร์ให้ผู้ใช้เข้าใจได้ง่ายขึ้น และน่าสนใจมากขึ้นด้วย
- พัฒนาอุปกรณ์ให้เป็นเครื่องมือช่วยในการทำงานที่จะอำนวยประโยชน์ต่อผู้อื่นในสังคม เช่น ผู้สูงอายุ หรือคนพิการ

บรรณานุกรม

[1] เอกชัย มะการ, รู้จักและเข้าใจ Chips Support I^2C BUS, บริษัท อีทีที จำกัด, 2545

[2] ประจิน พลังสันติกุล, การเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ AVR ด้วยภาษา C กับ WinAVR (C Compiler), บริษัท แอพซอฟต์เทค จำกัด, 2549

[3] http://www.vecthai.com/forums/index.php?topic=518.msg1483;topicseen

[4] http://www.thaieasyelec.com/Review-Product-Article/xbee-with-microcontroller-PIC16F877.html

[5] http://codeyoung.blogspot.com/2009/11/adxl345-accelerometer-breakout-board.html

[6] http://www.thaieasyelec.com/Review-Product-Article/step-by-step-to-use-xbee-from-digi.html

[7] http://www.thaieasyelec.com/Review-Product-Article/what-is-xbee.html

[8] เอกชัย มะการ, รู้จัก เข้าใจ ใช้งาน ไมโครคอนโทรลเลอร์ตระกูล AVR ด้วย Arduino, บริษัท อีทีที จำกัด, 2552

[9] http://social.msdn.microsoft.com/Forums/en/vbgeneral/thread/4ebccff7-baf6-4f2d-ae40-bb8733c2c805

[10] http://www.vbforums.com/showthread.php?t=402916

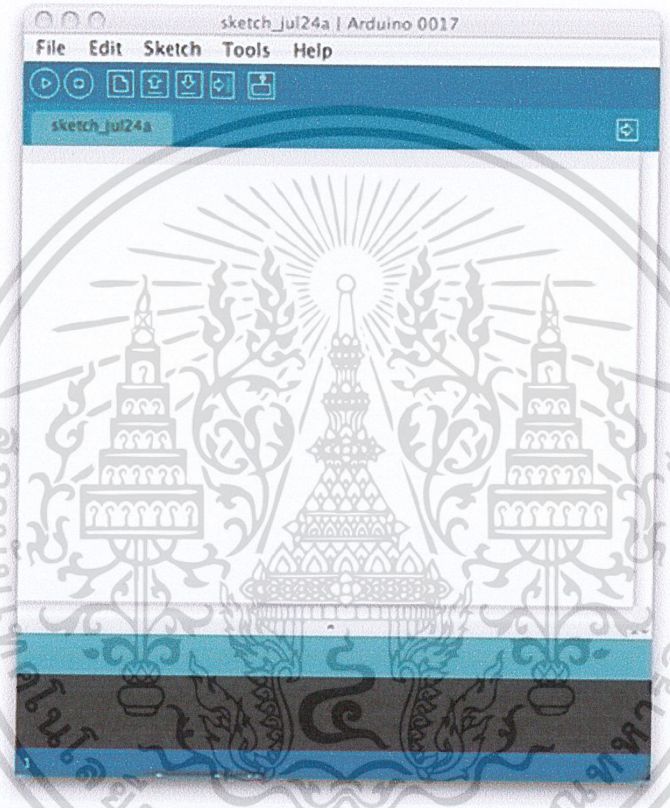
[11] http://www.dreamincode.net/forums/topic/59875-move-cursor-using-keyboard-keys-to-draw/

ภาคผนวก

ภาคผนวก ก.

Arduino IDE Basic Manual

เมื่อเรามี hardware ของ Arduino แล้ว ขั้นต่อไปคือมี software นั้นคือ Arduino IDE โดยสามารถ download ได้ที่ http://www.arduino.cc/en/Main/Software ซึ่งมี OS สามชนิด Windows,Linux และ Mac ซึ่งมาดูหน้าตา GUI ของ IDE ตัวนี้ทำจากภาษาจาวาจึงต้องมีการติดตั้ง Java Runtime เสมอ



MENU bar ของ Arduino จะประกอบด้วย

1. Menu File โดยทำหน้าที่บริหารจัดการข้อมูลที่เรียกว่า Sketch File

- New : สร้าง Sketch File ใหม่
- Sketchbook :

1. Open : เรียก Sketchbook ที่บันทึกไว้ก่อนหน้านี้
2. Example :เปิด Sketchbook ตัวอย่างที่มาจาก Library

- Save : บันทึก Sketch File ณ.ปัจจุบัน
- Save as : บันทึก Sketch File ในที่ อื่นหรือชื่อใหม่

- Upload to I/O board : การ burn สู่ MCU(AVR)หลังจากที่เขียนโปรแกรมเสร็จ
- Preference : ตั้งค่าต่าง ๆ ของArduino IDE

2. Edit Menu ทำหน้าที่แก้ไข wording

- Undo : ยกเลิกคำสั่งหรือการพิมพ์ครั้งล่าสุด
- Redo : ทำซ้ำคำสั่งหรือการพิมพ์ครั้งล่าสุด
- Cut : ตัดข้อความไว้ใน clipboard
- Copy : คัดลอกข้อความจาก clipboard
- Paste : แปะข้อความจาก clipboard
- Select all : เลือก(แรเงา)ข้อความทั้งหมด
- Find : ค้นหาข้อความ
- Find Next : ค้นหาข้อความถัดไป

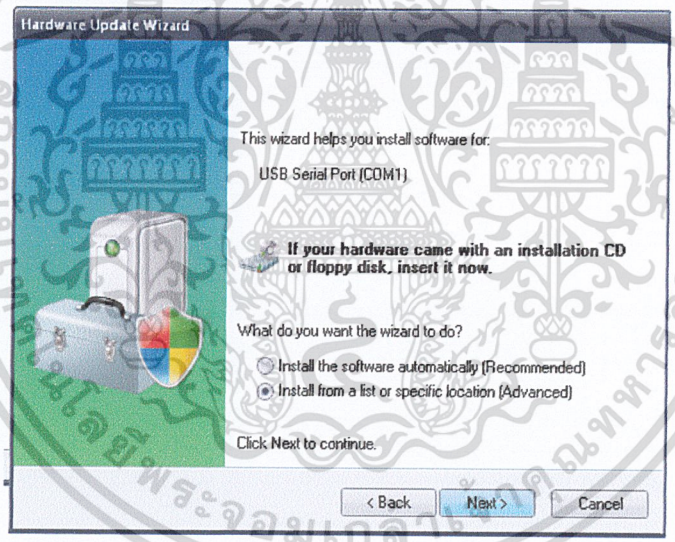
3. Sketch Menu เป็นเมนูที่ใช้คำสั่ง Compile ,เพิ่มLibrary

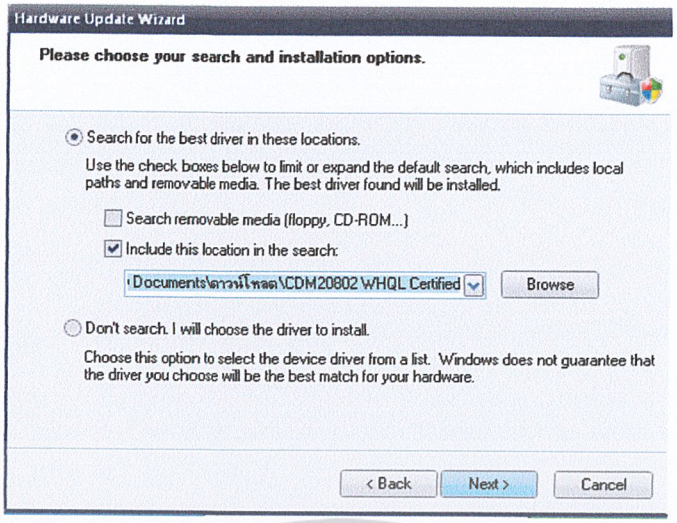
- Verify/Compile : Compile ภาษา c/c++ เป็นภาษา Machine
- Stop : หยุดการ Compile
- Import Library : เลือก/เพิ่ม Library โดยแทรกในบรรทัดแรก
- Show Sketch Folders : ทำการแสดงที่เก็บโปรแกรม
- Add File.. : คัดลอก File เลือก/เพิ่มมาบันทึกรวมใน Folder ปัจจุบัน

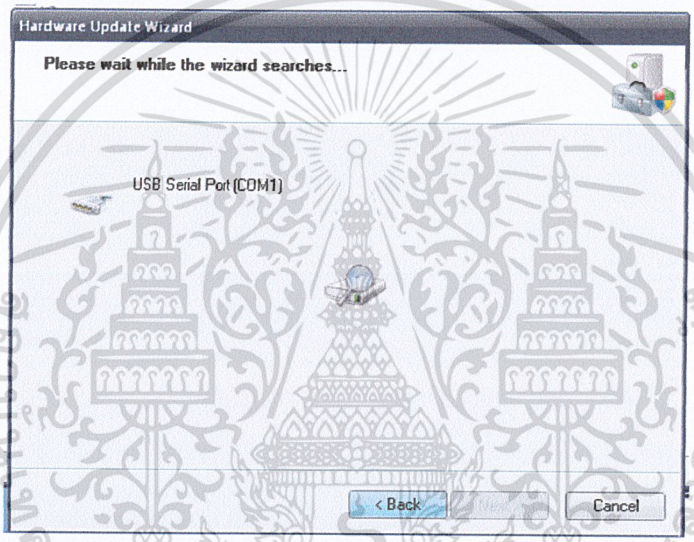
4. Tools Menu :ใช้จัด code ,เลือกเบอร์ Microcontroller ,เลือก port serial

- Auto Format : จัดวาง code ให้เป็นระเบียบสวยงาม ให้อ่านได้ง่ายโดยแยกสี code ที่เป็นคำสั่งและตัวแปร
- copy to Forum : คัดลอก code ลง clipboard
- Archive Sketch : สั่งให้บีบอัด Folder ปัจจุบันเป็นไฟล์บีบอัด มีประโยชน์ในการทำ CVS
- Board : เลือก Hardware Device ที่ใช้งานกับโปรแกรม เช่น Arduino Nano เป็นต้น
- Serial Ports : เลือก port เพื่อ communication ระหว่าง Hardware Device กับ Arduino IDE
- Burn Bootloader : เพื่อทำการ bootloadder ให้ Hardware Device ดดยต้องมีเครื่อง burn

