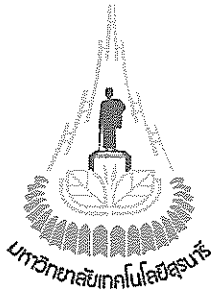


เครื่องวัดระดับน้ำมันโดยอาศัยหลักการสะท้อนคลื่นอัลตร้าโซนิค

โดย

นายธนวัฒน์	ควรรติกุล	รหัสนักศึกษา	B4603436
นายบูรนิติ	มาศอมรพันธุ์	รหัสนักศึกษา	B4604747
นางสาวรัชดาพร	มานุวงศ์	รหัสนักศึกษา	B4607373

รายงานนี้เป็นส่วนหนึ่งของการศึกษาวิชา 427499 โครงงานศึกษาวิศวกรรมโทรคมนาคม

ประจำภาคการศึกษาที่ 3 ปีการศึกษา 2550

หลักสูตรวิศวกรรมศาสตร์บัณฑิต สาขาวิชาวิศวกรรมโทรคมนาคม

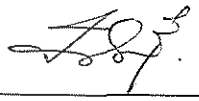
หลักสูตรปรับปรุงพ.ศ. 2546

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

เครื่องวัดระดับน้ำมันโดยอาศัยหลักการสะท้อนคลื่นอัลตร้าโซนิก

คณะกรรมการสอบโครงงาน

(ผู้ช่วยศาสตราจารย์ ดร. รังสรรค์ วงศ์สรรค์)
อาจารย์ที่ปรึกษาโครงงาน



(อาจารย์ ดร. มนต์ทิพย์ภา อุฑารสกุล)
กรรมการ

(อาจารย์ปิยาภรณ์ กระฉอดนอก)
กรรมการ

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับรายงานโครงงานฉบับนี้ เป็นส่วนหนึ่งของการศึกษาระดับปริญญาตรี สาขาวิชาวิศวกรรมโทรคมนาคม วิชา 427 494 โครงงานศึกษาวิศวกรรมโทรคมนาคม และวิชา 427 499 โครงงานวิศวกรรมโทรคมนาคม ประจำปีการศึกษา 2550

โครงงาน	เครื่องวัดระดับน้ำมันโดยอาศัยคลื่นหลักการสะท้อนอัลตร้าโซนิค			
ผู้ดำเนินงาน	1. นายธนวัฒน์	ควรรติกุล	รหัสนักศึกษา	B4603436
	2. นายบูรนิติ	มาศอมรพันธุ์	รหัสนักศึกษา	B4604747
	3. นางสาวรัชดาพร	มานุวงศ์	รหัสนักศึกษา	B4607373

อาจารย์ที่ปรึกษา ผศ.ดร.รังสรรค์ วงศ์สรรค์

สาขาวิชา วิศวกรรมโทรคมนาคม

ภาคการศึกษาที่ 1/2549

บทคัดย่อ

โครงงานฉบับนี้เป็นการวิเคราะห์และออกแบบวงจรคำนวณระดับน้ำมันในถังน้ำมัน รวมถึงการประเมินและทดสอบถึงประสิทธิภาพของเครื่องวัดระดับน้ำมันโดยอาศัยคลื่นอัลตร้าโซนิค เป็นตัวบ่งบอกถึงระดับน้ำมันในถังน้ำมัน โดยตัวที่บ่งบอกคือ แผงวงจรอิเล็คทรอนิกส์ที่ได้ผ่านการออกแบบและเขียนชุดคำสั่ง ซึ่งผลที่ได้คือความถี่ และนำความถี่ที่ได้นั้นมาประมวลและวิเคราะห์ผล เพื่อที่จะได้ทราบถึงระดับปริมาณน้ำมันในถัง ซึ่งเราสามารถนำผลที่ได้มาใช้ประโยชน์ในการวิเคราะห์ ถึงระดับน้ำมันในถังที่คงเหลืออยู่ในถังน้ำมัน เพื่อที่จะสามารถนำไปใช้ประโยชน์ได้จริง

กิตติกรรมประกาศ

ในการจัดทำโครงงานนี้จะมิอาจสำเร็จลุลวงไปได้ด้วยดี ถ้าหากมิได้รับความกรุณาจากอาจารย์ที่ปรึกษา ผศ.ดร.รังสรรค์ วงศ์สรรค์ ผู้ที่ให้แนวคิดแรกเริ่มของวงจรเชื่อมต่อคอมพิวเตอร์ช่วยสอนปฏิบัติการ ที่ให้ความช่วยเหลือในเรื่องของแนวคิด การติดตามเอาใจใส่ และชี้แนะข้อบกพร่องที่ข้าพเจ้าได้มองข้ามไปในบางส่วน

นอกจากนี้ยังต้องขอขอบพระคุณผู้ที่เกี่ยวข้อง ดังนี้

คณาจารย์ประจำสาขาวิชาวิศวกรรมโทรคมนาคมทุกท่าน ที่ได้สั่งสอนให้ความรู้

คุณประพล จาระตะคุ ผู้ที่ให้ความช่วยเหลือในเรื่องของการเบิกจ่ายงบประมาณ ตลอดจนอุปกรณ์ในการทดลองต่างๆ

คุณมณีรัตน์ ทุมพงษ์ เลขานุการประจำสาขาวิชาวิศวกรรมโทรคมนาคม ที่ให้ความช่วยเหลือในเรื่องของงานเอกสาร

นายไพโรจน์ บุญไทย ที่ช่วยให้คำแนะนำเกี่ยวกับอุปกรณ์ไมโครคอลโทรเลอร์

พี่ๆและเพื่อนๆ สาขาวิชาวิศวกรรมโทรคมนาคม ที่ให้ความช่วยเหลือในเรื่องของการเขียนโปรแกรมและการทดสอบโปรแกรมที่ใช้ในการวัดปริมาตรที่เหลือ

สุดท้ายนี้ คุณงานความดีที่เกิดจากโครงงานฉบับนี้ ขอมอบแก่บิดามารดา ผู้ที่คอยห่วงใย ให้กำลังใจ ให้โอกาส และให้การสนับสนุนทางการศึกษามาโดยตลอด

นายธนวัฒน์ ควรรติกุล
นายบูรนิติ มาศอมรพันธุ์
นางสาวรัชดาพร มานุวงศ์

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

หน้า

สารบัญตาราง

สารบัญกราฟ

บทที่ 1
บทนำ

1.1 ความเป็นมา

เนื่องจากในยุคปัจจุบันมีการติดต่อค้าขายกันมากขึ้น ยานพาหนะจึงมีความจำเป็นสำหรับการขนส่งสินค้า ซึ่งยานพาหนะจะขับเคลื่อนได้นั้น จำเป็นต้องใช้น้ำมัน และในยานพาหนะแต่ละชนิดต่างใช้น้ำมันในการขับเคลื่อนไม่เท่ากัน ดังนั้นหากทราบระดับน้ำมันในถังน้ำมันที่แน่นอน ก็จะสามารถทำให้คาดการณ์ว่าน้ำมันที่เหลืออยู่ในถังน้ำมันเพียงพอที่จะทำให้ถึงจุดหมายปลายทางได้หรือไม่ แต่สำหรับยานพาหนะรุ่นเก่าจะมีตัวบ่งชี้ปริมาณน้ำมันที่ไม่ละเอียดเท่าที่ควร

โครงงานนี้จึงนำความรู้เรื่องเซนเซอร์แบบอัลตราโซนิค (Ultrasonic Sensor) ซึ่งอาศัยหลักการการสะท้อนของเสียงที่ปล่อยออกไปยังเป้าหมายที่ต้องการ ทำให้ทราบถึงระยะห่างระหว่างต้นทางและปลายทาง เพื่อนำมาประยุกต์ใช้เป็นส่วนประกอบในการจัดทำเครื่องวัดปริมาณน้ำมันและนำผลที่ได้มาทำการวิเคราะห์และประมวลผล ซึ่งจะทำให้ทราบถึงระดับน้ำมันในถังบรรจุน้ำมันได้อย่างถูกต้องและมีความละเอียดยิ่งขึ้น เพื่อที่จะนำไปประยุกต์ใช้งานได้จริงอย่างมีประสิทธิภาพ

1.2 วัตถุประสงค์

1.2.1 เพื่อศึกษาวิธีการหลักการและวิธีการทำงานของเซนเซอร์แบบอัลตราโซนิค (Ultrasonic Sensor)

1.2.2 เพื่อศึกษาผลลัพธ์ที่ได้จากเซนเซอร์แบบอัลตราโซนิค (Ultrasonic Sensor) มาใช้ในการหาปริมาตรน้ำมันในถัง โดยใช้ไมโครคอนโทรเลอร์เป็นตัวประมวลผล และแสดงผลออกที่หน้าจอแสดงผล

1.3 ขอบเขตของการทำงาน

1.3.1 ศึกษาค้นคว้าข้อมูลเกี่ยวกับทฤษฎีคลื่นอัลตร้าโซนิค (Ultrasonic Wave)

1.3.2 ทำการทดสอบค่าความสูงที่วัดได้จากเซนเซอร์แบบอัลตราโซนิค (Ultrasonic Sensor) ทำการออกแบบโปรแกรมประมวลผลผลลัพธ์ที่ได้เซนเซอร์แบบอัลตราโซนิค (Ultrasonic Sensor) เพื่อคำนวณหาปริมาตรคงเหลือของน้ำมันที่มีในถังทดลอง

1.3.3 ปรับแต่งโปรแกรมเพื่อให้ทำงานได้ตามคุณสมบัติที่คาดหวังไว้

1.3.4 สามารถนำไปใช้งานได้จริงในการวัดระดับน้ำมัน

1.4 ขั้นตอนการดำเนินงาน

1.4.1 ศึกษาและทำการค้นคว้าเกี่ยวกับการทำงานของเซนเซอร์อัลตร้าโซนิค (Ultrasonic Sensor) ที่มี

1.4.2 นำผลลัพธ์ที่ได้จากเซนเซอร์แบบอัลตราโซนิค (Ultrasonic Sensor) มาทำการประมวลผลโดยใช้ไมโครคอลโทลเลอร์ (MCS 51) เพื่อคำนวณหาปริมาตรน้ำมันในถัง

1.4.3 ทำการปรับแต่งให้สามารถใช้งานได้ง่ายขึ้น

1.4.4 ทำการทดลองจริงกับน้ำมัน

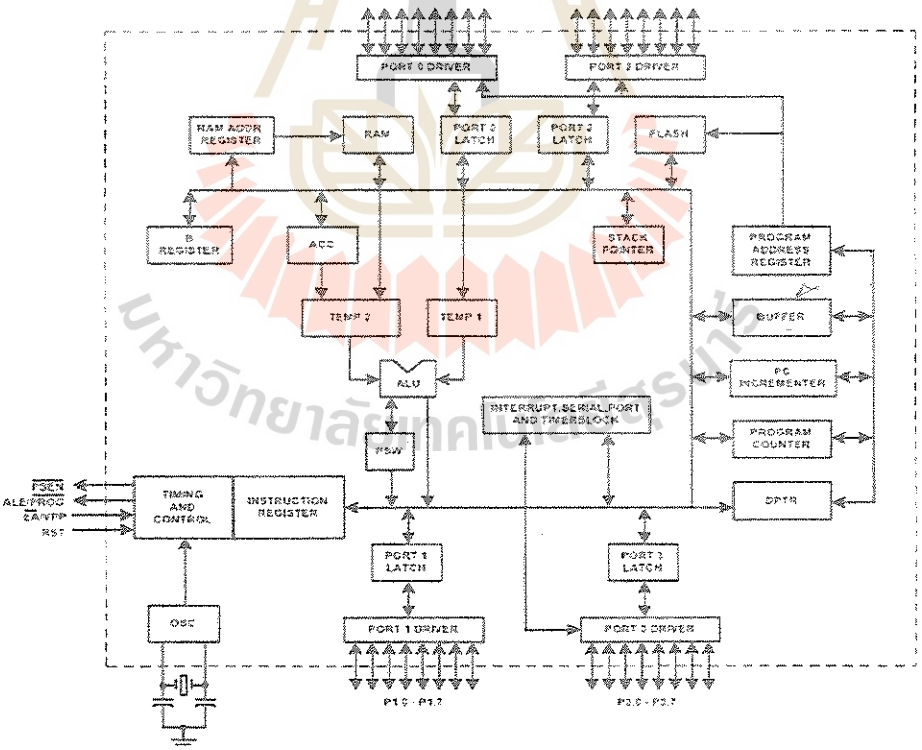
บทที่ 2
ทฤษฎีและหลักการ

2.1 ไมโครคอนโทรลเลอร์ MCS-51 [1]

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ได้จัดให้มีส่วนประกอบภายในเพื่ออำนวยความสะดวกแก่ผู้ใช้ เช่น ไทเมอร์ (Timer) เคาน์เตอร์ (Counter) พอร์ตอนุกรม (Serial port) และสำหรับไมโครคอนโทรลเลอร์เบอร์ใหญ่ ๆ ยังอาจมีส่วนอื่นเพิ่มเติมเข้ามาอีก เช่น เบอร์ 80C515, 80C535 จะมีวงจรแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล

2.1.1 โครงสร้างภายในของไมโครคอนโทรลเลอร์ตระกูล MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ใช้เทคโนโลยีการผลิตแบบ NMOS และ CMOS ซึ่งภายในได้รวมวงจรต่างไว้อย่างครบถ้วนพร้อมที่จะทำงานได้เมื่อจ่ายไฟเลี้ยงและสัญญาณนาฬิกา (Clock) ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ได้ถูกผลิตออกมามากมายหลายเบอร์โดยบริษัทต่าง ๆ ไม่ว่าจะเป็นเบอร์อะไรก็ตาม ถ้าเป็นไมโครคอนโทรลเลอร์ตระกูล MCS-51 แล้วจะมีโครงสร้างต่าง ๆ ที่คล้ายกัน จะต่างกันออกไปในส่วนของความสามารถพิเศษของแต่ละเบอร์ ยกตัวอย่างเช่นเบอร์ AT89C51 มีไทเมอร์ 2 ตัว ในขณะที่เบอร์ AT89C52 มีไทเมอร์ 3 ตัว เป็นต้น โดยที่โครงสร้างต่างๆ ภายในของไมโครคอนโทรลเลอร์ตระกูล MCS-51 ได้แสดงไว้ดังรูปที่ 2.1



รูปที่ 2.1 บล็อกไดอะแกรมของไมโครคอนโทรลเลอร์ตระกูล MCS-51

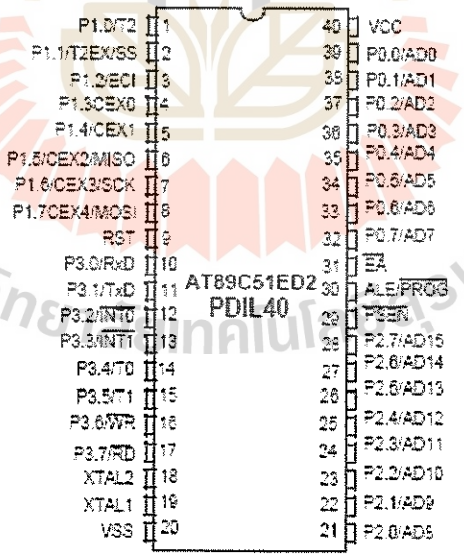
2.1.2 คุณสมบัติทั่วไปของไมโครคอนโทรลเลอร์ตระกูล MCS-51

เนื่องจากคุณสมบัติของไมโครคอนโทรลเลอร์แต่ละเบอร์นั้นมีความสามารถที่แตกต่างกันออกไปในรายละเอียดปลีกย่อย ดังนั้นจะขออ้างถึงเบอร์ AT89C51 ของบริษัท Atmel ซึ่งเป็นไมโครคอนโทรลเลอร์ขนาด 8 บิต ซึ่งมีคุณสมบัติดังนี้

- มีหน่วยความจำโปรแกรมแบบแฟลช (Flesh Memory) ขนาด 8 กิโลไบต์
- มีหน่วยความจำแบบ EEPROM ขนาด 2 กิโลไบต์ (kbyte)
- แหล่งจ่ายไฟกระแสตรงขนาด 5 โวลต์ (ทำงานในช่วง 4-6 โวลต์)
- ทำงานได้ด้วยสัญญาณนาฬิกาตั้งแต่ 0-24 เมกกะเฮิร์ต (MHz)
- มีหน่วยความจำข้อมูล (RAM) ขนาด 256 ไบต์ (byte)
- มีพอร์ต 32 พอร์ตอิสระสามารถเข้าถึงในระดับบิตได้
- มีไทเมอร์/เคาเตอร์ขนาด 16 บิต ทั้งหมด 3 ตัว
- รองรับการอินเตอร์รัปต์ได้ 8 แหล่ง
- สามารถสื่อสารข้อมูลแบบอนุกรมได้ด้วย UART Channel

2.1.3 การจัดขาของไมโครลคอนโทรลเลอร์ตระกูล MCS-51

ไมโครคอนโทรลเลอร์ในตระกูล MCS- 51 จะมีโครงสร้างของการจัดเรียงขาที่คล้าย ๆ กัน ได้แสดงไว้ดังรูปที่ 2.2 และมีส่วนประกอบของขาดังต่อไปนี้



รูปที่ 2.2 การจัดขาของ MCS-51

VCC ต่อไฟเลี้ยง

GND ต่อกราวด์

Port0 (P0.0-P0.7) เป็นพอร์ตแบบสองทิศทางขนาด 8 บิต สามารถทำงานได้ทั้งสองหน้าที่ คือ เป็นพอร์ตอินพุตและเอาท์พุตทั่วไป และใช้เป็นพอร์ตสำหรับติดต่อกับหน่วยความจำภายนอก คือรับและส่งข้อมูลพร้อมทั้งกำหนดแอดเดรส (Address) ไบต์ต่ำ

Port1 (P1.0-P1.7) เป็นพอร์ตแบบสองทิศทางขนาด 8 บิต มีการต่อความต้านทานพูลอัพ (pull-up resister) ไว้ภายใน ทำหน้าที่เป็นพอร์ตอินพุตและเอาท์พุตทั่วไป นอกจากนี้ยังใช้งานเป็นขาอินพุตเอาท์พุตของไทเมอร์ 2

Port2 (P2.0-P2.7) เป็นพอร์ตแบบสองทิศทางขนาด 8 บิต มีการต่อความต้านทานพูลอัพ (pull-up resister) ไว้ภายใน สามารถทำงานได้สองหน้าที่คือเป็นพอร์ตอินและพุตเอาท์พุตทั่วไป และใช้เป็นพอร์ตสำหรับติดต่อกับหน่วยความจำภายนอกคือกำหนดแอดเดรสไบต์สูง

Port3 เป็นพอร์ตแบบสองทิศทางขนาด 8 บิต มีการต่อความต้านทานพูลอัพ (pull-up resister) ไว้ภายใน ทำหน้าที่เป็นพอร์ตอินพุตและเอาท์พุตทั่วไป นอกจากนี้ยังใช้เป็นขาสัญญาณควบคุมการติดต่อกับหน่วยความจำการอินเตอร์รัปต์ และอื่น ๆ

RST เป็นขาอินพุตที่ใช้รับสัญญาณสำหรับรีเซตซีพียู โดยซีพียู (CPU) จะถูกรีเซต (Reset) เมื่อขานี้เป็นลอจิก “1” นาน 2 แมชินไซเคิล หรือ 24 ไซเคิลของสัญญาณนาฬิกา

ALE/PROG ทำหน้าที่เป็นขาเอาท์พุตเมื่อซีพียูต้องการติดต่อกับหน่วยความจำภายนอก คือจะทำการส่งสัญญาณพัลส์ (Pulse) ออกมาที่ขานี้เพื่อทำการแลกแอดเดรสไบต์ต่ำของหน่วยความจำภายนอก และขานี้จะเป็นอินพุตเมื่อยู่ในระหว่างโปรแกรมแฟรช

PSEN เป็นขาเอาท์พุต ใช้ในการติดต่อกับหน่วยความจำภายนอก คือเมื่อซีพียูทำการประมวลผลกับหน่วยความจำโปรแกรมภายนอกขานี้จะแอคทีฟสองครั้งในแต่ละแมชินไซเคิล

EA/VPP เป็นขาอินพุตและต้องการลอจิก “0” เพื่อยอมให้ซีพียูสามารถเข้าถึงหน่วยความจำโปรแกรมภายนอกได้ ซึ่งอยู่ที่ตำแหน่ง 0000H ถึง FFFFH นอกจากนี้แล้วขานี้ยังใช้รับไฟ 12 โวลต์เพื่อใช้ในระหว่างที่ทำการโปรแกรมแฟลช

XTAL1 เป็นขาอินพุตของวงจรออสซิลเลเตอร์แอมปลิไฟเออร์ และยังเป็นขาอินพุตของวงจรกำเนิดสัญญาณนาฬิกาภายใน

XTAL2 เป็นขาเอาท์พุตของวงจรออสซิลเลเตอร์แอมปลิไฟเออร์

2.2 คลื่นอัลตร้าโซนิค (Ultrasonic) [2]

ระบบอัลตร้าโซนิค (Ultrasonic) หมายถึง คลื่นเสียงที่มีความถี่สูงเกินกว่าที่หูมนุษย์จะได้ยิน โดยทั่วไปแล้วหูของมนุษย์โดยเฉลี่ยจะได้ยินเสียงสูงถึงเพียงแค่ประมาณ 15 KHz เท่านั้น ดังนั้นโดยปกติแล้วคำว่าอัลตร้าโซนิคหมายถึงคลื่นเสียงที่มีความถี่สูงกว่า 20 KHz ขึ้นไป สาเหตุที่มีการนำเอาคลื่นย่านอัลตร้าโซนิคมาใช้ก็เพราะว่าเป็นคลื่นที่สามารถกำหนดการส่งคลื่นไปยังเป้าหมายที่ต้องการได้โดยเจาะจง เรื่องนี้เป็นคุณสมบัติของคลื่นอย่างหนึ่ง ยิ่งคลื่นมีความถี่สูงขึ้นความยาวคลื่นก็จะยิ่งสั้นลง ถ้าความยาวคลื่นยาวกว่าช่องเปิด (ที่ให้เสียงนั้นออกมา) ของตัวกำเนิดเสียงความถี่นั้นเช่น คลื่นความถี่ 300 Hz ในอากาศจะมีความยาวถึงประมาณ 1000 กิโลเมตร ซึ่งจะยาวกว่าช่องที่ให้คลื่นเสียงออกมาจากตัวกำเนิดเสียงโดยทั่วไปมากมายคลื่นจะหักเหที่ขอบด้านนอกของตัวกำเนิดเสียงทำให้เกิดการกระจายทิศทางคลื่นแต่ถ้าความถี่สูงขึ้นมาอยู่ในย่านอุลตร้าโซนิค อย่างเช่น 40 KHz จะมีความยาวคลื่นในอากาศเพียงประมาณ 8 มม. เท่านั้นซึ่งเล็กกว่ารูเปิดของตัวที่ให้กำเนิดเสียงความถี่นี้มากคลื่นเสียงจะไม่มีการเลี้ยวเบนที่ขอบจึงพุ่งออกมาเป็นลำแคบ ๆ หรือที่เราเรียกว่ามีทิศทาง

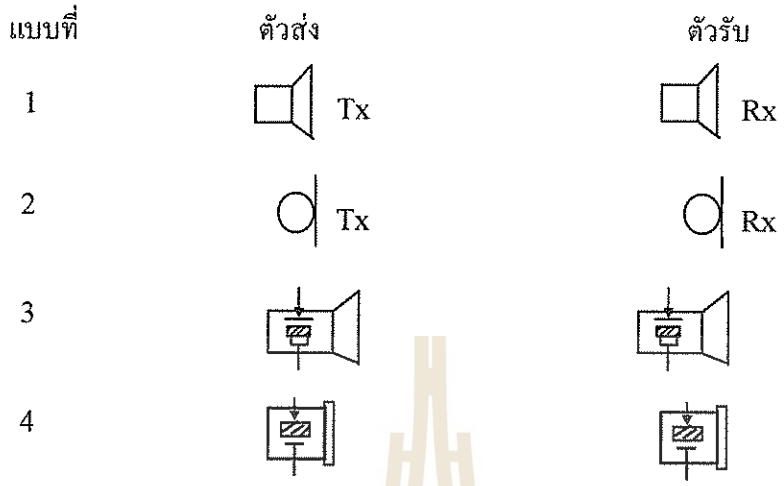
การมีทิศทางของคลื่นเสียงย่านอัลตร้าโซนิคทำให้เรานำไปใช้งานได้หลายอย่าง เช่น นำไปใช้ในเครื่องควบคุมระยะไกล (Ultrasonic remote control) โดยให้น้ำสั่นที่ความถี่สูง เครื่องวัดความหนาของวัตถุโดยสังเกตระยะเวลาที่คลื่นสะท้อนกลับมา เครื่องวัดความลึกและทำแผนที่ใต้ท้องทะเล เป็นต้น โดยความถี่ที่ใช้ขึ้นอยู่กับการใช้งาน เช่น คลื่นเสียงต้องเดินทางผ่านอากาศแล้ว ความถี่ที่ใช้ก็มักจะจำกัดอยู่เพียงไม่เกิน 50 KHz เพราะที่ความถี่สูงขึ้นกว่านี้อากาศจะดูดกลืนคลื่นเสียงเพิ่มขึ้นมาก ทำให้ระดับความแรงของคลื่นเสียงที่ระยะห่างออกไปลดลงอย่างรวดเร็ว ส่วนการใช้งานด้านการแพทย์ซึ่งต้องการรัศมีทำการสั้น ๆ ก็อาจใช้ความถี่ในช่วง 1 MHz ถึง 10 MHz ขณะที่ความถี่เป็น GHz (10^9 Hz) ก็มีใช้กันในหลายๆการใช้งานที่ตัวกลางที่คลื่นเสียงเดินทางผ่านไม่ใช่อากาศอุปกรณ์ที่สามารถแปลงพลังงานในรูปอื่นให้มาเป็นพลังงานทางกล โดยการสั่นไปมาซึ่งทำให้เกิดคลื่นเสียงย่านอุลตร้าโซนิคกระจายไปในอากาศได้หรือแปลงพลังงานทางกลให้มาเป็นพลังงานในรูปอื่นได้นั้น มีชื่อเรียกว่า อัลตร้าโซนิคทรานสดิวเซอร์ (Ultrasonic Transducer) ในปัจจุบันอุลตร้าโซนิคทรานสดิวเซอร์มีหลายแบบขึ้นอยู่กับหลักการที่ใช้ แบบที่นิยมใช้กันมากได้แก่

- แบบเปียโซอิเลคทริก (Piezo-electric Transducer) ซึ่งแปลงไปมาระหว่างพลังงานไฟฟ้าและ พลังงานทางกล โดยมีความถี่เรโซแนนซ์คงที่อยู่ค่าหนึ่ง

- แบบแมกนีโตสตริคทีฟ (Magnetostrictive Transducer) ซึ่งแปลงไปมาระหว่างพลังงานไฟฟ้าในขดลวดกับตำแหน่งความยาวของแกนเหล็กที่สวมขดลวดนั้นอยู่

- แบบอิเลคโตรสตริคทีฟ (Electrostrictive Transducer) ซึ่งแปลงไปมาระหว่างพลังงานไฟฟ้ากับพลังงานทางกล

โดยทั่วไป ลักษณะปกติของตัวส่ง Ultrasonic จะเปรียบเสมือนลำโพงส่งสัญญาณเสียง และตัวรับจะมีลักษณะเหมือน Microphone ดังนั้นสัญลักษณ์ที่ใช้จึงมีลักษณะคล้าย ๆ กัน ซึ่งมีหลายแบบ คือ



รูปที่2.3 สัญลักษณ์ของ Ultrasonic

ตัวส่ง คือ อัลตร้าโซนิคที่ถูกออกแบบเจาะจงมาให้แปลงสัญญาณไฟฟ้าที่ให้แก่ตัวมัน ให้ออกมาเป็นคลื่นเสียงย่านอัลตร้าโซนิค หน้าที่ของตัวส่งจึงคล้าย ๆ กับเป็นลำโพง

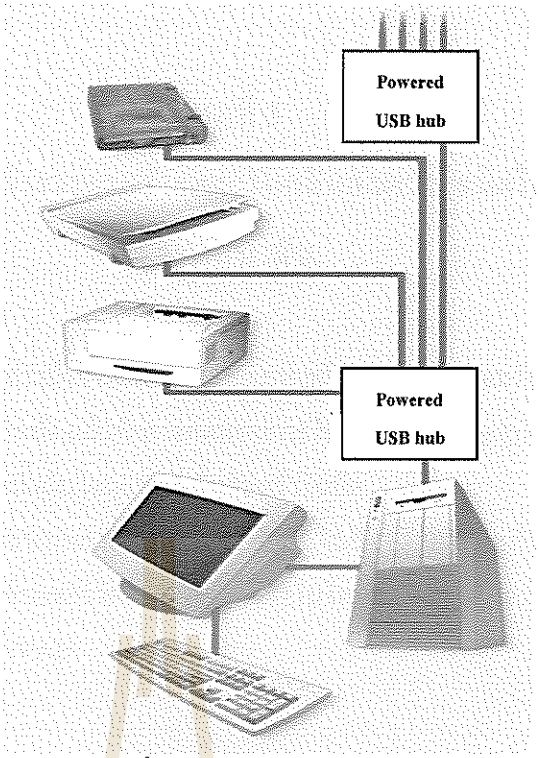
ตัวรับ คือ อัลตร้าโซนิคที่ถูกออกแบบเจาะจงมาให้แปลงคลื่นเสียงย่านอัลตร้าโซนิคที่มาตกกระทบตัวมันให้ออกมาเป็นสัญญาณไฟฟ้า หน้าที่ของตัวรับจึงคล้าย ๆ กับเป็นไมโครโฟน ด้วยเหตุนี้เวลาเขียนสัญลักษณ์ของอัลตร้าโซนิคจึงนิยมเขียนตามหน้าที่ของมันคือถ้าเป็นตัวส่งก็เขียนสัญลัษณ์เป็นลำโพง ถ้าเป็นตัวรับก็เขียนสัญลักษณ์เป็นไมโครโฟน

2.3 USB (Universal Serial Bus USB) [3]

การสื่อสารระหว่างคอมพิวเตอร์และอุปกรณ์ต่อพ่วงต่างๆจำเป็นที่จะต้องมี พอร์ตI/O เพื่อใช้ในการสื่อสาร (I/O = input , output) ในสมัยก่อน พอร์ตที่รู้จักกันดี ก็คือ พอร์ตอนุกรม (Serial Port)หรือเรียกกันว่า พอร์ตคอม (Com Port) และ พอร์ตขนาน (Parallel Port) แต่ข้อจำกัดของพอร์ต เหล่านี้นั้นมีมากพอสมควรไม่ว่าจะเป็นเรื่องของความเร็ว จำนวนอุปกรณ์จะนำมาต่อได้ และคุณสมบัติทางด้าน Plug & Play USB ก็เป็น พอร์ต I/O ที่ทำงานในลักษณะซีเรียลบัส (Serial Bus) เช่นกัน สามารถต่อได้กับอุปกรณ์ถึง 127 อุปกรณ์ ต่อ 1 พอร์ต และ สนับสนุนการทำงานแบบ Plug & Play อย่างเต็มรูปแบบ โดยจะทำการค้นหา และตั้งค่าของอุปกรณ์ที่นำมาต่อให้อย่างอัตโนมัติ ไม่จำเป็นต้องกำหนดค่า (assign) IRQ เพียงแต่ใส่แผ่นไดร์เวอร์ (Driver) ของอุปกรณ์ที่นำมาติดตั้ง และส่วนมากไม่จำเป็นต้องปิดแล้วเปิดเครื่องใหม่ (Restart) หลังจากติดตั้งไดร์เวอร์ด้วย ถ้าหากอุปกรณ์นั้นใช้กำลังไฟฟ้าไม่มากพอร์ต USB ก็จ่ายไฟเลี้ยงให้กับอุปกรณ์ต่างๆ ได้ด้วย

2.3.1 ระบบการจัดส่งข้อมูล USB

สำหรับ USB เราเรียกระบบบัสเช่นนี้ว่า การติดต่อแบบ "Host/Slave" หมายถึงตัวเครื่องคอมพิวเตอร์จะเป็นผู้จัดการ การจัดส่งข้อมูลทั้งหมด และอุปกรณ์ที่นำมาต่อพ่วงเพียงแต่พร้อมที่จะรับส่งข้อมูลเท่านั้น โดยตัว USB Host Controller หรือตัวควบคุมของ USB นั้น จะรวมอยู่ในตัวชิพเซต (Chipset) บนเมนบอร์ดแต่ถ้าหากเป็นชิพเซตรุ่นเก่าๆก็จะยังไม่มีตัวควบคุม USB ในการจัดส่งข้อมูลนั้น ภายในสาย USB จะมีสายภายในทั้งหมด 4 เส้น ในจำนวนนี้ 2 เส้นใช้เพื่อเป็นสายในการส่งข้อมูล อีก 2 เส้นใช้สำหรับจ่ายไฟเลี้ยง แรงดันบวก 5 โวลต์ และ กราวด์ ข้อมูลที่ถูกส่งผ่าน USB นั้นจะต่างไปจากพอร์ตขนาน และพอร์ตอนุกรมซึ่งทั้ง 2 ชนิดนี้นั้น จะส่งข้อมูลเป็นบิตเดี่ยวๆ แต่ USB จะส่งเป็นชุดข้อมูล อย่างเช่น ถ้าหากต้องการ เก็บข้อมูลไปยัง USB Zip drive PC จะทำการแบ่งข้อมูลออก มา 64 ไบต์ แล้วใส่ข้อมูลของแอดเดรสเข้าไปแล้วส่งไปยังพอร์ต USB จากนั้นก็จะทำเช่นนี้จนกระทั่งส่งข้อมูลครบสิ่งที่ทำให้ USB ส่งข้อมูลได้เร็วถึง 12 เมกกะบิตต่อวินาที (Mbits/s) นั้นอยู่ที่แม่นยำของการส่ง เพราะ สายที่นำมาใช้สำหรับ USB นั้น มีระดับสัญญาณรบกวน และ ความเพี้ยนของรูปสัญญาณนั้นน้อยกว่า ทั้งพอร์ตขนาน และพอร์ตอนุกรม และการส่งสัญญาณแบบ isochronous data delivery หมายถึง อุปกรณ์แต่ละชิ้นบนระบบบัสนั้น จะถูกจำกัดขนาดของแบนด์วิดธ์ (bandwidth) ให้แน่นอน อย่างไรก็ตามแบนด์วิดธ์โดยรวมแล้วก็ไม่เกิน 12 เมกกะบิตต่อวินาที นี่เป็นเหตุผลที่ทำให้ USB สามารถต่ออุปกรณ์ได้มากที่สุด 127 อุปกรณ์



รูปที่ 2.4 การเชื่อมต่อพอร์ต USB

จากรูปที่ 2.3 เป็นการต่ออุปกรณ์ต่อพ่วงออกจากเครื่องคอมพิวเตอร์ไปยังอุปกรณ์ต่าง ๆ เช่น จากพอร์ต USB แรกไปยัง Powered USB Hub แล้วต่อไปยังอุปกรณ์อื่น ๆ รวมถึงต่อไปยัง Powered USB Hub เพื่อต่ออุปกรณ์เพิ่มเข้าอีกได้เรื่อย ๆ ในลักษณะของการต่อแบบอนุกรม ส่วนพอร์ต USB ชุดที่ 2 หลังเครื่องนั้นสามารถต่อไปยังจอมอนิเตอร์ (Monitor) ได้โดยอิสระต่อกัน จะสังเกตได้ว่าถ้าหากการต่อมี USB Hub เข้ามาต่อพ่วงด้วยที่ตัว Hub จะต้องมีตัวจ่ายไฟด้วยเพราะลำพังไฟเลี้ยงจากเครื่องคอมพิวเตอร์เองคงไม่พอส่งไปเลี้ยงอุปกรณ์ต่าง ๆ

ดังนั้นเราอาจมองได้ว่าสิ่งที่จะต่อจากพอร์ต USB นั้นมี 2 ชนิด ก็คือ ต่อเข้าได้กับ Hubs หรือต่อกับ อุปกรณ์ Hubs นั้นมีหน้าที่ ทำให้ สามารถต่ออุปกรณ์ได้มากขึ้นบนระบบบัส บางครั้ง Hubs จะเข้าไป รวมกับอุปกรณ์ อย่างเช่น มอนิเตอร์ ที่มี USB Hubs ในตัว โดยจะ สามารถปรับค่าต่างๆของมอนิเตอร์ผ่านระบบปฏิบัติการ และ นอกจากนั้นมีหน้าที่เป็น Hubs ด้วย

2.4 สรุป

เครื่องวัดปริมาตรน้ำมันโดยอาศัยคลื่นอัลตร้าโซนิค มีองค์ประกอบที่สำคัญคือ เซนเซอร์โดยอาศัยคลื่นอัลตร้าโซนิค (Ultrasonic Sensor) ไมโครลคอนโทรลเลอร์ เบอร์ MCS-51 และระบบการจัดส่งข้อมูลผ่าน USB

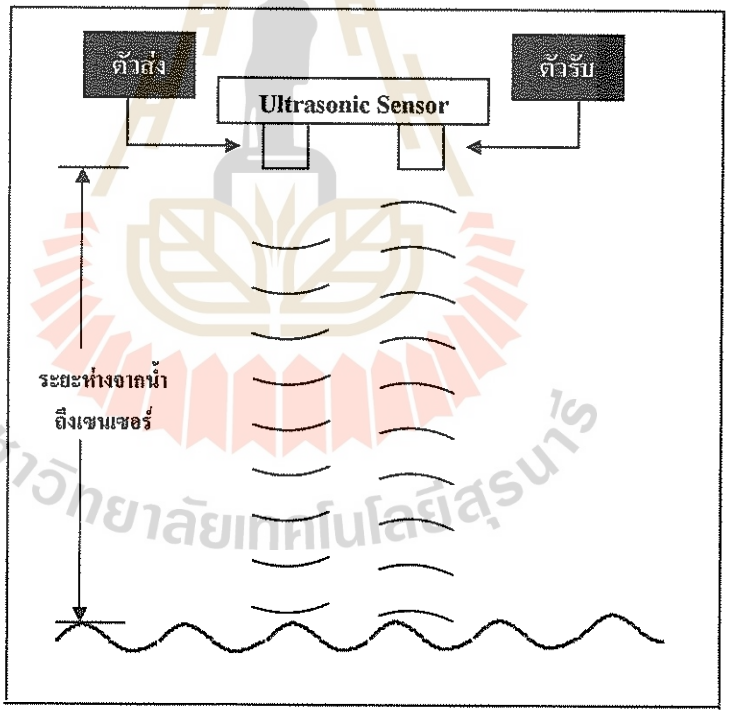
บทที่ 3
การออกแบบและสร้างวงจรสร้างและรับคลื่นอัลตร้าโซนิคในการตรวจวัดระดับน้ำมัน

3.1 กล่าวนำ

เนื้อหาในบทนี้จะกล่าวถึงการออกแบบและการสร้างวงจรสร้างและรับคลื่นอัลตร้าโซนิคในการตรวจวัดระดับน้ำมันในถัง แล้วทำการทดสอบการใช้งานได้จริงของอุปกรณ์แต่ละตัวที่นำมาประกอบเป็นเครื่องวัดระดับน้ำมันโดยอาศัยคลื่นอัลตร้าโซนิค