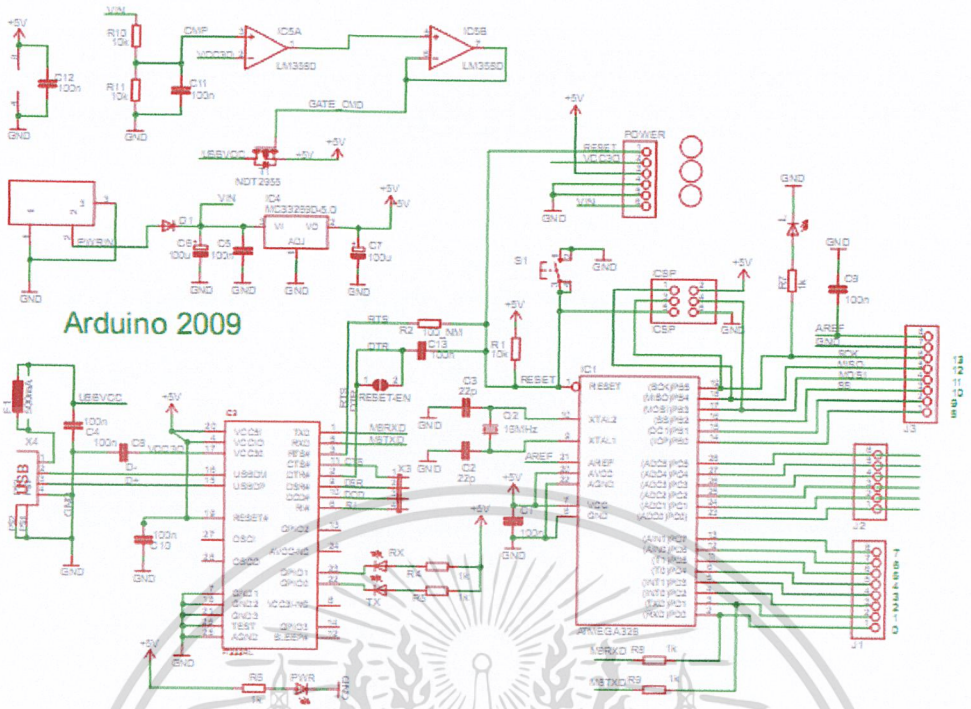
ติดตั้ง Driver USB

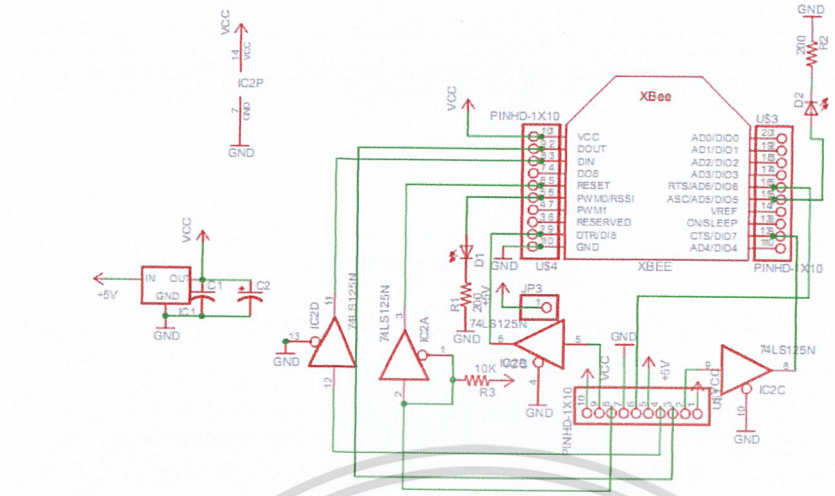


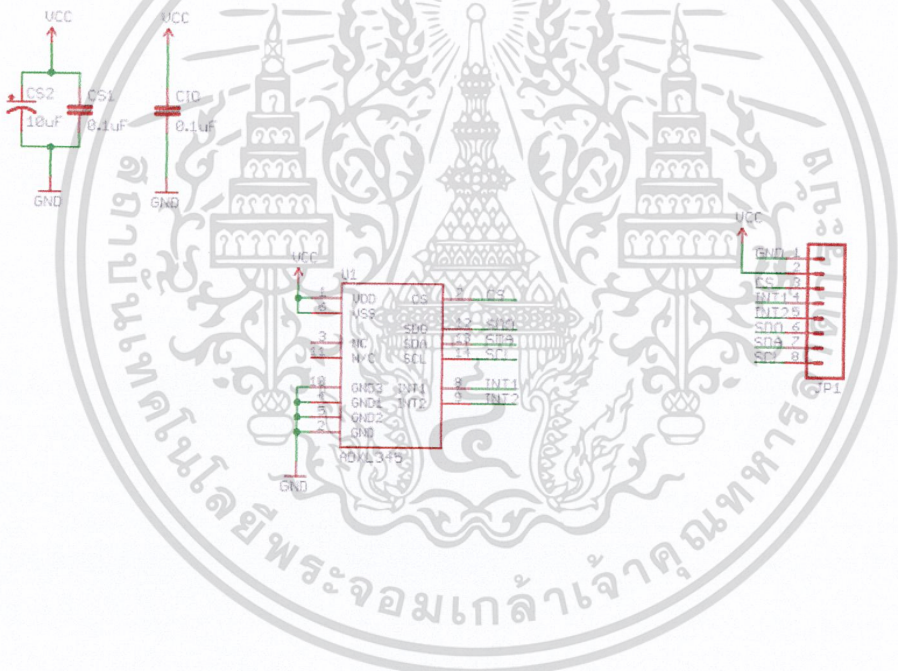




ภาคผนวก ข.







ภาคผนวก ค.

3-Axis, ±2 *g*/±4 *g*/±8 *g*/±16 *g* Digital Accelerometer

ADXL345

FEATURES

- Ultralow power: as low as 40 µA in measurement mode and 0.1 µA in standby mode at V_S = 2.5 V (typical)
- Power consumption scales automatically with bandwidth
- User-selectable resolution
  - Fixed 10-bit resolution
  - Full resolution, where resolution increases with *g* range, up to 13-bit resolution at ±16 *g* (maintaining 4 mg/LSB scale factor in all *g* ranges)
- Embedded, patent pending FIFO technology minimizes host processor load
- Tap/double tap detection
- Activity/inactivity monitoring
- Free-fall detection
- Supply voltage range: 2.0 V to 3.6 V
- I/O voltage range: 1.7 V to V_S
- SPI (3- and 4-wire) and I^2C digital interfaces
- Flexible interrupt modes mappable to either interrupt pin
- Measurement ranges selectable via serial command
- Bandwidth selectable via serial command
- Wide temperature range (−40°C to +85°C)
- 10,000 *g* shock survival
- Pb free/RoHS compliant
- Small and thin: 3 mm × 5 mm × 1 mm LGA package

APPLICATIONS

- Handsets
- Medical instrumentation
- Gaming and pointing devices
- Industrial instrumentation
- Personal navigation devices
- Hard disk drive (HDD) protection
- Fitness equipment

GENERAL DESCRIPTION

The ADXL345 is a small, thin, low power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16 *g*. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I^2C digital interface.

The ADXL345 is well suited for mobile device applications. It measures the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (4 mg/LSB) enables measurement of inclination changes less than 1.0°.

Several special sensing functions are provided. Activity and inactivity sensing detect the presence or lack of motion and if the acceleration on any axis exceeds a user-set level. Tap sensing detects single and double taps. Free-fall sensing detects if the device is falling. These functions can be mapped to one of two interrupt output pins. An integrated, patent pending 32-level first in, first out (FIFO) buffer can be used to store data to minimize host processor intervention.

Low power modes enable intelligent motion-based power management with threshold sensing and active acceleration measurement at extremely low power dissipation.

The ADXL345 is supplied in a small, thin, 3 mm × 5 mm × 1 mm, 14-lead, plastic package.

FUNCTIONAL BLOCK DIAGRAM

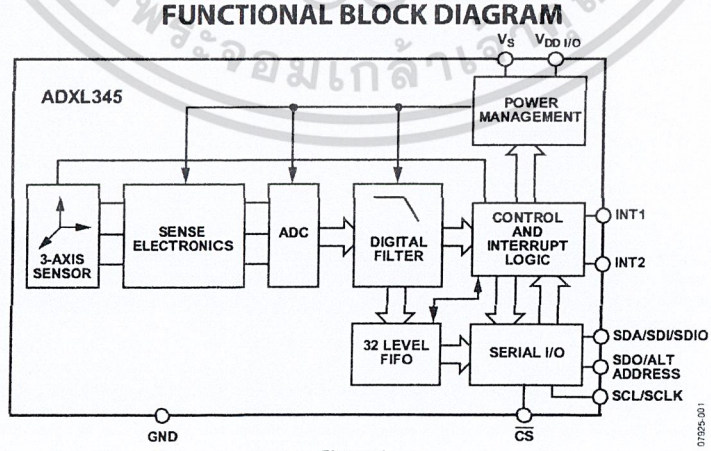


Figure 1.

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 www.analog.com
Fax: 781.461.3113

TABLE OF CONTENTS

REVISION HISTORY

5/09—Revision 0: Initial Version

SPECIFICATIONS

T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V, acceleration = 0 *g*, C_S = 1 µF tantalum, C_{IO} = 0.1 µF, unless otherwise noted.

Table 1. Specifications[1]

Parameter	Test Conditions	Min	Typ	Max	Unit
SENSOR INPUT	Each axis				
Measurement Range	User selectable		±2, ±4, ±8, ±16		*g*
Nonlinearity	Percentage of full scale		±0.5		%
Inter-Axis Alignment Error			±0.1		Degrees
Cross-Axis Sensitivity[2]			±1		%
OUTPUT RESOLUTION	Each axis				
All *g* Ranges	10-bit resolution		10		Bits
±2 *g* Range	Full resolution		10		Bits
±4 *g* Range	Full resolution		11		Bits
±8 *g* Range	Full resolution		12		Bits
±16 *g* Range	Full resolution		13		Bits
SENSITIVITY	Each axis				
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	±2 *g*, 10-bit or full resolution	232	256	286	LSB/*g*
Scale Factor at X_{OUT}, Y_{OUT}, Z_{OUT}	±2 *g*, 10-bit or full resolution	3.5	3.9	4.3	mg/LSB
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	±4 *g*, 10-bit resolution	116	128	143	LSB/*g*
Scale Factor at X_{OUT}, Y_{OUT}, Z_{OUT}	±4 *g*, 10-bit resolution	7.0	7.8	8.6	mg/LSB
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	±8 *g*, 10-bit resolution	58	64	71	LSB/*g*
Scale Factor at X_{OUT}, Y_{OUT}, Z_{OUT}	±8 *g*, 10-bit resolution	14.0	15.6	17.2	mg/LSB
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	±16 *g*, 10-bit resolution	29	32	36	LSB/*g*
Scale Factor at X_{OUT}, Y_{OUT}, Z_{OUT}	±16 *g*, 10-bit resolution	28.1	31.2	34.3	mg/LSB
Sensitivity Change Due to Temperature			±0.01		%/°C
0 *g* BIAS LEVEL	Each axis				
0 *g* Output for X_{OUT}, Y_{OUT}		−150	±40	+150	mg
0 *g* Output for Z_{OUT}		−250	±80	+250	mg
0 *g* Offset vs. Temperature for x-, y-Axes			±0.8		mg/°C
0 *g* Offset vs. Temperature for z-Axis			±4.5		mg/°C
NOISE PERFORMANCE					
Noise (x-, y-Axes)	Data rate = 100 Hz for ±2 *g*, 10-bit or full resolution		<1.0		LSB rms
Noise (z-Axis)	Data rate = 100 Hz for ±2 *g*, 10-bit or full resolution		<1.5		LSB rms
OUTPUT DATA RATE AND BANDWIDTH	User selectable				
Measurement Rate[3]		6.25		3200	Hz
SELF-TEST[4]	Data rate ≥ 100 Hz, 2.0 V ≤ V_S ≤ 3.6 V				
Output Change in x-Axis		0.20		2.10	*g*
Output Change in y-Axis		−2.10		−0.20	*g*
Output Change in z-Axis		0.30		3.40	*g*
POWER SUPPLY					
Operating Voltage Range (V_S)		2.0	2.5	3.6	V
Interface Voltage Range ($V_{DD\ I/O}$)	V_S ≤ 2.5 V	1.7	1.8	V_S	V
	V_S ≥ 2.5 V	2.0	2.5	V_S	V
Supply Current	Data rate > 100 Hz		145		µA
	Data rate < 10 Hz		40		µA
Standby Mode Leakage Current			0.1	2	µA
Turn-On Time[5]	Data rate = 3200 Hz		1.4		ms
TEMPERATURE					
Operating Temperature Range		−40		+85	°C
WEIGHT					
Device Weight			20		mg

[1] All minimum and maximum specifications are guaranteed. Typical specifications are not guaranteed.
[2] Cross-axis sensitivity is defined as coupling between any two axes.
[3] Bandwidth is half the output data rate.
[4] Self-test change is defined as the output (*g*) when the SELF_TEST bit = 1 (in the DATA_FORMAT register) minus the output (*g*) when the SELF_TEST bit = 0 (in the DATA_FORMAT register). Due to device filtering, the output reaches its final value after 4 × τ when enabling or disabling self-test, where τ = 1/(data rate).
[5] Turn-on and wake-up times are determined by the user-defined bandwidth. At a 100 Hz data rate, the turn-on and wake-up times are each approximately 11.1 ms. For other data rates, the turn-on and wake-up times are each approximately τ + 1.1 in milliseconds, where τ = 1/(data rate).

ABSOLUTE MAXIMUM RATINGS

Table 2.

Parameter	Rating
Acceleration	
Any Axis, Unpowered	10,000 *g*
Any Axis, Powered	10,000 *g*
V_S	−0.3 V to +3.6 V
$V_{DD\ I/O}$	−0.3 V to +3.6 V
Digital Pins	−0.3 V to $V_{DD\ I/O}$ + 0.3 V or 3.6 V, whichever is less
All Other Pins	−0.3 V to +3.6 V
Output Short-Circuit Duration (Any Pin to Ground)	Indefinite
Temperature Range	
Powered	−40°C to +105°C
Storage	−40°C to +105°C

Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

THERMAL RESISTANCE

Table 3. Package Characteristics

Package Type	θ_{JA}	θ_{JC}	Device Weight
14-Terminal LGA	150°C/W	85°C/W	20 mg

ESD CAUTION

ESD (electrostatic discharge) sensitive device. Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.

PIN CONFIGURATION AND FUNCTION DESCRIPTIONS

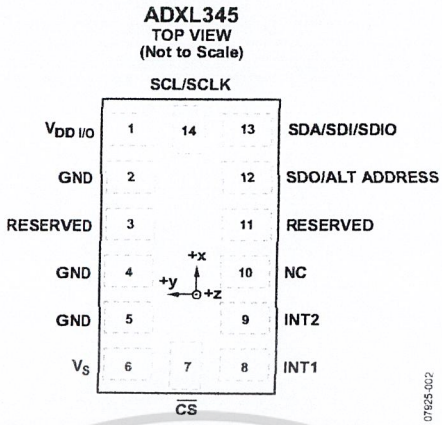


Figure 2. Pin Configuration

Table 4. Pin Function Descriptions

Pin No.	Mnemonic	Description
1	$V_{DD\ I/O}$	Digital Interface Supply Voltage.
2	GND	Must be connected to ground.
3	Reserved	Reserved. This pin must be connected to V_S or left open.
4	GND	Must be connected to ground.
5	GND	Must be connected to ground.
6	V_S	Supply Voltage.
7	$\overline{CS}$	Chip Select.
8	INT1	Interrupt 1 Output.
9	INT2	Interrupt 2 Output.
10	NC	Not Internally Connected.
11	Reserved	Reserved. This pin must be connected to ground or left open.
12	SDO/ALT ADDRESS	Serial Data Output/Alternate I^2C Address Select.
13	SDA/SDI/SDIO	Serial Data (I^2C)/Serial Data Input (SPI 4-Wire)/Serial Data Input and Output (SPI 3-Wire).
14	SCL/SCLK	Serial Communications Clock.

THEORY OF OPERATION

The ADXL345 is a complete 3-axis acceleration measurement system with a selectable measurement range of ±2 *g*, ±4 *g*, ±8 *g*, or ±16 *g*. It measures both dynamic acceleration resulting from motion or shock and static acceleration, such as gravity, which allows the device to be used as a tilt sensor.

The sensor is a polysilicon surface-micromachined structure built on top of a silicon wafer. Polysilicon springs suspend the structure over the surface of the wafer and provide a resistance against acceleration forces.