อุปกรณ์โดยรวมระบบของเครื่องวัดระน้ำมันในถังโดยอาศัยคลื่นอัลตร้าโซนิค มีอยู่ทั้งหมด 5 ส่วนด้วยกัน คือ เซ็นเซอร์วัดระยะทาง, วงจรไมโครคอนโทรลเลอร์, วงจรเชื่อมต่อพอร์ต USB, จอแสดงผลLCD, KEYPAD

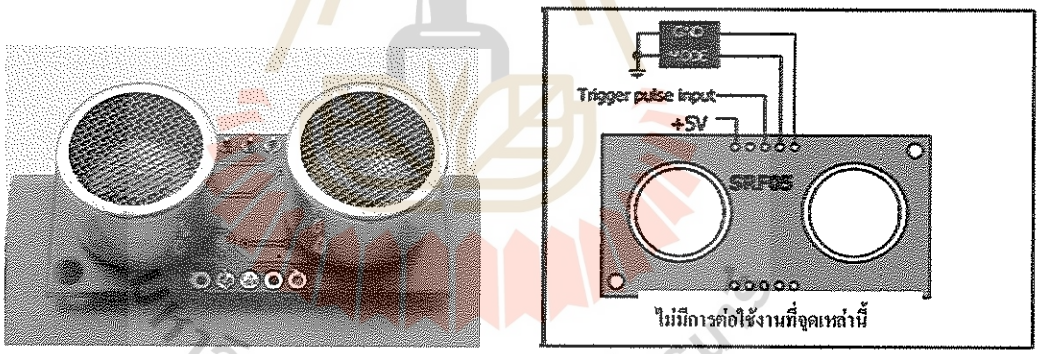
3.2 เซ็นเซอร์วัดระยะทาง Ultrasonic Distance Detector Module รุ่นSRF 05



รูปที่ 3.1 แสดงหลักการตรวจจับวัตถุโดยใช้สัญญาณความถี่เหนือเสียงหรืออัลตร้าโซนิก

เครื่องตรวจจับและวัดระยะทางด้วยอัลตร้าโซนิก (Ultrasonic Distance Detector Module) เป็นแผงวงจรวัดตรวจจับและวัดระยะทางด้วยอัลตร้าโซนิกที่มีแม่นยำและเที่ยงตรงสูง โดยสามารถวัดระยะทางได้ตั้งแต่ 1 เซนติเมตรไปจนถึง 4 เมตร โดยเครื่องตรวจจับและวัดระยะทางด้วยอัลตร้า-โซนิก (Ultrasonic Distance Detector Module) ถูกออกแบบมาให้ใช้งานกับไมโครคอนโทรลเลอร์ได้ง่ายโดยใช้ขาเชื่อมต่อเพียง 1 หรือ 2 ขา โดยจะทำการส่งสัญญาณคลื่นอัลตร้าโซนิกออกไปแล้ววัดระยะเวลาที่มีสัญญาณสะท้อนตอบกลับมา โดยเอาต์พุตที่ได้จะอยู่ในรูปของความกว้างพัลส์ซึ่งสัมพันธ์กับระยะทางของวัตถุที่ตรวจจับได้ โดยความถี่อัลตร้าโซนิกที่ถูกส่งออกไปในอากาศด้วยความเร็ว 1.125 ฟุตต่อมิลลิวินาที (ประมาณ 346 เมตรต่อวินาที) ดังนั้นเมื่อทราบความเร็วในการเคลื่อนที่ของคลื่น เวลาเริ่มส่งคลื่นและเวลาที่รับเสียงสะท้อนกลับมาจึงสามารถคำนวณหาค่าของระยะทางได้ ดังแสดงหลักการตรวจจับในรูปที่ 3.1

ระยะทางที่ได้นั้นจะต้องมีการคำนวณค่ากลับทางคณิตศาสตร์เมื่อใช้กับไมโครคอลโทรลเลอร์แล้วถือว่าเป็นเรื่องที่ยุ่งยากพอสมควร ดังนั้นเครื่องเครื่องตรวจจับและวัดระยะทางด้วยอัลตร้า-โซนิก (Ultrasonic Distance Detector Module) จึงประมวณผลค่าทางคณิตศาสตร์ต่างๆไว้เรียบร้อยแล้ว จากนั้นส่งผลลัพธ์ที่วัดได้ออกมาเป็นพัลส์ที่มีความกว้างสัมพันธ์กับระยะทางที่วัดได้ จะใช้สายสัญญาณเพียงเส้นเดียว จึงทำให้สะดวกมากในการนำมาเชื่อมต่อไมโครคอลโทรลเลอร์



รูปที่ 3.2 **(ก)** แสดงเซ็นเซอร์วัดระยะทาง Ultrasonic Distance Detector Module รุ่นSRF 05
(ข) แสดงขาสัญญาณของ SRF05 และการกำหนดโหมดทำงาน

3.2.1 การต่อใช้งานของ SRF05

มีจุดต่อสำหรับใช้งานอยู่ทั้งหมด 5 จุดตามรูปที่ 2

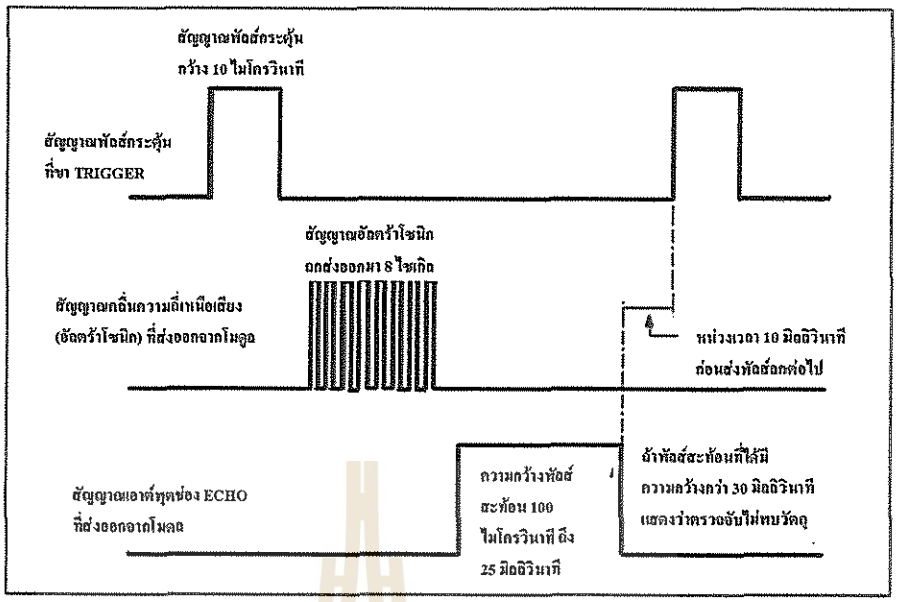
1. ขาไฟเลี้ยง (+5V) สำหรับต่อไฟเลี้ยงแรงดัน +5V

2. ขาEcho Pulse Output (ECHO) เป็นขาเอาต์พุตสำหรับส่งสัญญาณกัลส์ออกจาก SRF05 ซึ่งการใช้งานจะนำขานี้ไปต่อเข้ากับพอร์ตอินพุตของไมโครคอนโทรลเลอร์ เพื่อตรวจจับความกว้างของสัญญาณพัลส์ที่ส่งออกมาเพื่อแปลความหมายออกมาเป็นระยะทางอีกครั้งหนึ่ง

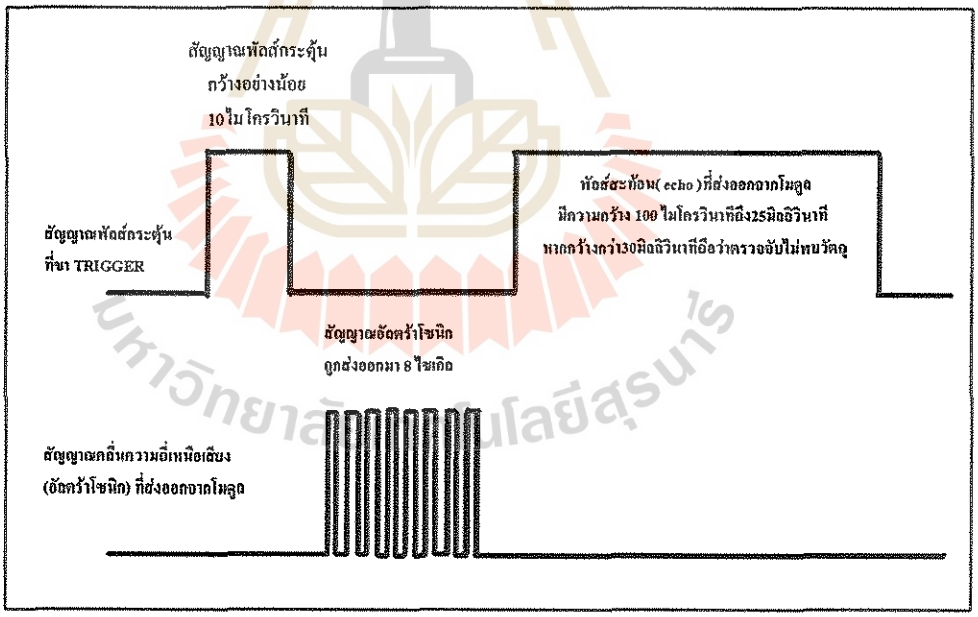
3. ขา Trigger Pulse Input (TRIGGER) เป็นขาอินพุตรับสัญญาณพัลส์ที่มีความกว้างอย่างน้อย 10 ไมโครวินาทีเพื่อกระตุ้นการสร้างคลื่นอัลตราโซนิกความถี่ 40 kHz ออกสู่อากาศจากตัวส่ง ดังนั้นเมื่อคลื่นความถี่ดังกล่าวนี้เคลื่อนที่กระทบสิ่งกีดขวางที่อยู่เบื้องหน้าก็จะเกิดการสะท้อนกลับเข้ามายังตัวรับและถูกแปลงออกมาเป็นความกว้างของสัญญาณพัลส์ที่จะส่งออกไปทางเขา Echo Pulse Output นอกจากนี้ในโหมด 1 สัญญาณ จะใช้จุดนี้เป็นจุดสื่อสารข้อมูลอนุกรมเพื่อรับส่งค่าการวัดกับไมโครคอนโทรลเลอร์

4. ขา MODE สำหรับเลือกรูปแบบการติดต่อกับ SRF05ปล่อยลอยไว้ (NC): เลือกใช้ติดต่อแบบ 2 สัญญาณ ผ่านจุดต่อถ ECHO และ TRIGGERต่อลงกราวด์: เลือกให้ติดต่อแบบ 1 สัญญาณ ผ่านจุดต่อ TRIGGER

5. ขา GND สำหรับต่อกราวด์



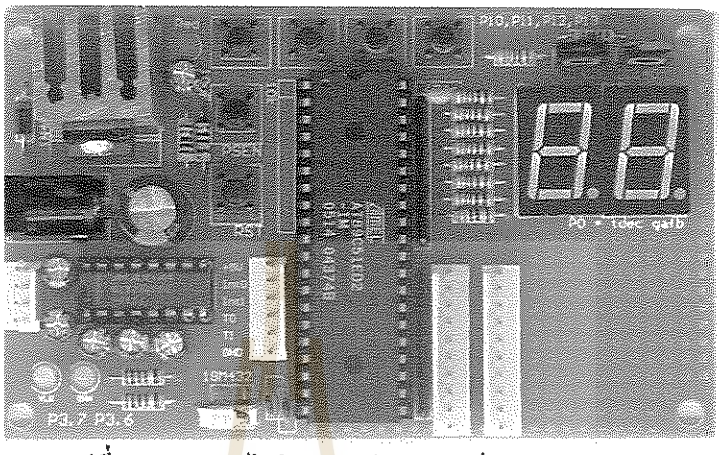
รูปที่ 3.3 ไดอะแกรมเวลาแสดงสัญญาณที่ส่งไปยังเครื่องตรวจจับและวัดระยะทางด้วยอัลตร้าโซนิก (Ultrasonic Distance Detector Module) และสัญญาณที่ตอบรับกลับมาจากเครื่องตรวจจับและวัดระยะทางด้วยอัลตร้าโซนิก (Ultrasonic Distance Detector Module) ในกรณีติดต่อแบบ 2 สัญญาณ



รูปที่ 3.4 ไดอะแกรมเวลาแสดงสัญญาณที่ส่งไปยังเครื่องตรวจจับและวัดระยะทางด้วยอัลตร้าโซนิก (Ultrasonic Distance Detector Module) และสัญญาณที่ตอบรับกลับมาจากเครื่องตรวจจับและวัดระยะทางด้วยอัลตร้าโซนิก (Ultrasonic Distance Detector Module)

3.3 วงจรไมโครคอนโทรเลอร์

ได้เลือกใช้วงจรไมโครคอนโทรเลอร์สำเร็จรูป ในตระกูล MCS-51 ที่ใช้ไอซีเบอร์ AT89C51ED2 ของ ATMEL ดังที่ได้แสดงวงจรรวมไว้ดังรูปที่ 3.5

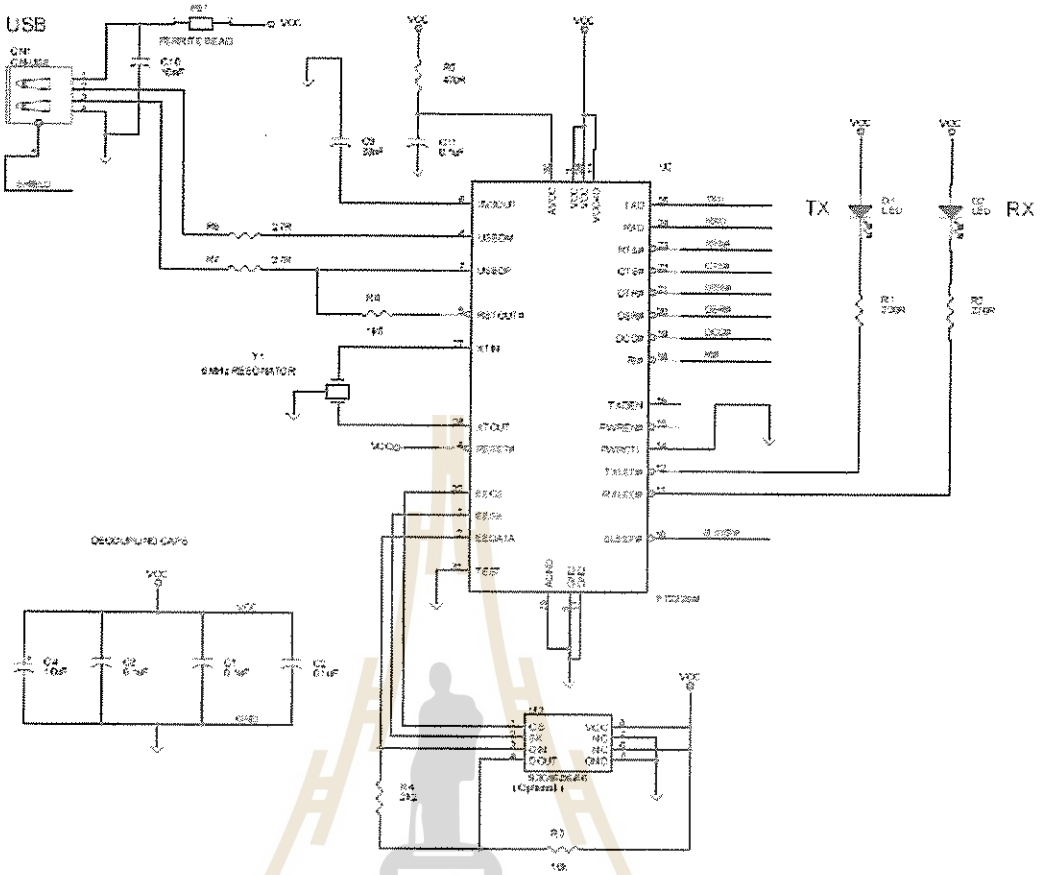


รูปที่ 3.5 ชุดวงจรไมโครคอนโทรลเลอร์ AT89C51ED2

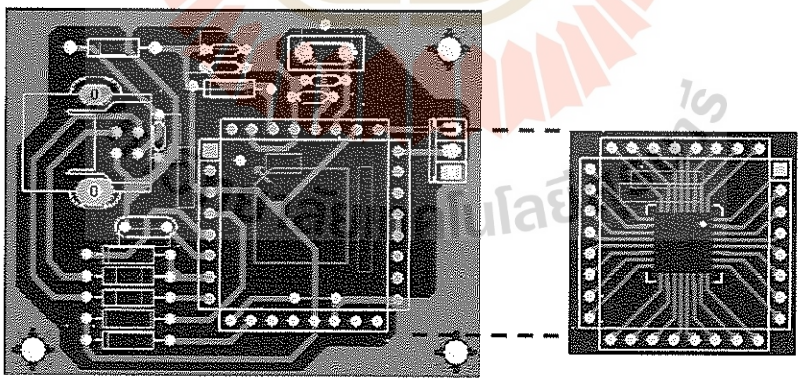
3.4 วงจรเชื่อมต่อ USB

3.4.1 การออกแบบลายวงจรเชื่อมต่อ USB ลงบนแผ่นวงจรพิมพ์

จากรูปที่ 3.6 ได้แสดงถึงวงจรเชื่อมต่อ USB ที่ได้ทำการออกแบบไว้โดยเลือกใช้ไอซีที่มีคุณสมบัติตามต้องการ จากนั้นทำการออกแบบลายวงจรด้วยโปรแกรม Protel 99SE เพื่อที่ได้ลายวงจรเป็นแผ่นลายทองแดงที่จะสามารถนำมาใช้งานได้ตามรูปที่ 3.7



รูปที่ 3.6 วงจรเชื่อมต่อ USB

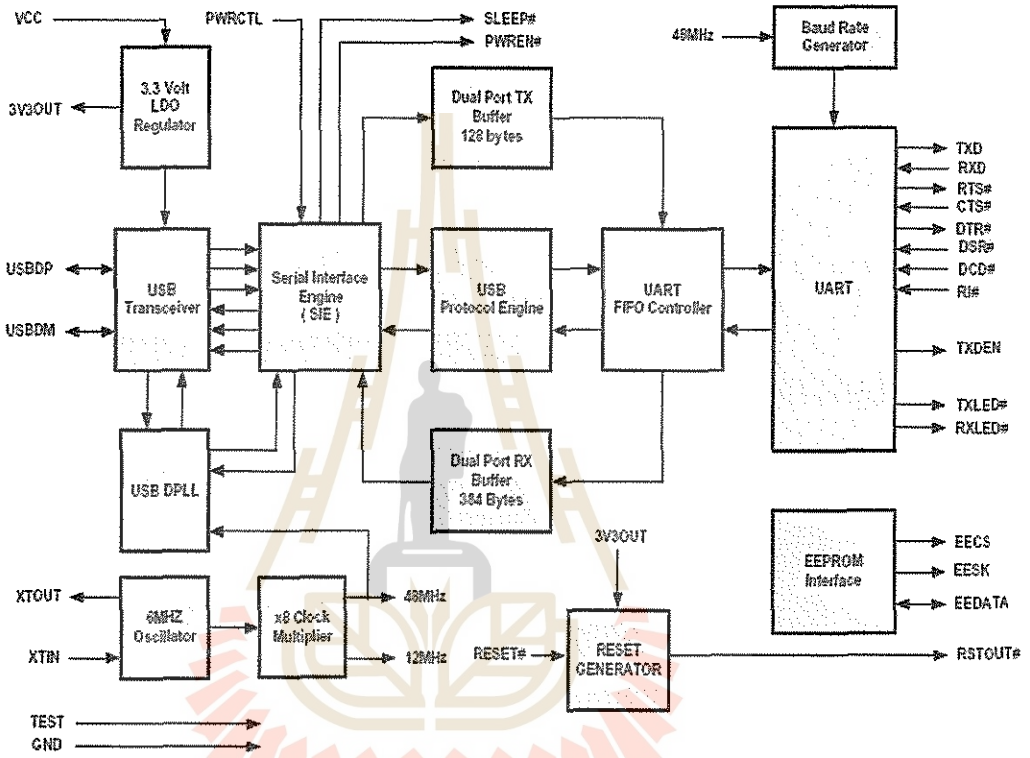


รูปที่ 3.7 ลายวงจรที่ได้จากโปรแกรม Protel99SE บนแผ่นวงจรพิมพ์

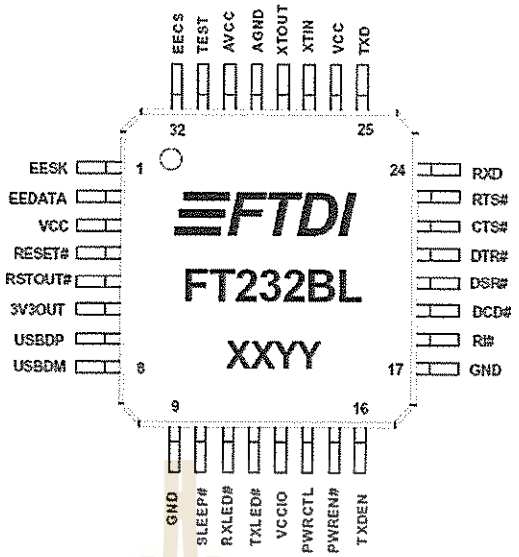
3.4.2 การเลือกใช้ไอซีในการออกแบบ

ในวงจรเชื่อมต่อ USB ได้เลือกใช้ไอซีจำนวน 1 ตัว คือ FT232BL

FT232BL มีคุณสมบัติเป็นตัวเชื่อมต่อระหว่างอุปกรณ์ที่ส่งสัญญาณแบบอนุกรม ให้สามารถเชื่อมต่อกับคอมพิวเตอร์โดยผ่านทางพอร์ต USB ได้ และทำหน้าที่แทน พอร์ต RS-232 ได้ ซึ่งมีแผนผังการทำงานดังรูปที่ 3.8 และมีการจัดเรียงขาของไอซี ดังรูปที่ 3.9



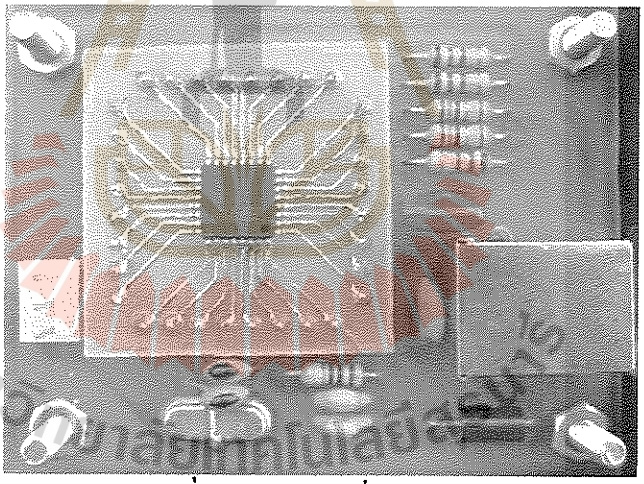
รูปที่ 3.8 USB-to-Serial (Single Channel) Controller Block Diagram



รูปที่ 3.9 Top View ของ FT232BL

3.4.3 การติดตั้งอุปกรณ์อิเล็คทรอนิคส์

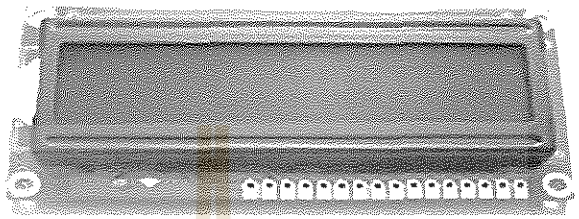
นำอุปกรณ์ที่ได้ออกแบบและจัดเตรียมมาประกอบลงบนแผ่นปริ้นท์ได้ดังรูปที่ 3.11



รูปที่ 3.10 ชุดวงจรเชื่อมต่อ USB

3.5 จอแสดงผลLCD

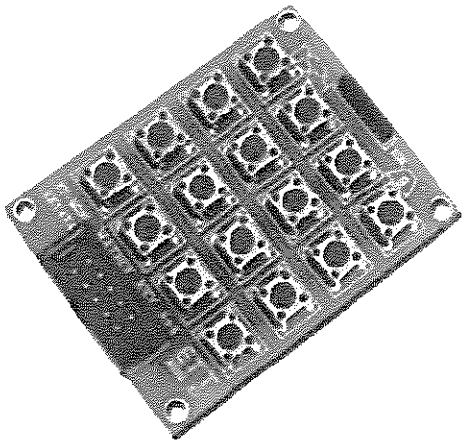
ได้เลือกใช้จอแสดงผล LCD รุ่น LCD 16 CHARS x1 LINE,AV DISPLAY โดยรุ่นที่นำมาใช้จะแสดงผล 1บรรทัดและ แสดง 16 ตัวอักษร ซึ่งเพียงพอต่อการนำมาใช้เป็นอุปกรณ์สำหรับแสดงผลต่างๆ แต่ละค่าในโครงการนี้



รูปที่ 3.11 จอแสดงผล LCD รุ่น LCD 16 CHARS x1 LINE,AV DISPLAY

3.6 KEYPAD

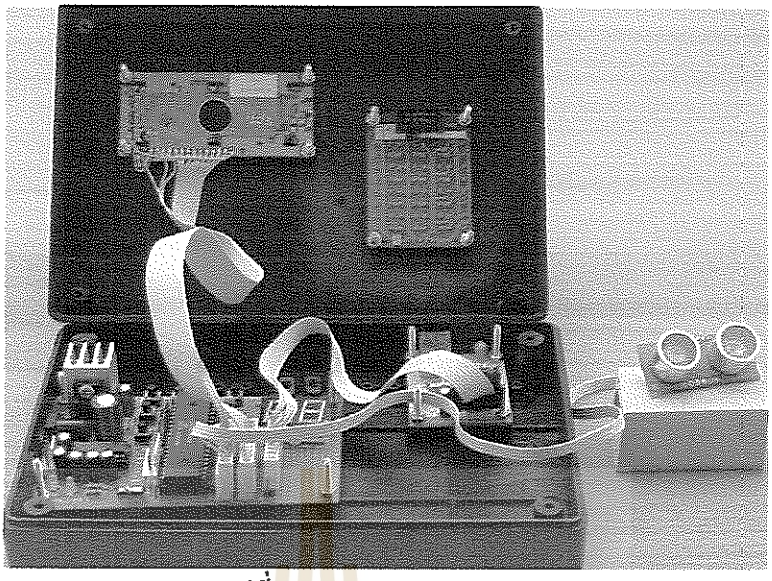
กลุ่มผู้ทดลองได้เลือกใช้ KEYPAD Module ET-MINIKEY 4x4 ซึ่งเป็นชุด Matrix 4x4 โดยจะมี PIN ใช้ต่อกับไมโครคอนโทรลเลอร์ทั้งหมด 8 PIN โดย PIN1-PIN4 (Colum) จะใช้ต่อเป็น Input เพื่อให้ MCU อ่านค่ารหัสของคีย์ที่ถูกกด ส่วน PIN5-PIN8 (ROW) ใช้ต่อเป็น Output เพื่อให้ MCU ส่งลอจิก “0” มาทำการสแกนคีย์ในแต่ละแถว การทำงานนั้น MCU จะอ่านสถานะทางลอจิกของคีย์แต่ละหลักเข้ามาทาง PIN1-PIN4 ซึ่งถ้าไม่มีการกดคียจะอ่านลอจิกได้ “1” ถ้ามีการกดคีย์ลอจิกที่อ่านได้ในหลักนั้นจะเป็น “0” แต่ก่อนที่จะอ่านค่าลอจิกแต่ละหลัก MCU จะต้องให้ลอจิก “0” แก่แถวของคีย์แต่ละแถว (PIN5-PIN8) ในการอ่านลอจิกเข้ามาแต่ละครั้งเสมอ



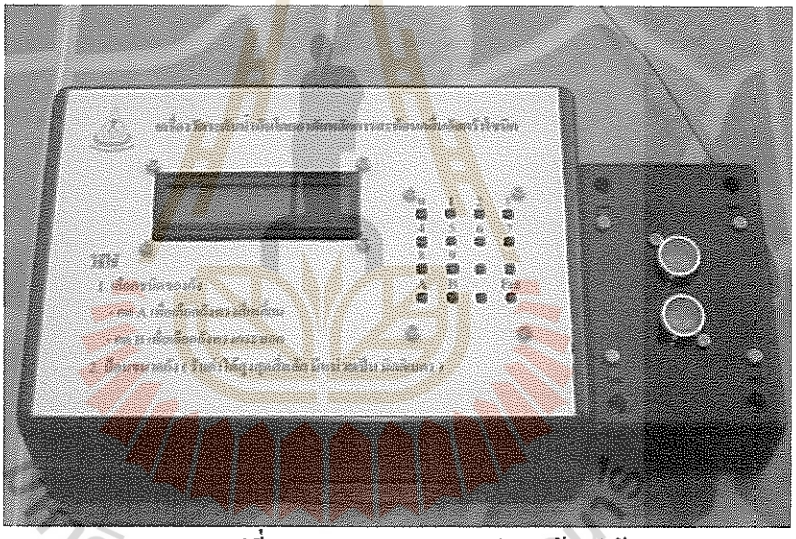
รูปที่ 3.12 ชุดปุ่มกด KEYPAD-Module ET-MINIKEY Matrix 4x4

3.7 สรุปเรื่องการออกแบบและสร้างวงจรสร้างและรับคลื่นอัลตร้าโซนิคในการตรวจวัดระดับน้ำมันในถังน้ำมัน

เมื่อทำการประกอบอุปกรณ์แต่ละอุปกรณ์เข้าด้วยกัน จากนั้นได้ทำการบรรจุลงในกล่องดังรูปที่ 3.14 แสดงรูปภายในกล่อง และรูปที่ 3.15 แสดงภายนอกกล่องพร้อมใช้งาน เพื่อแสดงความเป็นระเบียบและสะดวกต่อการนำไปใช้งานจริง และได้ทำการทดสอบโดยรวมขององค์ประกอบย่อยของแต่ละอุปกรณ์ เพื่อที่อุปกรณ์ต่างๆจะทำงานได้อย่างสมบูรณ์



รูปที่3.13 แสดงรูปภายในกล่อง

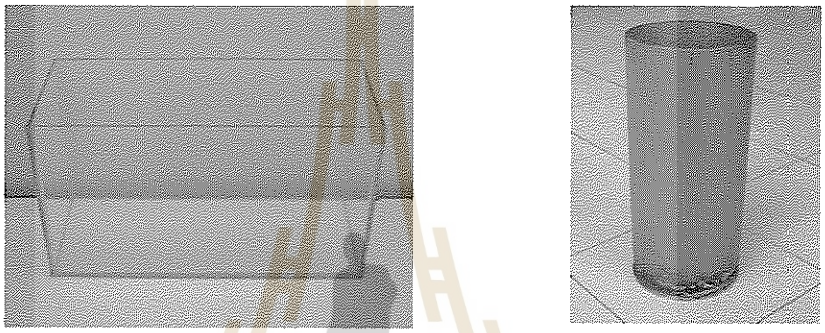


รูปที่3.14 แสดงภายนอกกล่องพร้อมใช้งาน

บทที่ 4
การเขียนโปรแกรมและการทดลองเครื่องวัดระดับน้ำมัน
โดยอาศัยคลื่นอัลตร้าโซนิค

4.1 กล่าวนำ

ในบทนี้อุปกรณ์ที่ได้ออกแบบและสร้างขึ้นมาประกอบเข้าเป็นเครื่องวัดระดับน้ำมันโดยอาศัยคลื่นอัลต้าโซนิค เพื่อทำการวัดระดับน้ำมันภายในถังที่มีรูปร่างและความจุที่แตกต่างกัน ได้แก่ ถังรูปทรงสี่เหลี่ยม ความจุ 10 ลิตร หรือถังรูปทรงกระบอกความจุ 7 ลิตร แสดงดังรูปที่ 4.1



(ก) (ข)

รูปที่ 4.1 (ก) ถังน้ำมันรูปทรงสี่เหลี่ยม
ขนาด 140X438X194 มิลลิเมตร
(ข) ถังรูปทรงทรงกระบอก
ขนาด รัศมี80 มิลลิเมตร X สูง185 มิลลิเมตร

4.2 การใช้โปรแกรม FLIP V2.2.4 และATMEL Microcontroller

ระบบของโปรแกรม FLIP (Flexible In-system Programmer) เป็นโปรแกรมสำหรับพัฒนาไมโครคอนโทรลเลอร์ของATMELโดยสามารถใช้สนับสนุนการพัฒนาโปรแกรมไมโครคอนโทรลเลอร์ในตระกูล MCS51ในกลุ่มที่ใช้การพัฒนาแบบISPซึ่งได้แก่เบอร์ AT89C51RD2 /AT89C51ED2และAT89C51AC2 โดยโปรแกรมทำงานภายใต้ระบบปฏิบัติการของ Windows9X/Me/NT/2000 และ Windows XP โดยสนับสนุนการเชื่อมต่อกับระบบฮาร์ดแวร์ที่ใช้การเชื่อมต่อแบบ RS232 หรือ CAN หรือ USB ซึ่งวิธีการเชื่อมต่อของโปรแกรม FLIP กับระบบฮาร์ดแวร์ของไมโครคอนโทรลเลอร์นั้น จะขึ้นอยู่กับความสามารถของตัวไมโครคอนโทรลเลอร์ ที่จะนำมาทำการพัฒนาว่าสามารถใช้การติดต่อสื่อสารด้วยวิธีใดได้บ้าง แต่สำหรับชุดไมโครคอนโทรลเลอร์อัน

ได้แก่ T89C51RD2/AT89C51RD2/AT89C51ED2 และAT89C51AC2 นั้นจะสามารถใช้การเชื่อมต่อผ่านทางพอร์ตอนุกรม RS232เท่านั้น ไม่สามารถเชื่อมต่อผ่านระบบการสื่อสารของ CANหรือ USB ได้โดยโปรแกรม FLIP จะใช้สำหรับดาวน์โหลด(download) ข้อมูลให้หน่วยความจำของไมโครคอนโทรลเลอร์ที่ทำงานในมอนิเตอร์โหมด (Monitor Mode) เพื่อให้ผู้ใช้สั่งจัดการกับหน่วยความจำภายในตัว CPU ไม่ว่าจะเป็นการ สั่งลบข้อมูล (Erase) สั่งตรวจสอบข้อมูลในหน่วยความจำ (Blank Check) สั่งโปรแกรมข้อมูลให้กับหน่วยความจำโปรแกรมของCPU (Program)โดยโปรแกรมสั่งเปรียบเทียบข้อมูลจากบัฟเพอร์ (Buffer) กับหน่วยความจำในตัว CPU (Verify) หรือสั่งอ่านข้อมูลจากหน่วยจำลองหน่วยจำของCPU (Read) เป็นต้น แต่เนื่องจากบอร์ดไมโครคอนโทรลเลอร์ตระกูล MCS51 ของบริษัท อีทีที จำกัด ในปัจจุบันนั้น จะมีอยู่ด้วยกันมากมายหลายรุ่น และ แต่ละรุ่นก็สามารถเลือกติดตั้งใช้งานกับ MCU ของATMELได้หลายเบอร์เนื่องจากมีโครงสร้างทางฮาร์ดแวร์และกาจัดเรียงขาสัญญาณที่เหมือนกัน ดังนั้นในขั้นตอนของการ Download Program ให้กับ MCU นั้น ผู้ใช้เองต้องเลือกกำหนดเบอร์ของMCU ให้กับโปรแกรม FLIP ให้ถูกต้องกับเบอร์ของ MCU ที่ติดตั้งใช้งานอยู่ภายในบอร์ดด้วยเพราะถ้าเลือกกำหนดเบอร์ไม่ถูกต้องจะทำให้การ Download Program เกิดความผิดพลาดขึ้นได้โดยทาง ATMEL เองก็ได้มีการผลิต MCUซึ่งมีรหัสเบอร์ใกล้เคียงกันแต่มีโครงสร้างในการ Download Program ที่แตกต่างกัน ซึ่งทำให้ผู้ใช้มักเกิดความสับสนและเข้าใจผิดในการเลือกกำหนดเบอร์ของ MCU เพื่อสั่ง Download Program อยู่บ่อยๆ ตัวอย่างดังเช่น T89C51RD2 และ AT89C51RD2 ซึ่งมีรหัสเบอร์ที่ใกล้เคียงกันมากแต่เป็น MCU ที่มีความแตกต่างกันทางด้าโครงสร้างของความจำและวิธีการDownload Program ที่แตกต่างกันซึ่งถ้าหากว่าในบอร์ดได้ทำการติดตั้ง MCU เบอร์ T89C51RD2 ไว้ แต่ผู้ใช้เลือกกำหนดเบอร์ MCU ให้กับโปรแกรม FLIP เป็น AT89C51RD2 ก็จะทำให้ไม่สามารดาวน์โหลดโปรมแกรมให้กับMCU ได้อย่างถูกต้อง ดังนี้เป็นต้น ซึ่งในที่นี้จึงขออธิบายถึงจุดสังเกต ในการดูเบอร์ของ MCU ของATMEL เพื่อเป็นแนวทางให้ผู้ใช้ในการเลือกกำหนดเบอร์ของ MCU ให้กับโปรแกรมFLIPดังนี้

- T89C51RD2 เป็น CPU ของ ATMEL Wireless ซึ่ง MCU เบอร์นี้จะผลิตขึ้นในนามของ ATMEL Wireless โดยจะสามารถสังเกตเห็นลักษณะการพิมพ์เบอร์ที่ตัวถังของ MCU โดยมีรหัส ATMEL และมีรหัสเบอร์เป็น 89C51RD2-CM โดยจะไม่มีตัวหนังสือ AT นำหน้าตัวเลขเบอร์ของ 89C51RD2-CM ด้วย ซึ่งถ้าติดตั้ง MCUรุ่นนี้ให้กับบอร์ดไว้ในขั้นตอนของการเลือกเบอร์ MCU จะต้องเลือกเป็น T89C51RD2 เท่านั้นถ้าเลือกเป็น AT89C51RD2 จะไม่ดาวน์โหลดโปรแกรมได้อย่างถูกต้อง

- AT89C51RD2 เป็น CPU ของ ATMEL โดยจะสามารถสังเกตเห็นลักษณะการพิมพ์เบอร์ที่ตัวถังของ MCU โดยมีรหัส ATMEL และมีรหัสเบอร์เป็น AT89C51RD2-IMซึ่งถ้าติดตั้ง MCU รุ่นนี้

ให้กับบอร์ดไว้ ในขั้นตอนของการเลือกเบอร์ MCU จะต้องเลือกเป็นAT89C51RD2 เท่านั้น ถ้าเลือกเป็น T89C51RD2 จะไม่สามารถดาวน์โหลดโปรแกรมได้อย่างถูกต้อง

- AT89C51ED2 เป็น CPU ของ ATMEL โดยจะสามารถสังเกตเห็นลักษณะการพิมพ์เบอร์ที่ตัวถังของ MCU โดยมีรหัส ATMEL และมีรหัสเบอร์เป็น AT89C51ED2-IM ซึ่งถ้าติดตั้ง MCU รุ่นนี้ให้กับบอร์ดไว้ ในขั้นตอนของการเลือกเบอร์ MCU จะต้องเลือกเป็น AT89C51ED2 เท่านั้น ถ้าเลือกเป็นเบอร์อื่นๆ จะไม่สามารถดาวน์โหลดโปรแกรมได้อย่างถูกต้อง

4.2.1วิธีการ Download Program ให้กับ MCU

สำหรับวิธีการดาวน์โหลด MCU ของ ATMELในกลุ่มเบอร์ที่ใช้วิธีการดาวน์โหลดแบบ ISP เช่น T89C51RD2/AT89C51RD2/AT89C51ED2 และAT89C51AC2นั้นสามารถใช้โปรแกรม FLIP ของATMELในการสั่งดาวน์โหลดได้ซึ่งวิธีการดาวน์โหลดโปรแกรมสามารถทำได้ 2แนวทาง

4.2.1.1 การ ดาวน์โหลดแบบธรรมดา (Manual Download)

สามารถใช้ได้กับบอร์ดไมโครคอนโทรลเลอร์ของบริษัท อีทีที จำกัด ทุกๆรุ่นที่มีพอร์ตสื่อสารอนุกรม RS232 แบบ 4 Pin และมีสวิตช์ RESET และสวิตช์ PSEN (บางรุ่นเรียกสวิตช์ LOAD) อยู่ภายในบอร์ดด้วย

4.2.1.2 การDownloadแบบอัตโนมัติ (Auto Download)

สามารถใช้งานได้กับ บอร์ดไมโครคอนโทรลเลอร์ของบริษัท อีทีที จำกัด ทุกๆรุ่นที่ขั้วดาวน์โหลดแบบ 5 Pin ซึ่งเรียกว่า “ETDOWNLOAD” (บางรุ่นอาจเรียก LOAD-RD2) อยู่ภายในบอร์ดด้วยซึ่งบอร์ไมโครคอนโทรลเลอร์ของ บริษัท อีทีที จำกัด ซึ่งสามารถทำการติดตั้งใช้งาน กับ MCU ของ ATMEL ในกลุ่มเบอร์ที่ใช้การดาวน์โหลดแบบ ISP ได้นั้นมีอยู่ด้วยกันมากมายหลายรุ่น โดยบางรุ่นก็สามารถเลือกติดตั้ง MCU ได้มากกว่า 1 เบอร์ ซึ่งการที่จะเลือกใช้ MCU เบอร์ใดนั้นขึ้นอยู่กับความต้องการของผู้ใช้เป็นหลัก ซึ่งได้แก่

- บอร์ดไมโครคอนโทรลเลอร์รุ่น ET-BASE51
- บอร์ดไมโครคอนโทรลเลอร์รุ่น ET-BASE51 V2
- บอร์ดไมโครคอนโทรลเลอร์รุ่น CP-JR51AC2 V1.0 / EXP
- บอร์ดไมโครคอนโทรลเลอร์รุ่น CP-JR51AC2 V2.0
- บอร์ดไมโครคอนโทรลเลอร์รุ่น CP-JR51RD2 และ CP-JR51RD2 EXP
- บอร์ดไมโครคอนโทรลเลอร์รุ่น CP-AT32 PLUS V2
- บอร์ดไมโครคอนโทรลเลอร์รุ่น CP-SPI/RD2 V1 / EXP
- บอร์ดไมโครคอนโทรลเลอร์รุ่น CP-SPI/RD2 V2

- บอร์ดไมโครคอนโทรลเลอร์รุ่น CP-SPI/RD2 V3
- บอร์ดไมโครคอนโทรลเลอร์รุ่น CP-89C51 V1 PLUS
- ชุดพัฒนาโปรแกรม MCS51 รุ่น ET-SPI51 V1 และ ET-SPI51 V2
- ชุดพัฒนาเรียนรู้หุ่นยนต์รุ่น ET-ROBOT RD2
- ชุดทดลอง MCS51 รุ่น ET-LAB3A (MCS51)

การ Download แบบ Manual

ในการดาวน์โหลดแบบนี้จะใช้กับสาย RS232 แบบ 4 Pin ในการสั่งดาวน์โหลดโปรแกรม ซึ่งสามารถใช้งานได้กับโปรแกรม FLIP ทุกรุ่น แต่ในปัจจุบันโปรแกรม FLIP ได้รับการปรับปรุงเป็น "FLIP V2.4.4" แล้วซึ่งเมื่อต้องการให้โปรแกรม FLIP ติดต่อกับ CPUในมอนิเตอร์โหมด (Monitor Mode) ด้วยวิธีการดาวน์โหลดแบบธรรมดา (Manual Download) นั้นจะต้องสั่ง Resetให้ CPUเข้าทำงานในมอนิเตอร์โหมด (Monitor Mode) ก่อน จึงจะสามารถสั่งงาน CPU ผ่านทางโปรแกรม FLIP ได้ ซึ่งหลักการสำหรับ Reset ให้ CPU เข้าทำงานใน มอนิเตอร์โหมด (Monitor Mode) จะต้องกำหนดให้ขาสัญญาณ PSEN มีสภาวะเป็น "0" ในขณะที่ CPU หลุดพ้นจากสภาวะของการ Reset ซึ่งตามปรกติแล้วหลังการ Reset ทุกครั้ง CPU จะตรวจสอบสภาวะของขาสัญญาณ PSEN ว่าเป็น "0" หรือไม่ถ้าไม่ใช่ก็จะทำงานในโหมการทำงานปรกติแต่ถ้าใช่ก็จะตรวจสอบสถานะของสัญญาณอื่นๆที่เกี่ยวข้องกับการทำงานในมอนิเตอร์โหมด (Monitor Mode) ถ้าเงื่อนไขอื่นๆถูกต้องก็จะเข้าทำงานในมอนิเตอร์โหมด (Monitor Mode) ทันที สำหรับบอร์ดของบริษัท อีทีที จำกัด รุ่นที่มีสวิตช์ RESET และสวิตช์ PSEN อยู่ภายในบอร์ดนั้นการที่จะสั่ง Reset ให้ CPU ของATMEL เข้าทำงานในมอนิเตอร์โหมด (Monitor Mode) ด้วยวิธีการแบบธรรมดานั้นจะต้องทำตามลำดับขั้นตอนดังต่อไปนี้

1. กดสวิตช์ PSEN (บางรุ่นเรียกสวิตช์ LOAD) ค้างไว้เพื่อกำหนดสถานะขาสัญญาณPSEN ให้เป็น "0"

2. กดสวิตช์ RESET เพื่อส่งสัญญาณ RESET ให้กับ CPU โดยสวิตช์ PSEN ต้องกดค้างอยู่เช่นเดิม

3. ปล่อยสวิตช์ RESET เพื่อปล่อยให้ CPU พ้นจากสภาวะการ Reset (สวิตช์ PSEN กดค้าง)

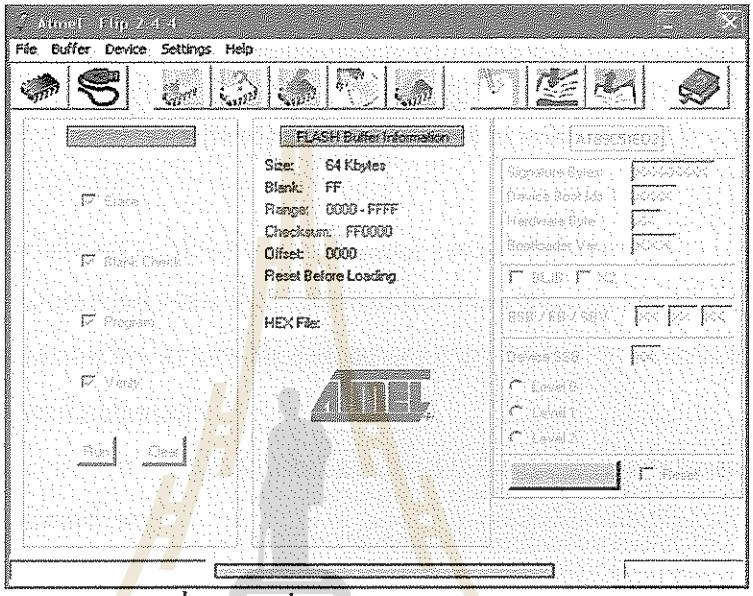
4. ปล่อยสวิตช์ PSEN เป็นลำดับสุดท้ายซึ่งผู้ใช้จะต้องทำการสั่ง Reset การทำงานของ MCU ให้เริ่มต้นในมอนิเตอร์โหมด (Monitor Mode) รอไว้ก่อนที่จะสั่ง Connect การเชื่อมต่อด้วยโปรแกรม FLIP เสมอ ไม่เช่นนั้นแล้วจะไม่สามารถสั่งดาวน์โหลด ได้อย่างถูกต้อง

ลำดับขั้นตอนการดาวน์โหลด HEX File แบบธรรมดา

1. ต่อสาย RS232 แบบ 4 PIN จาก Com Port ของเครื่องคอมพิวเตอร์ PC เข้ากับขั้ว RS232 แบบ 4 Pin ของบอร์ด

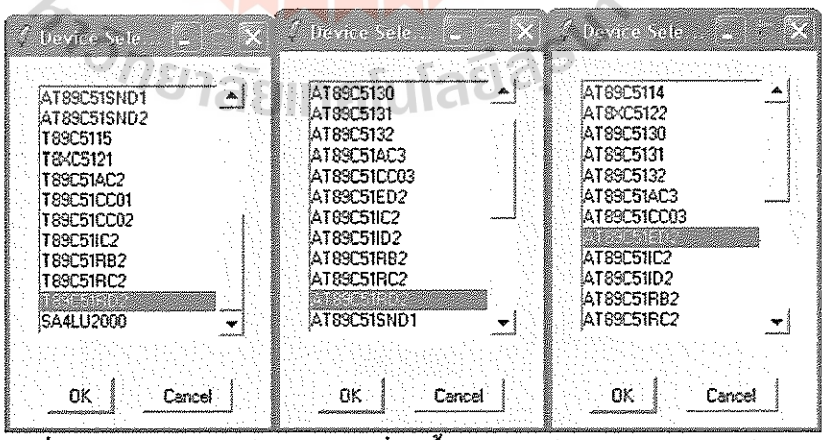
2. จ่ายไฟเลี้ยงวงจรให้บอร์ด ซึ่งจะสังเกตุเห็น LED แสดงสถานะของ PWR สีแดงติดสว่าง

3. สั่ง Run โปรแกรม FLIP V2.4.4 ซึ่งจะได้ผลดังรูปที่ 4.2



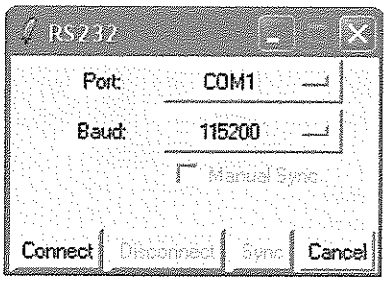
รูปที่ 4.2 การสั่ง Run โปรแกรม FLIP V2.4.4

4. สั่งเลือกกำหนดเบอร์ของ MCU ที่ติดตั้งไว้ในบอร์ด โดยเลือก Device ⟶ Select ซึ่งต้องเลือกกำหนดให้ตรงกับที่ทำการติดตั้งไว้จริงๆในบอร์ดด้วย ดังตัวอย่างในรูปที่ 4.3



รูปที่ 4.3 การสั่งเลือกกำหนดเบอร์ของ MCU ที่ติดตั้งไว้ในบอร์ดสำหรับการดาวน์โหลดโปรแกรม

5. คลิกเมาส์ที่คำสั่ง setting ⟶ Communication ⟶ RS232 จากนั้นเลือกกำหนดComport ให้ตรงกับที่ต่อสายไว้จริง ดังรูปที่ 4.4 (ในตัวอย่างใช้ COM1)



รูปที่ 4.4 การกำหนด Comport ให้ตรงกับสายที่ทำการต่อไว้

6. ทำการรีเซ็ต MCU ให้เข้าทำงานใน Monitor โดยมีลำดับขั้นตอนดังนี้

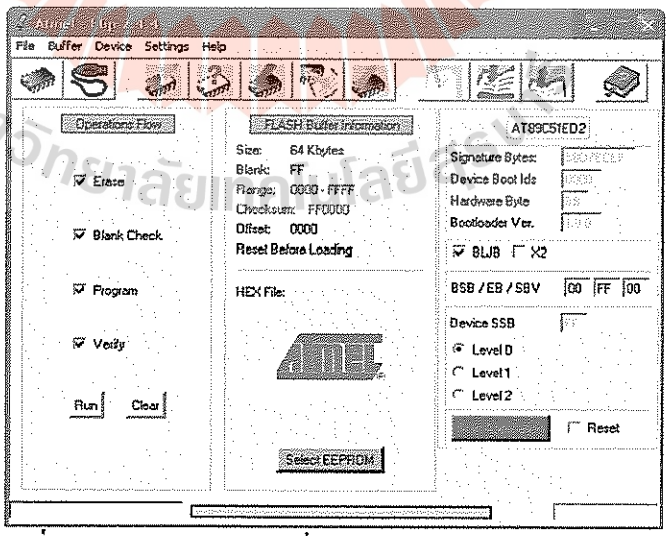
a) กดสวิตช์ PSEN ค้างไว้เพื่อกำหนดสถานะขาสัญญาณ PSEN ให้เป็น “0”

b) กดสวิตช์ RESET เพื่อส่งสัญญาณ RESET ให้กับ CPU โดยสวิตช์ PSEN ต้องกดค้างอยู่เช่นเดิม

c) ปล่อยสวิตช์ RESET เพื่อปล่อยให้ CPU พ้นจากสภาวะการ Reset (สวิตช์ PSEN ยังกดค้างอยู่)

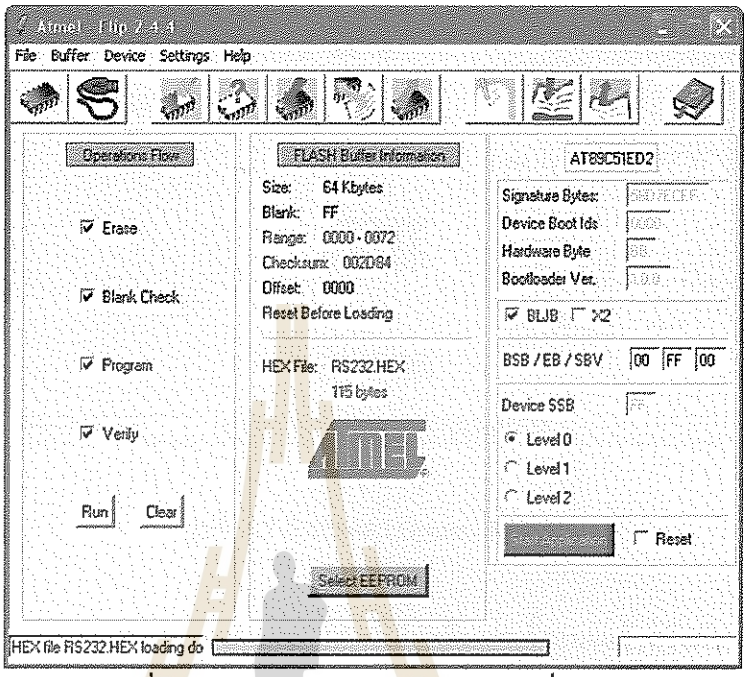
d) ปล่อยสวิตช์ PSEN เป็นลำดับสุดท้าย

7. คลิกเมาส์ที่ปุ่ม Connect เพื่อทำการติดต่อสื่อสารกับ MCU ใน Monitor Mode ซึ่งจะได้ผลดังรูปที่ 4.5



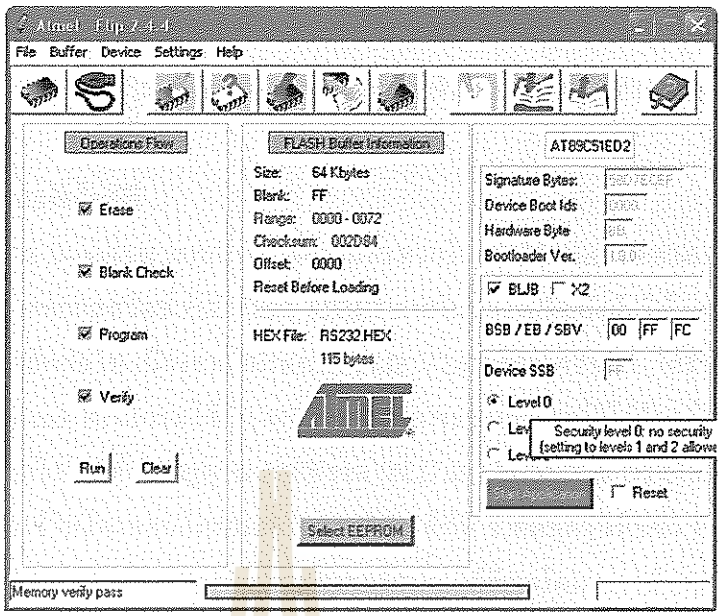
รูปที่ 4.5 ทำการติดต่อติดต่อสื่อสารกับ MCU ใน Monitor Mode

8. สั่งเปิด Hex File ที่ต้องการจะ Download ให้กับ MCU มารอไว้ใน Buffer ของโปรแกรม FLIP โดยใช้คำสั่ง File ⟶ Load Hex File… ดังรูปที่ 4.6



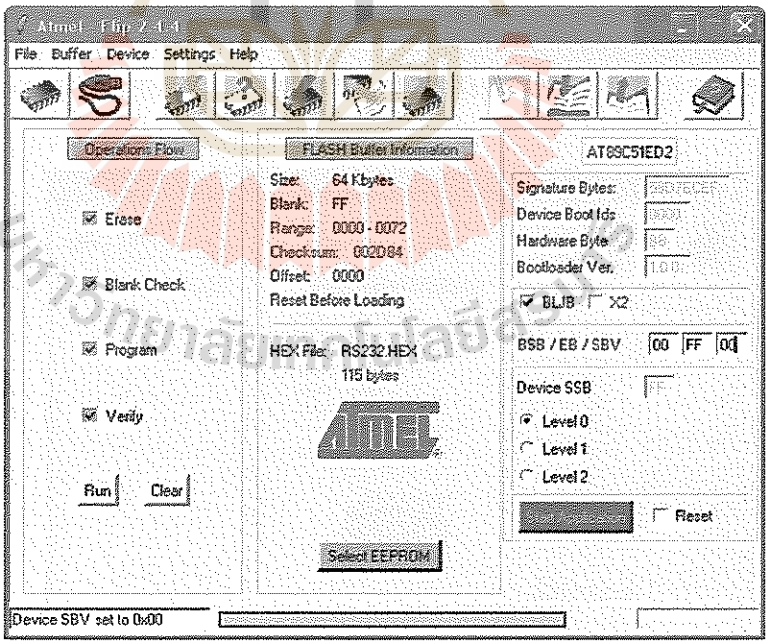
รูปที่ 4.6 การเปิด Hex File ที่ต้องการจะ Download ให้กับ MCU เพื่อให้รอใน Buffer ของโปรแกรม

9. คลิกเมาส์ที่หน้าตัวเลือกคำสั่งใน Tab ของ Operation Flow ให้ครบทุกคำสั่ง ซึ่งได้แก่ Erase, Bank Check, Program, Verify จากนั้นคลิกเมาส์ที่ปุ่มคำสั่ง Run และรอจนการทำงานของโปรแกรมเสร็จเรียบร้อยดังรูปที่ 4.7



รูปที่ 4.7 กำหนดค่า Operation Flow

10. ตรวจสอบค่า Device BSB และ SBV ว่ามีค่าเป็น 00 ทั้งหมดแล้วหรือยัง ซึ่งถ้ายังไม่เป็น 00 ให้ทำการแก้ไขค่าให้เป็น 00 โดยคลิกเมาส์ในช่องตัวเลขแล้วพิมพ์ค่า 00 แทนที่ลงไปทั้ง 2 ช่องดังรูป



รูปที่ 4.8 ตรวจสอบค่า Device BSB และ SBV

11. กดสวิตช์ Reset ให้กับบอร์ดเพื่อให้บอร์ดเริ่มต้นทำงานตามโปรแกรมที่ได้ทำการ Download ไปให้ ซึ่งถ้าไม่เกิดความผิดพลาดใดๆจะเห็น MCU เริ่มต้นทำงานทันที

การ Download แบบอัตโนมัติ

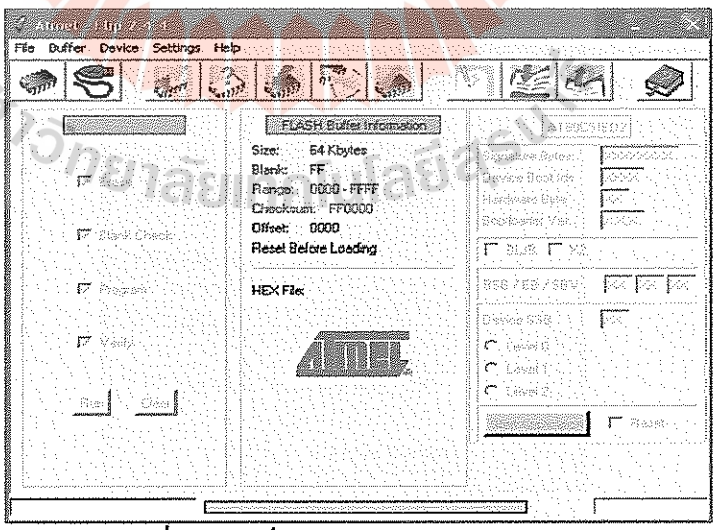
สำหรับการ Download แบบนี้จะเกิดความสะดวกในการใช้งานเป็นอย่างมาก เนื่องจากผู้ใช้ไม่ต้องเสียเวลามาคอยกดสวิตช์ PSEN และ RESET ภายในบอร์ดเพื่อบังคับให้ MCU เข้าทำงานใน Monitor ให้ยุ่งยากอีกต่อไป เนื่องจากโปรแกรม FLIP V2.4.4 ของ ATMEL ได้รับการปรับปรุงให้สามารถสั่ง Download โปรแกรมให้กับ MCU ด้วยวิธีการแบบอัตโนมัติได้ด้วย โดยจะใช้สัญญาณ RTS ในการกำหนดโลจิกให้กับสัญญาณ PSENของ MCU และใช้สัญญาณ DTRในการควบคุมการรีเซ็ของ MCUซึ่งบอร์ไมโครคอนโทรลเลอร์ของบริษัท อีทีที จำกัดทุกรุ่นที่มีขั้ว ET-DOWNLOAD (บางรุ่นเรียก LOAD-RD2) แบบ 5PIN อยู่ภายในบอร์ดนั้น ได้จัดเตรียมวงจรสำหรับการ Download แบบอัตโนมัติไว้ด้วยแล้ว โดยให้ใช้สาย RS232 รุ่น ET-DOWNLOADแบบ 5 Pin ต่อเข้ากับบอร์ดที่ขั้ว ET-DOWNLOAD ของบอร์ดเท่านั้น

ลำดับขั้นตอนการ Download HEX File แบบอัตโนมัติ

1. ต่อสาย RS232 Download แบบ 5 PIN จาก Com Port ของเครื่องคอมพิวเตอร์ PC เข้ากับขั้ว ET-DOWNLOAD (ในบางรุ่นอาจเรียกว่า LOAD-RD2) แบบ 5 Pin ของบอร์ด

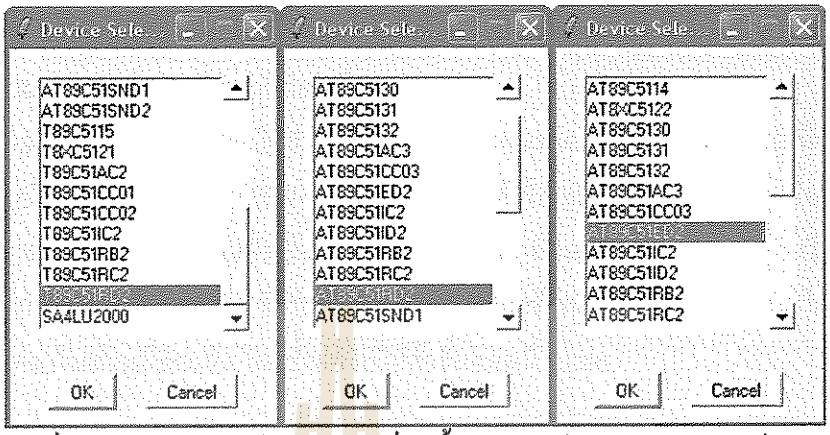
2. จ่ายไฟเลี้ยงวงจรให้บอร์ด ซึ่งจะสังเกตุเห็น LED แสดงสถานะของ PWR สีแดงสว่างอยู่

3. สั่ง Run โปรแกรม FLIP V2.4.4 ซึ่งจะได้ผลดังรูปที่ 4.9



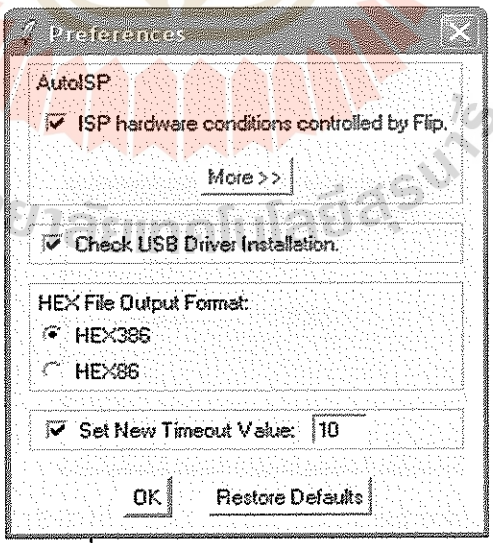
รูปที่ 4.9 การสั่ง Run โปรแกรม FLIP V2.4.4

4. สั่งเลือกกำหนดเบอร์ของ MCU ที่ติดตั้งไว้ในบอร์ด โดยเลือก Device ⟶ Select ซึ่งต้องเลือกกำหนดให้ตรงกับที่ทำการติดตั้งไว้จริงๆในบอร์ดด้วย ดังรูป 4.11



รูปที่ 4.10 การสั่งเลือกกำหนดเบอร์ของ MCU ที่ติดตั้งไว้ในบอร์ดสำหรับการดาวน์โหลดโปรแกรม

5. คลิกเมาส์ที่เมนูคำสั่ง Settings ⟶ Preferences... ในขั้นตอนเลือกกำหนดค่า Time-Out ของการสื่อสารโดยคลิกเมาส์เติมเครื่องหมาย (√) ที่หน้าตัวเลือกในหัวข้อ **"Set New Timeout Value"** พร้อมกับกำหนดค่าในช่องรับค่าเท่ากับ10 จากนั้นให้เลือกกำหนดการสื่อสารเป็นแบบ Auto ISP โดยให้คลิกเมาส์เติมเครื่องหมาย (√) ที่หน้าตัวเลือกในหัวข้อ **"ISP Hardware conditions controlled by Flip."** แล้วเลือกคลิกเมาส์ที่ปุ่มคำสั่ง **"More >>"** เพื่อเข้าไปกำหนดคุณสมบัติของสัญญาณควบคุม ดังรูปที่ 4.12

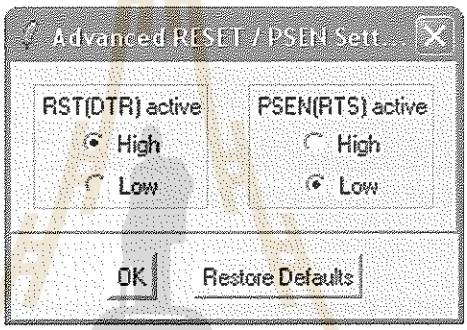


รูปที่ 4.11 กำหนดค่าต่างๆใน Preferences

ในขั้นตอนนี้จะเป็นการเลือกกำหนดคุณสมบัติของสัญญาณจาก Comport ที่จะใช้ในการควบคุมสัญญาณ RESET และ PSEN ของ MCU ในบอร์ด ให้เข้าทำงานใน Monitor Modeโดยอัตโนมัติ โดยให้เลือกกำหนดค่าตัวเลือกเป็นดังนี้

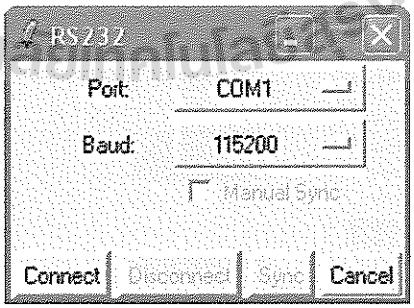
- **RST (DTR) active** จะใช้สำหรับเลือกกำหนดคุณสมบัติของสัญญาณ DTRที่จะใช้ในการควบคุมการ Reset ของ MCU โดยให้เลือกเป็น High

- **PSEN (RTS) active** จะใช้สำหรับเลือกกำหนดคุณสมบัติของสัญญาณ RTSที่จะใช้ในการกำหนดโลจิกให้กับขาสัญญาณ PSEN ของ MCU โดยให้เลือกกำหนดเป็นLowซึ่งเมื่อทำการเลือกกำหนดค่าต่างๆดังกล่าวข้างต้นเสร็จเรียบร้อยแล้วให้เลือกคลิกเมาส์ที่ปุ่มคำสั่ง Oเพื่อให้โปรแกรม FLIP ทำการบันทึกค่าตัวเลือกนี้ไว้ ซึ่งในครั้งต่อๆไป ที่เรียกใช้งานโปรแกรFLIP อีก ก็ไม่จำเป็นต้องเข้ามากำหนดค่าในส่วนนี้อีกแล้ว ในครั้งต่อไปให้ข้าขั้นตอนนี้ไปเลย



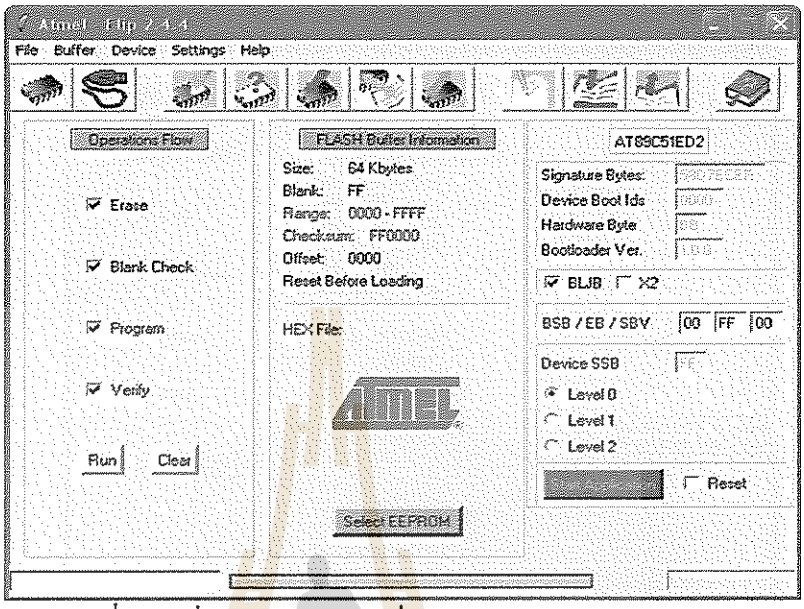
รูปที่ 4.12 กำหนดค่าที่จะใช้ในการควบคุมสัญญาณ RESET และ PSEN ของ MCU ในบอร์ด

6. คลิเมาส์ที่คำสั่ง setting ⟶ Communication ⟶ RS232 จากนั้นเลือกกำหนดComport ให้ตรงกับที่ต่อสายไว้จริง ดังรูปที่ 4.14



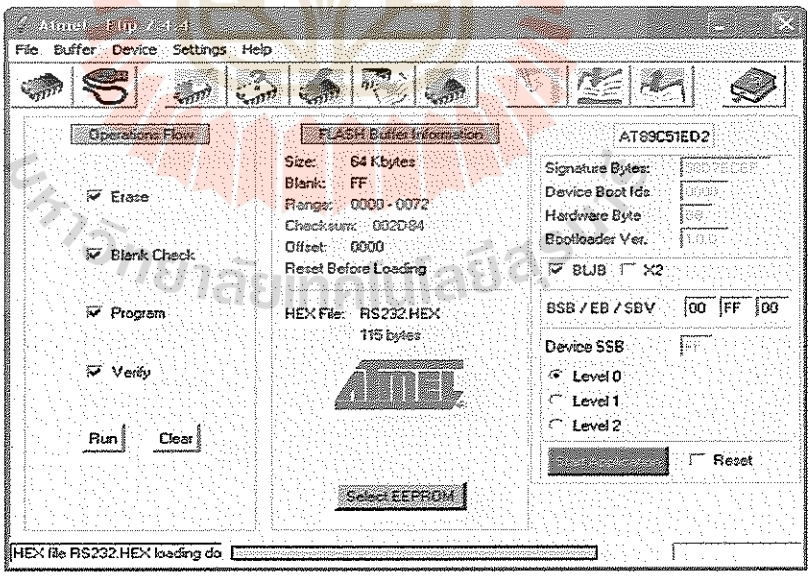
รูปที่ 4.13 การกำหนด Comport ให้ตรงกับสายที่ทำการต่อไว้

7. คลิกเมาส์ที่ปุ่ม Connect เพื่อให้โปรแกรม FLIP เริ่มต้นทำการติดต่อสื่อสารกับ MCU ใน Monitor Mode ซึ่งจะได้ผลดังรูปที่ 4.15



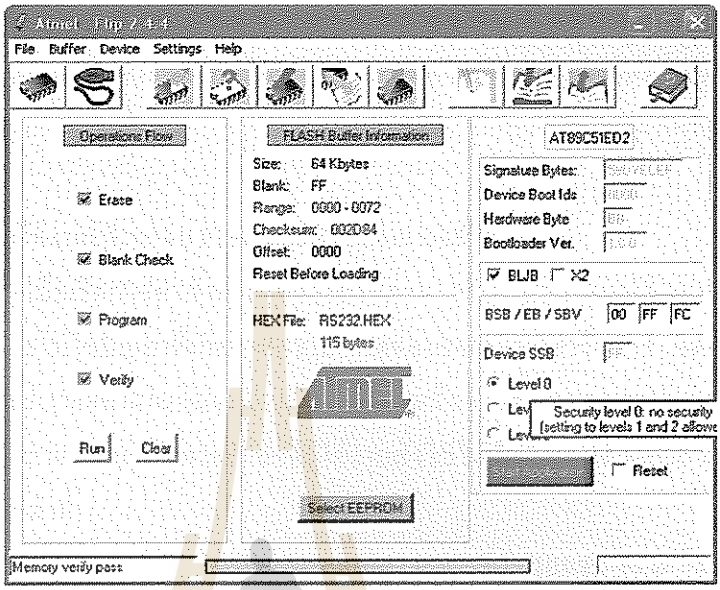
รูปที่ 4.14 เริ่มต้นทำการติดต่อสื่อสารกับ MCU ใน Monitor Mode

8. สั่งเปิด Hex File ที่ต้องการจะ Download ให้กับ MCU มารอไว้ใน Buffer ของโปรแกรม FLIP โดยใช้คำสั่ง File ⟶ Load Hex File... ดังรูปที่ 4.16



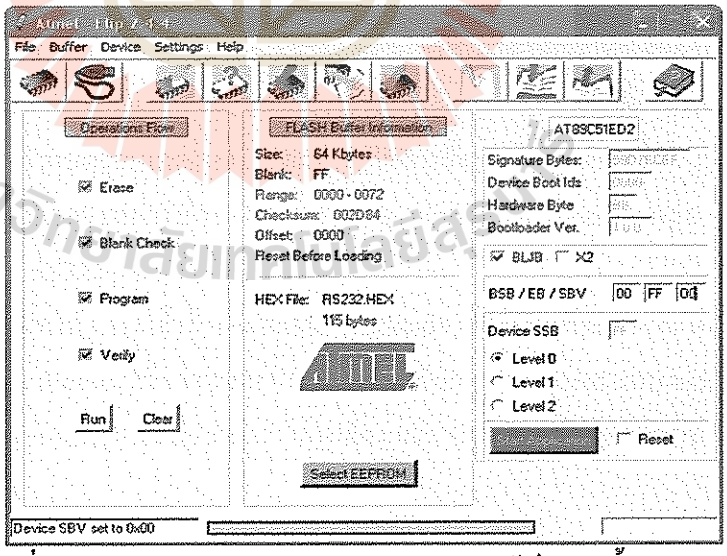
รูปที่ 4.15 การเปิด Hex File ที่ต้องการดาวน์โหลดให้กับ MCU เพื่อให้รอใน Buffer ของโปรแกรม

9. คลิกเมาส์ที่หน้าตัวเลือกคำสั่งใน Tab ของ Operation Flow ให้ครบทุกคำสั่ง ซึ่งได้แก่ Erase, Bank Check, Program, Verify จากนั้นคลิกเมาส์ที่ปุ่มคำสั่ง Run และรอจนการทำงานของโปรแกรมเสร็จเรียบร้อยดังรูป



รูปที่ 4.16 กำหนดค่าของ Operation Flow

10. ตรวจสอบค่า Device BSB และ SBV ว่ามีค่าเป็น 00 ทั้งหมดแล้วหรือยัง ซึ่งถ้ายังไม่เป็น 00 ให้ทำการแก้ไขค่าให้เป็น 00 โดยคลิกเมาส์ในช่องตัวเลขแล้วพิมพ์ค่า 00 แทนที่ลงไปทั้ง 2 ช่องดังรูป 4.18



รูปที่ 4.17 ตรวจสอบค่า Device BSB และ SBV ให้เป็น 00 ทั้งหมด

11. ทำการคลิกเมาส์ที่ปุ่ม Start Application กดสวิตช์ Reset ให้กับบอร์ดเพื่อให้บอร์ดเริ่มต้นทำงานตามโปรแกรมที่ได้ทำการ Download ไปให้ ซึ่งถ้าไม่เกิดความผิดพลาดใดๆจะเห็นMCU เริ่มต้นทำงานทันที และควรถอดสาย Download ออกจากบอร์ดด้วยเสมอเนื่องจากเมื่อจบการทางานจาก โปรแกรม FLIP ไปแล้วสถานะของสัญญาณ RTS และDSR ของพอร์ตสื่อสารอนุกรม RS232 อาจมีสถานะไม่แน่นอน ซึ่งอาจทำงานค้างอยู่ อันจะส่งผลทำให้ MCU ถูกรีเซ็ตอยู่ตลอดเวลาทำให้ MCU ไม่ทำงาน

ปัญหาต่างๆในขณะใช้งานโปรแกรม FLIP และแนวทางการแก้ไข

ในบางครั้งเมื่อเรียกใช้คำสั่งต่างๆของโปรแกรม FLIP แล้ว อาจเกิดความผิดพลาดบางประการขึ้น ซึ่งอาจไม่ใช่ปัญหาที่เกิดจากความบกพร่องของระบบฮาร์ดแวร์ แต่อาจเกิดการการกำหนดพารามิเตอร์บางอย่างในโปรแกรมไม่ถูกต้องหรือข้ามขั้นตอนบางประการไป ซึ่งเมื่อโปรแกรม FLIP ไม่สามารถปฏิบัติตามคำสั่งที่ผู้ใช้งานสั่งไปได้สำเร็จจะแสดงอาการ Error ต่างๆให้ทราบ ซึ่งพอสรุปได้ดังนี้

1. **Time Out Error** เป็นความผิดพลาดที่เกิดจากการที่โปรแกรม FLIP ไม่สามารถทำการสื่อสารกับ CPU ใน Monitor Mode ได้ ซึ่งอาจเกิด หลายสาเหตุ เช่น

- ใช้สายสัญญาณในการ Download ไม่ถูกต้อง ซึ่งถ้าใช้งานบอร์ดไมโครคอนโทรลเลอร์ของบริษัท อีทีที จำกัด รุ่นที่สามารถใช้งานกับโปรแกรม Flip นั้น ถ้าใช้สายสัญญาณRS232 แบบ 4Pin จะต้องใช้การ Download แบบ Manual เท่านั้น ส่วนถ้าต้องการDownload แบบ Auto จะต้องใช้สาย ET-DOWNLOAD (5Pin) ของ บริษัท อีทีที จำกัด