Deflection of the structure is measured using differential capacitors that consist of independent fixed plates and plates attached to the moving mass. Acceleration deflects the beam and unbalances the differential capacitor, resulting in a sensor output whose amplitude is proportional to acceleration. Phase-sensitive demodulation is used to determine the magnitude and polarity of the acceleration.

POWER SEQUENCING

Power can be applied to V_S or $V_{DD\ I/O}$ in any sequence without damaging the ADXL345. All possible power-on modes are summarized in Table 5. The interface voltage level is set with the interface supply voltage, $V_{DD\ I/O}$, which must be present to ensure that the ADXL345 does not create a conflict on the communication bus. For single-supply operation, $V_{DD\ I/O}$ can be the same as the main supply, V_S. In a dual-supply application, however, $V_{DD\ I/O}$ can differ from V_S to accommodate the desired interface voltage, as long as V_S is greater than $V_{DD\ I/O}$.

After V_S is applied, the device enters standby mode, where power consumption is minimized and the device waits for $V_{DD\ I/O}$ to be applied and for the command to enter measurement mode to be received. (This command can be initiated by setting the measure bit in the POWER_CTL register (Address 0x2D).) In addition, any register can be written to or read from to configure the part while the device is in standby mode. It is recommended to configure the device in standby mode and then to enable measurement mode. Clearing the measure bit returns the device to the standby mode.

Table 5. Power Sequencing

Condition	V_S	$V_{DD\ I/O}$	Description
Power Off	Off	Off	The device is completely off, but there is a potential for a communication bus conflict.
Bus Disabled	On	Off	The device is on in standby mode, but communication is unavailable and will create a conflict on the communication bus. The duration of this state should be minimized during power-up to prevent a conflict.
Bus Enabled	Off	On	No functions are available, but the device will not create a conflict on the communication bus.
Standby or Measurement	On	On	At power-up, the device is in standby mode, awaiting a command to enter measurement mode, and all sensor functions are off. After the device is instructed to enter measurement mode, all sensor functions are available.

POWER SAVINGS

Power Modes

The ADXL345 automatically modulates its power consumption in proportion to its output data rate, as outlined in Table 6. If additional power savings is desired, a lower power mode is available. In this mode, the internal sampling rate is reduced, allowing for power savings in the 12.5 Hz to 400 Hz data rate range but at the expense of slightly greater noise. To enter lower power mode, set the LOW_POWER bit (Bit 4) in the BW_RATE register (Address 0x2C). The current consumption in low power mode is shown in Table 7 for cases where there is an advantage for using low power mode. The current consumption values shown in Table 6 and Table 7 are for a V_S of 2.5 V. Current scales linearly with V_S.

Table 6. Current Consumption vs. Data Rate
(T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V)

Output Data Rate (Hz)	Bandwidth (Hz)	Rate Code	I_{DD} (μA)
3200	1600	1111	145
1600	800	1110	100
800	400	1101	145
400	200	1100	145
200	100	1011	145
100	50	1010	145
50	25	1001	100
25	12.5	1000	65
12.5	6.25	0111	55
6.25	3.125	0110	40

Table 7. Current Consumption vs. Data Rate, Low Power Mode
(T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V)

Output Data Rate (Hz)	Bandwidth (Hz)	Rate Code	I_{DD} (μA)
400	200	1100	100
200	100	1011	65
100	50	1010	55
50	25	1001	50
25	12.5	1000	40
12.5	6.25	0111	40

Auto Sleep Mode

Additional power can be saved if the ADXL345 automatically switches to sleep mode during periods of inactivity. To enable this feature, set the THRESH_INACT register (Address 0x25) and the TIME_INACT register (Address 0x26) each to a value that signifies inactivity (the appropriate value depends on the application), and then set the AUTO_SLEEP bit and the link bit in the POWER_CTL register (Address 0x2D). Current consumption at the sub-8 Hz data rates used in this mode is typically 40 μA for a V_S of 2.5 V.

Standby Mode

For even lower power operation, standby mode can be used. In standby mode, current consumption is reduced to 0.1 μA (typical). In this mode, no measurements are made. Standby mode is entered by clearing the measure bit (Bit 3) in the POWER_CTL register (Address 0x2D). Placing the device into standby mode preserves the contents of FIFO.

SERIAL COMMUNICATIONS

I²C and SPI digital communications are available. In both cases, the ADXL345 operates as a slave. I²C mode is enabled if the $\overline{CS}$ pin is tied high to $V_{DD\ I/O}$. The $\overline{CS}$ pin should always be tied high to $V_{DD\ I/O}$ or be driven by an external controller because there is no default mode if the $\overline{CS}$ pin is left unconnected. Therefore, not taking these precautions may result in an inability to communicate with the part. In SPI mode, the $\overline{CS}$ pin is controlled by the bus master. In both SPI and I²C modes of operation, data transmitted from the ADXL345 to the master device should be ignored during writes to the ADXL345.

SPI

For SPI, either 3- or 4-wire configuration is possible, as shown in the connection diagrams in Figure 3 and Figure 4. Clearing the SPI bit in the DATA_FORMAT register (Address 0x31) selects 4-wire mode, whereas setting the SPI bit selects 3-wire mode. The maximum SPI clock speed is 5 MHz with 100 pF maximum loading, and the timing scheme follows clock polarity (CPOL) = 1 and clock phase (CPHA) = 1.

$\overline{CS}$ is the serial port enable line and is controlled by the SPI master. This line must go low at the start of a transmission and high at the end of a transmission, as shown in Figure 5. SCLK is the serial port clock and is supplied by the SPI master. It is stopped high when $\overline{CS}$ is high during a period of no transmission. SDI and SDO are the serial data input and output, respectively. Data should be sampled at the rising edge of SCLK.

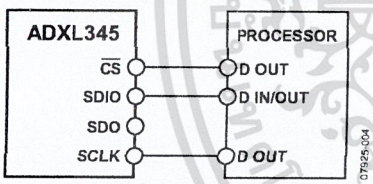


Figure 3. 3-Wire SPI Connection Diagram

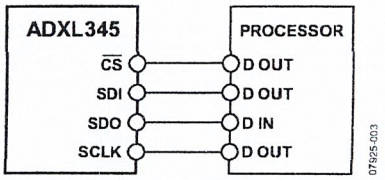


Figure 4. 4-Wire SPI Connection Diagram

To read or write multiple bytes in a single transmission, the multiple-byte bit, located after the R/$\overline{W}$ bit in the first byte transfer (MB in Figure 5 to Figure 7), must be set. After the register addressing and the first byte of data, each subsequent set of clock pulses (eight clock pulses) causes the ADXL345 to point to the next register for a read or write. This shifting continues until the clock pulses cease and $\overline{CS}$ is deasserted. To perform reads or writes on different, nonsequential registers, $\overline{CS}$ must be deasserted between transmissions and the new register must be addressed separately.

The timing diagram for 3-wire SPI reads or writes is shown in Figure 7. The 4-wire equivalents for SPI writes and reads are shown in Figure 5 and Figure 6, respectively.

Table 8. SPI Digital Input/Output Voltage

Parameter	Limit[1]	Unit
Digital Input Voltage		
Low Level Input Voltage (V_{IL})	$0.2 \times V_{DD\ I/O}$	V max
High Level Input Voltage (V_{IH})	$0.8 \times V_{DD\ I/O}$	V min
Digital Output Voltage		
Low Level Output Voltage (V_{OL})	$0.15 \times V_{DD\ I/O}$	V max
High Level Output Voltage (V_{OH})	$0.85 \times V_{DD\ I/O}$	V min

[1] Limits based on characterization results, not production tested.

Table 9. SPI Timing (T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V)[1]

	Limit[2, 3]			
Parameter	Min	Max	Unit	Description
f_{SCLK}		5	MHz	SPI clock frequency
t_{SCLK}	200		ns	1/(SPI clock frequency) mark-space ratio for the SCLK input is 40/60 to 60/40
t_{DELAY}	10		ns	$\overline{CS}$ falling edge to SCLK falling edge
t_{QUIET}	10		ns	SCLK rising edge to $\overline{CS}$ rising edge
t_{DIS}		100	ns	$\overline{CS}$ rising edge to SDO disabled
$t_{CS,DIS}$	250		ns	$\overline{CS}$ deassertion between SPI communications
t_S	$0.4 \times t_{SCLK}$		ns	SCLK low pulse width (space)
t_M	$0.4 \times t_{SCLK}$		ns	SCLK high pulse width (mark)
t_{SDO}		95	ns	SCLK falling edge to SDO transition
t_{SETUP}	10		ns	SDI valid before SCLK rising edge
t_{HOLD}	10		ns	SDI valid after SCLK rising edge

[1] The $\overline{CS}$, SCLK, SDI, and SDO pins are not internally pulled up or down; they must be driven for proper operation.
[2] Limits based on characterization results, characterized with f_{SCLK} = 5 MHz and bus load capacitance of 100 pF; not production tested.
[3] *The timing values are measured corresponding to the input thresholds (V_{IL} and V_{IH}) given in Table 8.*

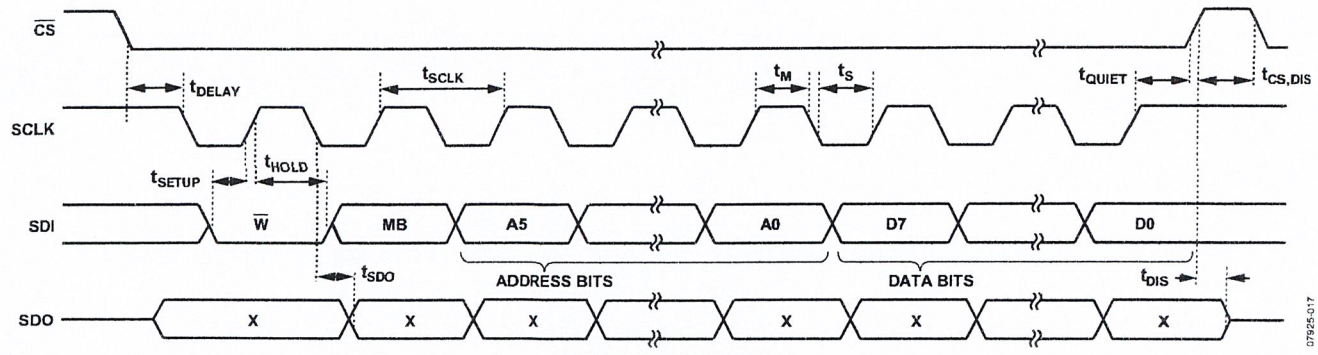


Figure 5. SPI 4-Wire Write

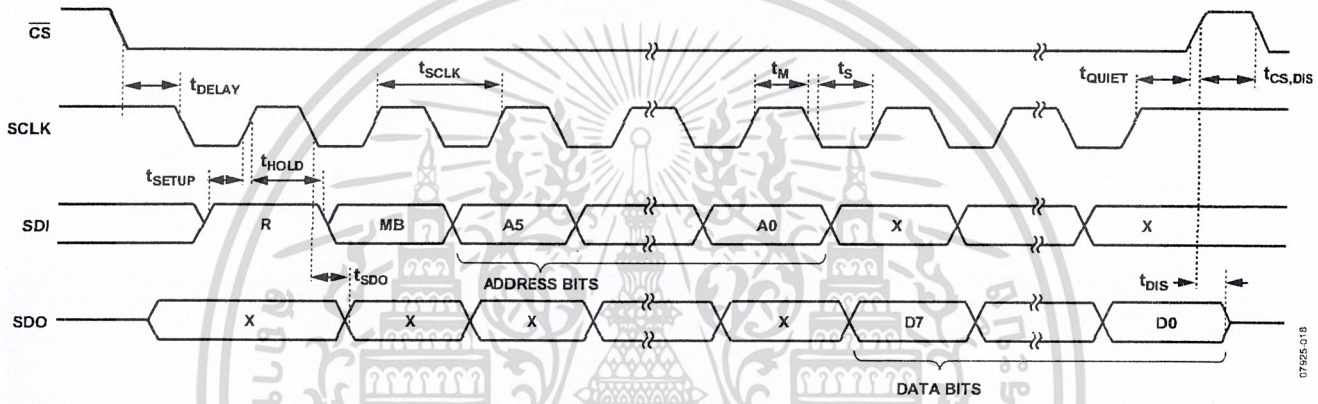
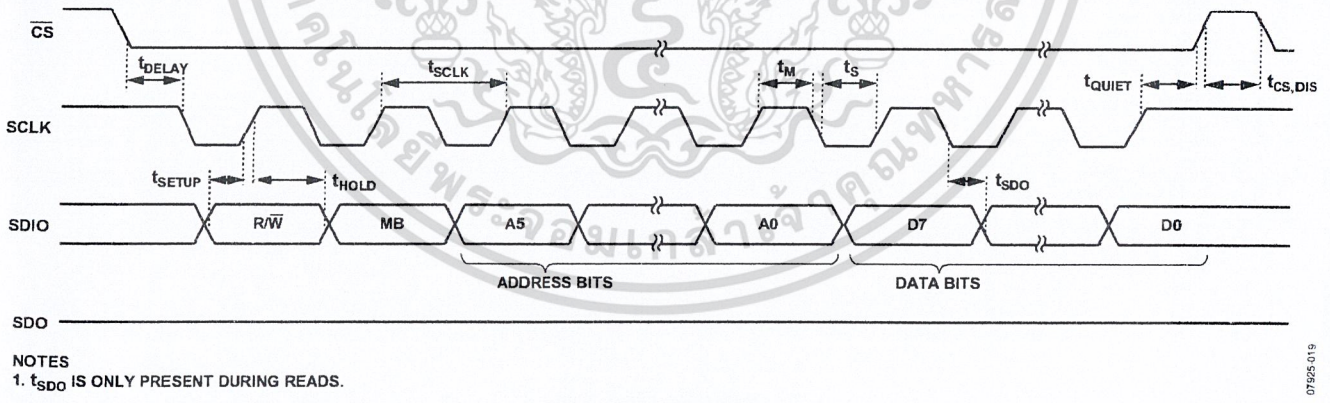


Figure 6. SPI 4-Wire Read



NOTES
1. t_{SDO} IS ONLY PRESENT DURING READS.