- การต่อสายสัญญาณระหว่างขั้วต่อ RS232 ของบอร์ดไมโครคอนโทรลเลอร์กับขั้วต่อพอร์ตสื่อสารอนุกรม RS232 ของคอมพิวเตอร์ยังไม่เรียบร้อย หรือ ต่อไม่ตรงกับที่กำหนดตัวเลือกไว้ในโปรแกรม หรือ การกำหนดรูปแบบและตัวเลือกต่างๆในการสื่อสารไม่ถูกต้องเมื่อพบปัญหานี้ให้ลองทำการตรวจสอบค่าต่างๆในการสื่อสารใน Setting ⟶ Preferences... และ Setting ⟶ Communication ⟶ RS232

- ยังไม่ได้รีเซ็ตให้ CPU เข้าทำงานใน Monitor Mode รอไว้ก่อนที่จะสั่งงานโปรแกรมในกรณี Download แบบ Manual หรือกำหนดค่า Option สำหรับการ Download แบบAuto ที่ **"ISP Hardware conditions controlled by Flip."** ไม่ถูกต้อง หรือ บอร์ดยังไม่พร้อมทำงาน เช่น ยังไม่ได้จ่ายไฟเลี้ยงให้บอร์ด

- กำหนดค่า Baud rate เร็วเกินไป ซึ่งในกรณีที่ใช้งานกับเครื่องคอมพิวเตอร์ที่มีความเร็วมากๆนั้น ควรกำหนดค่า Baud rate ในการสื่อสารให้ช้าลง ซึ่งอาจใช้ค่า 19200 หรือ 9600ก็พอ เพราะถ้ากำหนดให้ความเร็วมากเกินไป จะทำให้เกิดความผิดพลาดบ่อยครั้งขึ้น

2. **Software Security Bit Set. Cannot access device Data** เป็นความผิดพลาดที่เกิดจากการนำ CPU ที่มีการสั่ง Lock Bit ของ Security Bit ไว้ก่อนแล้ว จึงมาสั่ง Program หรือ Verifyหรือ Read ในภายหลังโดยยังไม่ได้สั่งลบข้อมูลเก่าออกเสียก่อน ซึ่งให้แก้ปัญหาด้วยการสั่งลบข้อมูล (Erase) เสียก่อนแล้วจึงสั่งเขียนข้อมูลใหม่อีกครั้งหนึ่ง

3. **The board reply is not correct** เป็นความผิดพลาดที่เกิดจากการสื่อสารข้อมูลระหว่างโปรแกรม FLIP กับ ไมโครคอนโทรลเลอร์ เกิดความผิดพลาดในลักษณะของ Framing Errorขึ้น ซึ่งปัญหาอาจเกิดจากการกำหนดค่า Baud rate ไม่ถูกต้องกับค่าความถี่ของ Crystal

4. **The RS232 port could not be opened** เป็นความผิดพลาดที่เกิดจากโปรแกรม FLIP ไม่สามารถสั่งเปิดการทำงานของพอร์ตสื่อสารอนุกรม RS232 ของเครื่องคอมพิวเตอร์ได้ ซึ่งอาจเกิดจากการกำหนดหมายเลข Comport ในโปรแกรมที่เลือกไว้ไม่มีอยู่จริง หรือมีโปรแกรมอื่นเรียกใช้งาน Comport นั้นค้างอยู่ หรือเรียกใช้งานโปรแกรม FLIP ในขณะที่กำลังมีการใช้งานComport อยู่ด้วย ซึ่งให้ลองปิดโปรแกรม FLIP แล้วสั่งเปิดโปรแกรมใหม่ดู ถ้ายังเกิดปัญหาเดิมอยู่อีกอาจลองตรวจสอบสาเหตุอื่นๆที่เกี่ยวข้องและทำการแก้ไข

5. **Check sum error** เป็นความผิดพลาดที่เกิดจากการที่ CPU รับข้อมูลที่ส่งไปจากคอมพิวเตอร์PC ไม่ครบถูกต้องทั้งหมด ซึ่งปัญหาอาจเกิดจากการกำหนดความเร็วในการสื่อสารBaud rateเร็วเกินไป หรือกำหนดไว้ไม่เหมาะสมกับค่าความถี่ Crystal ค่า 18.432MHz ให้ลองเปลี่ยนค่า Baud rate ให้ช้าลงกว่าเดิม ซึ่งค่าที่เหมาะสมได้แก่ 9600, 19200 และ 38400 แต่ถ้าคอมพิวเตอร์ไม่เร็วมากนักก็อาจกำหนดเป็น 57600 หรือ 115200 ก็ได้ แต่ถ้ากำหนดค่าสูงๆแล้วเกิด Error ควรลดค่า Baud rate ให้ช้าลงกว่าเดิม

6. การสั่ง Load HEX ไม่ได้ เป็นความผิดพลาดที่เกิดจากการที่โปรแกรม FLIP ไม่สามารถอ่านข้อมูลใน HEX File ออกมาได้ ซึ่งอาจเกิดจากไฟล์ที่สั่งโหลดนั้น ไม่ใช่ไฟล์แบบ Intel HEX เนื่องจากโปรแกรม FLIP สามารถใช้งานกับไฟล์แบบ Intel HEX เท่านั้น ส่วนไฟล์ในรูปแบบอื่นๆจะไม่สามารถนำมาใช้งานกับโปรแกรมนี้ได้ ส่วนปัญหาอีกประการหนึ่งที่มักพบอยู่บ่อยๆคือ โปรแกรม FLIP ไม่สามารถอ่าน HEX File ได้ทั้งๆที่ไฟล์ที่สั่งให้อ่านนั้นเป็นไฟล์แบบ Intel HEXอยู่แล้ว ซึ่งที่พบอยู่บ่อยๆก็ได้แก่ HEX File ที่สั่งแปลโดยใช้โปรแกรม Assembler ของSXA51.EXE เนื่องจาก HEX File ที่ได้จากการแปลของโปรแกรมตัวนี้จะเกิดบันทัดว่างอยู่ในไฟล์ในส่วนเริ่มต้มบรรทัดแรกด้วย 1 บรรทัด ซึ่งตามรูปแบบของ HEX File แล้ว ในแต่ละบรรทัดของไฟล์จะต้องเริ่มต้นด้วย

เครื่องหมายโคลอน (☺ แล้วตามด้วยข้อมูลต่างๆในบรรทัดนั้น แต่เมื่อบรรทัดแรกเป็นบรรทัดว่าง โปรแกรมจึงแสดง Error ว่าไม่ใช่ HEX File โดยโปรแกFLIPจะแสดง Error .ให้ทราบดังรูปที่ 4.21

รูปที่ 4.18 Error แสดงว่าไฟล์ที่เลือกไม่ใช่ HEX File

7. **เมื่อสั่งโปรแกรมข้อมูลให้กับ CPU เรียบร้อยแล้วหลังจากรีเซ็ตบอร์ดแล้วไม่ทำงาน** ซึ่งปัญหานี้อาจเกิดจากสาเหตุความผิดพลาดหลายประการ ซึ่งพอสรุปได้ 2 กรณี คือ

- โปรแกรมที่เขียนขึ้นไม่ถูกต้องยังไม่สามารถทำงานได้เอง ซึ่งปัญหานี้ผู้ใช้ต้องหาทางตรวจสอบและแก้ไขความผิดพลาดที่เกิดขึ้นเอง

- ยังไม่ได้มีการสั่ง Load HEX เข้ามารอไว้ยัง Buffer แล้วสั่งโปรแกรม (Program Device) ซึ่งโปรแกรม FLIP จะนำข้อมูลที่อยู่ใน Buffer เขียนไปยังหน่วยความจำของโปรแกรม

- สวิตช์ PSEN อาจถูกกดค้างอยู่ จึงทำให้การรีเซ็ตบอร์ดทุกๆครั้งนั้น CPU จะเข้าไปทำงานใน Monitor Mode เสมอ ซึ่งปัญหานี้สามารถตรวจสอบได้โดยการวัดระดับลอจิกที่ขาสัญญาณ PSEN ของ CPU ซึ่งควรมีสภาวะเป็น “1” ถ้าไม่มีการกดสวิตช์ PSEN ไว้และควรมีสภาวะเป็น “0” ถ้ามีการกดสวิตช์ PSEN ไว้

- สวิตช์ RESET อาจถูกกดค้างอยู่ จึงทำให้ CPU ไม่สามารถหลุดพ้นจากสภาวะการรีเซ็ตได้ ซึ่งปัญหานี้สามารถตรวจสอบได้โดยการวัดระดับลอจิกที่ขาสัญญาณ RESET ของCPU ซึ่งควรมีสภาวะเป็น “0” ถ้าไม่มีการกดสวิตช์ RESET ไว้ และควรมีสภาวะเป็น “1” ถ้ามีการกดสวิตช์ RESET ไว้

- ในกรณีของการ Download ด้วยวิธีการ Auto Download นั้น หลังการ Download เสร็จแล้วควรถอดสาย Download ออกด้วยทุกครั้ง เนื่องจากเมื่อจบการทำงานของโปรแกรมFLIP ไปแล้วสัญญาณ RTS และ DTR ของพอร์ตสื่อสาร RS232 อาจมีสถานะไม่แน่นอนทำให้สัญญาณทั้ง 2 ควบคุมการรีเซ็ตของ MCU อยู่ตลอดเวลาได้

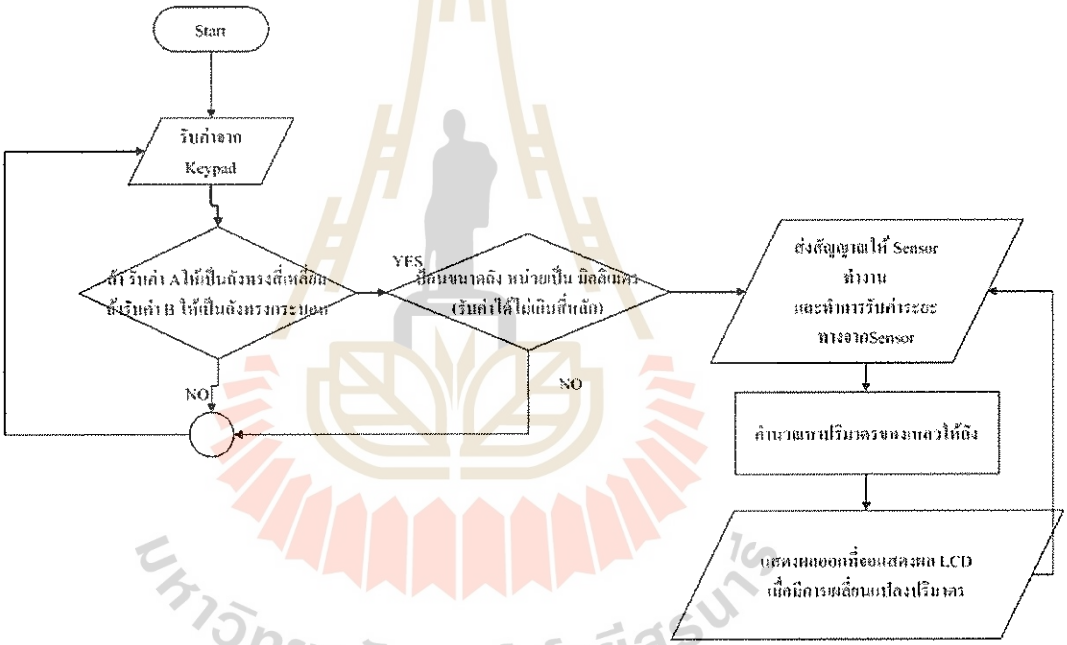
- ค่าของ Device BSB ยังไม่ได้ถูกกำหนดให้มีค่าเป็น 00H ไว้ ซึ่งจะทำให้โปรแกรมกระโดดไปทำงานยังตำแหน่งที่ชี้โดย Device SBV แทน ซึ่งถ้าค่าของ Device SBV ไม่ใช้ศูนย์ก็จเหมือนกับว่าโปรแกรมไม่ทำงาน ซึ่งการแก้ไข ปัญหานี้ หลังจากสั่งโปรแกรมข้อมูลให้กับ CPU เรียบร้อยแล้วควรกำหนดให้ค่าของ Device BSB และ Device SBV มีค่าเป็น 00H ไว้ทั้งคู่จะดีที่สุด

4.3 การเขียนชุดโปรแกรมคำสั่งควบคุมอุปกรณ์[4]

4.3.1 ลำดับความคิดในการเขียนชุดโประแกรมคำสั่งควบคุมอุปกรณ์ (Algorithms)

1. รับค่าจากชุดปุ่มกด เพื่อเลือกรูปแบบของถังน้ำมัน
2. รับค่าจากชุดปุ่มกด เพื่อกำหนดขนาดถังน้ำมันที่ได้เลือกในข้อที่ 1
3. สั่งให้ชุดเซนเซอร์ทำงาน โดยส่งสัญญาณคลื่นสี่เหลี่ยมที่มีความกว้างไม่ต่ำกว่า 10 ไมโครวินาที เพื่อที่หาความระยะห่างจากเซนเซอร์กับผิวของเหลว
4. ประมวณผลจากสูตรทางคณิตศาสตร์ โดยคำนึงถึงรูปแบบถังน้ำมันที่เลือก
5. แสดงผลที่ได้จาดการประมวลออกทางจอแสดงผล (LCD)
6. เมื่อระดับของเหลวในถังเปลี่ยนไป ทำการประมวลผลข้อที่ 4

จะสามรถเขียนเป็นแผนภาพแสดงความคิดในการเขียนชุดโประแกรมคำสั่งควบคุมอุปกรณ์ (Flowchart) ดังรูป 4.19



รูปที่ 4.19 แผนภาพแสดงความคิดในการเขียนชุดโประแกรมคำสั่งควบคุมอุปกรณ์

4.3.2 ชุดโปรแกรมคำสั่งควบคุมเซนเซอร์วัดระยะทางด้วยคลื่นอัลตร้าโซนิค SRF05

```
sbit echo = P3^6;
sbit trigger = P3^7;
void trig_pulse(void)
    { trigger = 1;
    delay_ms(1);
    trigger = 0;}
unsigned int distance()
{   unsigned int mc;
    trigger = 0;
    echo = 1;
            TMOD &= 0x0F;
            TMOD |= 0x10;
            TH1 = 0x00;
            TL1 = 0x00;
            TF1 = 0;
            TR1 = 0;
            trig_pulse();
while(!echo);
            TR1 = 1;
                while(echo);
            TR1 = 0;
            TF1 = 0;
            mc = TH1;
            mc <<=8;
            mc+=TL1;
            delay_ms(10);
            mc=mc*1.1;
return(mc/10);}
```

4.3.3 ชุดโปรแกรมคำสั่งควบคุมจอแสดงผล (LCD)

```
sbit e =P3^4 ;
sbit rs=P3^5 ;
void inti_lcd(void);
void lcd_comm(unsigned char com);
void delay(int z) ;
void lcd_display(unsigned char text);
void lcd_comm(unsigned char com)
    { rs = 0 ;
      e = 1 ;
      P2 =  com ;
      delay(5);
      e = 0 ;
     delay(5);     }
void inti_lcd( )
    { delay(100);
      lcd_comm(0x38) ;
      lcd_comm(0x0C) ;
      lcd_comm(0x01) ;  }
void delay(int z)
    { int x,y ;
      for(x=0;x<z;x++)
      {for(y=0;y<200;y++);}     }
 void lcd_display(unsigned char text)
    {  rs = 1 ;
      e = 1  ;
      P2 = text ;
      delay(5);
      e = 0 ;
      delay(5);   }
```

4.3.4 ชุดโปรแกรมคำสั่งควบคุมการรับค่าจากชุดปุ่มกด (Keypad)

```
unsigned int keypad( )
{   unsigned chardat_scankey[]={0x77,0xb7,0xd7,0xe7,0x7b,0xbb,0xdb,0xeb
    ,0x7d,0xbd,0xdd,0xed,0x7e,0xbe,0xde,0xee};
   unsigned char i ,j , x ,y,z=1,w ;
    while(z)
    { x=0xf7;
      for(i=0;i<4;i++)
      {P1 = x ;
       for(j=0;j<16;j++)
         {y = P1 ;
          if(y==dat_scankey[j])
           {delay(100);
            if(y==dat_scankey[j])
              {z=0;
            w=j;} } }
              x = x>>1;
              x =x|0xf0; }
   }return (w);   }
```

4.3.5 ชุดโปรแกรมคำสั่งควบคุมการคำนวณหาปริมาตร

```
#include <reg52.h>
#include <stdio.h>
#include <lcd.h>
#include <keypad.h>
#include <math.h>
#include <ult.h>
void show_error()
{       unsigned int i,x=0x82;
        unsigned char error[]="ERROR";
        inti_lcd();
        lcd_comm(0x80);
        for(i=0;i<5;i++)
        {lcd_display(error[i]);}
}
void show_length()
{       unsigned int i;
        unsigned char length1[]="Length";
        unsigned char num3[]="mm.";
        inti_lcd();
        lcd_comm(0x80);
        for(i=0;i<6;i++)
                {lcd_display(length1[i]);}
                lcd_comm(0xc4);
                for(i=0;i<3;i++)
                        {lcd_display(num3[i]);}
}
void show_width()
{       int i;
        unsigned char width1 []="Width";
```

```
        unsigned char num3[]="mm.";
        inti_lcd();
        lcd_comm(0x80);
        for(i=0;i<5;i++)
                {lcd_display(width1[i]);}
                 lcd_comm(0xc4);
                 for(i=0;i<3;i++)
                        {lcd_display(num3[i]);}
}
void show_high()
{       int i;
        unsigned char high1 []="High";
        unsigned char num3[]="mm.";
        inti_lcd();
        lcd_comm(0x80);
        for(i=0;i<4;i++)
                {lcd_display(high1[i]);}
                 lcd_comm(0xc4);
                 for(i=0;i<3;i++)
                        {lcd_display(num3[i]);}
}
void show_radiance()
{       int i;
        unsigned char radiance1[]="Radiance";
        unsigned char num3[]="mm.";
                inti_lcd();
                lcd_comm(0x80);
                for(i=0;i<8;i++)
                        {lcd_display(radiance1[i]);}
                        lcd_comm(0xc4);
```

```
                for(i=0;i<3;i++)
                        {lcd_display(num3[i]);}
}
unsigned int key_number(unsigned int n)
{       unsigned int key=0,q=0,x=0xc0,number=0;
        unsigned char number1[]="0123456789ABCDEF ";
                while(q<4)
                { key=keypad();
                  if((key>=0)&&(key<=9))
                        {lcd_comm(x);
                         lcd_display(number1[key]);
                         delay_ms(5);
                                if(q==0)
                        {number=key;}
                else {number=(number*10)+key;}
                        x++;  q++;
              }else if(key==15)
                   {    q=4;
                }else
                { show_error();
                  delay_ms(10);
                  if (n==1)
                {show_length();}
                else if (n==2)
                {show_width();}
                else if (n==3)
                {show_high();}
                else if (n==4)
                {show_radiance();}
```

```
            number=0;
            q=0;
            x=0xc0;}
            }return(number);
}
void show_volume(unsigned long value)
{           unsigned int i;
            unsigned char number[]="0123456789ABCDEF ";
            unsigned char num3[]="Lt.";
            unsigned char vol[]="VOL.";
            unsigned char a,b,c,d,e,f,g,h,k;
a=b=c=d=e=f=g=0;
                        k =(value%1000000000)/100000000;
                        h =(value%100000000)/10000000;
                        g =(value%10000000)/1000000;
                        f = (value%1000000)/100000;
                        e = (value%100000)/10000;
                        d = (value%10000)/1000;
                        c = (value%1000)/100;
                        b = (value%100)/10;
                        a = value%10;
if(k==0)
            {k=16;
if(h==0)
            {h=16;
if(g==0)
            {g=16;
if(f==0)
            {f=16;
```

```
if(e==0)
        {e=16;
if(d==0)
        {d=16;
                                        }
                                    }
                                }
                            }
                        }
                    }
        inti_lcd();
        lcd_comm(0x80);
        for(i=0;i<3;i++)
{lcd_display(vol[i]);}
 lcd_comm(0xC4);
 lcd_display(number[a]);
 lcd_comm(0xC3);
 lcd_display(number[b]);
 lcd_comm(0xC2);
 lcd_display(vol[3]);
 lcd_comm(0xC1);
 lcd_display(number[c]);
 lcd_comm(0xC0);
 lcd_display(number[d]);
 lcd_comm(0x87);
 lcd_display(number[e]);
 lcd_comm(0x86);
 lcd_display(number[f]);
 lcd_comm(0x85);
 lcd_display(number[g]);
```

```
 lcd_comm(0x84);
 lcd_display(number[h]);
 lcd_comm(0x83);
 lcd_display(number[k]);
 lcd_comm(0xC5);
        for(i=0;i<3;i++)
        {lcd_display(num3[i]);}
}
void cubic()
{        unsigned int length=0,width=0,high=0;
        unsigned long value,old_value;
 //*Recieve Length
                show_length();
                length= key_number(1) ;
 //*Recieve Width
                show_width();
                width =key_number(2) ;
//*recieve High
                show_high();
                high =key_number(3) ;
//*Calculate Volume
                while(1)
                        {       value=distance();
                                value=high-value;
                                value=(value*length*width)/10000;
        if(value != old_value)
        {   show_volume(value);
            delay_ms(20);
            old_value = value;
        }
```

```
        }
        }
void cylinder()
        {       float z;
                unsigned int radiance=0,high=0;
                unsigned long value,old_value;
        //Recieve Radiance
                 show_radiance();
                 radiance= key_number(4) ;
        //recieve High
                 show_high();
                 high =key_number(3) ;
        //Calculate Volume
        while(1){       value=distance();
                        value=high-value;
                        z=(radiance*radiance*value)/10000;
                                value=z*22/7;
                if(value != old_value)
         {      show_volume(value);
                delay_ms(20);
                old_value = value;
          }
                }
        }
        void main()
        {       unsigned int key,i;
                unsigned char start[]="Enter Type";
                while(1){
                        inti_lcd();
                        lcd_comm(0x80);
```

```
                for(i=0;i<8;i++)
                        {lcd_display(start[i]);}
                lcd_comm(0xc0);
                for(i=8;i<9;i++)
                        {lcd_display(start[i]);}
key=keypad();
                if(key==12)
                {       while(1)
                        {cubic();}
                }
                else if(key==13)
                {
                        while(1)
                        {cylinder();}
                }
                }
                else
                {show_error();
                 delay_ms(10);
                }
        }
}
```

4.4 การทดสอบเครื่องวัดระดับน้ำมันโดยอาศัยคลื่นอัลตร้าโซนิคในถังรูปแบบต่างๆ

เป็นการทดสอบเพื่อวัดปริมาตรระดับน้ำมันที่ทำการทดลองในถังทรงสี่เหลี่ยม และถังรูปทรงกระบอก เพื่อทดสอบประสิทธิภาพเครื่องวัดระดับน้ำมันโดยอาศัยคลื่นอัลตร้าโซนิค มีขั้นตอนดังนี้

4.4.1 ต่อชุดอุปกรณ์ต่างๆ

โดยเตรียมชุดอุปกรณ์ให้พร้อมต่อการใช้งาน

4.4.2 ป้อนรูปแบบถังบรรจุน้ำมันและขนาดถังบรรจุน้ำมัน

โดยเลือกรูปแบบถังจากปุ่มที่กำหนด โดยที่ A คือถังรูปทรงสี่เหลี่ยม และ B คือถังรูปทรงกระบอก และขนาดของถังบรรจุน้ำมันที่ป้อนเข้าไปต้องมีหน่วยเป็น มิลลิเมตร

โดยที่ถังรูปทรงสี่เหลี่ยมต้องป้อนค่าขนาดถัง 3ด้านคือ กว้าง ยาว และสูง

และถังรูปทรงกระบอกต้องป้อน 2ค่าคือ รัศมีปากท่อ และสูง

ได้ผลการทดลองตามตารางที่ 4.1 และตารางที่ 4.2 และการทดลองกับน้ำแทนการทดลองกับน้ำมันในตารางที่ 4.3

ตารางที่ 4.1 แสดงผลการทดสอบเครื่องวัดระดับน้ำมันโดยอาศัยคลื่นอัลตร้าโซนิคในถังรูปทรงสี่เหลี่ยม

ปริมาตรน้ำมันจริงที่เติมลงถังทดลอง(ลิตร)	ปริมาตรครั้งที่ 1 (ลิตร)	ปริมาตรครั้งที่ 2 (ลิตร)	ปริมาตรครั้งที่ 3 (ลิตร)	ค่าเฉลี่ย (ลิตร)	ค่าความผิดพลาด (%error)
0.50	0.48	0.49	0.48	0.48	4.00
1.00	0.99	1.00	0.98	0.99	1.00
1.50	1.47	1.49	1.47	1.47	1.47
2.00	2.00	2.00	1.98	1.99	0.50
2.50	2.46	2.48	2.46	2.47	1.20
3.00	2.96	2.96	2.96	2.96	1.30
3.50	3.47	3.43	3.47	3.46	1.10
4.00	3.97	3.97	3.97	3.97	0.75
4.50	4.43	4.47	4.47	4.46	0.80
5.00	4.96	4.96	4.96	4.96	0.80
5.50	5.43	5.47	5.45	5.45	0.91
6.00	5.91	5.97	5.95	5.94	1.00

6.50	6.43	6.43	6.43	6.45	0.76
7.00	6.95	6.95	6.95	6.95	0.71
7.50	7.48	7.45	7.47	7.47	0.47
8.00	7.97	7.98	8.00	7.98	0.25

ค่าความผิดพลาดเฉลี่ย = 1.09 %

ตารางที่ 4.2แสดงผลการทดสอบเครื่องวัดระดับน้ำมันโดยอาศัยคลื่นอัลตร้าโซนิคในถังรูปทรงกระบอก

ปริมาตรน้ำมันจริงที่เติมลงถังทดลอง(ลิตร)	ปริมาตรครั้งที่ 1 (ลิตร)	ปริมาตรครั้งที่ 2 (ลิตร)	ปริมาตรครั้งที่ 3 (ลิตร)	ค่าเฉลี่ย (ลิตร)	ค่าความผิดพลาด (%error)
0.50	0.50	0.50	0.48	0.49	2.00
1.00	0.99	0.98	0.98	0.98	2.00
1.50	1.47	1.47	1.47	1.47	2.00
2.00	1.98	2.00	1.98	1.98	1.00
2.50	2.48	2.47	2.47	2.47	1.20
3.00	2.98	2.98	2.97	2.98	0.67
3.50	3.46	3.47	3.48	3.47	0.85
4.00	3.95	3.98	3.97	3.97	0.75
4.50	4.41	4.46	4.49	4.45	1.11
5.00	4.98	4.98	4.98	4.98	0.40
5.50	5.47	5.49	5.48	5.48	0.36
6.00	5.95	5.98	5.98	5.97	0.50
6.50	6.47	6.49	6.47	6.48	0.36
7.00	6.97	6.98	6.98	6.98	0.28

ค่าความผิดพลาดเฉลี่ย = 0.96 %

ตารางที่ 4.3แสดงผลการทดสอบเครื่องวัดระดับน้ำมันโดยอาศัยคลื่นอัลตร้าโซนิคในถังรูปทรงสี่เหลี่ยม โดยใช้วัดน้ำแทนการวัดน้ำมัน

ปริมาตรน้ำมันจริงที่เติมลงถังทดลอง(ลิตร)	ปริมาตรครั้งที่ 1 (ลิตร)	ปริมาตรครั้งที่ 2 (ลิตร)	ปริมาตรครั้งที่ 3 (ลิตร)	ค่าเฉลี่ย (ลิตร)	ค่าความผิดพลาด (%error)
0.50	0.50	0.50	0.48	0.49	2.00
1.00	0.99	0.98	0.98	0.98	1.00
1.50	1.47	1.47	1.47	1.47	0.67
2.00	1.98	1.97	1.98	1.98	1.00
2.50	2.48	2.47	2.47	2.47	1.20
3.00	2.98	2.98	2.97	2.98	1.00
3.50	3.46	3.47	3.48	3.47	0.86
4.00	3.95	3.98	3.97	3.97	0.50
4.50	4.41	4.46	4.49	4.45	0.44
5.00	4.98	4.98	4.98	4.98	0.80
5.50	5.47	5.49	5.48	5.48	0.91
6.00	5.95	5.98	5.98	5.97	0.17
6.50	6.47	6.79	6.47	6.48	0.15
7.00	6.97	6.98	6.98	6.98	0.29
7.50	7.48	7.48	7.49	7.48	0.27
8.00	7.98	7.99	8.00	7.99	0.12
8.50	8.47	8.51	8.50	8.49	0.12
9.00	8.95	9.00	9.00	8.98	0.22
9.50	9.48	9.50	9.49	9.49	0.11
10.00	9.99	10.00	9.99	9.99	0.10

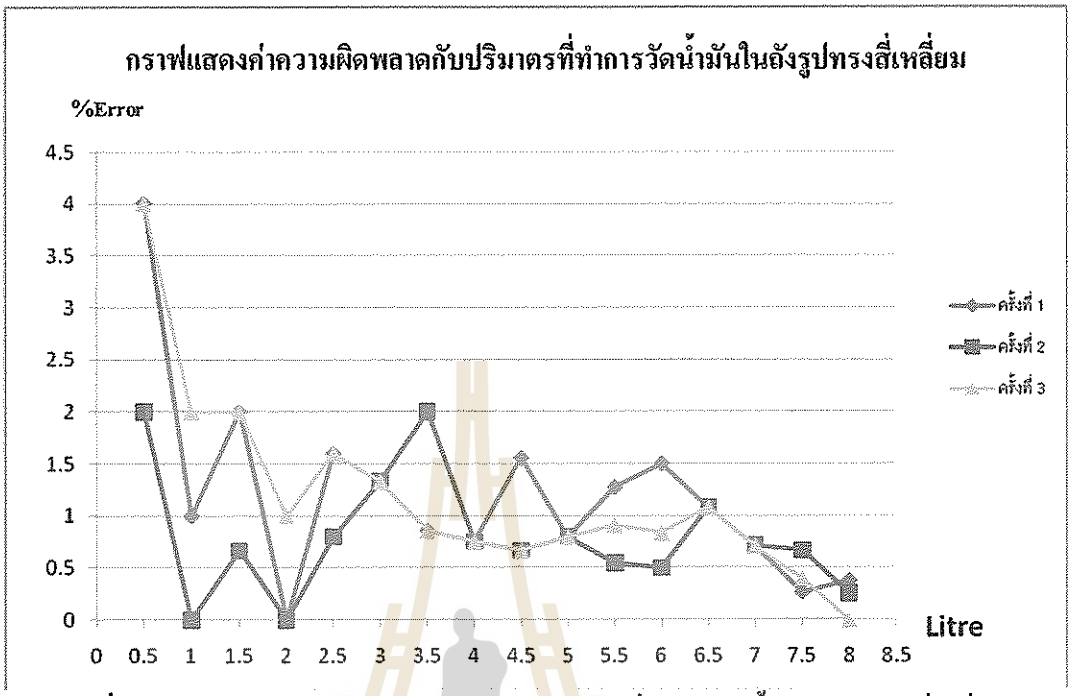
ค่าเฉลี่ยค่าความผิดพลาดรวม = 0.59 %

ตารางที่ 4.4 แสดงผลการทดสอบเครื่องวัดระดับน้ำมันโดยอาศัยคลื่นอัลตร้าโซนิคในถังรูปทรงกระบอก โดยใช้วัดน้ำแทนการวัดน้ำมัน

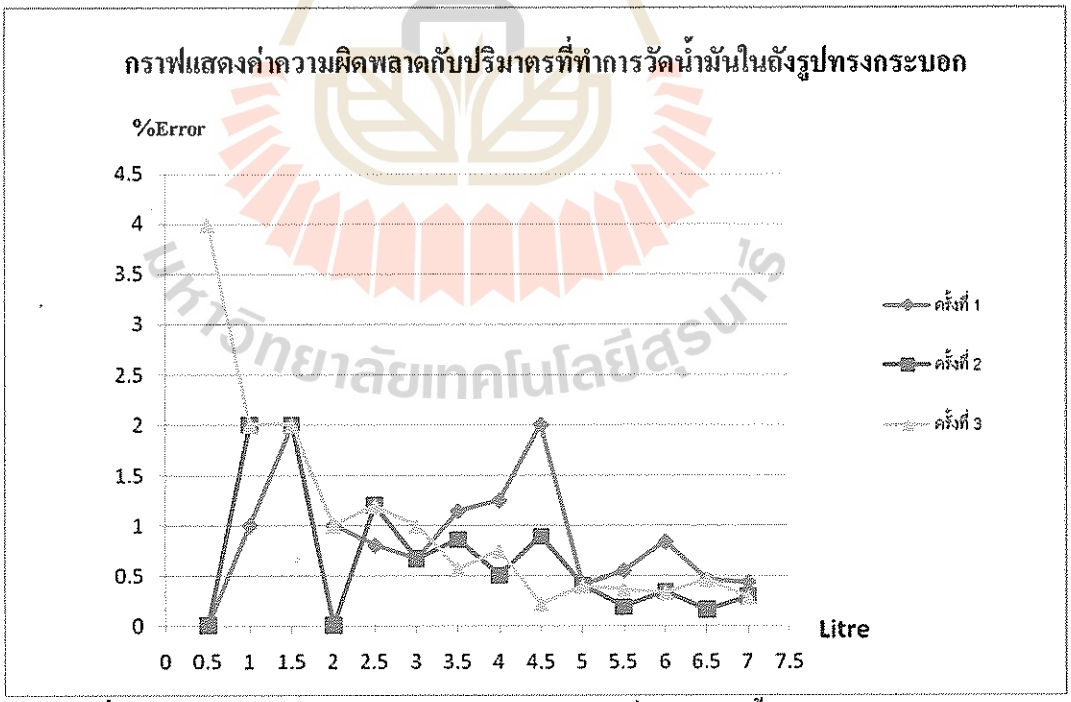
ปริมาตรน้ำมันจริงที่เติมลงถังทดลอง(ลิตร)	ปริมาตรครั้งที่ 1 (ลิตร)	ปริมาตรครั้งที่ 2 (ลิตร)	ปริมาตรครั้งที่ 3 (ลิตร)	ค่าเฉลี่ย (ลิตร)	ค่าความผิดพลาด (%error)
0.50	0.50	0.50	0.50	0.50	0.00
1.00	1.00	1.00	1.00	1.00	0.00
1.50	1.50	1.50	1.49	1.49	0.67
2.00	2.01	2.01	2.00	2.00	0.00
2.50	2.48	2.51	2.50	2.49	0.40
3.00	2.96	2.98	3.00	2.98	0.67
3.50	3.48	3.48	3.49	3.48	0.57
4.00	3.97	4.00	4.00	3.99	0.25
4.50	4.46	4.51	4.50	4.49	0.22
5.00	4.96	5.02	5.00	4.99	0.20
5.50	5.46	5.50	5.50	5.49	0.18
6.00	6.01	5.99	6.00	6.00	0.00
6.50	6.50	6.50	6.50	6.50	0.00
7.00	7.00	7.01	7.00	7.00	0.00

ค่าเฉลี่ยค่าความผิดพลาดรวม = 0.23 %

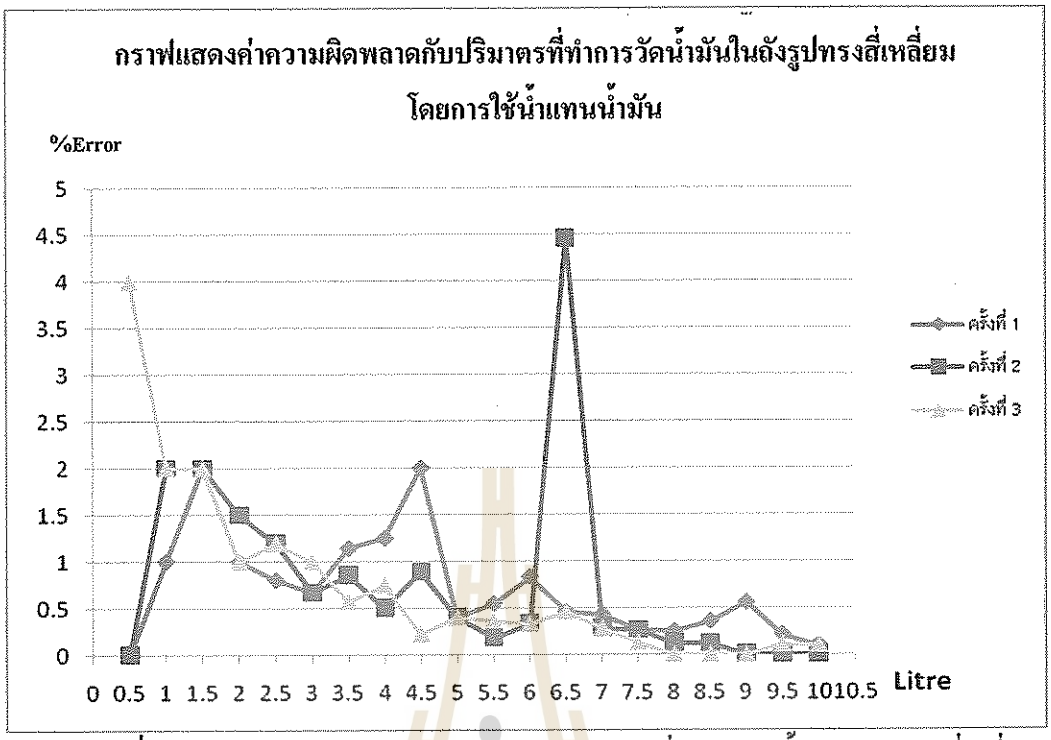
จากผลการทดลอง นำค่าที่ได้มาวาดกราฟระหว่างค่าความผิดพลาดกับปริมาตรที่ทำการวัดในถังทรงสี่เหลี่ยมและถังทรงกระบอก ดังรูปที่ 4.19และรูปที่ 4.20 ตามลำดับดังนี้



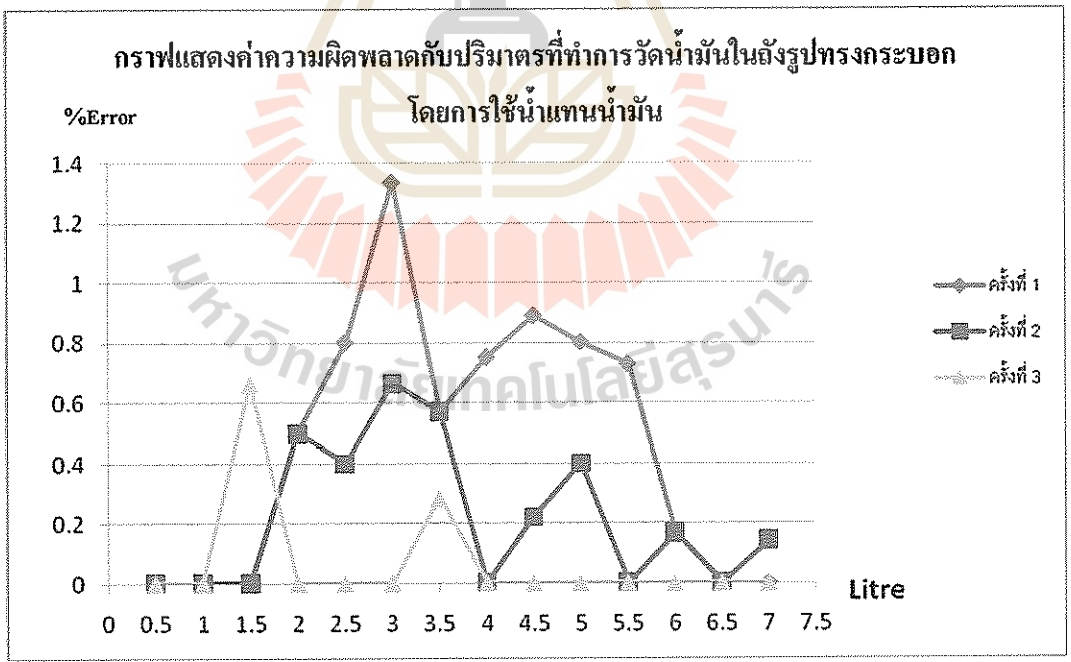
รูปที่ 4.19 กราฟระหว่างค่าความผิดพลาดกับปริมาตรที่ทำการวัดน้ำมันในถังทรงสี่เหลี่ยม