Figure 7. SPI 3-Wire Read/Write

I²C

With $\overline{CS}$ tied high to $V_{DD\ I/O}$, the ADXL345 is in I²C mode, requiring a simple 2-wire connection as shown in Figure 8. The ADXL345 conforms to the *UM10204 I²C-Bus Specification and User Manual*, Rev. 03—19 June 2007, available from NXP Semiconductor. It supports standard (100 kHz) and fast (400 kHz) data transfer modes if the timing parameters given in Table 11 and Figure 10 are met. Single- or multiple-byte reads/writes are supported, as shown in Figure 9. With the SDO/ALT ADDRESS pin high, the 7-bit I²C address for the device is 0x1D, followed by the R/$\overline{W}$ bit. This translates to 0x3A for a write and 0x3B for a read. An alternate I²C address of 0x53 (followed by the R/$\overline{W}$ bit) can be chosen by grounding the SDO/ALT ADDRESS pin (Pin 12). This translates to 0xA6 for a write and 0xA7 for a read.

If other devices are connected to the same I²C bus, the nominal operating voltage level of these other devices cannot exceed $V_{DD\ I/O}$ by more than 0.3 V. External pull-up resistors, R_P, are necessary for proper I²C operation. Refer to the *UM10204 I²C-Bus Specification and User Manual*, Rev. 03—19 June 2007, when selecting pull-up resistor values to ensure proper operation.

Table 10. I²C Digital Input/Output Voltage

Parameter	Limit[1]	Unit
Digital Input Voltage		
Low Level Input Voltage (V_{IL})	$0.25 \times V_{DD\ I/O}$	V max
High Level Input Voltage (V_{IH})	$0.75 \times V_{DD\ I/O}$	V min
Digital Output Voltage		
Low Level Output Voltage (V_{OL})[2]	$0.2 \times V_{DD\ I/O}$	V max

[1] Limits based on characterization results; not production tested.
[2] The limit given is only for $V_{DD\ I/O} < 2$ V. When $V_{DD\ I/O} > 2$ V, the limit is 0.4 V max.

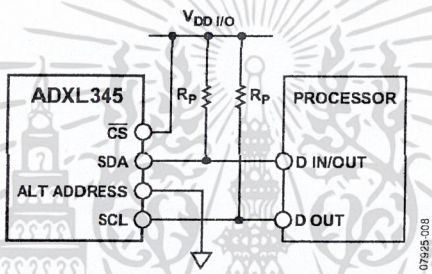


Figure 8. I²C Connection Diagram (Address 0x53)

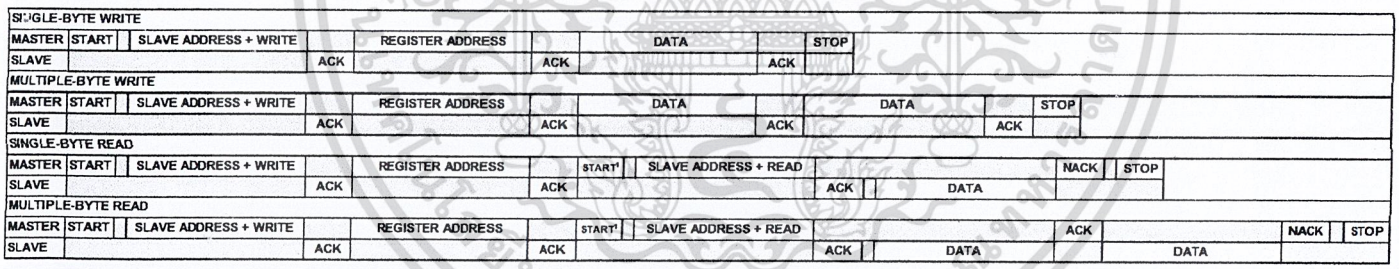


Figure 9. I²C Device Addressing

Table 11. I²C Timing (T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V)

Parameter	Limit[1, 2] Min	Max	Unit	Description
f_{SCL}		400	kHz	SCL clock frequency
t_1	2.5		µs	SCL cycle time
t_2	0.6		µs	t_{HIGH}, SCL high time
t_3	1.3		µs	t_{LOW}, SCL low time
t_4	0.6		µs	$t_{HD,\ STA}$, start/repeated start condition hold time
t_5	350		ns	$t_{SU,\ DAT}$, data setup time
t_6[3, 4, 5, 6]	0	0.65	µs	$t_{HD,\ DAT}$, data hold time
t_7	0.6		µs	$t_{SU,\ STA}$, setup time for repeated start
t_8	0.6		µs	$t_{SU,\ STO}$, stop condition setup time
t_9	1.3		µs	t_{BUF}, bus-free time between a stop condition and a start condition
t_{10}		300	ns	t_R, rise time of both SCL and SDA when receiving
	0		ns	t_R, rise time of both SCL and SDA when receiving or transmitting
t_{11}		250	ns	t_F, fall time of SDA when receiving
		300	ns	t_F, fall time of both SCL and SDA when transmitting
	20 + 0.1 C_b[7]		ns	t_F, fall time of both SCL and SDA when transmitting or receiveing
C_b		400	pF	Capacitive load for each bus line

[1] Limits based on characterization results, with f_{SCL} = 400 kHz and a 3 mA sink current; not production tested.
[2] All values referred to the V_{IH} and the V_{IL} levels given in Table 10.
[3] t_6 is the data hold time that is measured from the falling edge of SCL. It applies to data in transmission and acknowledge times.
[4] A transmitting device must internally provide an output hold time of at least 300 ns for the SDA signal (with respect to $V_{IH(min)}$ of the SCL signal) to bridge the undefined region of the falling edge of SCL.
[5] The maximum t_6 value must be met only if the device does not stretch the low period (t_3) of the SCL signal.
[6] The maximum value for t_6 is a function of the clock low time (t_3), the clock rise time (t_{10}), and the minimum data setup time ($t_{5(min)}$). This value is calculated as $t_{6(max)} = t_3 - t_{10} - t_{5(min)}$.
[7] C_b is the total capacitance of one bus line in picofarads.

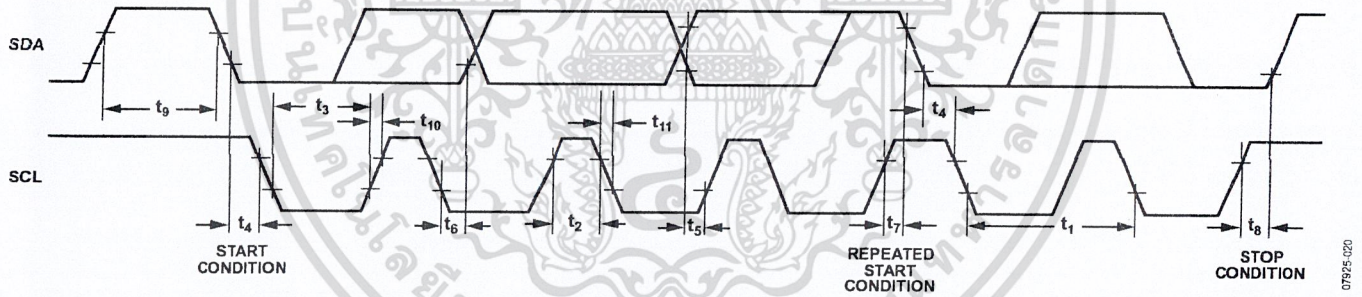


Figure 10. I²C Timing Diagram

INTERRUPTS

The ADXL345 provides two output pins for driving interrupts: INT1 and INT2. Each interrupt function is described in detail in this section. All functions can be used simultaneously, with the only limiting feature being that some functions may need to share interrupt pins. Interrupts are enabled by setting the appropriate bit in the INT_ENABLE register (Address 0x2E) and are mapped to either the INT1 or INT2 pin based on the contents of the INT_MAP register (Address 0x2F). It is recommended that interrupt bits be configured with the interrupts disabled, preventing interrupts from being accidentally triggered during configuration. This can be done by writing a value of 0x00 to the INT_ENABLE register. Clearing interrupts is performed either by reading the data registers (Address 0x32 to Address 0x37) until the interrupt condition is no longer valid for the data-related interrupts or by reading the INT_SOURCE register (Address 0x30) for the remaining interrupts. This section describes the interrupts that can be set in the INT_ENABLE register and monitored in the INT_SOURCE register.

DATA_READY

The DATA_READY bit is set when new data is available and is cleared when no new data is available.

SINGLE_TAP

The SINGLE_TAP bit is set when a single acceleration event that is greater than the value in the THRESH_TAP register (Address 0x1D) occurs for less time than is specified in the DUR register (Address 0x21).

DOUBLE_TAP

The DOUBLE_TAP bit is set when two acceleration events that are greater than the value in the THRESH_TAP register (Address 0x1D) occur for less time than is specified in the DUR register (Address 0x21), with the second tap starting after the time specified by the latent register (Address 0x22) but within the time specified in the window register (Address 0x23). See the Tap Detection section for more details.

Activity

The activity bit is set when acceleration greater than the value stored in the THRESH_ACT register (Address 0x24) is experienced.

Inactivity

The inactivity bit is set when acceleration of less than the value stored in the THRESH_INACT register (Address 0x25) is experienced for more time than is specified in the TIME_INACT register (Address 0x26). The maximum value for TIME_INACT is 255 sec.

FREE_FALL

The FREE_FALL bit is set when acceleration of less than the value stored in the THRESH_FF register (Address 0x28) is experienced for more time than is specified in the TIME_FF register (Address 0x29). The FREE_FALL interrupt differs from the inactivity interrupt as follows: all axes always participate, the timer period is much smaller (1.28 sec maximum), and the mode of operation is always dc-coupled.

Watermark

The watermark bit is set when the number of samples in FIFO equals the value stored in the samples bits (Register FIFO_CTL, Address 0x38). The watermark bit is cleared automatically when FIFO is read, and the content returns to a value below the value stored in the samples bits.

Overrun

The overrun bit is set when new data replaces unread data. The precise operation of the overrun function depends on the FIFO mode. In bypass mode, the overrun bit is set when new data replaces unread data in the DATAX, DATAY, and DATAZ registers (Address 0x32 to Address 0x37). In all other modes, the overrun bit is set when FIFO is filled. The overrun bit is automatically cleared when the contents of FIFO are read.

FIFO

The ADXL345 contains patent pending technology for an embedded 32-level FIFO that can be used to minimize host processor burden. This buffer has four modes: bypass, FIFO, stream, and trigger (see Table 19). Each mode is selected by the settings of the FIFO_MODE bits in the FIFO_CTL register (Address 0x38).

Bypass Mode

In bypass mode, FIFO is not operational and, therefore, remains empty.

FIFO Mode

In FIFO mode, data from measurements of the x-, y-, and z-axes are stored in FIFO. When the number of samples in FIFO equals the level specified in the samples bits of the FIFO_CTL register (Address 0x38), the watermark interrupt is set. FIFO continues accumulating samples until it is full (32 samples from measurements of the x-, y-, and z-axes) and then stops collecting data. After FIFO stops collecting data, the device continues to operate; therefore, features such as tap detection can be used after FIFO is full. The watermark interrupt continues to occur until the number of samples in FIFO is less than the value stored in the samples bits of the FIFO_CTL register.

Stream Mode

In stream mode, data from measurements of the x-, y-, and z-axes are stored in FIFO. When the number of samples in FIFO equals the level specified in the samples bits of the FIFO_CTL register (Address 0x38), the watermark interrupt is set. FIFO continues accumulating samples and holds the latest 32 samples from measurements of the x-, y-, and z-axes, discarding older data as new data arrives. The watermark interrupt continues occurring until the number of samples in FIFO is less than the value stored in the samples bits of the FIFO_CTL register.

Trigger Mode

In trigger mode, FIFO accumulates samples, holding the latest 32 samples from measurements of the x-, y-, and z-axes. After a trigger event occurs and an interrupt is sent to the INT1 or INT2 pin (determined by the trigger bit in the FIFO_CTL register), FIFO keeps the last n samples (where n is the value specified by the samples bits in the FIFO_CTL register) and then operates in FIFO mode, collecting new samples only when FIFO is not full. A delay of at least 5 μs should be present between the trigger event occurring and the start of reading data from the FIFO to allow the FIFO to discard and retain the necessary samples. Additional trigger events cannot be recognized until the trigger mode is reset. To reset the trigger mode, set the device to bypass mode and then set the device back to trigger mode. Note that the FIFO data should be read first because placing the device into bypass mode clears FIFO.

Retrieving Data from FIFO

The FIFO data is read through the DATAX, DATAY, and DATAZ registers (Address 0x32 to Address 0x37). When the FIFO is in FIFO, stream, or trigger mode, reads to the DATAX, DATAY, and DATAZ registers read data stored in the FIFO. Each time data is read from the FIFO, the oldest x-, y-, and z-axes data are placed into the DATAX, DATAY and DATAZ registers.