รูปที่ 4.20 กราฟระหว่างค่าความผิดพลาดกับปริมาตรที่ทำการวัดน้ำมันในถังทรงกระบอก



รูปที่ 4.21 กราฟระหว่างค่าความผิดพลาดกับปริมาตรที่ทำการวัดน้ำมันในถังทรงสี่เหลี่ยม โดยการใช้น้ำแทนน้ำมัน



รูปที่ 4.22 กราฟระหว่างค่าความผิดพลาดกับปริมาตรที่ทำการวัด ในถังทรงกระบอก โดยการใช้น้ำแทนน้ำมัน

4.5 สรุปการทดลองเครื่องวัดระดับน้ำมันโดยอาศัยคลื่นอัลตร้าโซนิค

เครื่องวัดระดับน้ำมันโดยอาศัยคลื่นอัลตร้าโซนิคที่ออกแบบและสร้างขึ้นมานี้ สามารถวัดระดับน้ำมันโดยอาศัยคลื่นอัลตร้าโซนิคที่ส่งคลื่นกระทบผิวของน้ำมันที่คงอยู่ในถัง 2รูปแบบคือ รูปทรงสี่เหลี่ยมและรูปทรงกระบอก แล้วทำการประมวลผลทางไมโครคอนโทรเลอร์ จากการทดสอบซ้ำหลายๆครั้งในถังทั้ง 2รูปแบบ พบว่าในแต่ละครั้งที่ทดสอบจะมีผลที่ใกล้เคียงกันของแต่ละปริมาตรที่ได้ทำการทดสอบ ดังนั้นจึงพบว่าเครื่องวัดระดับน้ำมันโดยอาศัยคลื่นอัลตร้าโซนิคนี้ค่อนข้างที่จะมีความเที่ยงตรงในการวัดปริมาตร สามารถนำมาใช้งานในการวัดระดับน้ำมันในถังทั้ง 2 รูปแบบคือ ถังรูปทรงสี่เหลี่ยมและถังรูปทรงกระบอกได้

บทที่ 5
บทสรุป และข้อเสนอแนะ

5.1 กล่าวนำ

ในบทนี้จะกล่าวถึงระบบรวมของเครื่องวัดระดับน้ำมันโดยอาศัยคลื่นอัลตร้าโซนิค โดยได้อธิบายปัญหาที่พบในระหว่างการทำโครงงาน วิธีแก้ปัญหา ข้อเสนอแนะ แนวทางการพัฒนาต่อไป และบทสรุปของโครงงานที่จัดทำขึ้น

5.2 ปัญหาที่พบในระหว่างการทำโครงงานและวิธีแก้ปัญหา

ตารางที่ 5.1 แสดงรายละเอียดของปัญหาที่พบ และวิธีแก้ปัญหาของโครงงาน

ปัญหาที่พบ	สาเหตุและแนวทางแก้ไข
1. ปริมาตรน้ำมันที่วัดได้ไม่ตรงกับปริมาตรน้ำมันที่ตวงใส่ลงไปขณะทำการทดลอง	สาเหตุ เนื่องจากขณะที่ทำการทดลองได้มีการสูญเสียน้ำไปบางส่วนเนื่องจากการรวมตัวของน้ำแล้วไปเกาะตัวที่ผนังถ้วยตวงและผนังถังบรรจุน้ำมัน แนวทางแก้ไข พยายามเทน้ำมันลงถังที่ทำการทดลองอย่างระมัดระวัง เพื่อป้องกันน้ำมันเกาะข้างผิวผนังถังน้ำมัน
2. ลายวงจรที่กัดแผ่น Print มีขนาดเล็กมาก	สาเหตุ เนื่องจากลายวงจรที่ใช้ในการทำตัวเชื่อมต่อ USB มีความละเอียดสูง ลายของวงจรจึงมีขนาดเล็กมาก แนวทางแก้ไข ในการออกแบบลายวงจรบนแผ่นวงจรพิมพ์ จะใช้โปรแกรม Protel99SE ที่สามารถทำให้ลายวงจรมีขนาดเล็กและคมตามที่ต้องการ
3. อุปกรณ์เกิดการชำรุดและเสียบ่อย	สาเหตุ เนื่องจากอุปกรณ์อิเล็คทรอนิกส์มีความบอบบางและบางตัวต้องมีการบัดกรี และขา IC สั้นมากทำให้ยากต่อการบัดกรี แล้วการทดลองต้องถอดเข้า-ออกบ่อยจึงเกิดการชำรุดได้ง่าย แนวทางแก้ไข ถ้า IC ตัวใดมีขาที่สั้น จะใช้การเป่าลมร้อนในการบัดกรีแทนการใช้หัวแร้ง และไม่ถอดอุปกรณ์เข้า-ออกบ่อย

5.3 ข้อเสนอแนะ

5.3.1 ลายวงจรบนแผ่นวงจรพิมพ์ที่ออกแบบเสร็จแล้ว ควรที่จะใช้เครื่อง Network วัดเพื่อจะดูว่าอินพุตและเอาท์พุต Matching ที่ 50 โอห์ม หรือไม่

5.3.2 เนื่องจาก IC มีขนาดเล็ก ควรที่จะระมัดระวังในการบัดกรี เพราะเวลาบัดกรีขาของ IC อาจติดกันทำให้เกิดการ Short Circuit ได้ และควรระมัดระวังเรื่องความร้อนในการบัดกรีด้วย เช่น ไม่ควรจี้ขา IC เป็นเวลานานๆ เพราะอาจทำให้ IC พังได้

5.3.3 เนื่องจากเกิดการรวมตัวของน้ำมัน และเกาะตามผนังของถังน้ำมัน ทำให้ปริมาตรเวลาทำการวัดคลาดเคลื่อนได้ จึงควรใช้สารเคลือบผิวภาชนะเพื่อป้องกันการเกาะตัวของน้ำมันที่ผนังของถังน้ำมัน

5.4 แนวทางการพัฒนาต่อไป

5.4.1 ปรับปรุงชุดอุปกรณ์อัลตร้าโซนิคที่สามารถส่ง-รับคลื่นอัลต้าโซนิค ที่มีความสูงมากกว่า 4 เมตรและมีความละเอียดสูงได้

5.4.2 ปรับปรุงชุดกล่องอุปกรณ์เครื่องวัดระดับน้ำมันโดยอาศัยคลื่นอัลตร้าโซนิค ให้สามารถใช้งานโดยไม่ใช้ Adapter เป็นตัวจ่ายแรงดันให้เครื่องทำงานได้ โดยอาจจะใช้ถ่านแบต-เตอรี่แทน

5.4.3 ปรับปรุงชุดอุปกรณ์ที่ใช้ในการส่งข้อมูลจากตัวเซนเซอร์ที่อยู่ถังน้ำมัน ไปยังจอแสดงผลระดับน้ำมันจากที่ใช้สาย เป็นไร้สาย

5.5 บทสรุป

ระบบรวมของเครื่องวัดปริมาตรน้ำมันโดยอาศัยคลื่นอัลตร้าโซนิคประกอบไปด้วย

- การวัดปริมาตรน้ำมันโดยอาศัยคลื่นอัลตร้าโซนิคในถังรูปทรงสี่เหลี่ยม
- การวัดปริมาตรน้ำมันโดยอาศัยคลื่นอัลตร้าโซนิคในถังรูปทรงกระบอก

ก่อนที่จำทำการต่อชุดอุปกรณ์และวงจรต่างๆในระบบ จะทำการทดสอบชุดอุปกรณ์และวงจรข้างต้น ซึ่งได้ผลตามบทที่ 3 และบทที่ 4

จากการทดสอบเมื่อทำการต่อชุดอุปกรณ์ต่างๆในระบบรวมทั้งหมด แล้วทำการทดลองการววัดระดับน้ำมันโดยอาศัยคลื่นอัลตร้าโซนิค สามารถวัดระดับน้ำมันโดยอาศัยคลื่นอัลตร้าโซ-นิคที่ส่งคลื่นกระทบผิวสูงสุดของระดับน้ำมันที่คงอยู่ในถัง 2 รูปแบบคือ รูปทรงสี่เหลี่ยมและรูปทรงกระบอก แล้วทำการประมวลผลทางไมโครคอนโทรเลอร์ เมื่อทำการทดสอบได้ทำการทดสอบซ้ำหลายๆรอบในถังทั้ง 2 รูปแบบ พบว่าในแต่ละครั้งที่ได้ทำการทดสอบ และเมื่อผลการทดสอบออกมามีผลที่คล้ายคลึงกันในแต่ละค่าที่ได้ทำการทดสอบในแต่ละครั้ง แสดงว่าเครื่องวัด

ระดับน้ำมันโดยอาศัยคลื่นอัลตร้าโซนิคนี้มีความเที่ยงตรงและแม่นยำ สามารถนำมาใช้งานได้จริง ในการวัดระดับน้ำมันในถังทั้ง 2 รูปแบบคือ ถังทรงสี่เหลี่ยมและถังทรงกระบอกได้

บรรณานุกรม

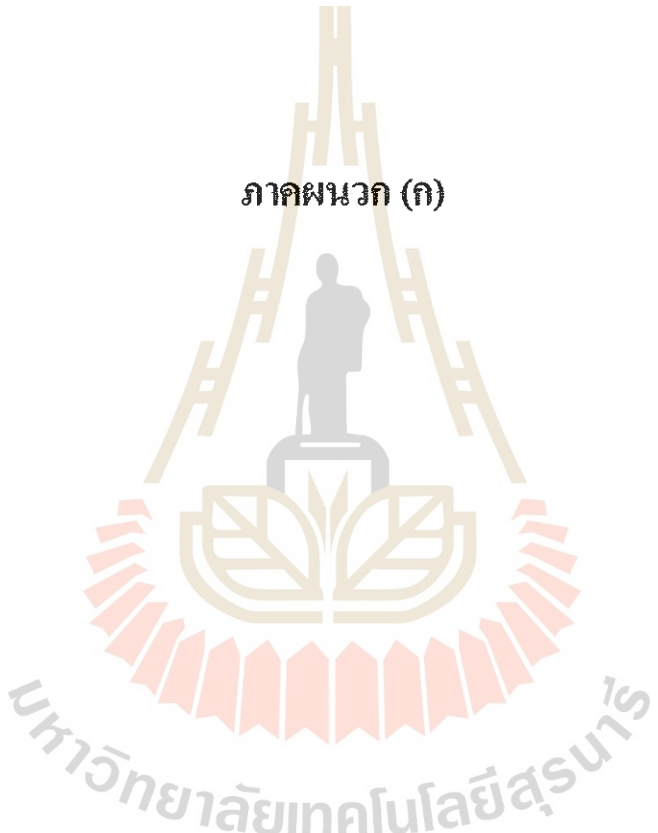
[1] สันติ นุราชและคณะ. , **"เรียนรู้ไมโครคอนโทรลเลอร์ MCS-51 ฉบับภาษา C"**, Micro Research Technology. Ltd, Part, ปทุมธานี.

[2] http://fivedots.coe.psu.ac.th/Software.coe/LAB/Lab_y3/ULTRASO.DOC

[3] http://www.ftdichip.com/Documents/DataSheets.htm

[4] ขจร อนุดิตย์ **"การเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ MCS-51 ด้วยภาษาซี."** พิมพ์ครั้งที่ 1, เพียร์สัน เอ็ดดูเคชั่น อินโดไชน่า, กรุงเทพฯ: 2547

ภาคผนวก (ก)



Datasheet

SRF05

Ultrasonic Distance Detector Module

โมดูลตรวจจับและวัดระยะทางด้วยอัลตร้าโซนิก

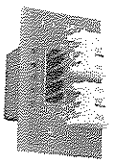
Distributed by Innovative Experiment Co.,Ltd., Thailand

คุณสมบัติ

- ใช้ไฟเลี้ยง +5V ต้องการกระแสไฟฟ้า 30mA
- ใช้ตัวรับและส่งคลื่นอัลตร้าโซนิก ใช้ความถี่ 40kHz ในการทำงาน
- วัดระยะทางในช่วง 1 เซนติเมตรถึง 4 เมตร
- สัญญาณพัลส์สำหรับกระตุ้นการทำงาน ต้องมีความกว้างอย่างน้อย 10 ไมโครวินาที
- ให้ผลลัพธ์จากการวัดระยะเป็นค่าความกว้างพัลส์ซึ่งเป็นสัดส่วนกับระยะทางที่วัดได้
- มีขนาดเล็กคือ 43 มม. x 20 มม. x 17 มม. (กว้างxยาวxสูง)
- สื่อสารกับไมโครคอนโทรลเลอร์ยอดนิยมได้ทุกตระกูล อาทิ เบสิกแสตมป์ 2SX/2P, PIC, MCS-51, PSoC, 68HC11
- สามารถติดต่อได้ 2 แบบคือ แบบ 2 สัญญาณ (Echo กับ Trigger) เหมือนกับ SRF04 และแบบอนุกรมสัญญาณเส้นเดียว
- สามารถใช้ทดแทน SRF04 ได้

อุปกรณ์เสริม

- บอร์ด ADX-SRF04 ซึ่งเป็นบอร์ดอะแดปเตอร์สำหรับอำนวยสะดวกในการเชื่อมต่อกับบอร์ดไมโครคอนโทรลเลอร์และแผงต่อวงจรหรือเบรดบอร์ด



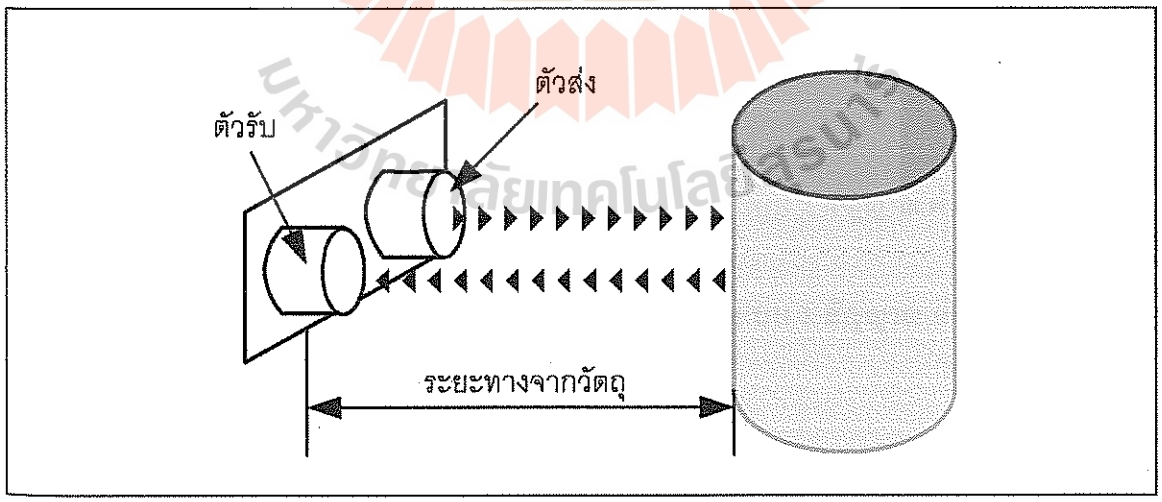
- สาย PCB3A สำหรับเชื่อมต่อกับบอร์ดควบคุมหุ่นยนต์ของ i-nex

SRF05 เป็นแผงวงจรวัดตรวจจับและวัดระยะทางด้วยคลื่นอัลตร้าโซนิกที่มีความเที่ยงตรงสูง โดยสามารถวัดระยะได้ตั้งแต่ 1 เซนติเมตรไปจนถึง 4 เมตร SRF04 ถูกออกแบบมาให้ใช้งานกับไมโครคอนโทรลเลอร์ได้ง่ายโดยใช้ขาเชื่อมต่อเพียง 1 หรือ 2 ขา ขึ้นอยู่กับการกำหนดรูปแบบการทำงานทางฮาร์ดแวร์ เหมาะอย่างยิ่งกับการประยุกต์ใช้งานด้านหุ่นยนต์

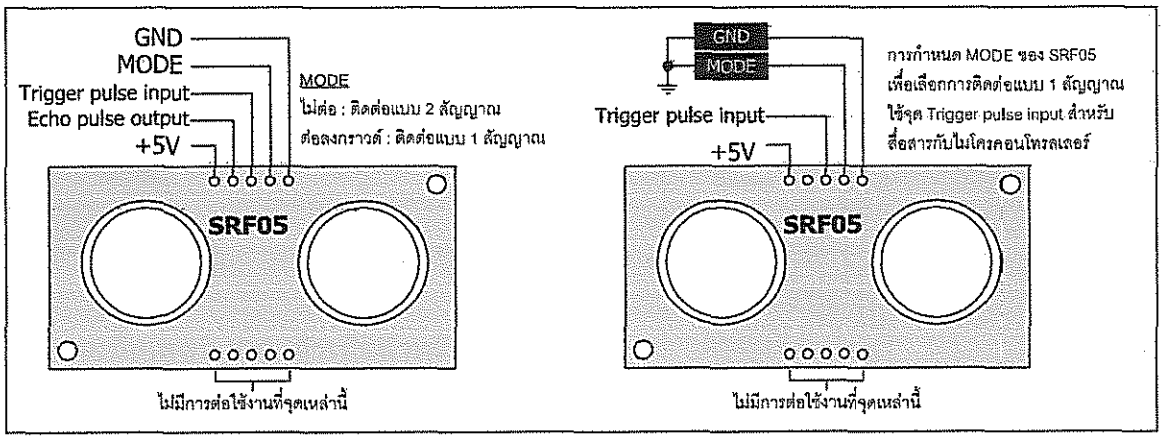
SRF05 จะทำการส่งสัญญาณคลื่นอัลตร้าโซนิกออกไป แล้ววัดระยะเวลาที่มีสัญญาณสะท้อนตอบกลับมา เอาต์พุตที่ได้จาก SRF05 จะอยู่ในรูปของความกว้างพัลส์ซึ่งสัมพันธ์กับระยะทางของวัตถุที่ตรวจจับได้ ความถี่สัญญาณอัลตร้าโซนิกของ SRF05 คือ 40kHz ถูกส่งออกไปในอากาศด้วยความเร็ว 1.125 ฟุตต่อมิลลิวินาที (ประมาณ 346 เมตรต่อวินาที) ดังนั้นเมื่อทราบความเร็วในการเคลื่อนที่ของคลื่น, เวลาเริ่มต้นส่งคลื่น และเวลาที่รับเสียงสะท้อนกลับมา จึงสามารถคำนวณหาค่าของระยะทางได้ ดังแสดงหลักการตรวจจับในรูปที่ 1

ระยะทางที่ได้นั้นจะต้องมีการคำนวณค่ากลับทางคณิตศาสตร์ เมื่อใช้กับไมโครคอนโทรลเลอร์แล้วถือว่าเป็นเรื่องยุ่งยากพอสมควร ดังนั้น SRF05 จึงประมวลผลค่าทางคณิตศาสตร์ต่าง ๆ เหล่านี้ไว้เรียบร้อยแล้ว จากนั้นส่งผลลัพธ์ที่วัดได้ออกมาเป็นพัลส์ที่มีความกว้างสัมพันธ์กับระยะทางที่วัดได้

การส่งผลลัพธ์ที่วัดได้ออกมาเป็นในเชิงความกว้างของสัญญาณพัลส์อาจจะดูว่ายุ่งยากกว่าการส่งเป็นข้อมูลดิจิตอลออกมา แต่การส่งออกมาเป็นข้อมูลดิจิตอลอาจต้องใช้สายสัญญาณจำนวนมาก ซึ่งทำให้ต้องใช้ขาพอร์ตในการเชื่อมต่อเป็นจำนวนมากตามไปด้วย ดังนั้นหากส่งผลลัพธ์ออกมาในรูปของสัญญาณพัลส์ จะใช้สายสัญญาณเพียงเส้นเดียว จึงทำให้สะดวกมากในการนำมาเชื่อมต่อกับไมโครคอนโทรลเลอร์



รูปที่ 1 แสดงหลักการตรวจจับวัตถุโดยใช้สัญญาณความถี่เหนือเสียงหรืออัลตร้าโซนิก



รูปที่ 2 แสดงขาสัญญาณของ SRF05 และการกำหนดโหมดทำงาน

1. จุดต่อใช้งานของ SRF05

มีจุดต่อสำหรับใช้งานอยู่ทั้งหมด 5 จุดตามรูปที่ 2

1. **ขาไฟเลี้ยง (+5V)** สำหรับต่อไฟเลี้ยงแรงดัน +5V

2. **ขา Echo Pulse Output (ECHO)** เป็นขาเอาต์พุตสำหรับส่งสัญญาณพัลส์ออกจาก SFR05 ซึ่งการใช้งานจะนำขานี้ไปต่อเข้ากับพอร์ตอินพุตของไมโครคอนโทรลเลอร์ เพื่อตรวจจับความกว้างของสัญญาณพัลส์ที่ส่งออกมาเพื่อแปลความหมายออกมาเป็นระยะทางอีกครั้งหนึ่ง

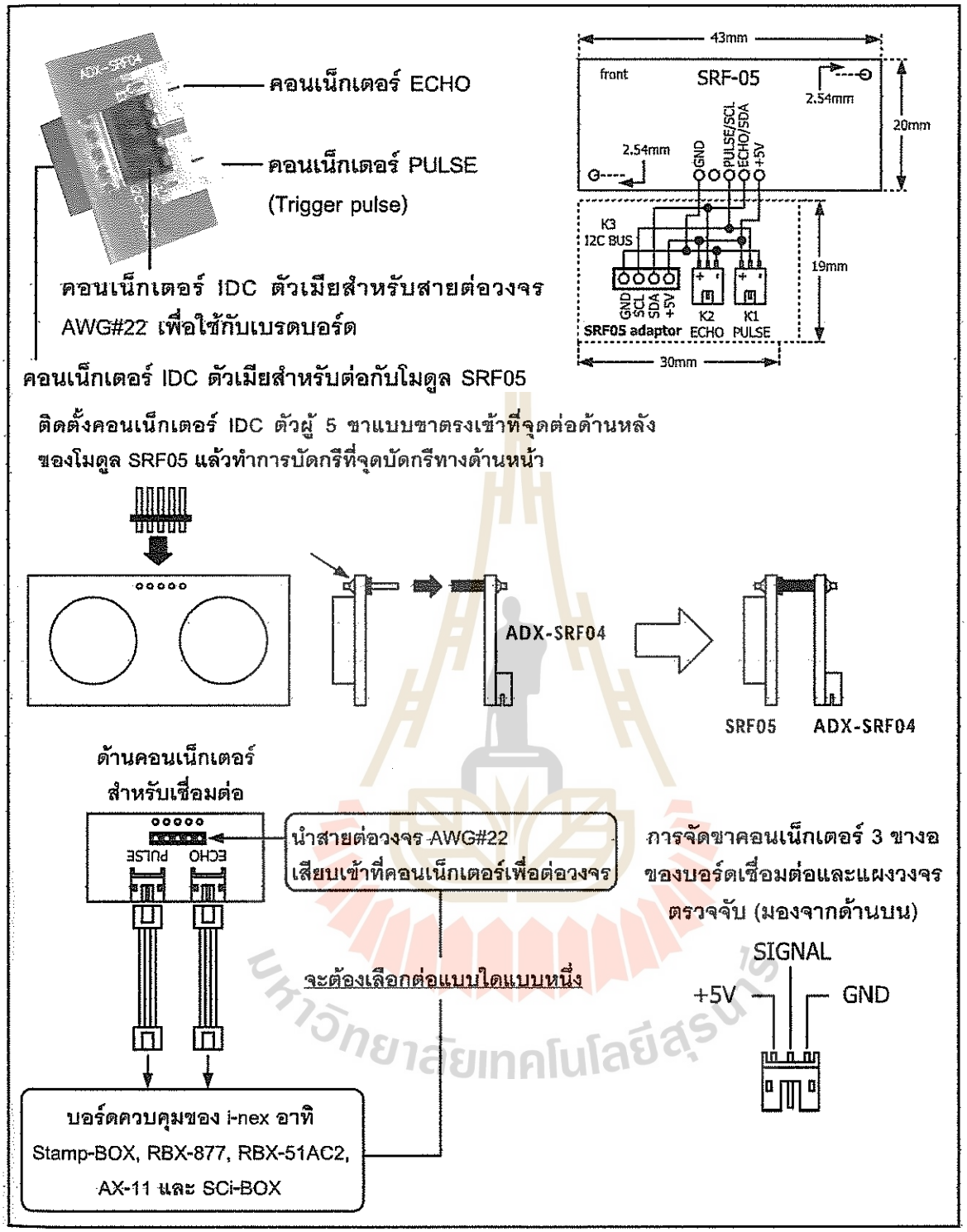
3. **ขา Trigger Pulse Input (TRIGGER)** เป็นขาอินพุตรับสัญญาณพัลส์ที่มีความกว้างอย่างน้อย 10 ไมโครวินาทีเพื่อกระตุ้นการสร้างคลื่นอัลตร้าโซนิกความถี่ 40kHz ออกสู่อากาศจากตัวส่ง ดังนั้นเมื่อคลื่นความถี่ดังกล่าวนี้เคลื่อนที่ไปกระทบสิ่งกีดขวางที่อยู่เบื้องหน้าก็จะเกิดการสะท้อนกลับเข้ามายังตัวรับ และถูกแปลงออกมาเป็นความกว้างของสัญญาณพัลส์ที่จะส่งออกไปทางขา Echo Pulse Output นอกจากนี้ในโหมด 1 สัญญาณ จะใช้จุดนี้เป็นจุดสื่อสารข้อมูลอนุกรมเพื่อรับส่งค่าการวัดกับไมโครคอนโทรลเลอร์

4. **ขา MODE** สำหรับเลือกรูปแบบการติดต่อกับ SRF05

ปล่อยลอยไว้ (NC) : เลือกให้ติดต่อแบบ 2 สัญญาณ ผ่านจุดต่อ ECHO และ TRIGGER

ต่อลงกราวด์ : เลือกให้ติดต่อแบบ 1 สัญญาณ ผ่านจุดต่อ TRIGGER

5. **ขา GND** สำหรับต่อกราวด์



รูปที่ 3 แสดงวงจรของบอร์ด ADX-SRF04 และการเชื่อมต่อกับโมดูล SRF05

2. บอร์ด ADX-SRF04

เพื่ออำนวยความสะดวกแก่ผู้ใช้งานกับบอร์ดควบคุมหุ่นยนต์ของบริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด (i-nex : เป็นตัวแทนจำหน่ายสินค้าของ Devantech ในประเทศไทยอย่างเป็นทางการ) จึงได้พัฒนาบอร์ดอะแดปเตอร์รุ่น ADX-SRF04 เพื่อให้นำโมดูล SRF04 หรือ SRF05 มาติดตั้ง (จัดมาพร้อมกับสายเชื่อมต่อในชุดของ SRF05) เพื่อให้ใช้งานได้สะดวกขึ้น

บนบอร์ด ADX-SRF04 ได้จัดเตรียมคอนเน็กเตอร์ PCB 3 ขาตัวผู้ 2 ตัวแยกกันระหว่างสัญญาณ ECHO และ TRIGGER สำหรับเชื่อมต่อกับบอร์ดควบคุมหุ่นยนต์ และคอนเน็กเตอร์ IDC ตัวเมียแถวเดี่ยว 4 ขาสำหรับเสียบสายต่อวงจรเบอร์ AWG#22 เพื่อต่อกับแผงต่อวงจรหรือเบรดบอร์ด โดยวงจรของบอร์ด ADX-SRF04 และการติดตั้งเพื่อใช้งานแสดงในรูปที่ 3

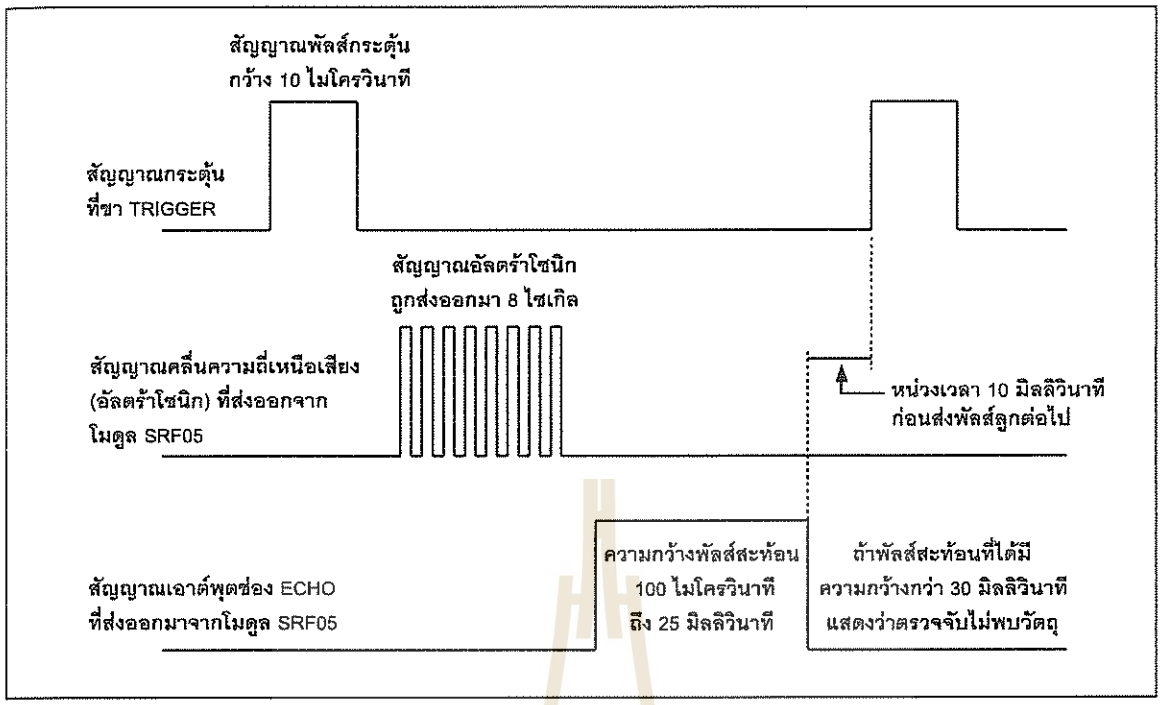
3. การใช้งานกับไมโครคอนโทรลเลอร์

3.1 ฟังก์ชั่นโปรแกรมภาษา C สำหรับติดต่อระหว่าง P89V51RD2 กับโมดูล SRF05 ในโหมดการติดต่อแบบ 2 สัญญาณ

ในการเขียนโปรแกรมภาษา C สำหรับไมโครคอนโทรลเลอร์ P89V51RD2 เพื่อใช้งานโมดูล SRF05 สิ่งที่ควรทำคือ สร้างฟังก์ชั่นในการติดต่อเพื่ออ่านค่าและประมวลผล ตัวอย่างที่นำมาเสนอในบทนี้คือ **ฟังก์ชั่น `distance`**

ฟังก์ชั่นนี้นำมาใช้อ่านค่าระยะทางที่วัดได้จากโมดูล SRF05 เริ่มต้นด้วยการกำหนดให้ขาพอร์ต P1.0 ของไมโครคอนโทรลเลอร์ P89V51RD2 ต่อกับขา ECHO ของโมดูล SRF05 และกำหนดให้เป็นพอร์ตอินพุต โดยกำหนดให้มีลอจิก "1" เพื่อเริ่มต้นทำงาน ส่วนขา TRIGGER ของโมดูล SRF05 ต่อเข้ากับขาพอร์ต P1.1 ดังนั้นจึงต้องกำหนดให้เป็นเอาต์พุต เพื่อสร้างสัญญาณพัลส์บวกที่มีความกว้าง 5 ถึง 10 ไมโครวินาที เพื่อเป็นกระตุ้นให้โมดูล SRF05 เริ่มต้นกระบวนการวัดระยะทาง

ทันทีที่โมดูล SRF05 ได้รับสัญญาณพัลส์บวกเข้าที่ขา TRIGGER โมดูล SRF05 จะดำเนินการสร้างขบวนพัลส์ความถี่ 40 kHz ออกสู่อากาศ ขบวนพัลส์ความถี่เหนือเสียงหรืออัลตร้าโซนิกกลุ่มนี้จะเดินทางเป็นเส้นตรง จนกระทั่งกระทบวัตถุที่ขวางอยู่เบื้องหน้า ทำให้ขบวนพัลส์อัลตราโซนิกสะท้อนกลับเข้ามายังตัวรับของโมดูล SRF05 หลังจากนั้นหน่วยประมวลผลภายในโมดูล SRF05 จะวิเคราะห์และให้สัญญาณเอาต์พุตออกมาเป็นความกว้างพัลส์บวกที่เป็นสัดส่วนโดยตรงกับระยะทางที่ตรวจจับได้

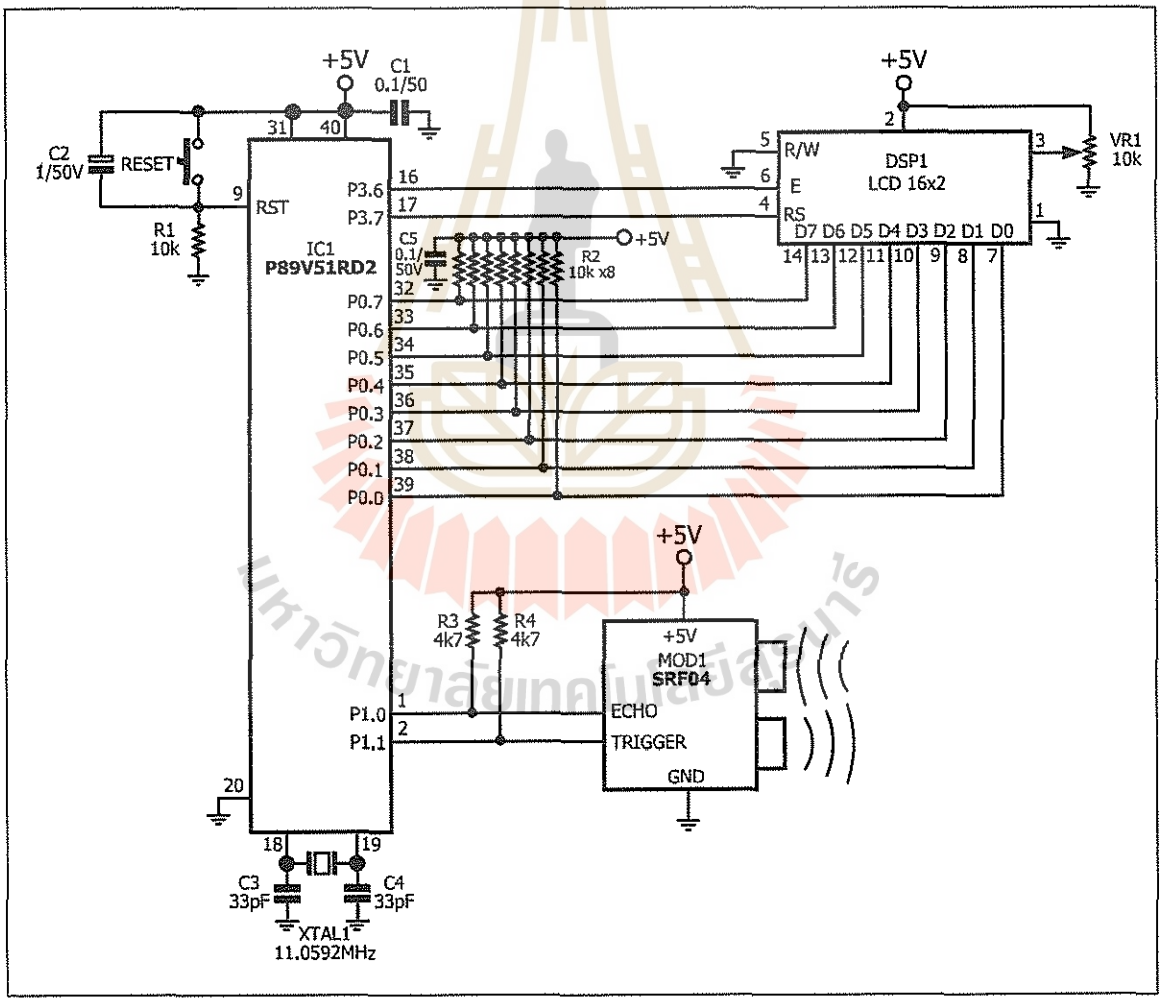


รูปที่ 4 ไดอะแกรมเวลาแสดงสัญญาณที่ส่งไปยัง SRF05 และสัญญาณที่ตอบรับกลับมาจาก SRF05 ในกรณีติดต่อแบบ 2 สัญญาณ

รูปที่ 5 ไดอะแกรมเวลาแสดงสัญญาณที่ส่งไปยัง SRF05 และสัญญาณที่ตอบรับกลับมาจาก SRF05 ในกรณีติดต่อแบบ 1 สัญญาณ (ขุดต่อ MODE ต้องต่อลงกราวด์)

สำหรับการนับความกว้างพัลส์ที่ขา ECHO จะใช้ไทเมอร์ 1 ในโหมด 16 บิต เป็นตัวนับเวลาความกว้างของสัญญาณพัลส์ โดยจะเริ่มต้นนับ เมื่อพบว่า มีการเปลี่ยนแปลงลอจิกแบบขาขึ้น จาก “0” ไปเป็น “1” และหยุดการนับเมื่อพบการเปลี่ยนแปลงลอจิกแบบขาลง จาก “1” ไปเป็น “0”

หลังจากนั้นค่าที่ได้จากการนับนำไปหารด้วย 114 จะเป็นค่าประมาณที่ใกล้เคียงจากการแปลงข้อมูลระยะทางที่ได้จากโมดูล SRF05 ไทเมอร์ของไมโครคอนโทรลเลอร์ที่ใช้นั้นจะอ้างอิงสัญญาณนาฬิกา ความถี่ 11.0592 MHz และไมโครคอนโทรลเลอร์ P89V51RD2 ทำงานด้วยความเร็ว 6 สัญญาณนาฬิกาต่อไซเกิล



รูปที่ 6 วงจรทดลองติดต่อไมโครคอนโทรลเลอร์ P89V51RD2 กับ SRF05 โมดูลวัดระยะทาง

```
/*-------------------------------------------------------------------------*/
// Description   : Distance detector by SFR05 display value on LCD
// Filename      : V51SRF05.C
// C compiler    : RIDE 51 V6.4.35
/*-------------------------------------------------------------------------*/
#include <REGLV51RD2.h>        // Header file register for P89V51RD2BN
#include <lcdV51.h>            // Library for LCD display
#include <intrins.h>           // Include library for nop function
sbit echo = P1^0;              // Define received pulse pin(SDA)
sbit trigger = P1^1;           // Define trigger pulse pin(SCL)

/***************** Function Trigger pulse for start process *****************/
void trigger_pulse(void)
{
  unsigned char i;             // Variable for counter
  trigger = 1;                 // Start positive pulse
  for(i=0;i<10;i++)            // Loop 10 times
  _nop_();                     // Delay 1 microseconnd function
  trigger = 0;                 // End of positive pulse
}

/******************** Distance reading function ****************************/
unsigned int distance()
{
  unsigned int mc;             // Variable for internal function
   trigger = 0;                // Initial logic low
  echo = 1;                    // Initial logic low
  TMOD &=0x0F;                 // Configuration Timer1 mode 2(16-bit counter)
  TMOD |=0x10;
  TH1 = 0x00;                  // Initial Timer1 counter value to zero
  TL1 = 0x00;
  TF1 = 0;                     // Clear overflow flag
  TR1 = 0;                     // Start Timer1
  trigger_pulse();             // Send trigger pulse
  while(!echo);                // Detect rising pulse
  TR1 = 1;                     // Start timer count
  while(echo);                 // Detect falling pulse
  TR1 = 0;                     // Stop timer
  TF1 = 0;                     // Clear overflow flag
  mc = TH1;                    // Keep high byte
  mc <<= 8;                    // Shift to high byte
  mc += TL1;                   // Keep low byte
  delay(10);                   // Delay 10 millisecond
  return(mc/114);              // Return distance value in centimetre unit
}

/************************* Main Program ***********************************/
void main()
{
  unsigned int value,old_value;          // Variable for storing distance value
  lcd_init();                            // Initial LCD
  lcd_puts(0x80,"SRF-05 Demo...");       // Display message
  lcd_puts(0xC0,"Distance:      cm");    // Show distance value on screen
  while(1)                               // Infinite loop
```

โปรแกรมที่ 1 โปรแกรมภาษา C สำหรับทดลองการใช้งานไมโครคอนโทรลเลอร์ P89V51RD2 กับ SRF05 โมดูลวัดระยะทาง (มีต่อ)

```
    {
      value = distance();              // Read distance
      if(value != old_value)
                                 // Compare new and previous value to update display
      {
         lcd_puts(0xCA,"    ");        // Clear previous value
         inttolcd(0xCA,value);         // Update new value
         delay(500);                   // Delay for displaying
      }
      old_value = value;          // Keep previous value for comparison next time
    }
}
```

คำอธิบายโปรแกรม

การทำงานของโปรแกรมนี้เป็นการติดต่อกับโมดูลวัดระยะทางด้วยอัลตราโซนิก SRF04 ของไมโครคอนโทรลเลอร์ P89V51RD2 โดยภายในโปรแกรมหลักเป็นการวนอ่านค่าระยะทางที่วัดได้มาแสดงผลอยู่ตลอดเวลาภายใต้ `while(1){}` ที่เงื่อนไขเป็นจริง ส่วนการแสดงผลจะเปรียบเทียบระหว่างข้อมูลเดิมและใหม่ที่อ่านได้ ถ้ามีการเปลี่ยนแปลงจึงจะนำค่าใหม่ไปแสดงผล

โปรแกรมที่ 1 โปรแกรมภาษา C สำหรับทดลองการใช้งานไมโครคอนโทรลเลอร์ P89V51RD2 กับ SRF05 โมดูลวัดระยะทาง (จบ)

ในการทดสอบนี้เป็นการเขียนโปรแกรมภาษา C ควบคุมไมโครคอนโทรลเลอร์ P89V51RD2 เพื่อติดต่อกับ SRF05 โมดูลตรวจจับและวัดระยะทางด้วยคลื่นอัลตร้าโซนิก

1. ทำการติดตั้งโมดูล SRF05 เข้ากับบอร์ด ADX-SRF04

2. ต่อวงจรตามรูปที่ 6

3. เขียนโปรแกรมที่ 1 แล้วทำการแปลงให้เป็นไฟล์ .hex ด้วย RC51 ผ่านกระบวนการสร้างไฟล์โปรเจ็กต์ของ Rkit-51 โดยใช้ RIDE ได้ไฟล์ v51srf05.hex ดาวน์โหลดลงในไมโครคอนโทรลเลอร์ P89V51RD2 โดยต้องลบข้อมูลก่อนโปรแกรมลงไปใหม่

4. จ่ายไฟให้แก่วงจร นำวัตถุมาวางขวางที่ด้านหน้าของโมดูล SRF05ในระยะ 1 เซนติเมตรถึง 4 เมตร แล้วสังเกตผลที่โมดูล LCD

ที่โมดูล LCD แสดงค่าระยะทางที่วัดได้ โดยมีรูปแบบคือ

SRF-05 Demo...