If a single-byte read operation is performed, the remaining bytes of data for the current FIFO sample are lost. Therefore, all axes of interest should be read in a burst (or multiple-byte) read operation. To ensure that the FIFO has completely popped (that is, that new data has completely moved into the DATAX, DATAY, and DATAZ registers), there must be at least 5 μs between the end of reading the data registers and the start of a new read of the FIFO or a read of the FIFO_STATUS register (Address 0x39). The end of reading a data register is signified by the transition from Register 0x37 to Register 0x38 or by the $\overline{CS}$ pin going high.

For SPI operation at 1.6 MHz or less, the register addressing portion of the transmission is a sufficient delay to ensure that the FIFO has completely popped. For SPI operation greater than 1.6 MHz, it is necessary to deassert the $\overline{CS}$ pin to ensure a total delay of 5 μs; otherwise, the delay will not be sufficient. The total delay necessary for 5 MHz operation is at most 3.4 μs. This is not a concern when using I^2C mode because the communication rate is low enough to ensure a sufficient delay between FIFO reads.

SELF-TEST

The ADXL345 incorporates a self-test feature that effectively tests its mechanical and electronic systems simultaneously. When the self-test function is enabled (via the SELF_TEST bit in the DATA_FORMAT register, Address 0x31), an electrostatic force is exerted on the mechanical sensor. This electrostatic force moves the mechanical sensing element in the same manner as acceleration, and it is additive to the acceleration experienced by the device. This added electrostatic force results in an output change in the x-, y-, and z-axes. Because the electrostatic force is proportional to V_S^2, the output change varies with V_S. The self-test feature of the ADXL345 also exhibits a bimodal behavior that depends on which phase of the clock self-test is enabled. However, the limits shown in Table 1 and Table 12 to Table 15 are valid for all potential self-test values across the entire allowable voltage range. Use of the self-test feature at data rates less than 100 Hz may yield values outside these limits. Therefore, the part should be placed into a data rate of 100 Hz or greater when using self-test.

Table 12. Self-Test Output in LSB for ±2 *g*, Full Resolution

Axis	Min	Max	Unit
X	50	540	LSB
Y	−540	−50	LSB
Z	75	875	LSB

Table 13. Self-Test Output in LSB for ±4 *g*, 10-Bit Resolution

Axis	Min	Max	Unit
X	25	270	LSB
Y	−270	−25	LSB
Z	38	438	LSB

Table 14. Self-Test Output in LSB for ±8 *g*, 10-Bit Resolution

Axis	Min	Max	Unit
X	12	135	LSB
Y	−135	−12	LSB
Z	19	219	LSB

Table 15. Self-Test Output in LSB for ±16 *g*, 10-Bit Resolution

Axis	Min	Max	Unit
X	6	67	LSB
Y	−67	−6	LSB
Z	10	110	LSB

REGISTER MAP

Table 16. Register Map

Address					
Hex	Dec	Name	Type	Reset Value	Description
0x00	0	DEVID	R	11100101	Device ID.
0x01 to 0x01C	1 to 28	Reserved			Reserved. Do not access.
0x1D	29	THRESH_TAP	R/$\overline{W}$	00000000	Tap threshold.
0x1E	30	OFSX	R/$\overline{W}$	00000000	X-axis offset.
0x1F	31	OFSY	R/$\overline{W}$	00000000	Y-axis offset.
0x20	32	OFSZ	R/$\overline{W}$	00000000	Z-axis offset.
0x21	33	DUR	R/$\overline{W}$	00000000	Tap duration.
0x22	34	Latent	R/$\overline{W}$	00000000	Tap latency.
0x23	35	Window	R/$\overline{W}$	00000000	Tap window.
0x24	36	THRESH_ACT	R/$\overline{W}$	00000000	Activity threshold.
0x25	37	THRESH_INACT	R/$\overline{W}$	00000000	Inactivity threshold.
0x26	38	TIME_INACT	R/$\overline{W}$	00000000	Inactivity time.
0x27	39	ACT_INACT_CTL	R/$\overline{W}$	00000000	Axis enable control for activity and inactivity detection.
0x28	40	THRESH_FF	R/$\overline{W}$	00000000	Free-fall threshold.
0x29	41	TIME_FF	R/$\overline{W}$	00000000	Free-fall time.
0x2A	42	TAP_AXES	R/$\overline{W}$	00000000	Axis control for tap/double tap.
0x2B	43	ACT_TAP_STATUS	R	00000000	Source of tap/double tap.
0x2C	44	BW_RATE	R/$\overline{W}$	00001010	Data rate and power mode control.
0x2D	45	POWER_CTL	R/$\overline{W}$	00000000	Power-saving features control.
0x2E	46	INT_ENABLE	R/$\overline{W}$	00000000	Interrupt enable control.
0x2F	47	INT_MAP	R/$\overline{W}$	00000000	Interrupt mapping control.
0x30	48	INT_SOURCE	R	00000010	Source of interrupts.
0x31	49	DATA_FORMAT	R/$\overline{W}$	00000000	Data format control.
0x32	50	DATAX0	R	00000000	X-Axis Data 0.
0x33	51	DATAX1	R	00000000	X-Axis Data 1.
0x34	52	DATAY0	R	00000000	Y-Axis Data 0.
0x35	53	DATAY1	R	00000000	Y-Axis Data 1.
0x36	54	DATAZ0	R	00000000	Z-Axis Data 0.
0x37	55	DATAZ1	R	00000000	Z-Axis Data 1.
0x38	56	FIFO_CTL	R/$\overline{W}$	00000000	FIFO control.
0x39	57	FIFO_STATUS	R	00000000	FIFO status.

REGISTER DEFINITIONS

Register 0x00—DEVID (Read Only)

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	0	0	1	0	1

The DEVID register holds a fixed device ID code of 0xE5 (345 octal).

Register 0x1D—THRESH_TAP (Read/Write)

The THRESH_TAP register is eight bits and holds the threshold value for tap interrupts. The data format is unsigned, so the magnitude of the tap event is compared with the value in THRESH_TAP. The scale factor is 62.5 mg/LSB (that is, 0xFF = +16 *g*). A value of 0 may result in undesirable behavior if tap/double tap interrupts are enabled.

Register 0x1E, Register 0x1F, Register 0x20—OFSX, OFSY, OFSZ (Read/Write)

The OFSX, OFSY, and OFSZ registers are each eight bits and offer user-set offset adjustments in twos complement format with a scale factor of 15.6 mg/LSB (that is, 0x7F = +2 *g*).

Register 0x21—DUR (Read/Write)

The DUR register is eight bits and contains an unsigned time value representing the maximum time that an event must be above the THRESH_TAP threshold to qualify as a tap event. The scale factor is 625 µs/LSB. A value of 0 disables the tap/double tap functions.

Register 0x22—Latent (Read/Write)

The latent register is eight bits and contains an unsigned time value representing the wait time from the detection of a tap event to the start of the time window (defined by the window register) during which a possible second tap event can be detected. The scale factor is 1.25 ms/LSB. A value of 0 disables the double tap function.

Register 0x23—Window (Read/Write)

The window register is eight bits and contains an unsigned time value representing the amount of time after the expiration of the latency time (determined by the latent register) during which a second valid tap can begin. The scale factor is 1.25 ms/LSB. A value of 0 disables the double tap function.

Register 0x24—THRESH_ACT (Read/Write)

The THRESH_ACT register is eight bits and holds the threshold value for detecting activity. The data format is unsigned, so the magnitude of the activity event is compared with the value in the THRESH_ACT register. The scale factor is 62.5 mg/LSB. A value of 0 may result in undesirable behavior if the activity interrupt is enabled.

Register 0x25—THRESH_INACT (Read/Write)

The THRESH_INACT register is eight bits and holds the threshold value for detecting inactivity. The data format is unsigned, so the magnitude of the inactivity event is compared with the value in the THRESH_INACT register. The scale factor is 62.5 mg/LSB. A value of 0 mg may result in undesirable behavior if the inactivity interrupt is enabled.

Register 0x26—TIME_INACT (Read/Write)

The TIME_INACT register is eight bits and contains an unsigned time value representing the amount of time that acceleration must be less than the value in the THRESH_INACT register for inactivity to be declared. The scale factor is 1 sec/LSB. Unlike the other interrupt functions, which use unfiltered data (see the Threshold section), the inactivity function uses filtered output data. At least one output sample must be generated for the inactivity interrupt to be triggered. This results in the function appearing unresponsive if the TIME_INACT register is set to a value less than the time constant of the output data rate. A value of 0 results in an interrupt when the output data is less than the value in the THRESH_INACT register.

Register 0x27—ACT_INACT_CTL (Read/Write)

D7	D6	D5	D4
ACT ac/dc	ACT_X enable	ACT_Y enable	ACT_Z enable
D3	**D2**	**D1**	**D0**
INACT ac/dc	INACT_X enable	INACT_Y enable	INACT_Z enable

ACT AC/DC and INACT AC/DC Bits

A setting of 0 selects dc-coupled operation, and a setting of 1 enables ac-coupled operation. In dc-coupled operation, the current acceleration magnitude is compared directly with THRESH_ACT and THRESH_INACT to determine whether activity or inactivity is detected.

In ac-coupled operation for activity detection, the acceleration value at the start of activity detection is taken as a reference value. New samples of acceleration are then compared to this reference value, and if the magnitude of the difference exceeds the THRESH_ACT value, the device triggers an activity interrupt.

Similarly, in ac-coupled operation for inactivity detection, a reference value is used for comparison and is updated whenever the device exceeds the inactivity threshold. After the reference value is selected, the device compares the magnitude of the difference between the reference value and the current acceleration with THRESH_INACT. If the difference is less than the value in THRESH_INACT for the time in TIME_INACT, the device is considered inactive and the inactivity interrupt is triggered.

ACT_x Enable Bits and INACT_x Enable Bits

A setting of 1 enables x-, y-, or z-axis participation in detecting activity or inactivity. A setting of 0 excludes the selected axis from participation. If all axes are excluded, the function is disabled.

Register 0x28—THRESH_FF (Read/Write)

The THRESH_FF register is eight bits and holds the threshold value, in unsigned format, for free-fall detection. The root-sum-square (RSS) value of all axes is calculated and compared with the value in THRESH_FF to determine if a free-fall event occurred. The scale factor is 62.5 mg/LSB. Note that a value of 0 mg may result in undesirable behavior if the free-fall interrupt is enabled. Values between 300 mg and 600 mg (0x05 to 0x09) are recommended.

Register 0x29—TIME_FF (Read/Write)

The TIME_FF register is eight bits and stores an unsigned time value representing the minimum time that the RSS value of all axes must be less than THRESH_FF to generate a free-fall interrupt. The scale factor is 5 ms/LSB. A value of 0 may result in undesirable behavior if the free-fall interrupt is enabled. Values between 100 ms and 350 ms (0x14 to 0x46) are recommended.

Register 0x2A—TAP_AXES (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	Suppress	TAP_X enable	TAP_Y enable	TAP_Z enable

Suppress Bit

Setting the suppress bit suppresses double tap detection if acceleration greater than the value in THRESH_TAP is present between taps. See the Tap Detection section for more details.

TAP_x Enable Bits

A setting of 1 in the TAP_X enable, TAP_Y enable, or TAP_Z enable bit enables x-, y-, or z-axis participation in tap detection. A setting of 0 excludes the selected axis from participation in tap detection.

Register 0x2B—ACT_TAP_STATUS (Read Only)

D7	D6	D5	D4	D3	D2	D1	D0
0	ACT_X source	ACT_Y source	ACT_Z source	Asleep	TAP_X source	TAP_Y source	TAP_Z source

ACT_x Source and TAP_x Source Bits

These bits indicate the first axis involved in a tap or activity event. A setting of 1 corresponds to involvement in the event, and a setting of 0 corresponds to no involvement. When new data is available, these bits are not cleared but are overwritten by the new data. The ACT_TAP_STATUS register should be read before clearing the interrupt. Disabling an axis from participation clears the corresponding source bit when the next activity or tap/double tap event occurs.

Asleep Bit

A setting of 1 in the asleep bit indicates that the part is asleep, and a setting of 0 indicates that the part is not asleep. See the Register 0x2D—POWER_CTL (Read/Write) section for more information on autosleep mode.

Register 0x2C—BW_RATE (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	LOW_POWER	Rate			

LOW_POWER Bit

A setting of 0 in the LOW_POWER bit selects normal operation, and a setting of 1 selects reduced power operation, which has somewhat higher noise (see the Power Modes section for details).

Rate Bits

These bits select the device bandwidth and output data rate (see Table 6 and Table 7 for details). The default value is 0x0A, which translates to a 100 Hz output data rate. An output data rate should be selected that is appropriate for the communication protocol and frequency selected. Selecting too high of an output data rate with a low communication speed results in samples being discarded.

Register 0x2D—POWER_CTL (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
0	0	Link	AUTO_SLEEP	Measure	Sleep	Wakeup	

Link Bit

A setting of 1 in the link bit with both the activity and inactivity functions enabled delays the start of the activity function until inactivity is detected. After activity is detected, inactivity detection begins, preventing the detection of activity. This bit serially links the activity and inactivity functions. When this bit is set to 0, the inactivity and activity functions are concurrent. Additional information can be found in the Link Mode section.

When clearing the link bit, it is recommended that the part be placed into standby mode and then set back to measurement mode with a subsequent write. This is done to ensure that the device is properly biased if sleep mode is manually disabled; otherwise, the first few samples of data after the link bit is cleared may have additional noise, especially if the device was asleep when the bit was cleared.

AUTO_SLEEP Bit

If the link bit is set, a setting of 1 in the AUTO_SLEEP bit sets the ADXL345 to switch to sleep mode when inactivity is detected (that is, when acceleration has been below the THRESH_INACT value for at least the time indicated by TIME_INACT). A setting of 0 disables automatic switching to sleep mode. See the description of the sleep bit in this section for more information.

When clearing the AUTO_SLEEP bit, it is recommended that the part be placed into standby mode and then set back to measurement mode with a subsequent write. This is done to ensure that the device is properly biased if sleep mode is manually disabled; otherwise, the first few samples of data after the AUTO_SLEEP bit is cleared may have additional noise, especially if the device was asleep when the bit was cleared.

Measure Bit

A setting of 0 in the measure bit places the part into standby mode, and a setting of 1 places the part into measurement mode. The ADXL345 powers up in standby mode with minimum power consumption.

Sleep Bit

A setting of 0 in the sleep bit puts the part into the normal mode of operation, and a setting of 1 places the part into sleep mode. Sleep mode suppresses DATA_READY, stops transmission of data to FIFO, and switches the sampling rate to one specified by the wakeup bits. In sleep mode, only the activity function can be used.

When clearing the sleep bit, it is recommended that the part be placed into standby mode and then set back to measurement mode with a subsequent write. This is done to ensure that the device is properly biased if sleep mode is manually disabled; otherwise, the first few samples of data after the sleep bit is cleared may have additional noise, especially if the device was asleep when the bit was cleared.

Wakeup Bits

These bits control the frequency of readings in sleep mode as described in Table 17.

Table 17. Frequency of Readings in Sleep Mode

Setting		
D1	**D0**	**Frequency (Hz)**
0	0	8
0	1	4
1	0	2
1	1	1

Register 0x2E—INT_ENABLE (Read/Write)

D7	D6	D5	D4
DATA_READY	SINGLE_TAP	DOUBLE_TAP	Activity
D3	**D2**	**D1**	**D0**
Inactivity	FREE_FALL	Watermark	Overrun

Setting bits in this register to a value of 1 enables their respective functions to generate interrupts, whereas a value of 0 prevents the functions from generating interrupts. The DATA_READY, watermark, and overrun bits enable only the interrupt output; the functions are always enabled. It is recommended that interrupts be configured before enabling their outputs.

Register 0x2F—INT_MAP (Read/Write)

D7	D6	D5	D4
DATA_READY	SINGLE_TAP	DOUBLE_TAP	Activity
D3	**D2**	**D1**	**D0**
Inactivity	FREE_FALL	Watermark	Overrun

Any bits set to 0 in this register send their respective interrupts to the INT1 pin, whereas bits set to 1 send their respective interrupts to the INT2 pin. All selected interrupts for a given pin are ORed.

Register 0x30—INT_SOURCE (Read Only)

D7	D6	D5	D4
DATA_READY	SINGLE_TAP	DOUBLE_TAP	Activity
D3	**D2**	**D1**	**D0**
Inactivity	FREE_FALL	Watermark	Overrun

Bits set to 1 in this register indicate that their respective functions have triggered an event, whereas a value of 0 indicates that the corresponding event has not occurred. The DATA_READY, watermark, and overrun bits are always set if the corresponding events occur, regardless of the INT_ENABLE register settings, and are cleared by reading data from the DATAX, DATAY, and DATAZ registers. The DATA_READY and watermark bits may require multiple reads, as indicated in the FIFO mode descriptions in the FIFO section. Other bits, and the corresponding interrupts, are cleared by reading the INT_SOURCE register.

Register 0x31—DATA_FORMAT (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
SELF_TEST	SPI	INT_INVERT	0	FULL_RES	Justify	Range	

The DATA_FORMAT register controls the presentation of data to Register 0x32 through Register 0x37. All data, except that for the ±16 *g* range, must be clipped to avoid rollover.

SELF_TEST Bit

A setting of 1 in the SELF_TEST bit applies a self-test force to the sensor, causing a shift in the output data. A value of 0 disables the self-test force.

SPI Bit

A value of 1 in the SPI bit sets the device to 3-wire SPI mode, and a value of 0 sets the device to 4-wire SPI mode.

INT_INVERT Bit

A value of 0 in the INT_INVERT bit sets the interrupts to active high, and a value of 1 sets the interrupts to active low.

FULL_RES Bit

When this bit is set to a value of 1, the device is in full resolution mode, where the output resolution increases with the *g* range set by the range bits to maintain a 4 mg/LSB scale factor. When the FULL_RES bit is set to 0, the device is in 10-bit mode, and the range bits determine the maximum *g* range and scale factor.

Justify Bit

A setting of 1 in the justify bit selects left (MSB) justified mode, and a setting of 0 selects right justified mode with sign extension.

Range Bits

These bits set the *g* range as described in Table 18.

Table 18. *g* Range Setting

Setting		
D1	**D0**	***g* Range**
0	0	±2 *g*
0	1	±4 *g*
1	0	±8 *g*
1	1	±16 *g*

Register 0x32 to Register 0x37—DATAX0, DATAX1, DATAY0, DATAY1, DATAZ0, DATAZ1 (Read Only)

These six bytes (Register 0x32 to Register 0x37) are eight bits each and hold the output data for each axis. Register 0x32 and Register 0x33 hold the output data for the x-axis, Register 0x34 and Register 0x35 hold the output data for the y-axis, and Register 0x36 and Register 0x37 hold the output data for the z-axis. The output data is twos complement, with DATAx0 as the least significant byte and DATAx1 as the most significant byte, where x represent X, Y, or Z. The DATA_FORMAT register (Address 0x31) controls the format of the data. It is recommended that a multiple-byte read of all registers be performed to prevent a change in data between reads of sequential registers.

Register 0x38—FIFO_CTL (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
FIFO_MODE		Trigger	Samples				

FIFO_MODE Bits

These bits set the FIFO mode, as described in Table 19.

Table 19. FIFO Modes

Setting			
D7	D6	Mode	Function
0	0	Bypass	FIFO is bypassed.
0	1	FIFO	FIFO collects up to 32 values and then stops collecting data, collecting new data only when FIFO is not full.
1	0	Stream	FIFO holds the last 32 data values. When FIFO is full, the oldest data is overwritten with newer data.
1	1	Trigger	When triggered by the trigger bit, FIFO holds the last data samples before the trigger event and then continues to collect data until full. New data is collected only when FIFO is not full.

Trigger Bit

A value of 0 in the trigger bit links the trigger event of trigger mode to INT1, and a value of 1 links the trigger event to INT2.

Samples Bits

The function of these bits depends on the FIFO mode selected (see Table 20). Entering a value of 0 in the samples bits immediately sets the watermark status bit in the INT_SOURCE register, regardless of which FIFO mode is selected. Undesirable operation may occur if a value of 0 is used for the samples bits when trigger mode is used.

Table 20. Samples Bits Functions

FIFO Mode	Samples Bits Function
Bypass	None.
FIFO	Specifies how many FIFO entries are needed to trigger a watermark interrupt.
Stream	Specifies how many FIFO entries are needed to trigger a watermark interrupt.
Trigger	Specifies how many FIFO samples are retained in the FIFO buffer before a trigger event.

0x39—FIFO_STATUS (Read Only)

D7	D6	D5	D4	D3	D2	D1	D0
FIFO_TRIG	0	Entries					

FIFO_TRIG Bit

A 1 in the FIFO_TRIG bit corresponds to a trigger event occurring, and a 0 means that a FIFO trigger event has not occurred.

Entries Bits

These bits report how many data values are stored in FIFO. Access to collect the data from FIFO is provided through the DATAX, DATAY, and DATAZ registers. FIFO reads must be done in burst or multiple-byte mode because each FIFO level is cleared after any read (single- or multiple-byte) of FIFO. FIFO stores a maximum of 32 entries, which equates to a maximum of 33 entries available at any given time because an additional entry is available at the output filter of the device.

APPLICATIONS INFORMATION

POWER SUPPLY DECOUPLING

A 1 μF tantalum capacitor (C_S) at V_S and a 0.1 μF ceramic capacitor (C_{IO}) at $V_{DD\ I/O}$ placed close to the ADXL345 supply pins is used for testing and is recommended to adequately decouple the accelerometer from noise on the power supply. If additional decoupling is necessary, a resistor or ferrite bead, no larger than 100 Ω, in series with V_S may be helpful. Additionally, increasing the bypass capacitance on V_S to a 10 μF tantalum capacitor in parallel with a 0.1 μF ceramic capacitor may also improve noise.

Care should be taken to ensure that the connection from the ADXL345 ground to the power supply ground has low impedance because noise transmitted through ground has an effect similar to noise transmitted through V_S. It is recommended that V_S and $V_{DD\ I/O}$ be separate supplies to minimize digital clocking noise on the V_S supply. If this is not possible, additional filtering of the supplies as previously mentioned may be necessary.

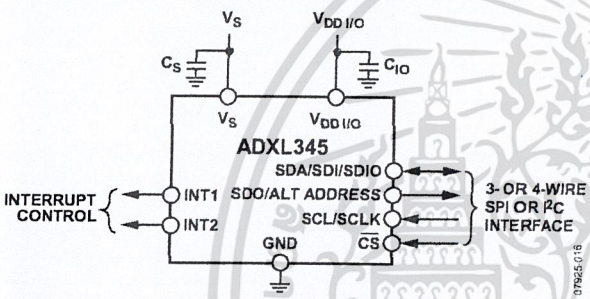


Figure 11. Application Diagram

MECHANICAL CONSIDERATIONS FOR MOUNTING

The ADXL345 should be mounted on the PCB in a location close to a hard mounting point of the PCB to the case. Mounting the ADXL345 at an unsupported PCB location, as shown in Figure 12, may result in large, apparent measurement errors due to undampened PCB vibration. Locating the accelerometer near a hard mounting point ensures that any PCB vibration at the accelerometer is above the accelerometer's mechanical sensor resonant frequency and, therefore, effectively invisible to the accelerometer.

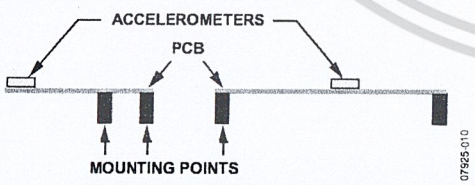


Figure 12. Incorrectly Placed Accelerometers

TAP DETECTION

The tap interrupt function is capable of detecting either single or double taps. The following parameters are shown in Figure 13 for a valid single and valid double tap event:

- The tap detection threshold is defined by the THRESH_TAP register (Address 0x1D).
- The maximum tap duration time is defined by the DUR register (Address 0x21).
- The tap latency time is defined by the latent register (Address 0x22) and is the waiting period from the end of the first tap until the start of the time window, when a second tap can be detected, which is determined by the value in the window register (Address 0x23).
- The interval after the latency time (set by the latent register) is defined by the window register. Although a second tap must begin after the latency time has expired, it need not finish before the end of the time defined by the window register.

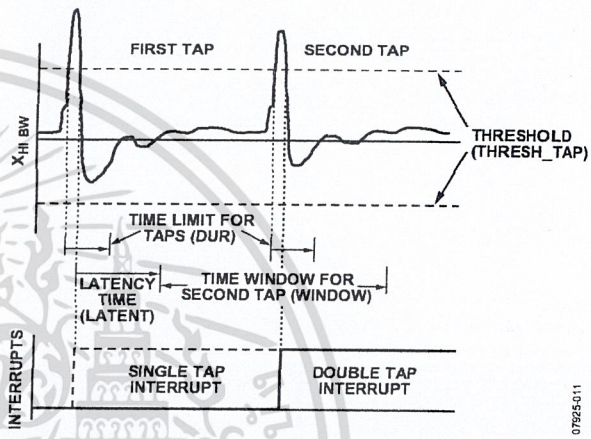


Figure 13. Tap Interrupt Function with Valid Single and Double Taps

If only the single tap function is in use, the single tap interrupt is triggered when the acceleration goes below the threshold, as long as DUR has not been exceeded. If both single and double tap functions are in use, the single tap interrupt is triggered when the double tap event has been either validated or invalidated.

Several events can occur to invalidate the second tap of a double tap event. First, if the suppress bit in the TAP_AXES register (Address 0x2A) is set, any acceleration spike above the threshold during the latency time (set by the latent register) invalidates the double tap detection, as shown in Figure 14.

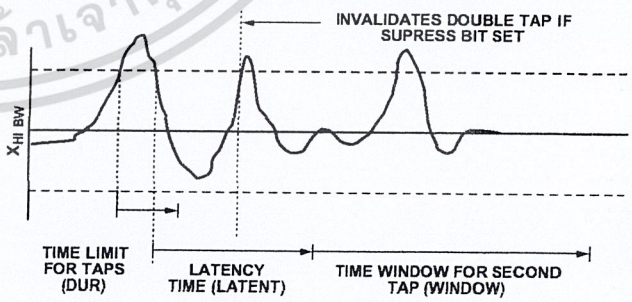


Figure 14. Double Tap Event Invalid Due to High g Event When the Suppress Bit Is Set

A double tap event can also be invalidated if acceleration above the threshold is detected at the start of the time window for the second tap (set by the window register). This results in an invalid double tap at the start of this window, as shown in Figure 15. Additionally, a double tap event can be invalidated if an accel-

eration exceeds the time limit for taps (set by the DUR register), resulting in an invalid double tap at the end of the DUR time limit for the second tap event, also shown in Figure 15.