Distance: xxx cm

5. จากนั้นทดลองวัดระยะห่างโดยใช้ไม้บรรทัดเพื่อเปรียบเทียบผลการทดลองที่ได้

3.2 เขียนโปรแกรมเพื่ออ่านค่าจากโมดูล SRF05 ใช้การติดต่อแบบ 2 สัญญาณสำหรับไมโครคอนโทรลเลอร์ PIC16F877 โดยใช้ PICBASIC PRO compiler

ต่อขา RC4 ต่อเข้ากับขา ECHO และ ขา RC3 ต่อเข้ากับขา TRIGGER ดังวงจรรูปที่ 7

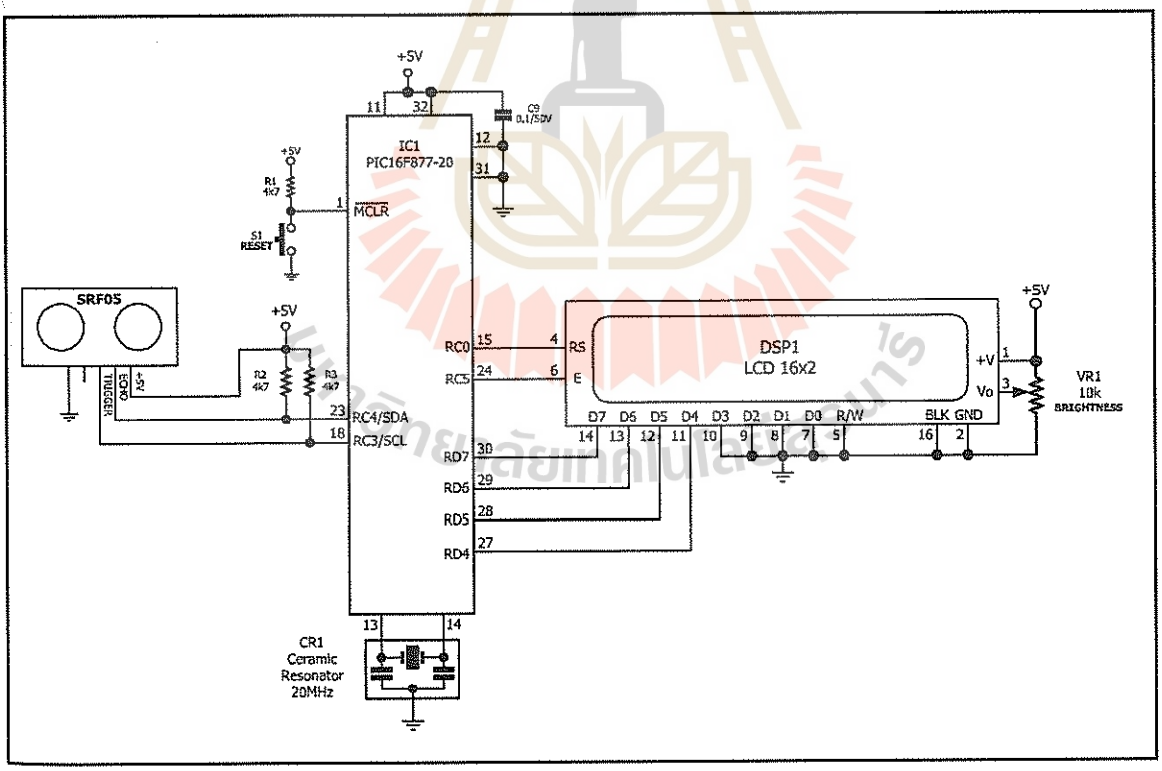
โปรแกรมที่ 2 เป็นโปรแกรมตัวอย่างที่ใช้อ่านค่าระยะทางที่วัดได้จาก SRF05 นำไปแสดงที่โมดูล LCD การทำงานเริ่มจากกำหนดให้ขาพอร์ตที่ใช้สำหรับส่งสัญญาณกระตุ้นไปยัง SRF05 มีลอจิกเป็น "0" ก่อน จากนั้นเรียกโปรแกรมย่อยอ่านข้อมูลจาก SRF05 โดยโปรแกรมย่อยจะทำงานดังนี้

1. วนลูป 5 รอบ เพื่ออ่านค่าทั้งหมด 5 ครั้งแล้วนำมาเฉลี่ย

2. ส่งพัลส์ 10 ไมโครวินาที โดยใช้คำสั่ง PULSOUT หน่วยของความกว้างพัลส์สำหรับคำสั่งนี้คือ 2 ไมโครวินาทีที่ความถี่สัญญาณนาฬิกาหลัก 20MHz ถ้าต้องการพัลส์ 10 ไมโครวินาทีต้องกำหนดค่าให้กับคำสั่ง PULSOUT เท่ากับ 5

3. อ่านค่าความกว้างพัลส์เพื่อตรวจสอบระยะที่ SRF05 อ่านค่าได้ โดยใช้คำสั่ง PULSIN คำสั่งนี้จะวัดค่าความกว้างพัลส์ช่วงบวกในหน่วย 2 ไมโครวินาที เก็บค่าที่ได้ไว้ในตัวแปร RESULT

4. หน่วงเวลา 10 มิลลิวินาทีก่อนที่จะเริ่มต้นวนลูปทำงานตามข้อ 1 ใหม่จนครบ 5 รอบ แล้วออกจากโปรแกรมย่อย ผลลัพธ์จากการวนลูป 5 รอบเก็บในตัวแปร RAW



รูปที่ 7 วงจรทดลองติดต่อไมโครคอนโทรลเลอร์ PIC16F877 กับ SRF05 โมดูลวัดระยะทาง

เมื่อออกจากโปรแกรมย่อย โปรแกรมหลักจะนำค่าที่อ่านได้ มาหารด้วย 30 ผลลัพธ์ที่ได้มีหน่วยเป็นเซนติเมตร แล้วนำมาแสดงที่โมดูล LCD จากนั้นทำการแปลงค่าที่อ่านได้เป็นนิ้ว โดยใช้ความสัมพันธ์ 1 เซนติเมตรเท่ากับ 0.3937 นิ้ว เพื่อให้การแสดงค่าได้ผลลัพธ์ที่ละเอียดขึ้น ควรให้ละเอียดเป็น 0.1 นิ้ว จึงต้องคูณด้วย 3.937 แทน

```
@ DEVICE  PIC16F877,HS_OSC' Use PIC16F877 and HS Oscilator
DEFINE OSC 20
DEFINE LCD_DREG  PORTD     ' Set LCD Data port
DEFINE LCD_DBIT  4         ' Set starting Data bit for 4-bit interface
DEFINE LCD_RSREG PORTC     ' Set LCD Register Select port
DEFINE LCD_RSBIT 0         ' Set LCD Register Select bit
DEFINE LCD_EREG  PORTC     ' Set LCD Enable port
DEFINE LCD_EBIT  5         ' Set LCD Enable bit
DEFINE LCD_BITS  4         ' Set LCD bus size (4 or 8 bits)
DEFINE LCD_LINES 2         ' Set number of lines on LCD
DEFINE LCD_COMMANDUS 2000 ' Set command delay time in us
DEFINE LCD_DATAUS   50     ' Set data delay time in us
TRIGGERVAR PORTC.3
ECHO       VAR PORTC.4
' =====[ Variables ]=============================================
I       VAR BYTE          ' loop counter
RESULT  VAR WORD          ' pulse width from sensor
RAW     VAR WORD          ' filtered measurment
CM      VAR WORD          ' centimeters
INCH    VAR WORD
' =====[ Initialization ]=======================================
SETUP: LOW TRIGGER
' =====[ Program Code ]==========================================
MAIN:
   GOSUB GET_SONAR                              ' Take Sonar Reading
   CM = RAW /30                                 ' Convert to Centimeter
   LCDOUT $FE,$01,"CM = " , DEC CM , " CM"      ' Show on LCD
   INCH = CM */ $03EF                           ' x 3.937 (to 0.1 inches)
   LCDOUT $FE,$C0,"INCH = ",DEC INCH/10,".",DEC1 INCH," INCH"   ' Show on LCD
   GOTO MAIN
   END
' =====[ Subroutines ]===========================================
GET_SONAR:
   RAW = 0                                      ' Clear Measurement
   FOR I = 1 TO 5                               ' Take 5 Samples
       PULSOUT TRIGGER,5                        ' 10 uS Trigger Pulse
       PULSIN ECHO,1,RESULT                     ' Measure Pulse
       RAW = RAW + (RESULT / 5)                 ' Simple Digital Filter
       PAUSE 10                                 ' Minimum Period Between
   NEXT
   RETURN
```

***หมายเหตุ** : การคูณค่าด้วยตัวเลขที่เป็นทศนิยม จะใช้คำสั่งคูณเลขแล้วนำเอาผลลัพธ์ 16 บิตตรงกลางมาใช้ ค่า 3.937 เมื่อแปลงเป็นตัวเลขฐานสิบหกจะมีค่าเท่ากับ 03EF โดย 03 คือตัวเลขในหลักหน่วย ส่วน EF คือค่าตัวเลขทศนิยม*

โปรแกรมที่ 2 ตัวอย่างโปรแกรมภาษาเบสิกของ PIC16F877 ที่ใช้ PICBASIC PRO compiler สำหรับตรวจสอบการทำงานของ SRF05 โมดูลตรวจจับวัตถุด้วยคลื่นอัลตร้าโซนิก

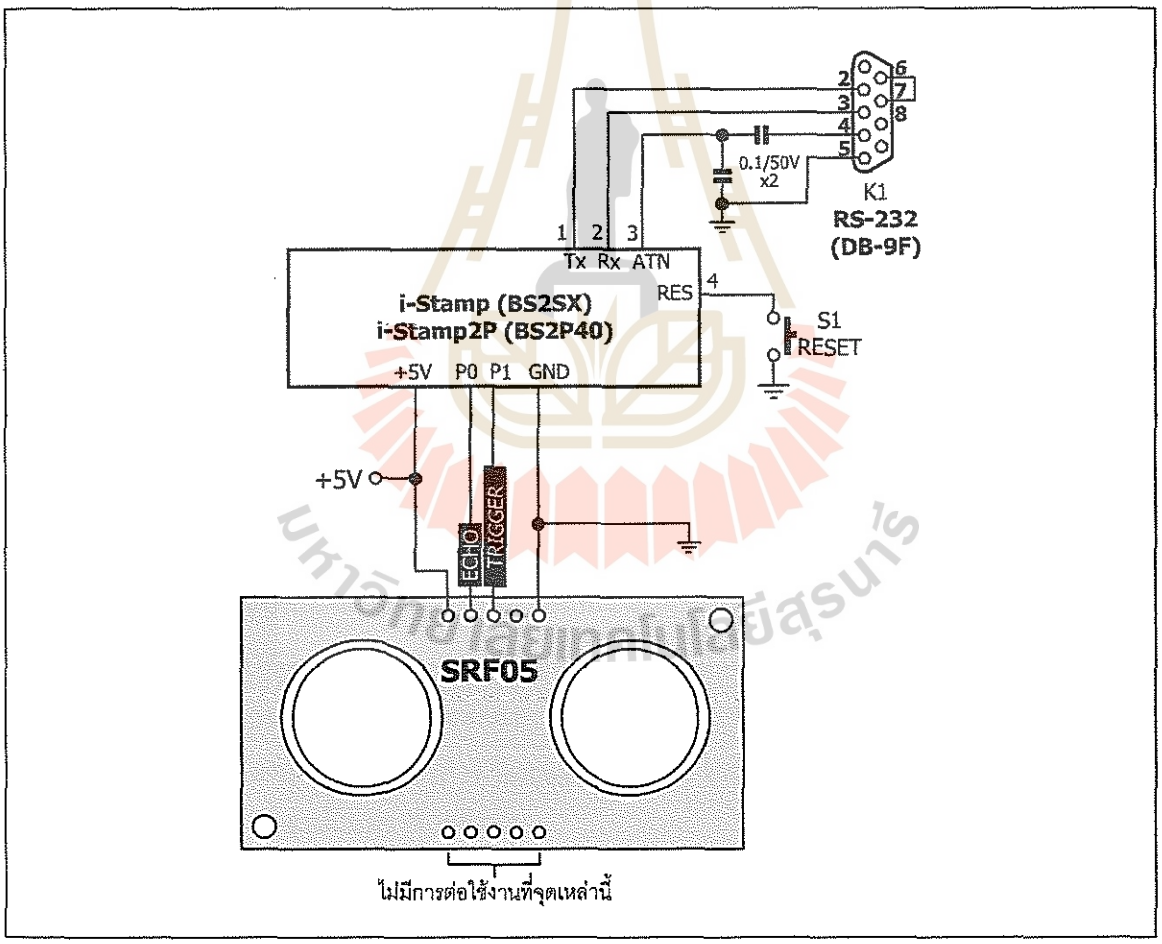
3.3 เขียนโปรแกรมเพื่ออ่านค่าจากโมดูล SRF05 ใช้การติดต่อแบบ 2 สายสัญญาณ สำหรับ i-Stamp (เบสิกแสตมป์ 2SX) และ i-Stamp2P (เบสิกแสตมป์ 2P)

1. ต่อขา P0 ของ i-Stamp หรือ i-Stamp2P เข้ากับจุดต่อ ECHO

2. ต่อขา P0 ของ i-Stamp หรือ i-Stamp2P เข้ากับจุดต่อ TRIGGER

3. ต่อสายสัญญาณที่จำเป็น ดังในวงจรรูปที่ 8

โปรแกรมที่ 3 เป็นโปรแกรมตัวอย่างที่ใช้อ่านค่าระยะทางที่วัดได้จาก SRF05 นำไปแสดงที่หน้าต่าง Debug Terminal ของโปรแกรมเบสิกแสตมป์เอดิเตอร์ โดยในโปรแกรมนี้สามารถใช้ได้กับทั้ง i-Stamp (BS2SX : เบสิกแสตมป์ 2SX), i-Stamp2P (BS2P : เบสิกแสตมป์ 2P) และ BS2 (เบสิกแสตมป์ 2 รุ่นมาตรฐาน) โดยจะแยกด้วยไดเร็กตีฟที่กำหนดในส่วนหัวของโปรแกรม



รูปที่ 8 วงจรทดลองติดต่อ i-Stamp และ i-Stmp2P กับ SRF05 โมดูลวัดระยะทาง

```
' {$STAMP BS2sx}
' {$PBASIC 2.5}
'*****************************************************************************
' SRF05 Ultra-Sonic Ranger demonstration code on BASIC Stamp 2,2e,2sx,2p,2pe,2px
' Using mode 1 (Mode pin leave no connect)
' Connect Trigger pin to transmit, Echo pin to receive
'*****************************************************************************
Trig_Pulse PIN  1
Echo       PIN  0
Distance   VAR  Word

LOW Trig_Pulse
DO
  #IF ($STAMP = BS2SX) OR ($STAMP = BS2P) OR ($STAMP = BS2PX) #THEN
    PULSOUT Trig_Pulse,13          ' Pulse out 10.8 us
    PULSIN  Echo,1,Distance        ' Read positive pulse 0.8us per unit
    Distance = Distance */ $0161  ' Multiply by 1.379 convert unit as Millimeter
    DEBUG HOME,DEC5 Distance," mm.",CR
    PAUSE 100
  #ELSE ' Case $Stamp BS2 , BS2E , BS2PE
    PULSOUT Trig_Pulse,5           ' Pulse out 10 us
    PULSIN  Echo,1,Distance        ' Read positive pulse 2us per unit
    IF Distance < 19000 THEN
        Distance = Distance */ $0373   ' Multiply by 3.448 convert unit as
Millimeter
      DEBUG HOME,DEC5 Distance," mm.",CR
    ELSE
      Distance = Distance / 29   ' Divide by 29 convert unit as Centimeter
      DEBUG HOME,DEC5 Distance," cm.",CR
    ENDIF
    PAUSE 100
  #ENDIF
LOOP
```

โปรแกรมที่ 3 ตัวอย่างโปรแกรมภาษาพีเบสิกของ i-Stamp และ i-Stamp2P สำหรับตรวจสอบการทำงานของ SRF05 โมดูลวัดระยะทางด้วยคลื่นอัลตร้าโซนิกในโหมดการติดต่อแบบ 2 สายสัญญาณ

การทำงานเริ่มจากกำหนดขาพอร์ตที่ใช้ติดต่อกับโมดูล SRF05 จากนั้นส่งพัลส์กระตุ้นออกไปยังโมดูล SRF05 จากนั้นอ่านค่าสัญญาณพัลส์กลับเข้ามาด้วยคำสั่ง PULSIN มาเก็บไว้ที่ตัวแปร Distance ทำการคำนวณค่าระยะทางในหน่วยมิลลิเมตรออกมาด้วยคำสั่ง

```
Distance = Distance */ $0161        กรณีใช้กับ i-Stamp และ i-Stamp2P
```

แล้วให้แสดงออกมาที่หน้าต่าง Debug Terminal จากนั้นจะทำการคำนวณเพื่อแสดงค่าระยะทางออกมาในหน่วยเซนติเมตร ด้วยคำสั่ง

```
Distance = Distance / 58            กรณีใช้กับ i-Stamp และ i-Stamp2P
```

แล้วให้แสดงออกมาที่หน้าต่าง Debug Terminal เช่นกัน จากนั้นหน่วงเวลา 100 มิลลิวินาที ก่อนวนกลับไปอ่านค่าอีกครั้ง

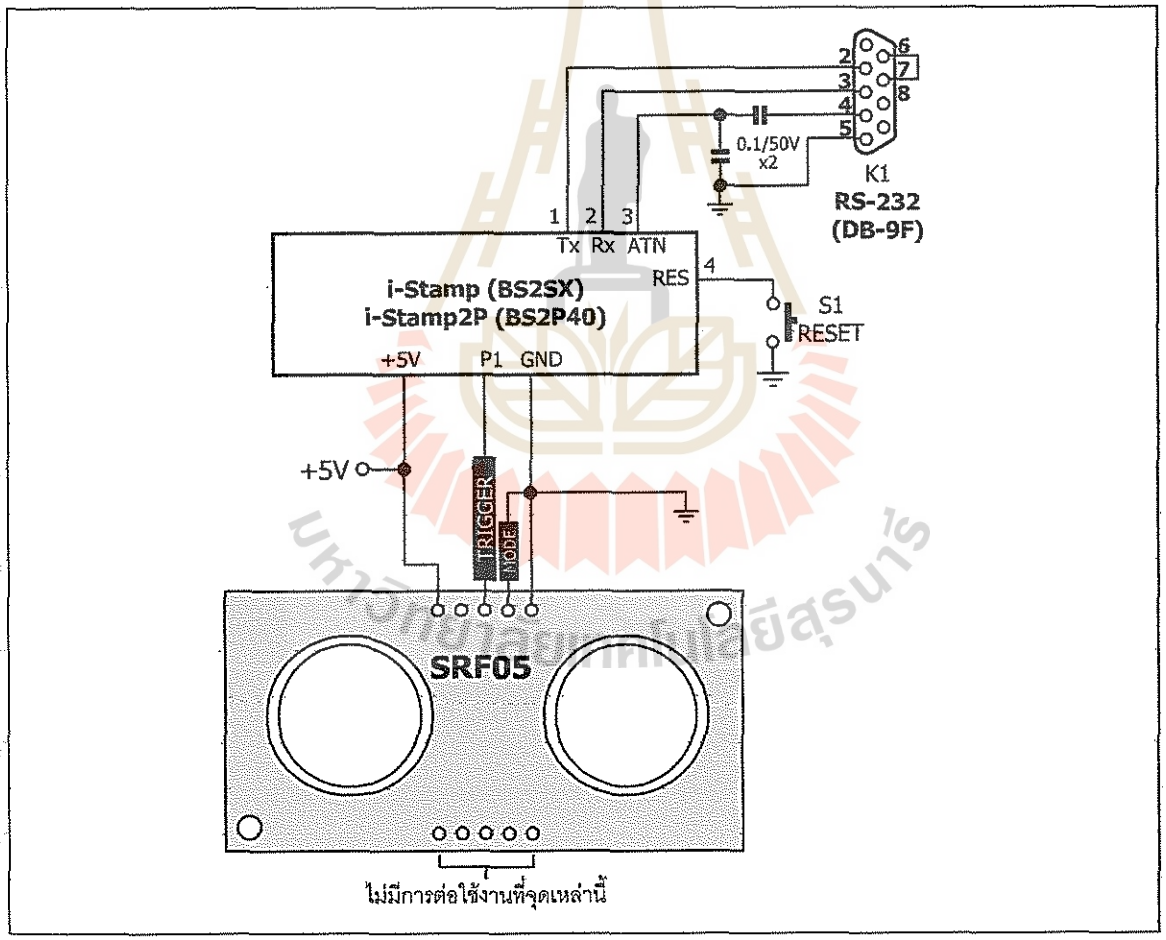
3.4 เขียนโปรแกรมเพื่ออ่านค่าจากโมดูล SRF05 ใช้การติดต่อแบบ 1 สายสัญญาณ สำหรับ i-Stamp (เบสิกแสตมป์ 2SX) และ i-Stamp2P (เบสิกแสตมป์ 2P)

1. ต่อขา P0 ของ i-Stamp หรือ i-Stamp2P เข้ากับจุดต่อ TRIGGER

2. ต่อจุดต่อ MODE ของ SRF05 ลงกราวด์

3. ต่อสายสัญญาณที่จำเป็น ดังในวงจรรูปที่ 9

โปรแกรมที่ 4 เป็นโปรแกรมตัวอย่างอ่านค่าระยะทางที่วัดได้จาก SRF05 ซึ่งใช้การติดต่อแบบ 1 สายสัญญาณ นำไปแสดงที่หน้าต่าง Debug Terminal ของโปรแกรมเบสิกแสตมป์เอดิเตอร์ โดยในโปรแกรมนี้สามารถใช้ได้กับทั้ง i-Stamp (BS2SX : เบสิกแสตมป์ 2SX), i-Stamp2P (BS2P : เบสิกแสตมป์ 2P) และ BS2 (เบสิกแสตมป์ 2 รุ่นมาตรฐาน) โดยจะแยกด้วยไดเร็กตีฟที่กำหนดในส่วนหัวของโปรแกรม



รูปที่ 9 วงจรทดลองติดต่อ i-Stamp และ i-Stmp2P กับ SRF05 โมดูลวัดระยะทางในแบบ 1 สายสัญญาณ

```
' {$STAMP BS2sx}
'*****************************************************************************
' SRF05 Ultra-Sonic Ranger demonstration code on BASIC Stamp 2,2e,2sx,2p,2pe,2px
' Using mode 2 (Mode pin tied to GND)
' Connect Trigger pin both transmit and receive
'*****************************************************************************
Trig_Echo PIN  1
Distance   VAR  Word

LOW Trig_Echo

DO
  #IF ($STAMP = BS2SX) OR ($STAMP = BS2P) OR ($STAMP = BS2PX) #THEN
    PULSOUT Trig_Echo,13                    ' Pulse out 10.8 us
    PULSIN  Trig_Echo,1,Distance            ' Read positive pulse 0.8us per unit
   Distance = Distance */ $0161  ' Multiply by 1.379 convert unit as Millimeter
    DEBUG HOME,DEC5 Distance," mm.",CR
    PAUSE 100
  #ELSE ' Case $Stamp BS2 , BS2E , BS2PE
    PULSOUT Trig_Echo,5                     ' Pulse out 10 us
    PULSIN  Trig_Echo,1,Distance            ' Read positive pulse 2us per unit
    IF Distance < 19000 THEN
      Distance = Distance */ $0373
                              ' Multiply by 3.448 convert unit as Millimeter
      DEBUG HOME,DEC5 Distance," mm.",CR
    ELSE
      Distance = Distance / 29
                              ' Divide by 29 convert unit as Centimeter
      DEBUG HOME,DEC5 Distance," cm.",CR
    ENDIF
    PAUSE 100
  #ENDIF
LOOP
```

โปรแกรมที่ 4 ตัวอย่างโปรแกรมภาษาพีเบสิกของ I-Stamp และ I-Stamp2P สำหรับตรวจสอบการทำงานของ SRF05 โมดูลวัดระยะทางด้วยคลื่นอัลตร้าโซนิกในโหมดการติดต่อแบบ 1 สายสัญญาณ

บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด
3133/53 ซ.สุขุมวิท 101/2 ถ. สุขุมวิท แขวงบางนา เขตบางนา กรุงเทพฯ 10260
โทรศัพท์ 0-2747-7001-4 โทรสาร 0-2747-7005
URL : www.inex.co.th e-mail : tech@inex.co.th

The FT232BL is the lead free version of the 2nd generation of FTDI's popular USB UART I.C. This device not only adds extra functionality to its FT8U232AM predecessor and reduces external component count, but also maintains a high degree of pin compatibility with the original, making it easy to upgrade or cost reduce existing designs as well as increasing the potential for using the device in new application areas.

1.0 Features

HARDWARE FEATURES

- Single Chip USB ⇔Asynchronous Serial Data Transfer
- Full Handshaking & Modem Interface Signals
- UART I/F Supports 7 / 8 Bit Data, 1 / 2 Stop Bits and Odd/Even/Mark/Space/No Parity
- Data rate 300 => 3M Baud (TTL)
- Data rate 300 => 1M Baud (RS232)
- Data rate 300 => 3M Baud (RS422/RS485)
- 384 Byte Receive Buffer / 128 Byte Transmit Buffer for high data throughput
- Adjustable RX buffer timeout
- Fully Assisted Hardware or X-On / X-Off Handshaking
- In-built support for event characters and line break condition
- Auto Transmit Buffer control for RS485
- Support for USB Suspend / Resume through SLEEP# and RI# pins
- Support for high power USB Bus powered devices through PWREN# pin
- Integrated level converter on UART and control signals for interfacing to 5V and 3.3V logic
- Integrated 3.3V regulator for USB IO
- Integrated Power-On-Reset circuit
- Integrated 6MHz – 48Mhz clock multiplier PLL
- USB Bulk or Isochronous data transfer modes
- 4.35V to 5.25V single supply operation
- UHCI / OHCI / EHCI host controller compatible
- USB 1.1 and USB 2.0 compatible
- USB VID, PID, Serial Number and Product Description strings in external EEPROM
- EEPROM programmable on-board via USB
- Compact Lead free RoHS compliant 32-LD LQFP package.

VIRTUAL COM PORT (VCP) DRIVERS for

- Windows 98 and Windows 98 SE
- Windows 2000 / ME / Server 2003 / XP
- Windows XP 64 Bit
- Windows XP Embedded
- Windows CE 4.2
- MAC OS-8 and OS-9
- MAC OS-X
- Linux 2.40 and greater

D2XX (USB Direct Drivers + DLL S/W Interface)

- Windows 98 and Windows 98 SE
- Windows 2000 / ME / Server 2003 / XP
- Windows XP 64 Bit
- Windows XP Embedded
- Windows CE 4.2
- Linux 2.40 and greater

APPLICATION AREAS

- USB ⇔RS232 Converters
- USB ⇔ RS422 / RS485 Converters
- Upgrading RS232 Legacy Peripherals to USB
- Cellular and Cordless Phone USB data transfer cables and interfaces
- Interfacing MCU based designs to USB
- USB Audio and Low Bandwidth Video data transfer
- PDA ⇔ USB data transfer
- USB Smart Card Readers
- Set Top Box (S.T.B.) PC - USB interface
- USB Hardware Modems
- USB Wireless Modems
- USB Instrumentation
- USB Bar Code Readers

2.0 Enhancements

This section summarises the enhancements of the 2nd generation device compared to its FT8U232AM predecessor. For further details, consult the device pin-out description and functional descriptions.

- **Integrated Power-On-Reset (POR) Circuit**
  The device now incorporates an internal POR function. The existing RESET# pin is maintained in order to allow external logic to reset the device where required, however for many applications this pin can now simply be hard wired to VCC. In addition, a new reset output pin (RSTOUT#) is provided in order to allow the new POR circuit to provide a stable reset to external MCU and other devices. RSTOUT# was the TEST pin on the previous generation of devices.

- **Integrated RCCLK Circuit**
  In the previous devices, an external RC circuit was required to ensure that the oscillator and clock multiplier PLL frequency was stable prior to enabling the clock internal to the device. This circuit is now embedded on-chip – the pin assigned to this function is now designated as the TEST pin and should be tied to GND for normal operation.

- **Integrated Level Converter on UART interface and control signals**
  The previous devices would drive the UART and control signals at 5V CMOS logic levels. The new device has a separate VCC-IO pin allowing the device to directly interface to 3.3V and other logic families without the need for external level converter I.C.'s

- **Improved Power Management control for USB Bus Powered, high current devices**
  The previous devices had a USBEN pin, which became active when the device was enumerated by USB. To provide power control, this signal had to be externally gated with SLEEP# and RESET#. This gating is now done on-chip - USBEN has now been replaced with the new PWREN# signal which can be used to directly drive a transistor or P-Channel MOSFET in applications where power switching of external circuitry is required. A new EEPROM based option makes the device pull gently down its UART interface lines when the power is shut off (PWREN# is High). In this mode, any residual voltage on external circuitry is bled to GND when power is removed thus ensuring that external circuitry controlled by PWREN# resets reliably when power is restored.

- **Lower Suspend Current**
  Integration of RCCLK within the device and internal design improvements reduce the suspend current of the FT232BL to under 200uA (excluding the 1.5k pull-up on USBDP) in USB suspend mode. This allows greater margin for peripherals to meet the USB Suspend current limit of 500uA.

- **Support for USB Isochronous Transfers**
  Whilst USB Bulk transfer is usually the best choice for data transfer, the scheduling time of the data is not guaranteed. For applications where scheduling latency takes priority over data integrity such as transferring audio and low bandwidth video data, the new device now offers an option of USB Isochronous transfer via an option bit in the EEPROM.

Programmable Receive Buffer Timeout

In the previous device, the receive buffer timeout used to flush remaining data from the receive buffer was fixed at 16ms timeout. This timeout is now programmable over USB in 1ms increments from 1ms to 255ms, thus allowing the device to be better optimised for protocols requiring faster response times from short data packets.

TXDEN Timing fix

TXDEN timing has now been fixed to remove the external delay that was previously required for RS485 applications at high baud rates. TXDEN now works correctly during a transmit send-break condition.

Relaxed VCC Decoupling

The 2nd generation devices now incorporate a level of on-chip VCC decoupling. Though this does not eliminate the need for external decoupling capacitors, it significantly improves the ease of PCB design requirements to meet FCC, CE and other EMI related specifications.

Improved PreScaler Granularity

The previous version of the Prescaler supported division by (n + 0), (n + 0.125), (n + 0.25) and (n + 0.5) where n is an integer between 2 and 16,384 (2^{14}). To this we have added (n + 0.375), (n + 0.625), (n + 0.75) and (n+ 0.875) which can be used to improve the accuracy of some baud rates and generate new baud rates which were previously impossible (especially with higher baud rates).

Bit Bang Mode

The 2nd generation device has a new option referred to as "Bit Bang" mode. In Bit Bang mode, the eight UART interface control lines can be switched between UART interface mode and an 8-bit Parallel IO port. Data packets can be sent to the device and they will be sequentially sent to the interface at a rate controlled by the prescaler setting. As well as allowing the device to be used stand-alone as a general purpose IO controller for example controlling lights, relays and switches, some other interesting possibilities exist. For instance, it may be possible to connect the device to an SRAM configurable FPGA as supplied by vendors such as Altera and Xilinx. The FPGA device would normally be un-configured (i.e. have no defined function) at power-up. Application software on the PC could use Bit Bang Mode to download configuration data to the FPGA which would define its hardware function, then after the FPGA device is configured the FT232BL can switch back into UART interface mode to allow the programmed FPGA device to communicate with the PC over USB. This approach allows a customer to create a "generic" USB peripheral who's hardware function can be defined under control of the application software. The FPGA based hardware can be easily upgraded or totally changed simply by changing the FPGA configuration data file. Application notes, software and development modules for this application area will be available from FTDI and other 3rd parties.

PreScaler Divide By 1 Fix

The previous device had a problem when the integer part of the divisor was set to 1. In the 2nd generation device setting the prescaler value to 1 gives a baud rate of 2 million baud and setting it to zero gives a baud rate of 3 million baud. Non-integer division is not supported with divisor values of 0 and 1.

Less External Support Components

As well as eliminating the RCCLK RC network, and for most applications the need for an external reset circuit, we have also eliminated the requirement for a 100K pull-up on EECS to select 6MHz operation. When the FT232BL is being used without the configuration EEPROM, EECS, EESK and EEDATA can now be left n/c. For circuits requiring a long reset time (where the device is reset externally using a reset generator I.C., or reset is controlled by the IO port of a MCU, FPGA or ASIC device) an external transistor circuit is no longer required as the 1.5k pull-up resistor on USBDP can be wired to the RSTOUT# pin instead of to 3.3V. Note : RSTOUT# drives out at 3.3V level, not at 5V VCC level. This is the preferred configuration for new designs.

Extended EEPROM Support

The previous generation of devices only supported EEPROM of type 93C46 (64 x 16 bit). The new devices will also work with EEPROM type 93C56 (128 x 16 bit) and 93C66 (256 x 16 bit). The extra space is not used by the device, however it is available for use by other external MCU / logic whilst the FT232BL is being held in reset.

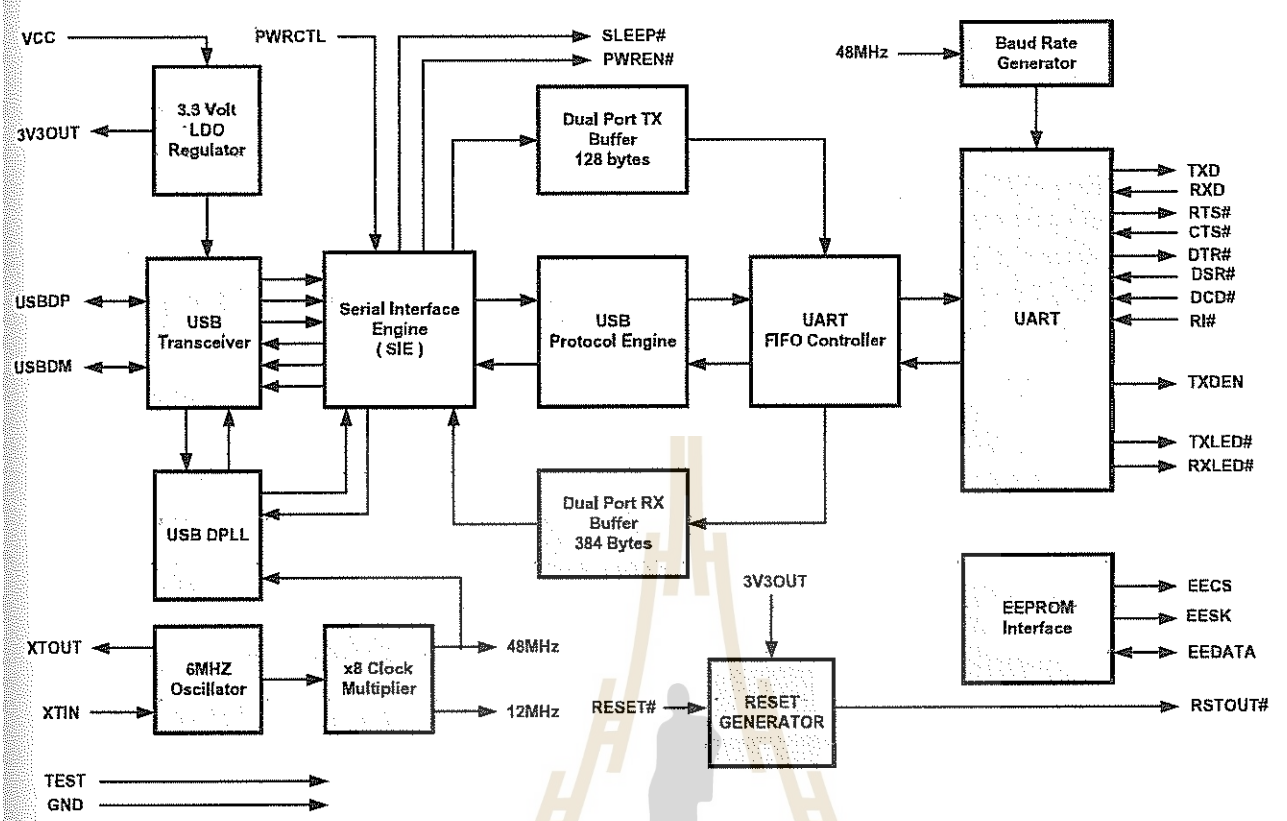
USB 2.0 (full speed option)

A new EEPROM based option allows the FT232BL to return a USB 2.0 device descriptor as opposed to USB 1.1. Note : The device would be a USB 2.0 Full Speed device (12Mb/s) as opposed to a USB 2.0 High Speed device (480Mb/s).