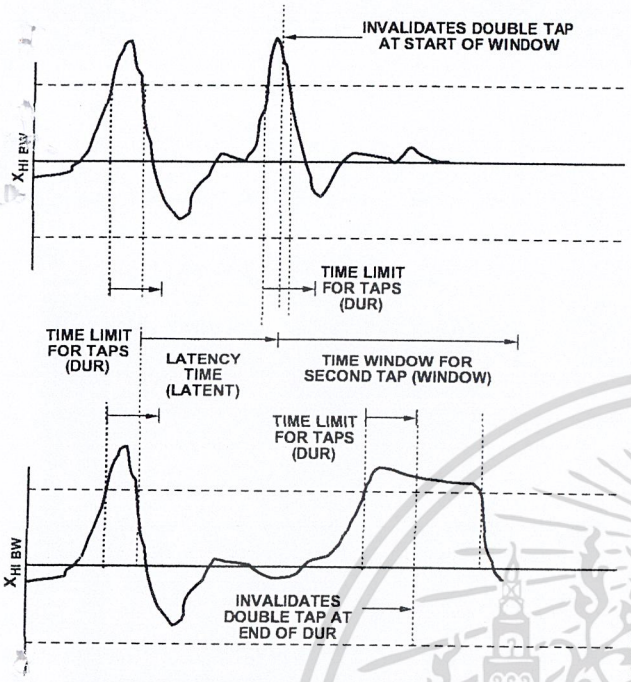


Figure 15. Tap Interrupt Function with Invalid Double Taps

Single taps, double taps, or both can be detected by setting the respective bits in the INT_ENABLE register (Address 0x2E). Control over participation of each of the three axes in single tap/double tap detection is exerted by setting the appropriate bits in the TAP_AXES register (Address 0x2A). For the double tap function to operate, both the latent and window registers must be set to a nonzero value.

Every mechanical system has somewhat different single tap/double tap responses based on the mechanical characteristics of the system. Therefore, some experimentation with values for the latent, window, and THRESH_TAP registers is required. In general, a good starting point is to set the latent register to a value greater than 0x10, to set the window register to a value greater than 0x10, and to set the THRESH_TAP register to be greater than 3 *g*. Setting a very low value in the latent, window, or THRESH_TAP register may result in an unpredictable response due to the accelerometer picking up echoes of the tap inputs.

After a tap interrupt has been received, the first axis to exceed the THRESH_TAP level is reported in the ACT_TAP_STATUS register (Address 0x2B). This register is never cleared, but is overwritten with new data.

THRESHOLD

The lower output data rates are achieved by decimating a common sampling frequency inside the device. The activity, free-fall, and single tap/double tap detection functions are performed using unfiltered data. Since the output data is filtered, the high frequency and high *g* data that is used to determine activity, free-fall, and single tap/double tap events may not be present if the output of the accelerometer is examined. This may result in trigger events being detected when acceleration does not appear to trigger an event because the unfiltered data may have exceeded a threshold or remained below a threshold for a certain period of time while the filtered output data has not exceeded such a threshold.

LINK MODE

The function of the link bit is to reduce the number of activity interrupts that the processor must service by setting the device to look for activity only after inactivity. For proper operation of this feature, the processor must still respond to the activity and inactivity interrupts by reading the INT_SOURCE register (Address 0x30) and, therefore, clearing the interrupts. If an activity interrupt is not cleared, the part cannot go into autosleep mode. The asleep bit in the ACT_TAP_STATUS register (Address 0x2B) indicates if the part is asleep.

SLEEP MODE VS. LOW POWER MODE

In applications where a low data rate is sufficient and low power consumption is desired, it is recommended that the low power mode be used in conjunction with the FIFO. The sleep mode, while offering a low data rate and low average current consumption, suppresses the DATA_READY interrupt, preventing the accelerometer from sending an interrupt signal to the host processor when data is ready to be collected. In this application, setting the part into low power mode (by setting the LOW_POWER bit in the BW_RATE register) and enabling the FIFO in FIFO mode to collect a large value of samples reduces the power consumption of the ADXL345 and allows the host processor to go to sleep while the FIFO is filling up.

USING SELF-TEST

The self-test change is defined as the difference between the acceleration output of an axis with self-test enabled and the acceleration output of the same axis with self-test disabled (see Endnote 4 of Table 1). This definition assumes that the sensor does not move between these two measurements, because if the sensor moves, a non–self-test related shift corrupts the test.

Proper configuration of the ADXL345 is also necessary for an accurate self-test measurement. The part should be set with a data rate greater than or equal to 100 Hz. This is done by ensuring that a value greater than or equal to 0x0A is written into the rate bits (Bit D3 through Bit D0) in the BW_RATE register (Address 0x2C). It is also recommended that the part be set to full-resolution, 16 *g* mode to ensure that there is sufficient dynamic range for the entire self-test shift. This is done by setting Bit D3 of the DATA_FORMAT register (Address 0x31) and writing a value of 0x03 to the range bits (Bit D1 and Bit D0) of the DATA_FORMAT register (Address 0x31). This results in a high dynamic range for measurement and a 3.9 mg/LSB scale factor.

After the part is configured for accurate self-test measurement, several samples of x-, y-, and z-axis acceleration data should be retrieved from the sensor and averaged together. The number of

samples averaged is a choice of the system designer, but a recommended starting point is 0.1 sec worth of data, which corresponds to 10 samples at 100 Hz data rate. The averaged values should be stored and labeled appropriately as the self-test disabled data, that is, X_{ST_OFF}, Y_{ST_OFF}, and Z_{ST_OFF}.

Next, self-test should be enabled by setting Bit D7 of the DATA_FORMAT register (Address 0x31). The output needs some time (about four samples) to settle after enabling self-test. After allowing the output to settle, several samples of the x-, y-, and z-axis acceleration data should be taken again and averaged. It is recommended that the same number of samples be taken for this average as was previously taken. These averaged values should again be stored and labeled appropriately as the value with self-test enabled, that is, X_{ST_ON}, Y_{ST_ON}, and Z_{ST_ON}. Self-test can then be disabled by clearing Bit D7 of the DATA_FORMAT register (Address 0x31).

With the stored values for self-test enabled and disabled, the self-test change is as follows:

$$X_{ST} = X_{ST\_ON} - X_{ST\_OFF}$$

$$Y_{ST} = Y_{ST\_ON} - Y_{ST\_OFF}$$

$$Z_{ST} = Z_{ST\_ON} - Z_{ST\_OFF}$$

Because the measured output for each axis is expressed in LSBs, X_{ST}, Y_{ST}, and Z_{ST} are also expressed in LSBs. These values can be converted to *g*'s of acceleration by multiplying each value by the 3.9 mg/LSB scale factor, if configured for full-resolution, 16 *g* mode. Additionally, Table 12 through Table 15 correspond to the self-test range converted to LSBs and can be compared with the measured self-test change. If the part was placed into full-resolution, 16 *g* mode, the values listed in Table 12 should be used. Although the fixed 10-bit mode or a range other than 16 *g* can be used, a different set of values, as indicated in Table 13 through Table 15, would need to be used. Using a range below 8 *g* may result in insufficient dynamic range and should be considered when selecting the range of operation for measuring self-test. In addition, note that the range in Table 1 and the values in Table 12 through Table 15 take into account all possible supply voltages, V_S, and no additional conversion due to V_S is necessary.

If the self-test change is within the valid range, the test is considered successful. Generally, a part is considered to pass if the minimum magnitude of change is achieved. However, a part that changes by more than the maximum magnitude is not necessarily a failure.

AXES OF ACCELERATION SENSITIVITY

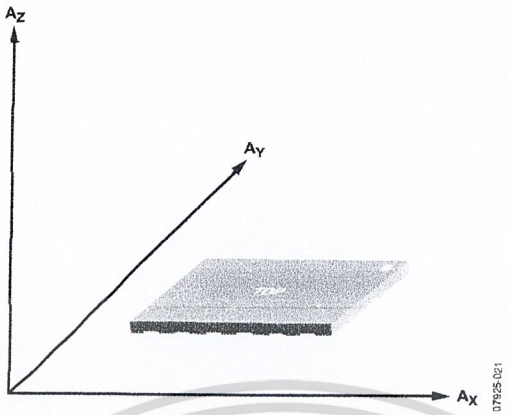


Figure 16. Axes of Acceleration Sensitivity (Corresponding Output Voltage Increases When Accelerated Along the Sensitive Axis)

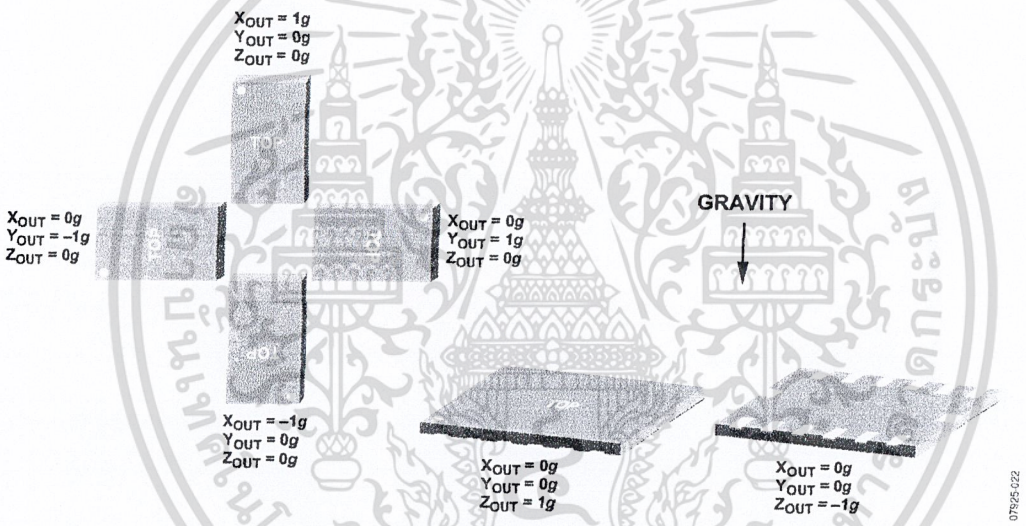


Figure 17. Output Response vs. Orientation to Gravity

LAYOUT AND DESIGN RECOMMENDATIONS

Figure 18 shows the recommended printed wiring board land pattern. Figure 19 and Table 21 provide details about the recommended soldering profile.

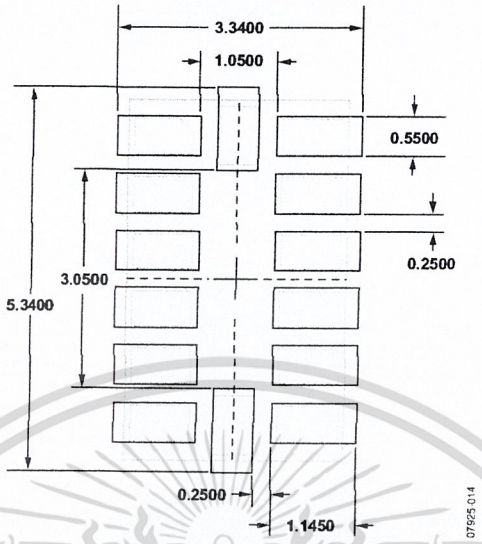


Figure 18. Recommended Printed Wiring Board Land Pattern (Dimensions shown in millimeters)

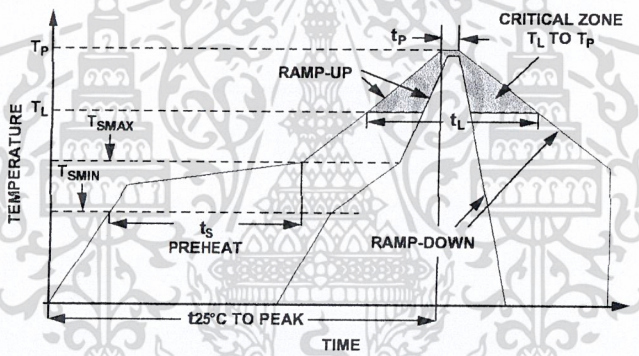


Figure 19. Recommended Soldering Profile

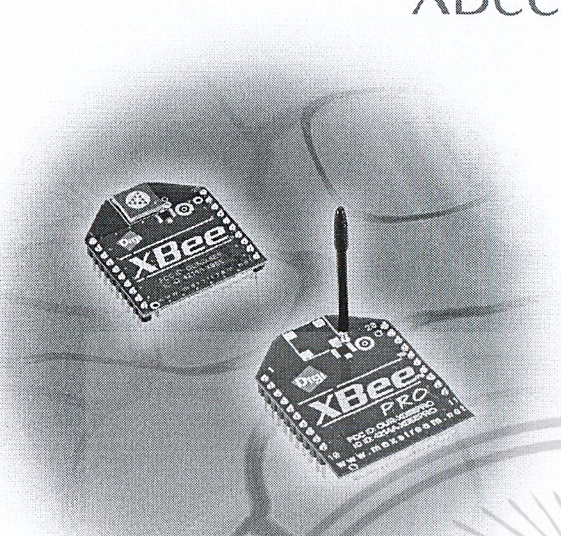
Table 21. Recommended Soldering Profile[1, 2]

Profile Feature	Condition	
	Sn63/Pb37	Pb-Free
Average Ramp Rate from Liquid Temperature (T_L) to Peak Temperature (T_P)	3°C/sec max	3°C/sec max
Preheat		
Minimum Temperature (T_{SMIN})	100°C	150°C
Maximum Temperature (T_{SMAX})	150°C	200°C
Time from T_{SMIN} to T_{SMAX} (t_S)	60 sec to 120 sec	60 sec to 180 sec
T_{SMAX} to T_L Ramp-Up Rate	3°C/sec max	3°C/sec max
Liquid Temperature (T_L)	183°C	217°C
Time Maintained Above T_L (t_L)	60 sec to 150 sec	60 sec to 150 sec
Peak Temperature (T_P)	240 + 0/−5°C	260 + 0/−5°C
Time of Actual T_P − 5°C (t_P)	10 sec to 30 sec	20 sec to 40 sec
Ramp-Down Rate	6°C/sec max	6°C/sec max
Time 25°C to Peak Temperature	6 minutes max	8 minutes max

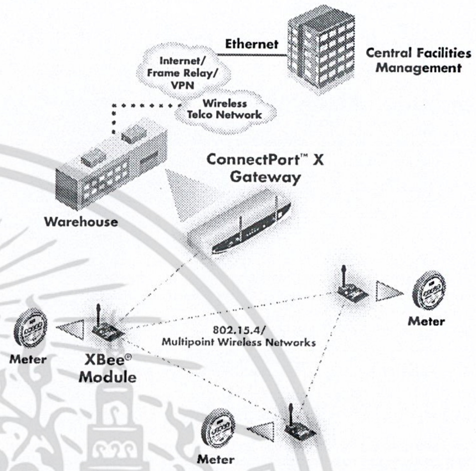
[1] Based on JEDEC Standard J-STD-020D.1.
[2] For best results, the soldering profile should be in accordance with the recommendations of the manufacturer of the solder paste used.