Multiple Device Support without EEPROM

When no EEPROM (or a blank or invalid EEPROM) is attached to the device, the FT232BL no longer gives a serial number as part of its USB descriptor. This allows multiple devices to be simultaneously connected to the same PC. However, we still highly recommend that EEPROM is used, as without serial numbers a device can only be identified by which hub port in the USB tree it is connected to which can change if the end user re-plugs the device into a different port.

3.0 Block Diagram (Simplified)



3.1 Functional Block Descriptions

- **3.3V LDO Regulator**

  The 3.3V LDO Regulator generates the 3.3 volt reference voltage for driving the USB transceiver cell output buffers. It requires an external decoupling capacitor to be attached to the 3V3OUT regulator output pin. It also provides 3.3V power to the RSTOUT# pin. The main function of this block is to power the USB Transceiver and the Reset Generator Cells rather than to power external logic. However, external circuitry requiring 3.3V nominal at a current of not greater than 5mA could also draw its power from the 3V3OUT pin if required.

- **USB Transceiver**

  The USB Transceiver Cell provides the USB 1.1 / USB 2.0 full-speed physical interface to the USB cable. The output drivers provide 3.3 volt level slew rate control signalling, whilst a differential receiver and two single ended receivers provide USB data in, SEO and USB Reset condition detection.

- **USB DPLL**

  The USB DPLL cell locks on to the incoming NRZI USB data and provides separate recovered clock and data signals to the SIE block.

- **6MHz Oscillator**

  The 6MHz Oscillator cell generates a 6MHz reference clock input to the x8 Clock multiplier from an external 6MHz crystal or ceramic resonator.

- **x8 Clock Multiplier**
  The x8 Clock Multiplier takes the 6MHz input from the Oscillator cell and generates a 12MHz reference clock for the SIE, USB Protocol Engine and UART FIFO controller blocks. It also generates a 48MHz reference clock for the USB DPPL and the Baud Rate Generator blocks.

- **Serial Interface Engine (SIE)**
  The Serial Interface Engine (SIE) block performs the Parallel to Serial and Serial to Parallel conversion of the USB data. In accordance to the USB 2.0 specification, it performs bit stuffing / un-stuffing and CRC5 / CRC16 generation / checking on the USB data stream.

- **USB Protocol Engine**
  The USB Protocol Engine manages the data stream from the device USB control endpoint. It handles the low level USB protocol (Chapter 9) requests generated by the USB host controller and the commands for controlling the functional parameters of the UART.

- **Dual Port TX Buffer (128 bytes)**
  Data from the USB data out endpoint is stored in the Dual Port TX buffer and removed from the buffer to the UART transmit register under control of the UART FIFO controller.

- **Dual Port RX Buffer (384 bytes)**
  Data from the UART receive register is stored in the Dual Port RX buffer prior to being removed by the SIE on a USB request for data from the device data in endpoint.

- **UART FIFO Controller**
  The UART FIFO controller handles the transfer of data between the Dual Port RX and TX buffers and the UART transmit and receive registers.

- **UART**
  The UART performs asynchronous 7 / 8 bit Parallel to Serial and Serial to Parallel conversion of the data on the RS232 (RS422 and RS485) interface. Control signals supported by the UART include RTS, CTS, DSR , DTR, DCD and RI. The UART provides a transmitter enable control signal (TXDEN) to assist with interfacing to RS485 transceivers. The UART supports RTS/ CTS, DSR/DTR and X-On/X-Off handshaking options. Handshaking, where required, is handled in hardware to ensure fast response times. The UART also supports the RS232 BREAK setting and detection conditions.

- **Baud Rate Generator**
  The Baud Rate Generator provides a x16 clock input to the UART from the 48MHz reference clock and consists of a 14 bit prescaler and 3 register bits which provide fine tuning of the baud rate (used to divide by a number plus a fraction). This determines the Baud Rate of the UART which is programmable from 183 baud to 3 million baud.

- **RESET Generator**
  The Reset Generator Cell provides a reliable power-on reset to the device internal circuitry on power up. An additional RESET# input and RSTOUT# output are provided to allow other devices to reset the FT232BL or the FT232BL to reset other devices respectively. During reset, RSTOUT# is driven low, otherwise it drives out at the 3.3V provided by the onboard regulator. RSTOUT# can be used to control the 1.5k pull-up on USBDP directly where delayed USB enumeration is required. It can also be used to reset other devices. RSTOUT# will stay high-impedance for approximately 5ms after VCC has risen above 3.5V AND the device oscillator is running AND RESET# is high. RESET# should be tied to VCC unless it is a requirement to reset the device from external logic or an external reset generator i.c.

- **EEPROM Interface**

  Though the FT232BL will work without the optional EEPROM, an external 93C46 (93C56 or 93C66) EEPROM can be used to customise the USB VID, PID, Serial Number, Product Description Strings and Power Descriptor value of the FT232BL for OEM applications. Other parameters controlled by the EEPROM include Remote Wake Up, Isochronous Transfer Mode, Soft Pull Down on Power-Off and USB 2.0 descriptor modes.
  The EEPROM should be a 16 bit wide configuration such as a MicroChip 93LC46B or equivalent capable of a 1Mb/s clock rate at VCC = 4.35V to 5.25V. The EEPROM is programmable-on board over USB using a utility available from FTDI's web site (http://www.ftdichip.com). This allows a blank part to be soldered onto the PCB and programmed as part of the manufacturing and test process.
  If no EEPROM is connected (or the EEPROM is blank), the FT232BL will use its built-in default VID, PID Product Description and Power Descriptor Value. In this case, the device will not have a serial number as part of the USB descriptor.

4.0 Device Pin-Out

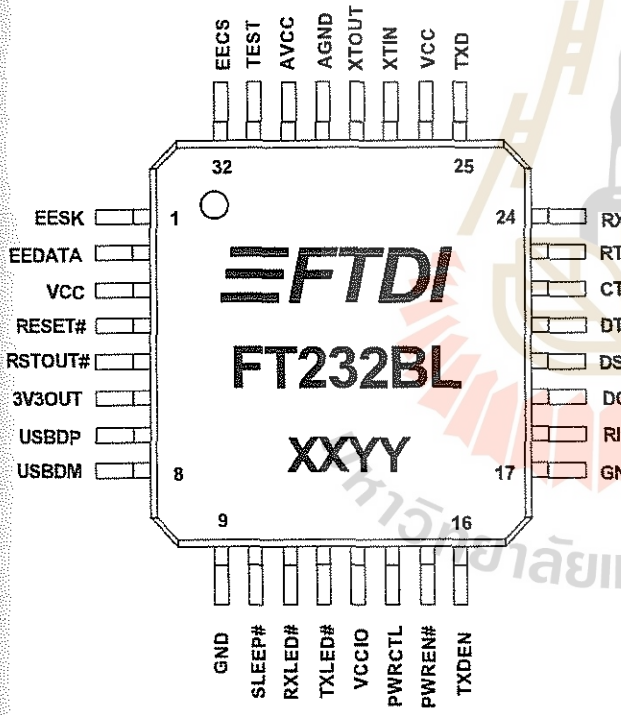


Figure 1
Pin-Out
(Lead free LQFP-32 Package)

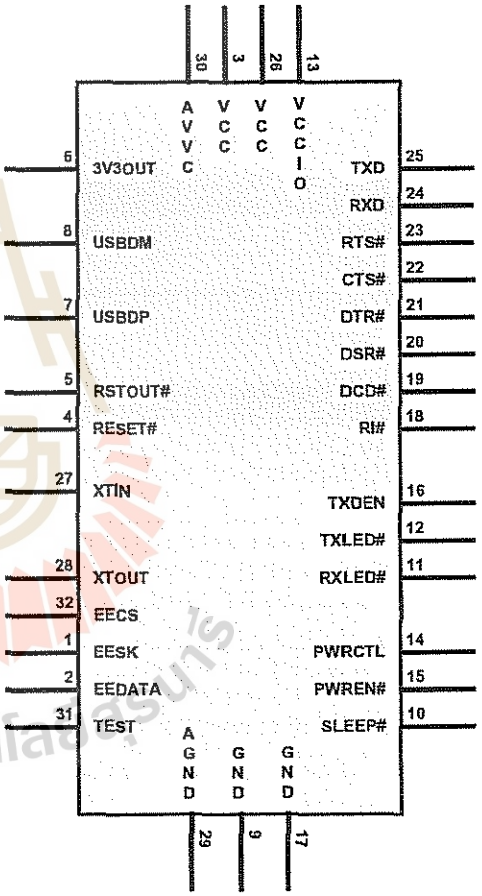


Figure 2
Pin-Out
(Schematic Symbol)

4.1 Signal Descriptions

Table 1 - FT232BL - PINOUT DESCRIPTION

UART INTERFACE GROUP

Pin#	Signal	Type	Description
25	TXD	OUT	Transmit Asynchronous Data Output
24	RXD	IN	Receive Asynchronous Data Input
23	RTS#	OUT	Request To Send Control Output / Handshake signal
22	CTS#	IN	Clear To Send Control Input / Handshake signal
21	DTR#	OUT	Data Terminal Ready Control Output / Handshake signal
20	DSR#	IN	Data Set Ready Control Input / Handshake signal
19	DCD#	IN	Data Carrier Detect Control Input
18	RI#	IN	Ring Indicator Control Input. When the Remote Wakeup option is enabled in the EEPROM, taking RI# low can be used to resume the PC USB Host controller from suspend.
16	TXDEN	OUT	Enable Transmit Data for RS485

USB INTERFACE GROUP

Pin#	Signal	Type	Description
7	USBDP	I/O	USB Data Signal Plus (Requires 1.5k pull-up to 3V3OUT or RSTOUT#)
8	USBDM	I/O	USB Data Signal Minus

EEPROM INTERFACE GROUP

Pin#	Signal	Type	Description
32	EECS	I/O	EEPROM – Chip Select. For 48MHz operation pull EECS to GND using a 10K resistor. For 6MHz operation no resistor is required. Tri-State during device reset. **Note 1
1	EESK	OUT	Clock signal to EEPROM. Tri-State during device reset, else drives out. Adding a 10K pull down resistor onto EESK will cause the FT232BL to use USB Product ID 6004 (hex) instead of 6001 (hex). All of the other USB device descriptors are unchanged.**Note 1
2	EEDATA	I/O	EEPROM – Data I/O Connect directly to Data-In of the EEPROM and to Data-Out of the EEPROM via a 2.2K resistor. Also, pull Data-Out of the EEPROM to VCC via a 10K resistor for correct operation. Tri-State during device reset. **Note 1

POWER CONTROL GROUP

Pin#	Signal	Type	Description
10	SLEEP#	OUT	Goes Low during USB Suspend Mode. Typically used to power-down an external TTL to RS232 level converter i.c. in USB <=> RS232 converter designs.
15	PWREN#	OUT	Goes Low after the device is configured via USB, then high during USB suspend. Can be used to control power to external logic using a P-Channel Logic Level MOSFET switch. Enable the Interface Pull-Down Option in EEPROM when using the PWREN# pin in this way.
14	PWRCTL	IN	Bus Powered – Tie Low / Self Powered – Tie High (to VCCIO)

MISCELLANEOUS SIGNAL GROUP

Pin#	Signal	Type	Description
4	RESET#	IN	Can be used by an external device to reset the FT232BL. If not required, tie to VCC.
5	RSTOUT#	OUT	Output of the internal Reset Generator. Stays high impedance for ~ 5ms after VCC > 3.5V and the internal clock starts up, then clamps its output to the 3.3v output of the internal regulator. Taking RESET# low will also force RSTOUT# to drive low. RSTOUT# is NOT affected by a USB Bus Reset.
12	TXLED#	O.C.	LED Drive - Pulses Low when Transmitting Data via USB
11	RXLED#	O.C.	LED Drive - Pulses Low when Receiving Data via USB
27	XTIN	IN	Input to 6MHz Crystal Oscillator Cell. This pin can also be driven by an external 6MHz clock if required. Note : Switching threshold of this pin is VCC/2, so if driving from an external source, the source must be driving at 5V CMOS level or a.c. coupled to centre around VCC/2.
28	XTOUT	OUT	Output from 6MHz Crystal Oscillator Cell. XTOUT stops oscillating during USB suspend, so take care if using this signal to clock external logic.
31	TEST	IN	Puts device in I.C. test mode – must be tied to GND for normal operation.

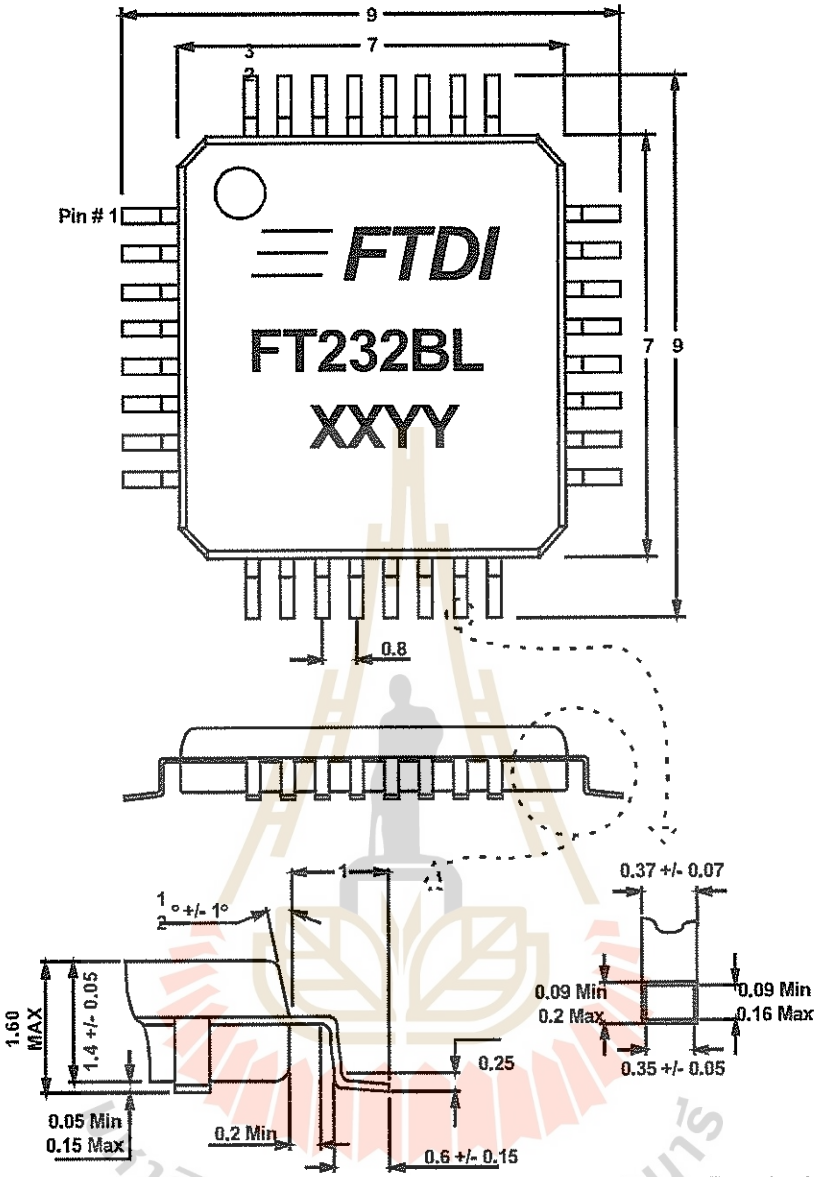
POWER AND GND GROUP

Pin#	Signal	Type	Description
6	3V3OUT	OUT	3.3 volt Output from the integrated L.D.O. regulator This pin should be decoupled to GND using a 33nF ceramic capacitor in close proximity to the device pin. Its prime purpose is to provide the internal 3.3V supply to the USB transceiver cell and the RSTOUT# pin. A small amount of current (<= 5mA) can be drawn from this pin to power external 3.3v logic if required.
3,26	VCC	PWR	+4.35 volt to +5.25 volt VCC to the device core, LDO and non-UART interface pins.
13	VCCIO	PWR	+3.0 volt to +5.25 volt VCC to the UART interface pins 10..12, 14..16 and 18..25. When interfacing with 3.3V external logic in a bus powered design connect VCCIO to a 3.3V supply generated from the USB bus. When interfacing with 3.3V external logic in a self powered design connect VCCIO to the 3.3V supply of the external logic. Otherwise connect to VCC to drive out at 5V CMOS level.
9,17	GND	PWR	Device - Ground Supply Pins
30	AVCC	PWR	Device - Analog Power Supply for the internal x8 clock multiplier
29	AGND	PWR	Device - Analog Ground Supply for the internal x8 clock multiplier

**Note 1 - During device reset, these pins are tri-state but pulled up to VCC via internal 200K resistors.

5.0 Package Outline

Figure 3 – 32 LD Lead Free LQFP Package Dimensions



The FT232BL is supplied in a 32 pin lead free LQFP package. This package has a 7mm x 7mm body (9mm x 9mm including leads) with leads on a 0.8mm pitch. The above drawing shows the LQFP-32 package – all dimensions are in millimetres. Note that there are two date code formats used - XXYY = Date Code where XX = 2 digit year number, YY = 2 digit week number; or XYY-1 where X = 1 digit year. number, YY = 2 digit week number.

The FT232BL is fully compliant with the European Union RoHS directive.

An alternative 5mm x 5mm leadless QFN32 package is also available for projects where package area is critical. Part numeber for this version is FT232BQ. The FT232BQ is also a lead free package. See their seperate datasheets for package dimensions.

6.0 Absolute Maximum Ratings

These are the absolute maximum ratings for the FT232BL device in accordance with the Absolute Maximum Rating System (IEC 60134). Exceeding these may cause permanent damage to the device.

Parameter	Value	Units
Storage Temperature	–65°C to + 150°C	Degrees C
Floor Life (Out of Bag) at Factory Ambient (30°C/60% Relative Humidity)	192 Hours (Level 3 Compliant) **Note 2	
Ambient Temperature (Power Applied)	0°C to + 70°C	Degrees C
M.T.B.F. (at 35°C)	247484 Hours ≈ 28 Years	
VCC Supply Voltage	-0.5 to +6.00	V
D.C. Input Voltage - USBDP and USBDM	-0.5 to +3.8	V
D.C. Input Voltage - High Impedance Bidirectionals	-0.5 to +(Vcc +0.5)	V
D.C. Input Voltage - All other Inputs	-0.5 to +(Vcc +0.5)	V
DC Output Current – Outputs	24	mA
DC Output Current – Low Impedance Bidirectionals	24	mA
Power Dissipation (VCC = 5.25V)	500	mW
Electrostatic Discharge Voltage (Human Body Model) (I < 1uA)	+/- 3000	V
Latch Up Current (Vi = +/- 10V maximum, for 10 ms)	+/-200	mA

**Note 2 – *If devices are stored out of the packaging beyond this time limit the devices should be baked before use. The devices should be ramped up to a temperature of 110°C and baked for 8 to 10 hours.*

6.1 D.C. Characteristics

DC Characteristics (Ambient Temperature = 0 to 70°C)

Operating Voltage and Current

Parameter	Description	Min	Typ	Max	Units	Conditions
Vcc1	VCC Operating Supply Voltage	4.35	5.0	5.25	V	
Vcc2	VCCIO Operating Supply Voltage	3.0	-	5.25	V	
Icc1	Operating Supply Current	-	25	-	mA	Normal Operation
Icc2	Operating Supply Current	-	180	200	uA	USB Suspend **Note 3

**Note 3 – *Supply current excludes the 200uA nominal drawn by the external pull-up resistor on USBDP.*

UART IO Pin Characteristics (VCCIO = 5.0V)

Parameter	Description	Min	Typ	Max	Units	Conditions
Voh	Output Voltage High	3.2	4.1	4.9	V	I source = 2mA
Vol	Output Voltage Low	0.3	0.4	0.6	V	I sink = 2mA
Vin	Input Switching Threshold	1.3	1.6	1.9	V	**Note 4
VHys	Input Switching Hysteresis	50	55	60	mV	

UART IO Pin Characteristics (VCCIO = 3.0 - 3.6V)

Parameter	Description	Min	Typ	Max	Units	Conditions
Voh	Output Voltage High	2.2	2.7	3.2	V	I source = 1mA
Vol	Output Voltage Low	0.3	0.4	0.5	V	I sink = 2 mA
Vin	Input Switching Threshold	1.0	1.2	1.5	V	**Note 4
VHys	Input Switching Hysteresis	20	25	30	mV	

**Note 4 – *Inputs have an internal 200K pull-up resistor to VCCIO.*

XTIN / XTOUT Pin Characteristics

Parameter	Description	Min	Typ	Max	Units	Conditions
Voh	Output Voltage High	4.0	-	5.0	V	Fosc = 6MHz
Vol	Output Voltage Low	0.1	-	1.0	V	Fosc = 6MHz
Vin	Input Switching Threshold	1.8	2.5	3.2	V	

RESET#, TEST, EECS, EESK, EEDATA Pin Characteristics

Parameter	Description	Min	Typ	Max	Units	Conditions
Voh	Output Voltage High	3.2	4.1	4.9	V	I source = 2mA
Vol	Output Voltage Low	0.3	0.4	0.6	V	I sink = 2 mA
Vin	Input Switching Threshold	1.3	1.6	1.9	V	**Note 5
VHys	Input Switching Hysteresis	50	55	60	mV	

**Note 5 – *EECS, EESK and EEDATA pins have an internal 200K pull-up resistor to VCC*

RSTOUT Pin Characteristics

Parameter	Description	Min	Typ	Max	Units	Conditions
Voh	Output Voltage High	3.0	-	3.6	V	I source = 2mA
Vol	Output Voltage Low	0.3	-	0.6	V	I sink = 2mA

USB IO Pin Characteristics

Parameter	Description	Min	Typ	Max	Units	Conditions
UVoh	IO Pins Static Output (High)	2.8		3.6	V	RI = 1.5K to 3V3Out (D+) RI = 15K to GND (D-)
UVol	IO Pins Static Output (Low)	0		0.3	V	RI = 1.5K to 3V3Out (D+) RI = 15K to GND (D-)
UVse	Single Ended Rx Threshold	0.8		2.0	V	
UCom	Differential Common Mode	0.8		2.5	V	
UVDif	Differential Input Sensitivity	0.2			V	
UDrvZ	Driver Output Impedance	29		44	Ohm	**Note 5

**Note 5 – *Driver Output Impedance includes the external 27R series resistors on USBDP and USBDM pins.*

7.0 Device Configuration Examples

7.1 Oscillator Configurations

Figure 4
3-Pin Ceramic Resonator Configuration

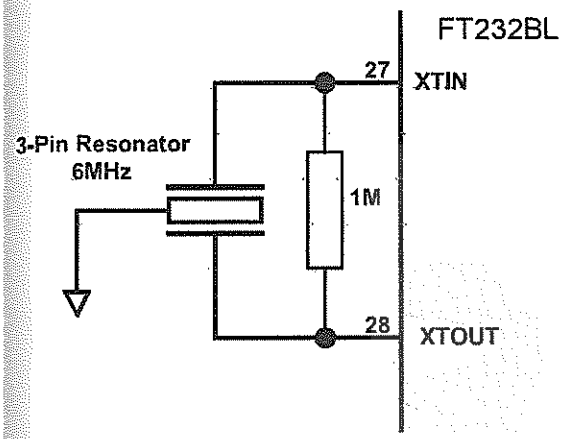


Figure 5
Crystal or 2-Pin Ceramic Resonator Configuration

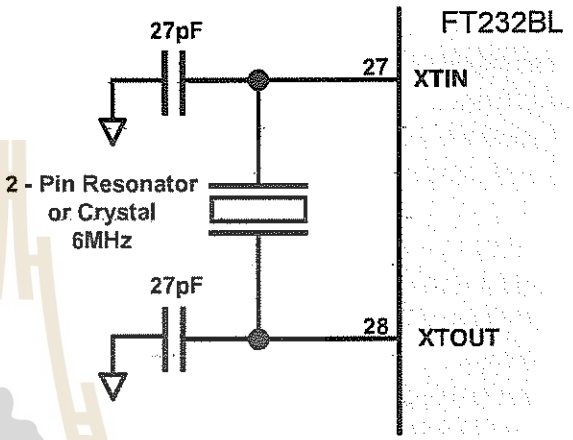


Figure 4 illustrates how to use the FT232BL with a 3-Pin Ceramic Resonator. A suitable part would be a ceramic resonator from Murata's CERALOCK range. (Murata Part Number CSTCR6M00G15), or equivalent. 3-Pin ceramic resonators have the load capacitors built into the resonator so no external loading capacitors are required. This makes for an economical configuration. The accuracy of this Murata ceramic resonator is +/- 0.1% and it is specifically designed for USB full speed applications. A 1 MegaOhm loading resistor across XTIN and XTOUT is recommended in order to guarantee this level of accuracy.

Other ceramic resonators with a lesser degree of accuracy (typically +/- 0.5%) are technically out-with the USB specification, but it has been calculated that using such a device will work satisfactorily in practice with a FT232BL design.

Figure 5 illustrates how to use the FT232BL with a 6MHz Crystal or 2-Pin Ceramic Resonator. In this case, these devices do not have in-built loading capacitors so these have to be added between XTIN, XTOUT and GND as shown. A value of 27pF is shown as the capacitor in the example – this will be good for many crystals and some resonators but do select the value based on the manufacturers recommendations wherever possible. If using a crystal, use a parallel cut type. If using a resonator, see the previous note on frequency accuracy.

7.2 EEPROM Configuration

Figure 6
EEPROM Configuration

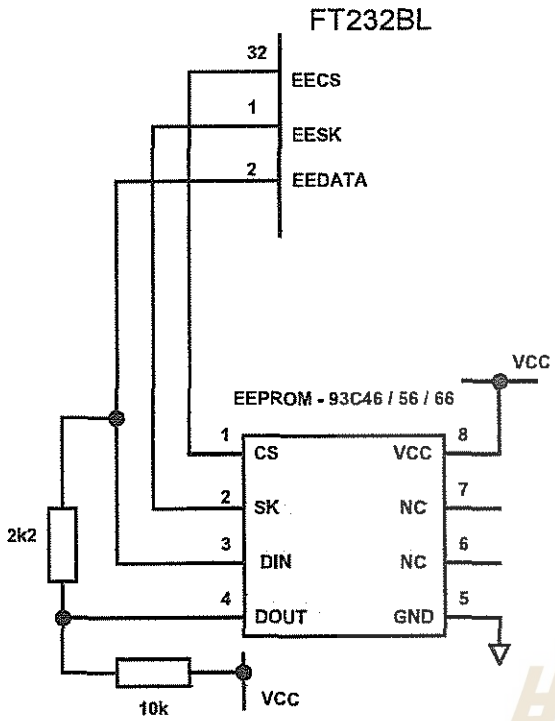


Figure 6 illustrates how to connect the FT232BL to the 93C46 (93C56 or 93C66) EEPROM. EECS (pin 32) is directly connected to the chip select (CS) pin of the EEPROM. EESK (pin 1) is directly connected to the clock (SK) pin of the EEPROM. EEDATA (pin 2) is directly connected to the Data In (Din) pin of the EEPROM. There is a potential condition whereby both the Data Output (Dout) of the EEPROM can drive out at the same time as the EEDATA pin of the FT232BL. To prevent potential data clash in this situation, the Dout of the EEPROM is connected to EEDATA of the FT232BL via a 2.2K resistor.

Following a power-on reset or a USB reset, the FT232BL will scan the EEPROM to find out (a) if an EEPROM is attached to the Device and (b) if the data in the device is valid. If both of these are the case, then the FT232BL will use the data in the EEPROM, otherwise it will use its built-in default values. When a valid command is issued to the EEPROM from the FT232BL, the EEPROM will acknowledge the command by pulling its Dout pin low. In order to check for this condition, it is necessary to pull Dout high using a 10K resistor. If the command acknowledge doesn't happen then EEDATA will be pulled high by the 10K resistor during this part of the cycle and the device will detect an invalid command or no EEPROM present.

There are two varieties of these EEPROM's on the market – one is configured as being 16 bits wide, the other is configured as being 8 bits wide. These are available from many sources such as Microchip, STMicro, ISSI etc. The FT232BL requires EEPROM's with a 16-bit wide configuration such as the Microchip 93LC46B device. The EEPROM must be capable of reading data at a 1Mb clock rate at a supply voltage of 4.35V to 5.25V. Most available parts are capable of this.

Check the manufacturers data sheet to find out how to connect pins 6 and 7 of the EEPROM. Some devices specify these as no-connect, others use them for selecting 8 / 16 bit mode or for test functions. Some other parts have their pinout rotated by 90° so please select the required part and its options carefully.

It is possible to "share" the EEPROM between the FT232BL and another external device such as an MCU. However, this can only be done when the FT232BL is in its reset condition as it tri-states its EEPROM interface at that time. A typical configuration would use four bit's of an MCU IO Port. One bit would be used to hold the FT232BL reset (using RESET#) on power-up, the other three would connect to the EECS, EESK and EEDATA pins of the FT232BL in order to read / write data to the EEPROM at this time. Once the MCU has read / written the EEPROM, it would take RESET# high to allow the FT232BL to configure itself and enumerate over USB.

7.3 USB Bus Powered and Self Powered Configuration

Figure 7
USB Bus Powered Configuration

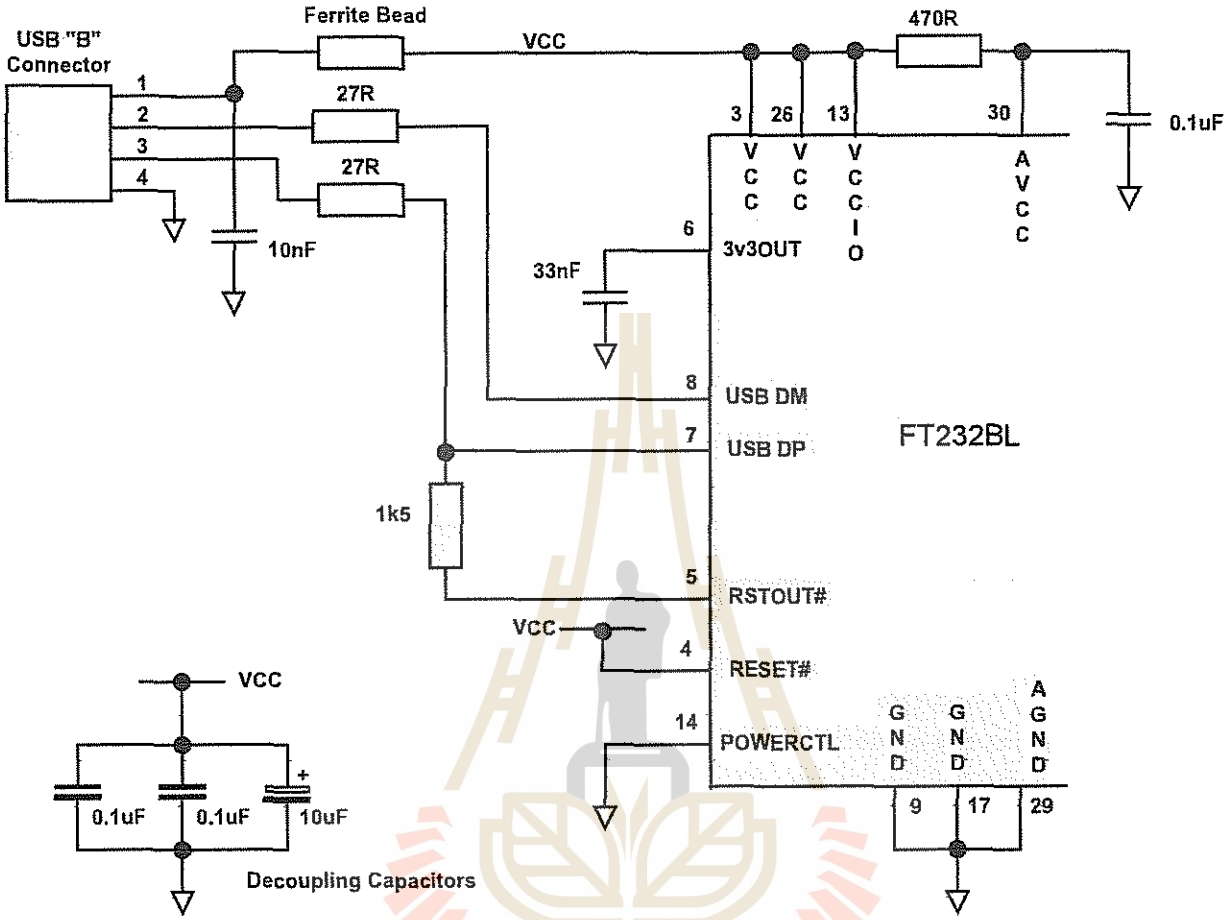


Figure 7 illustrates a typical USB bus powered configuration. A USB Bus Powered device gets its power from the USB bus. Basic rules for USB Bus power devices are as follows –

a) On plug-in, the device must draw no more than 100mA
b) On USB Suspend the device must draw no more than 500uA.
c) A Bus Powered High Power Device (one that draws more than 100mA) should use the PWREN# pin to keep the current below 100mA on plug-in and 500uA on USB suspend.
d) A device that consumes more than 100mA can not be plugged into a USB Bus Powered Hub
e) No device can draw more that 500mA from the USB Bus.

PWRCTL (pin 14) is pulled low to tell the device to use a USB Bus Power descriptor. The power descriptor in the EEPROM should be programmed to match the current draw of the device.

A Ferrite Bead is connected in series with USB power to prevent noise from the device and associated circuitry (EMI) being radiated down the USB cable to the Host. The value of the Ferrite Bead depends on the total current required by the circuit – a suitable range of Ferrite Beads is available from Steward (www.steward.com) for example Steward Part # MI0805K400R-00 also available as DigiKey Part # 240-1035-1.

Figure 8
USB Self Powered Configuration

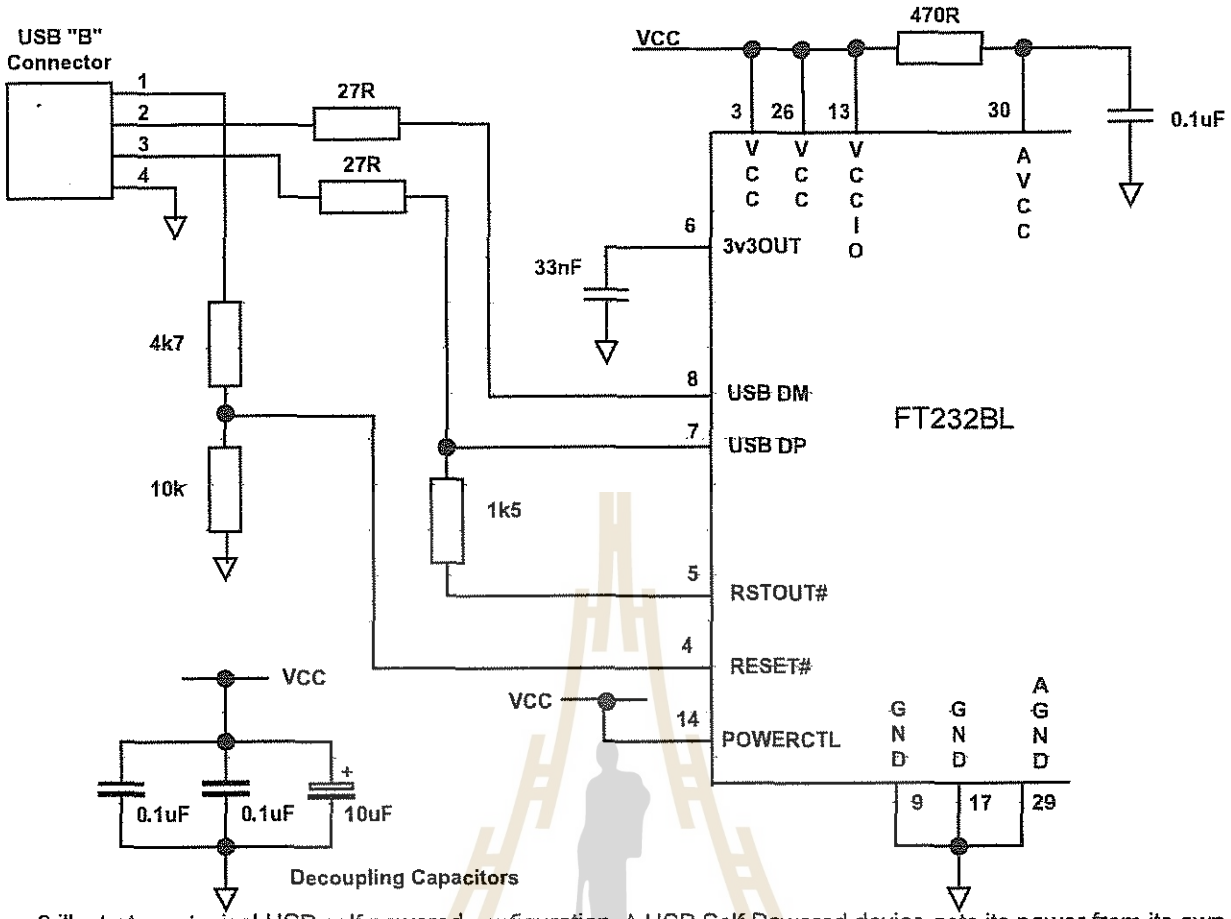


Figure 8 illustrates a typical USB self powered configuration. A USB Self Powered device gets its power from its own POWER SUPPLY and does not draw current from the USB bus. The basic rules for USB Self power devices are as follows –

a) A Self-Powered device should not force current down the USB bus when the USB Host or Hub Controller is powered down.

b) A Self Powered Device can take as much current as it likes during normal operation and USB suspend as it has its own POWER SUPPLY.

c) A Self Powered Device can be used with any USB Host and both Bus and Self Powered USB Hubs

PWRCTL (pin 14) is pulled high to tell the device to use a USB Bus Power descriptor. The power descriptor in the EEPROM should be programmed to a value of zero. The USB power descriptor option in the EEPROM should be programmed to a value of zero (self powered).

To meet requirement a) the 1.5K pull-up resistor on USBDP is connected to RSTOUT# as per the bus-power circuit. However, the USB Bus Power is used to control the RESET# Pin of the FT232BL device. When the USB Host or Hub is powered up RSTOUT# will pull the 1.5K resistor on USBDP to 3.3V, thus identifying the device as a full speed device to USB. When the USB Host or Hub power is off, RESET# will go low and the device will be held in reset. As RESET# is low, RSTOUT# will also be low, so no current will be forced down USBDP via the 1.5K pull-up resistor when the host or hub is powered down. Failure to do this may cause some USB host or hub controllers to power up erratically.