XBee® Multipoint RF Modules

Embedded RF Modules for OEMs



Providing critical end-point connectivity to Digi's Drop-in Networking product family, XBee multipoint RF modules are low-cost and easy to deploy.



Features/Benefits

- 802.15.4/Multipoint network topologies
- 2.4 GHz for worldwide deployment
- 900 MHz for long-range deployment
- Fully interoperable with other Digi Drop-in Networking products, including gateways, device adapters and extenders
- Common XBee footprint for a variety of RF modules
- Low-power sleep modes
- Multiple antenna options
- Industrial temperature rating (-40° C to 85° C)
- Low power and long range variants available

Overview

XBee Product Family

The XBee family of embedded RF modules provides OEMs with a common footprint shared by multiple platforms, including multipoint and ZigBee/Mesh topologies, and both 2.4 GHz and 900 MHz solutions. OEMs deploying the XBee can substitute one XBee for another, depending upon dynamic application needs, with minimal development, reduced risk and shorter time-to-market.

Why XBee Multipoint RF Modules?

XBee multipoint RF modules are ideal for applications requiring low latency and predictable communication timing. Providing quick, robust communication in point-to-point, peer-to-peer, and multipoint/star configurations, XBee multipoint products enable robust end-point connectivity with ease. Whether deployed as a pure cable replacement for simple serial communication, or as part of a more complex hub-and-spoke network of sensors, XBee multipoint RF modules maximize wireless performance and ease of development.

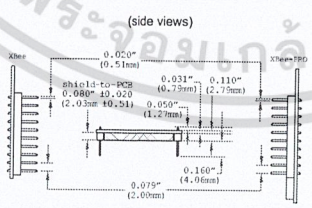
Drop-in Networking End-Point Connectivity

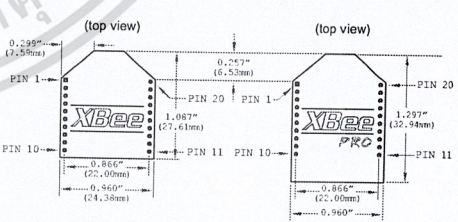
XBee OEM RF modules are part of Digi's Drop-in Networking family of end-to-end connectivity solutions. By seamlessly interfacing with compatible gateways, device adapters and extenders, XBee embedded RF modules provide developers with true beyond-the-horizon connectivity.

Platform	XBee® 802.15.4 (Series 1)	XBee-PRO® 802.15.4 (Series 1)	XBee-PRO® XSC
Performance			
RF Data Rate	250 kbps	250 kbps	10 kbps / 9.6 kbps
Indor/Urban Range	100 ft (30 m)	300 ft (100 m)	Up to 1200 ft (370 m)
Outdoor/RF Line-of-Sight Range	300 ft (100 m)	1 mi (1.6 km)	Up to 6 mi (9.6 km)
Transmit Power	1 mW (+0 dBm)	60 mW (+18 dBm)*	100 mW (+20 dBm)
Receiver Sensitivity (1% PER)	-92 dBm	-100 dBm	-106 dBm
Features			
Serial Data Interface	3.3V CMOS UART	3.3V CMOS UART	3.3V CMOS UART (5V Tolerant)
Configuration Method	API or AT Commands, local or over-the-air	API or AT Commands, local or over-the-air	AT Commands
Frequency Band	2.4 GHz	2.4 GHz	902 MHz to 928 MHz
Interference Immunity	DSSS (Direct Sequence Spread Spectrum)	DSSS (Direct Sequence Spread Spectrum)	FHSS (Frequency Hopping Spread Spectrum)
Serial Data Rate	1200 bps - 250 kbps	1200 bps - 250 kbps	1200 bps - 57.6 kbps
ADC Inputs	(6) 10-bit ADC inputs	(6) 10-bit ADC inputs	None
Digital I/O	8	8	None
Antenna Options	Chip, Wire Whip, U.FL, & RPSMA	Chip, Wire Whip, U.FL, & RPSMA	Wire Whip, U.FL, RPSMA
Networking & Security			
Encryption	128-bit AES	128-bit AES	No
Reliable Packet Delivery	Retries/Acknowledgments	Retries/Acknowledgments	Retries/Acknowledgements
IDs and Channels	PAN ID, 64-bit IEEE MAC, 16 Channels	PAN ID, 64-bit IEEE MAC, 12 Channels	PAN ID, 32-bit Address, 7 Channels
Power Requirements			
Supply Voltage	2.8 - 3.4VDC	2.8 - 3.4VDC	3.0 - 3.6VDC
Transmit Current	45 mA @ 3.3VDC	215 mA @ 3.3VDC	265 mA typical
Receive Current	50 mA @ 3.3VDC	55 mA @ 3.3VDC	65 mA typical
Power-Down Current	<10 uA @ 25° C	<10 uA @ 25° C	45 uA pin Sleep
Regulatory Approvals			
FCC (USA)	OUR-XBEE	OUR-XBEEPRO	MCQ-XBEEXSC
IC (Canada)	4214A-XBEE	4214A-XBEEPRO	1846A-XBEEXSC
ETSI (Europe)	Yes	Yes* Max TX 10 mW	No
C-TICK Australia	Yes	Yes	No
Telec (Japan)	Yes	Yes*	No

* XBee-PRO 802.15.4 TX Power restricted to 10 mW in Europe and Japan.

802.15.4 – Star





Please visit www.digi.com for part numbers.

DIGI SERVICE AND SUPPORT - You can purchase with confidence knowing that Digi is here to support you with expert technical support and a one-year warranty. www.digi.com/support

WHEN RELIABILITY MATTERS™

Digi International
11001 Bren Road E.
Minnetonka, MN 55343
U.S.A.
PH: 877-912-3444
952-912-3444
FX: 952-912-4952
email: info@digi.com

Digi International France
31 rue des Poissonniers
92200 Neuilly sur Seine
PH: +33-1-55-61-98-98
FX: +33-1-55-61-98-99
www.digi.fr

Digi International KK
NES Building South 8F
22-14 Sakuragaoka-cho,
Shibuya-ku
Tokyo 150-0031, Japan
PH: +81-3-5428-0261
FX: +81-3-5428-0262
www.digi-intl.co.jp

Digi International (HK) Limited
Suite 1703-05, 17/F.,
K Wah Centre
191 Java Road
North Point, Hong Kong
PH: +852-2833-1008
FX: +852-2572-9989
www.digi.cn

Digi International, the leader in device networking for business, develops reliable products and technologies to connect and securely manage local or remote electronic devices over the network or via the web. With over 20 million ports shipped worldwide since 1985, Digi offers the highest levels of performance, flexibility and quality.

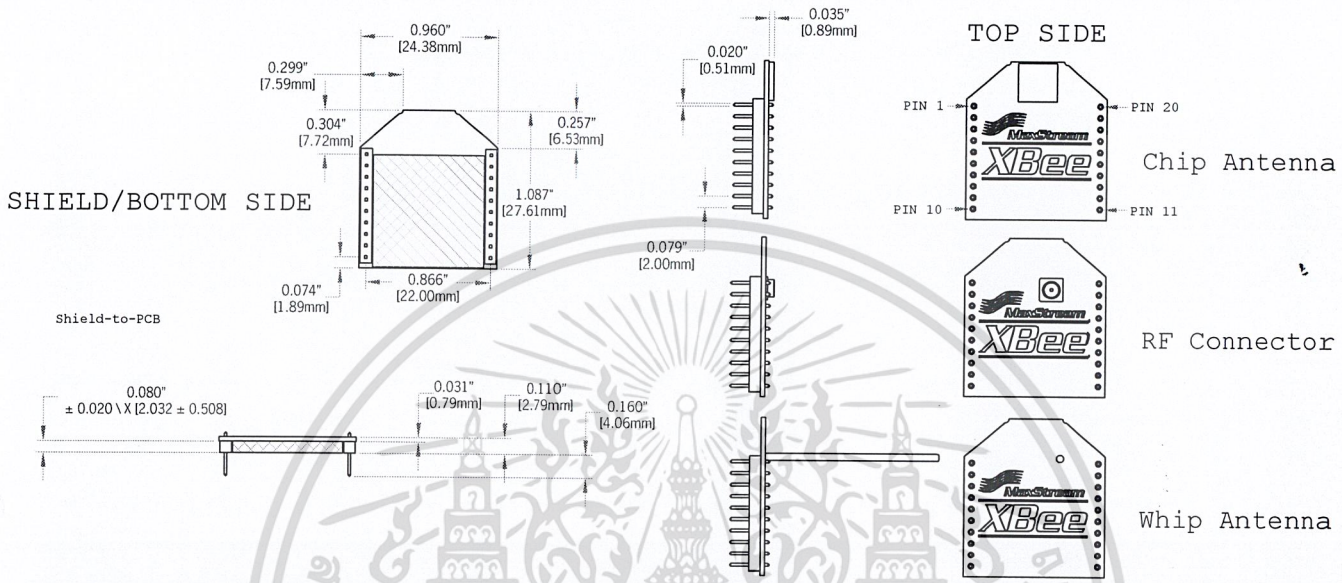
www.digi.com

91001412
B1/308

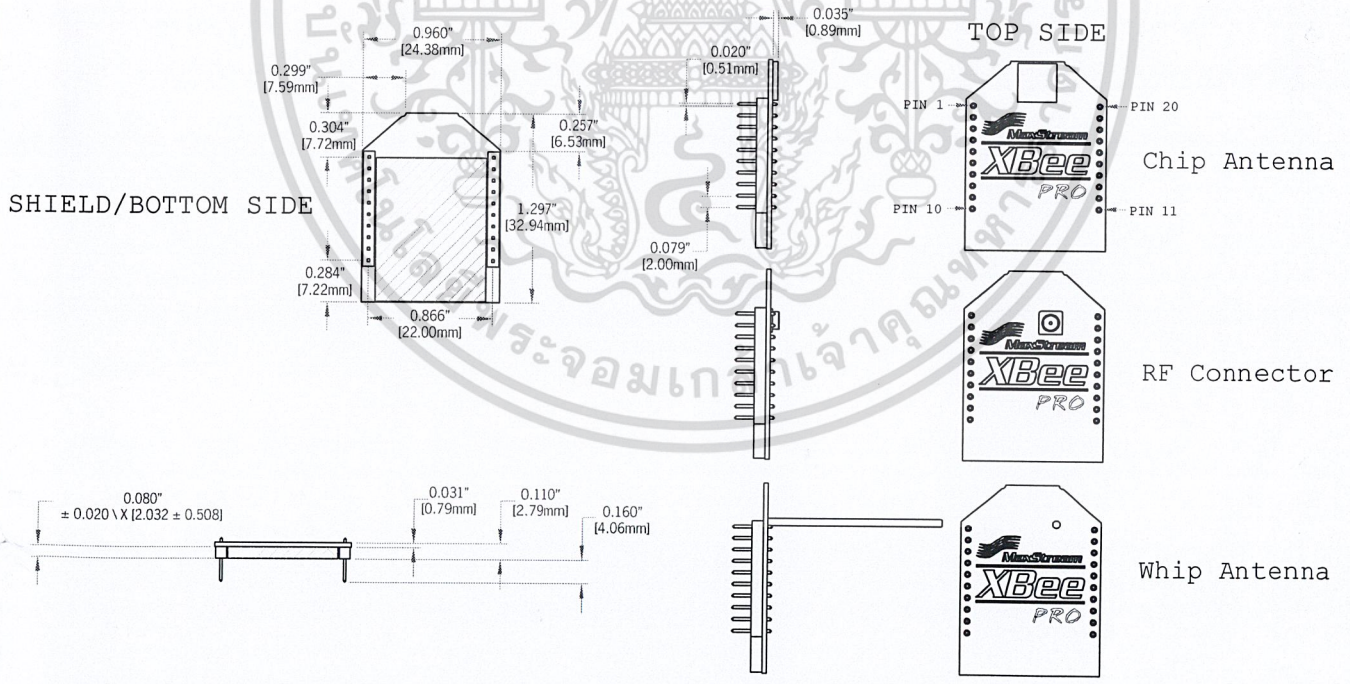
XBee Mechanical Drawings

XBee and XBee-PRO OEM RF Modules are pin-for-pin compatible with each other.

XBee OEM RF Module



XBee-PRO OEM RF Module



M100427 [2006.06.28]

MaxStream®
www.maxstream.net

Phone. (866) 765-9885 Toll-free in U.S. & Canada
(801) 765-9885 Worldwide
Live Chat. www.maxstream.net
E-mail. rf-xperts@maxstream.net