Note : When the FT232BL is in reset, the UART interface pins all go tri-state. These pins have internal 200K pull-up resistors to VCCIO, so they will gently pull high unless driven by some external logic.

7.4 UART Interface Configuration

Figure 9
USB <=> RS232 Converter Configuration

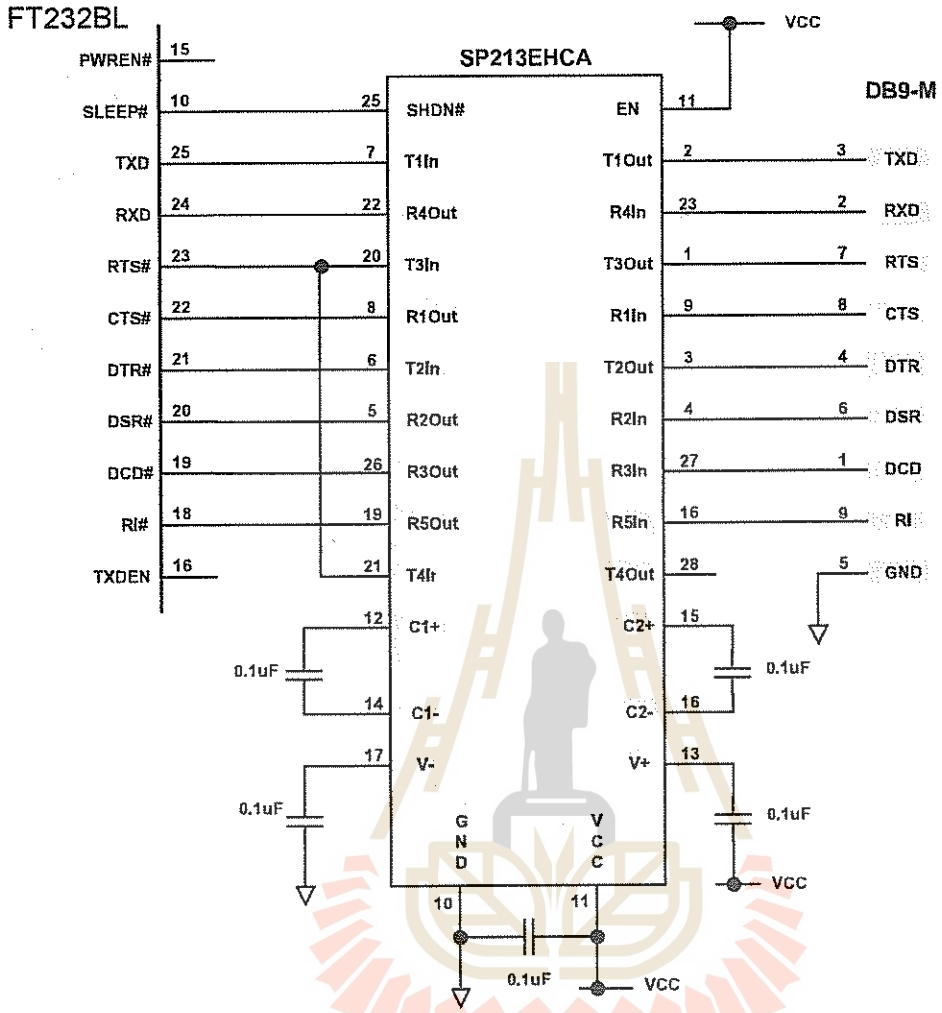


Figure 9 illustrates how to connect the UART interface of the FT232BL to a TTL – RS232 Level Converter I.C. to make a USB <=> RS232 converter using the popular "213" series of TTL to RS232 level converters. These devices have 4 transmitters and 5 receivers in a 28 LD SSOP package and feature an in-built voltage converter to convert the 5v (nominal) VCC to the +/- 9 volts required by RS232. An important feature of these devices is the SHDN# pin which can power down the device to a low quiescent current during USB suspend mode

The device used in the example is a Sipex SP213EHCA which is capable of RS232 communication at up to 500K baud. If a lower baud rate is acceptable, then several pin compatible alternatives are available such as Sipex SP213ECA , Maxim MAX213CAI and Analog Devices ADM213E which are good for communication at up to 115,200 baud. If a higher baud rate is desired, use a Maxim MAX3245CAI part which is capable of RS232 communication at rates of up to 1M baud. The MAX3245 is not pin compatible with the 213 series devices, also its SHDN pin is active high so connect this to PWREN# instead of SLEEP#.

Figure 10
USB <=> RS422 Converter Configuration

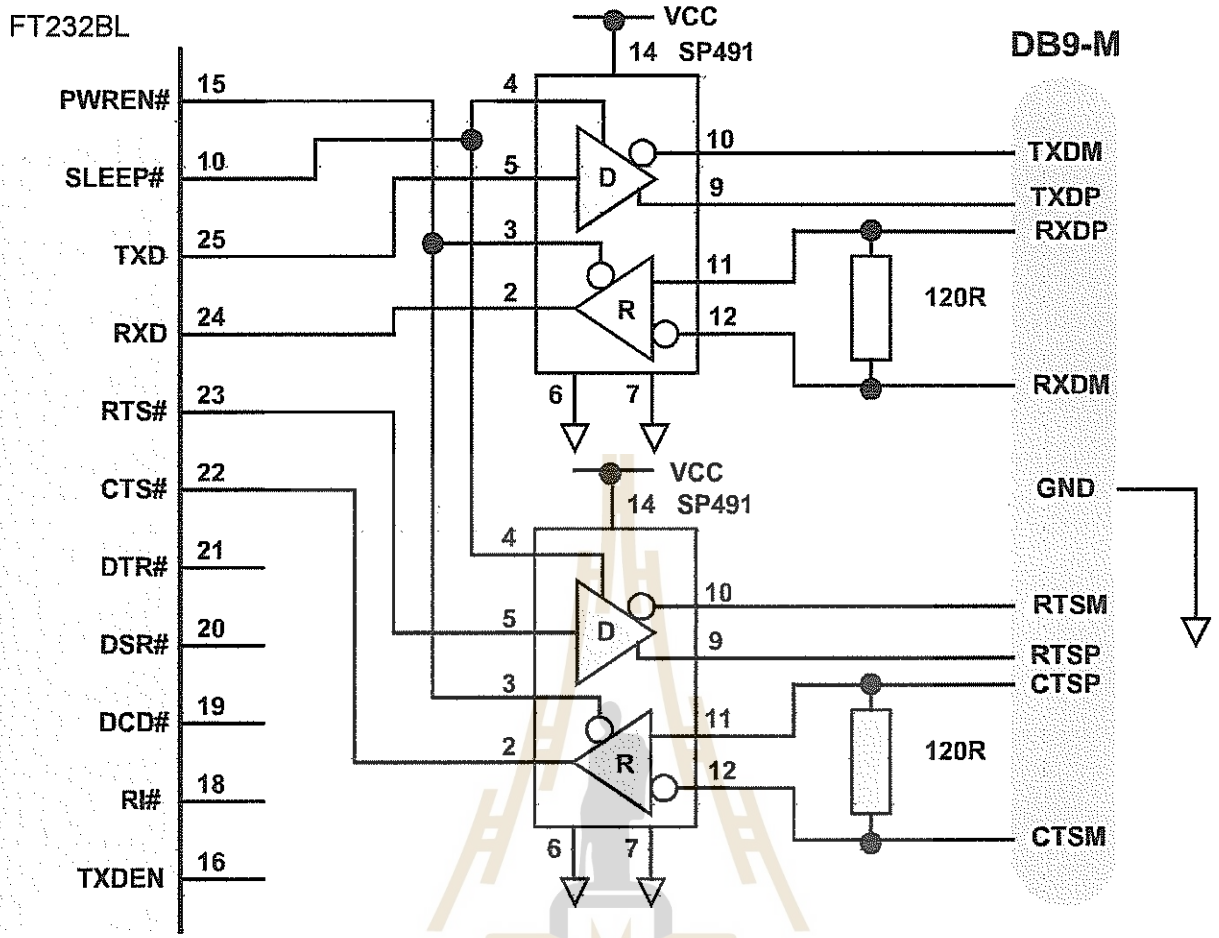


Figure 10 illustrates how to connect the UART interface of the FT232BL to a TTL – RS422 Level Converter I.C. to make a USB <=> RS422 converter. There are many such level converter devices available – this example uses Sipex SP491 devices which have enables on both the transmitter and receiver. Because the transmitter enable is active high, it is connected to the SLEEP# pin. The receiver enable is active low and is connected to the PWREN# pin. This ensures that both the transmitters and receivers are enabled when the device is active, and disabled when the device is in USB suspend mode. If the design is USB BUS powered, it may be necessary to use a P-Channel logic level MOSFET (controlled by PWREN#) in the VCC line of the SP491 devices to ensure that the USB standby current of 500uA is met.

The SP491 is good for sending and receiving data at a rate of up to 5M Baud – in this case the maximum rate is limited to 3M Baud by the FT232BL.

Figure 11
USB <=> RS485 Converter Configuration

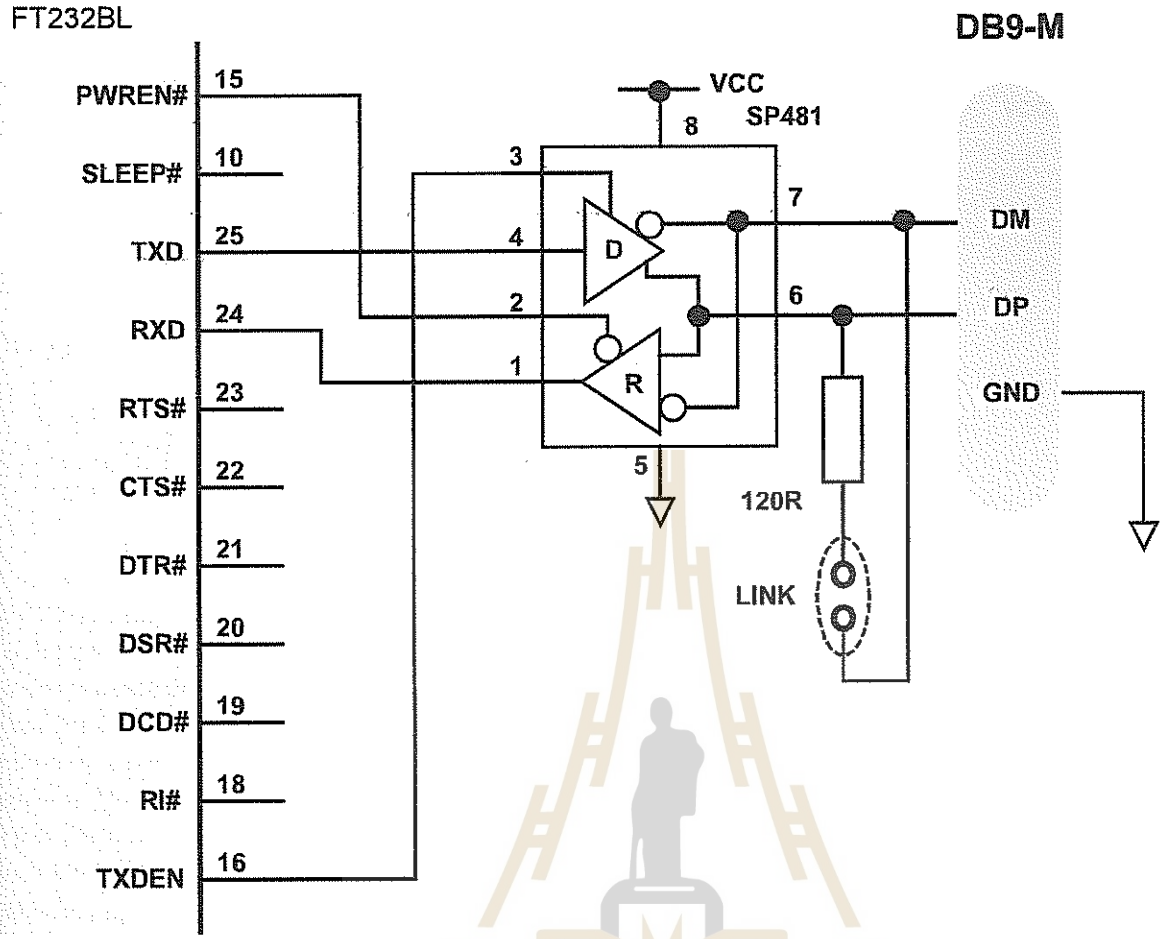


Figure 11 illustrates how to connect the UART interface of the FT232BL to a TTL – RS485 Level Converter I.C. to make a USB => RS485 converter. This example uses the Sipex SP481 device but there are similar parts available from Maxim and Analog Devices amongst others. The SP481 is a RS485 device in a compact 8 pin SOP package. It has separate enables on both the transmitter and receiver. With RS485, the transmitter is only enabled when a character is being transmitted from the UART. The TXDEN pin on the FT232BL is provided for exactly that purpose and so the transmitter enable is wired to TXDEN. The receiver enable is active low, so it is wired to the PWREN# pin to disable the receiver when in USB suspend mode.

RS485 is a multi-drop network – i.e. many devices can communicate with each other over a single two wire cable connection. The RS485 cable requires to be terminated at each end of the cable. A link is provided to allow the cable to be terminated if the device is physically positioned at either end of the cable.

In this example the data transmitted by the FT232BL is also received by the device that is transmitting. This is a common feature of RS485 and requires the application software to remove the transmitted data from the received data stream. With the FT232BL it is possible to do this entirely in hardware – simply modify the schematic so that RXD of the FT232BL is the logical OR of the SP481 receiver output with TXDEN using an HC32 or similar logic gate.

7.5 LED Interface

Figure 12
Dual LED Configuration

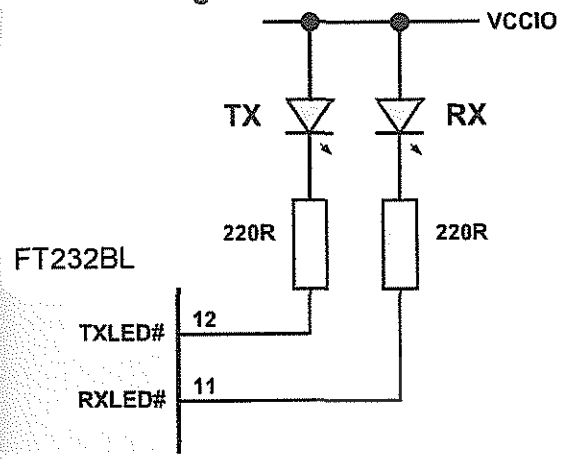
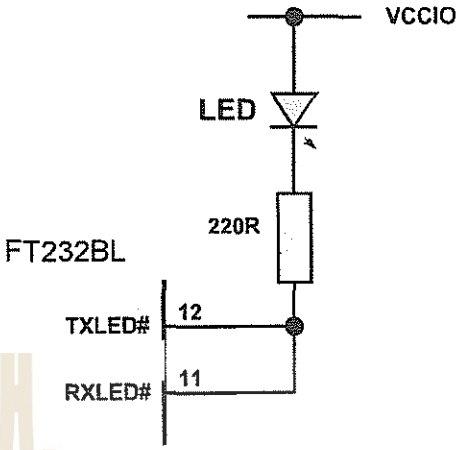


Figure 13
Single LED Configuration



The FT232BL has two IO pins dedicated to controlling LED status indicators, one for transmitted data the other for received data. When data is being transmitted / received the respective pins drive from tri-state to low in order to provide indication on the LEDs of data transfer. A digital one-shot timer is used so that even a small percentage of data transfer is visible to the end user. Figure 12 shows a configuration using two individual LED's – one for transmitted data the other for received data. In Figure 13, the transmit and receive LED indicators are wire-OR'ed together to give a single LED indicator which indicates any transmit or receive data activity.
Another possibility (not shown here) is to use a 3 pin common anode tri-color LED based on the circuit in Figure 13 to have a single LED that can display activity in a variety of colors depending on the ratio of transmit activity compared to receive activity.

Note that the LED's are connected to VCCIO.

7.6 Interfacing to 3.3v Logic

Figure 14
Bus Powered Circuit with 3.3V logic drive / supply voltage

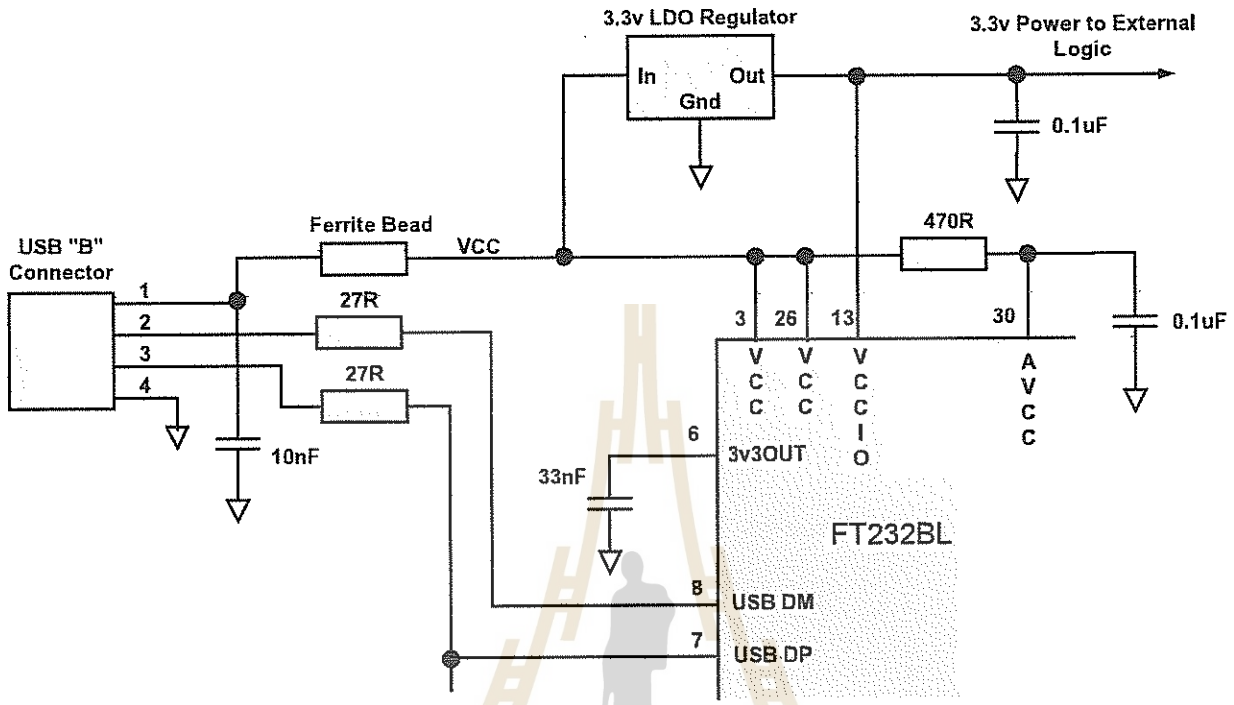


Figure 14 shows how to configure the FT232BL to interface with a 3.3V logic device. In this example, a discrete 3.3V regulator is used to supply the 3.3V logic from the USB supply. VCCIO is connected to the output of the 3.3V regulator, which in turn will cause the UART interface IO pins to drive out at 3.3V level. For USB bus powered circuits some considerations have to be taken into account when selecting the regulator –

a) The regulator must be capable of sustaining its output voltage with an input voltage of 4.35 volts. A Low Drop Out (LDO) regulator must be selected.
b) The quiescent current of the regulator must be low in order to meet the USB suspend total current requirement of <= 500uA during USB suspend.

An example of a regulator family that meets these requirements is the MicroChip (Telcom) TC55 Series. These devices can supply up to 250mA current and have a quiescent current of under 1uA.

In some cases, where only a small amount of current is required (< 5mA) , it may be possible to use the in-built regulator of the FT232BL to supply the 3.3v without any other components being required. In this case, connect VCCIO to the 3v3OUT pin of the FT232BL.

Note : It should be emphasised that the 3.3V supply for VCCIO in a bus powered design with a 3.3V logic interface should come from an LDO which is supplied by the USB bus, or from the 3V3OUT pin of the FT232BL, and not from any other source.

Figure 15
Self Powered Circuit with 3.3V logic drive / supply voltage

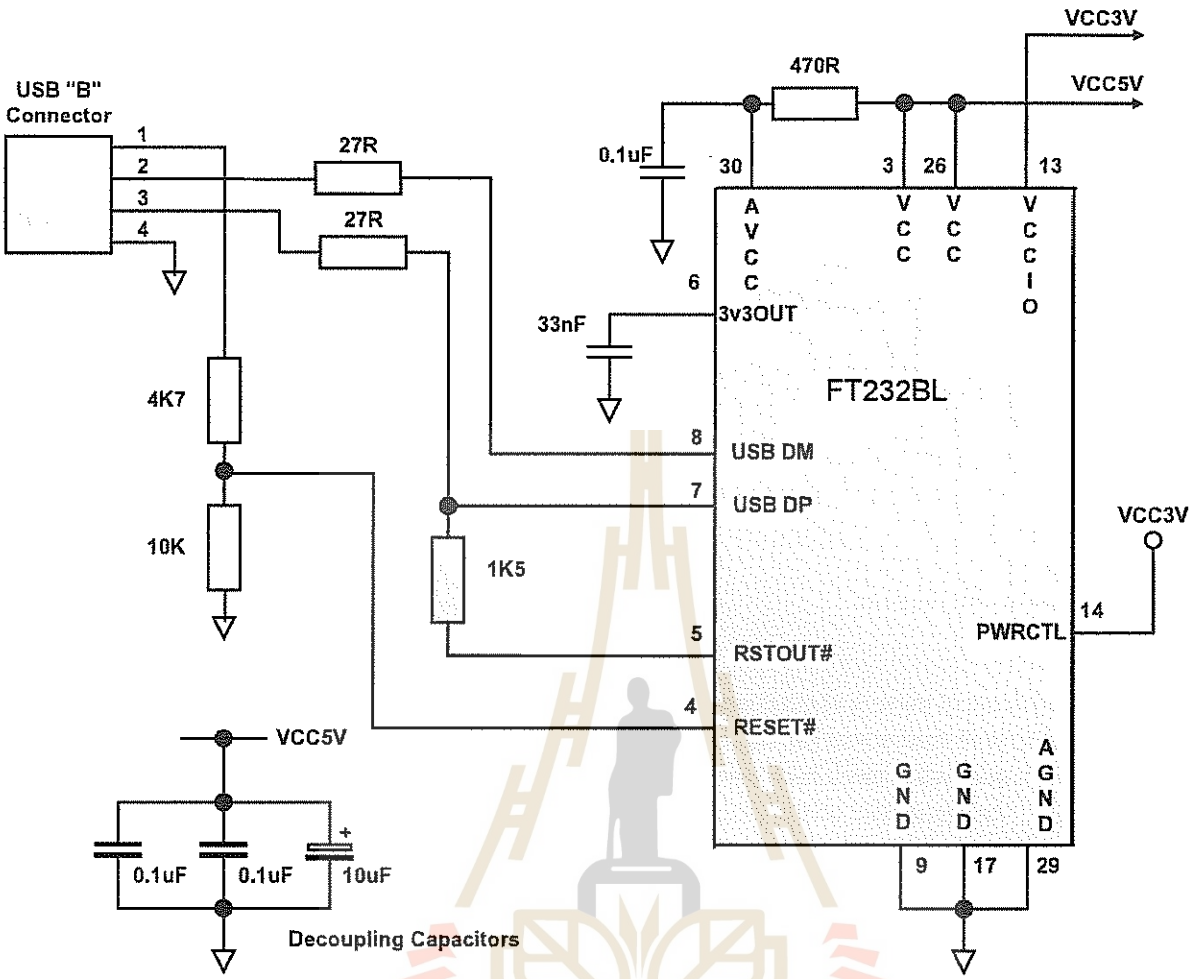


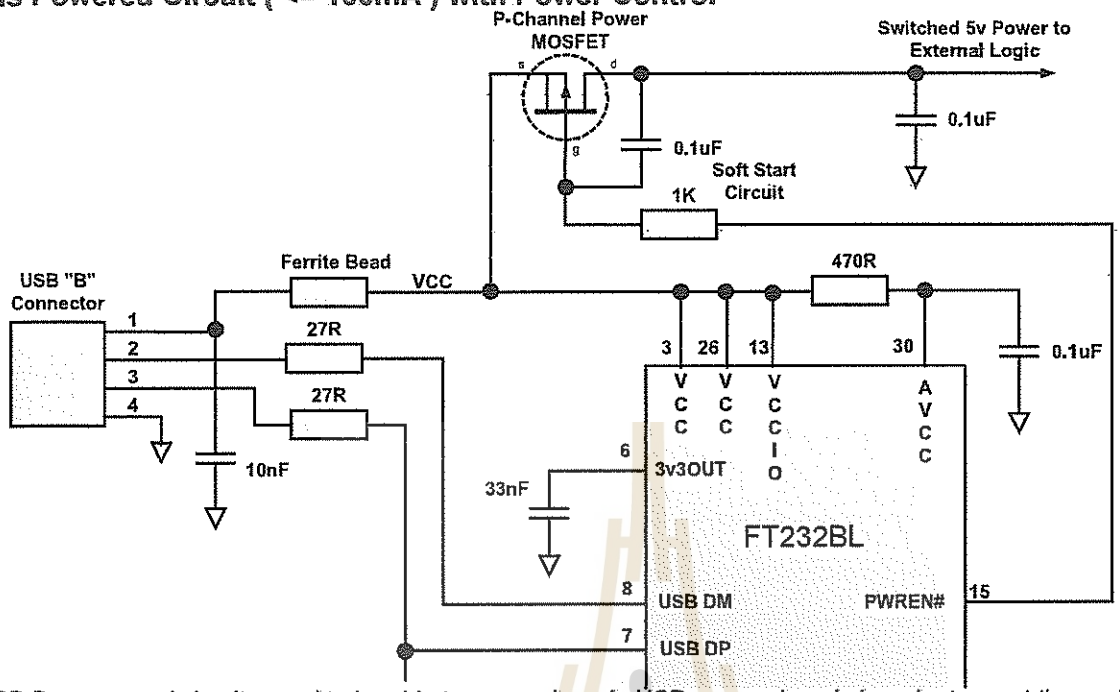
Figure 15 is an example of a USB self powered design with 3.3V interface. In this case VCCIO is supplied by an external 3.3V supply in order to make the device IO pins drive out at 3.3V logic level, thus allowing it to be connected to a 3.3V MCU or other external logic. A USB self powered design uses its own power supplies, and does not draw any of its power from the USB bus. In such cases, no special care need be taken to meet the USB suspend current (0.5 mA) as the device does not get its power from the USB port.

As with bus powered 3.3V interface designs, in some cases, where only a small amount of current is required (<5mA), it may be possible to use the in-built regulator of the FT232BL to supply the 3.3V without any other components being required. In this case, connect VCCIO to the 3v3OUT pin of the FT232BL.

Note that in this case PWRCTL is pulled up to VCCIO, not VCC.

7.7 Power Switching

Figure 16
Bus Powered Circuit (<= 100mA) with Power Control



USB Bus powered circuits need to be able to power down in USB suspend mode in order to meet the <= 500uA total suspend current requirement (including external logic). Some external logic can power itself down into a low current state by monitoring the PWREN# pin. For external logic that cannot power itself down in that way, the FT232BL provides a simple but effective way of turning off power to external circuitry during USB suspend.

Figure 16 shows how to use a discrete P-Channel Logic Level MOSFET to control the power to external logic circuits. A suitable device could be a Fairchild NDT456P, or International Rectifier IRLML6402, or equivalent. It is recommended that a "soft start" circuit consisting of a 1K series resistor and a 0.1 uF capacitor are used to limit the current surge when the MOSFET turns on. Without the soft start circuit there is a danger that the transient power surge of the MOSFET turning on will reset the FT232BL, or the USB host / hub controller. The values used here allow attached circuitry to power up with a slew rate of ~12.5 V per millisecond, in other words the output voltage will transition from GND to 5V in approximately 400 microseconds.

Alternatively, a dedicated power switch i.c. with inbuilt "soft-start" can be used instead of a MOSFET. A suitable power switch I.C. for such an application would be a Micrel (www.micrel.com) MIC2025-2BL or equivalent.

Please note the following points in connection with power controlled designs –

a) The logic to be controlled must have its own reset circuitry so that it will automatically reset itself when power is re-applied on coming out of suspend.
b) Set the Pull-down on Suspend option in the FT232BL's EEPROM.
c) For USB high-power bus powered device (one that consumes greater than 100 mA, and up to 500 mA of current from the USB bus), the power consumption of the device should be set in the max power field in the EEPROM. A high-power bus powered device must use this descriptor in the EEPROM to inform the system of its power requirements.
d) For 3.3V power controlled circuits VCCIO must not be powered down with the external circuitry (PWREN# gets its VCC supply from VCCIO). Either connect the power switch between the output of the 3.3V regulator and the external 3.3V logic OR if appropriate power VCCIO from the 3V3OUT pin of the FT232BL.

8.0 Document Revision History

DS232B Version 1.0 – Initial document created 30 April 2002.

DS232B Version 1.1 – Updated 04 August 2002

- Section 4.1 RESET# Pin description corrected (RESET# does not have an internal 200k pull-up to VCC as previously stated).
- Figure 2 pin-out corrected (EECS = Pin 32).

DS232B Version 1.2 – Updated 27 October 2003

- Pin and package naming made consistent throughout data sheet.
- Section 1.0 Updated to reflect availability of Mac OS X driver.
- Section 2.0 Minor corrections.
- Section 3.1 Minor changes to functional block descriptions of SIE, RESET Generator, and EEPROM interface.
- Section 4.1 Note added to EEPROM interface group.
- Section 4.1 RSTOUT# Pin description amended.
- Section 6.1 Minimum operating supply voltage adjusted.
- Section 6.1 EESK added to Note 3.
- Section 6.1 UART IO pin characteristics amended.
- Section 6.1 RESET#, TEST, EECS, EESK, and EEDATA pin characteristics amended.
- Section 6.1 RSTOUT pin characteristics amended.
- Section 7.1 Updated recommended ceramic resonator part number and circuit configuration.
- Section 7.3 "USB Self Powered Configuration (1)" (original figure 8 removed). Recommended circuit for USB self powered designs updated. Subsequent figure numbers have changed as a result.
- Section 7.6 Note added to description of Bus powered circuit with 3.3V logic drive / supply voltage.
- Section 7.6 Self Powered Circuit with 3.3V logic drive / supply voltage added (new figure 16).

DS232B Version 1.3 – Updated 10 December 2003

- Section 5.0 Package drawing amended
- Section 6.0 Floor Life / Relative Humidity specification added. ESD and Latch Up specifications amended.
- Section 7.1 Required resonator / crystal accuracy corrected.

DS232B Version 1.4 – Updated 10 February 2004

- Grammar Corrections
- Section 10.0 FTDI Address Updated
- Section 2.0 Extended EEPROM Support corrected
- Section 4.1 VCCIO Pin description amended.
- Section 7.4 RS485 Example Sipex SP481 part number corrected.

DS232B Version 1.5 – Updated April 2004

- Section 4.1 EESK Pin Description amended.
- Section 7.6 Figure 16 PWRCTL Pin number corrected.
- Section 7.7 Figure 15 PWREN# Pin number corrected.

DS232B Version 1.6 – Updated November 2004

- Section 1.0 WinCE drivers now available.
- Section 5.0 Date code format updated.
- Section 6.0 Absolute Maximum Ratings table reformated.

DS232B Version 1.7 – Updated February 2005

- Section 1.0 D2XX drivers for Linux and Windows CE now available.
- Section 5.0 FT232BL (lead Free) and FT232BQ (lead free QFN package) now available.

DS232B Version 1.8 – Updated December 2005

- Section 1.0 Driver OS Support updated.
- Section 6.0 USB Data line absolute maximum rating added.

9.0 Disclaimer

10.0 Contact Information

Head Office -

Future Technology Devices International Ltd.
373 Scotland Street,
Glasgow G5 8QB,
United Kingdom

Tel. : +(44) 141 429 2777
Fax. : +(44) 141 429 2758

E-Mail (Sales) : sales1@ftdichip.com
E-Mail (Support) : support1@ftdichip.com
E-Mail (General Enquiries) : admin1@ftdichip.com

Regional Sales Offices -

Future Technology Devices International Ltd. (Taiwan)
4F, No 16-1,
Sec. 6 Mincyuan East Road,
Neihu District,
Taipei 114,
Taiwan, R.o.C.

Tel.: +886 2 8791 3570
Fax: +886 2 8791 3576

E-Mail (Sales): tw.sales@ftdichip.com
E-Mail (Support): tw.support@ftdichip.com
E-Mail (General Enquiries): tw.admin@ftdichip.com

Future Technology Devices International Ltd. (USA)
5285 NE Elam Young Parkway, Suite B800
Hillsboro,
OR 97124-6499
USA

Tel.: +1 (503) 547-0988
Fax: +1 (503) 547-0987

E-Mail (Sales): us.sales@ftdichip.com
E-Mail (Support): us.support@ftdichip.com
E-Mail (General Enquiries): us.admin@ftdichip.com

Website URL : http://www.ftdichip.com

Agents and Sales Representatives

At the time of writing our Sales Network covers over 40 different countries world-wide. Please visit the Sales Network page of our Web site for the contact details our distributor(s) in your country.

SHEN ZHEN AV-DISPLAY ELECTRONICS CO.,LTD

SPECIFICATIONS FOR

LIQUID CRYSTAL DISPLAY MODULE

MODEL NO: HY-1601C-802 DATE:2004-6-1

Approved	Checked	epartment

CONTENTS

Ⅰ.General Specifications

1. General

The AV-DISPLAY dot matrix LCD module consist of the liquid crystal display C-MOS driver and C-MOS LSI controller. the module utilizes 5*7 dot matrix characters to provide full alphanumeric capability. All control, refresh and display functions are executed by a dedicated on-board controller. the module is capable of displaying the full 160-character JIS font set .data interfacing is via the 4-bit or 8-bit bi-directional data bus by using of simple control commands the data can be selective written to the data register.

2. Features

A. Built-In Controller LSI.

B. 5*7 Dot Matrix With Cursor.

C. Micro-Processor Compatible Data-Bus Interface(4-Bit Or 8-Bit).

D. Character Generator ROM Built-In

5*8 Dot : ----------208 Character Fonts

5*10 Dot : ----------32 Character Fonts

E. Character Generator RAM--------Customer Rewritable

5*8 Font:8 Characters

F. Powerful Control Command

(1) Display Clear

(2) Return Home

(3) Cursor Preset

(4) Cursor On/Off Or Cursor Blinking

(5) Cursor Display Shift

(6) Display Shift

(7) Display On/Off Control

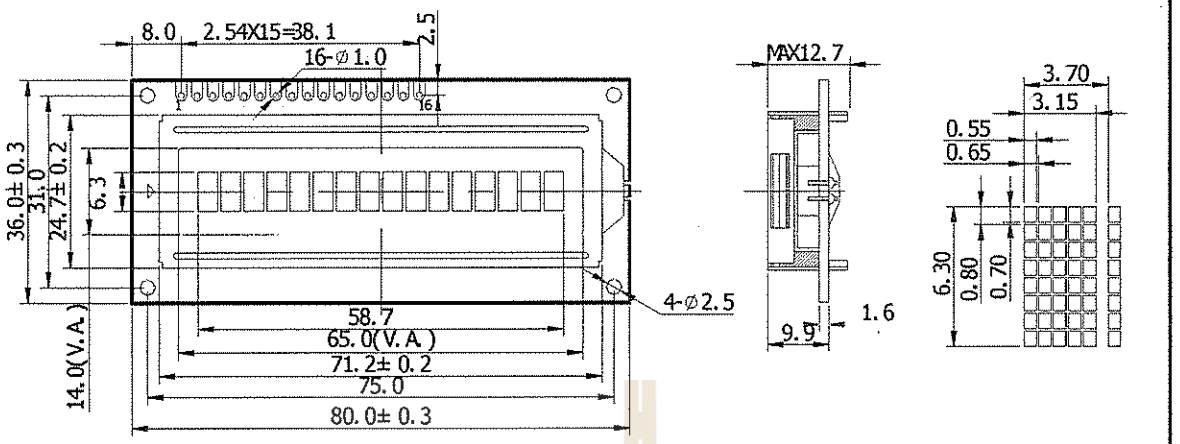
(8) Display Data Read/Write

G. Low power consumption 5.0v power supply

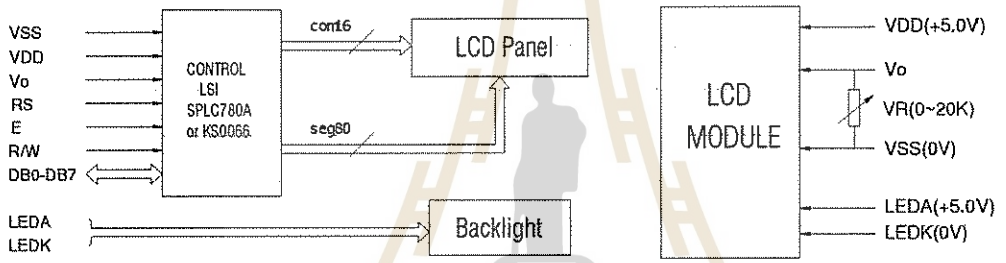
H. Attaching drawing and general description

QiuTian,ShiJia **HY-1601C-802** 16x1 CHARACTERS 1/16DUTY,1/5BIAS

1.0 DIMENSIONAL DRAWING



2.0 BLOCK DIAGRAM & POWER SUPPLY



3.0 MECHANICAL SPECIFICATIONS & FEATURE

Item	Nominal Dimensions(mm)	FEATURE	
		LCD Type	STN
Module Size (W*H*T)	80.0x36.0x12.7max	LCD Colour	Blue
View Area (W*H)	65.0X14.0	View Angle	12 O'clock
Character Pitch(W*H)	3.70X6.30	Display Type	Negative Type
Character Size(WxH)	3.15X6.30	Rear polarizer	Transmissive
Character Font	5x8	Operating Temperature	-20i℃ ~70i℃
Dot Pitch (W*H)	0.65X0.80	Storage Temperature	-30i℃ ~80i℃
Dot Size (W*H)	0.55X0.70	Blacklight	LED£ White£ ©

4.0 ELECTRICAL CHARACTERISTICS

Item	Symbol	Test Condition	Min.	Typ.	Max.	Unit
Operating Voltage	Vdd	Ta=25i℃	----	5.0	---	V
Operating Voltage for LCD	Vlcd	Ta=25i℃	---	4.5	---	V
Supply Current	Idd	Ta=25i℃, Vdd=5.0V	---	2.0	3.0	mA
Supply Current for Blacklight	If	Ta=25i℃, Vf=3.5V	----	20	---	mA

5.0 INTERFACE PIN CONNECTIONS

Pin No	Symbol	Level	Description
1	VSS	--	GND
2	VDD	--	Power supply for Logic(+5.0v)
3	Vo	--	Power supply for LCD drive
4	RS	H/L	Register selection (H:Data registor, L:Instruction registor)
5	R/W	H/L	Read/write selection (H:Read,L:Write)
6	E	H/H—L	Enable signal for LCM,The fall of E is for active
7-14	DB0-DB7	H/L	3-State I/O Data Bus
15	LEDA	--	Power supply for Blacklight(+5v)
16	LEDK	--	Power supply for Blacklight(0v)

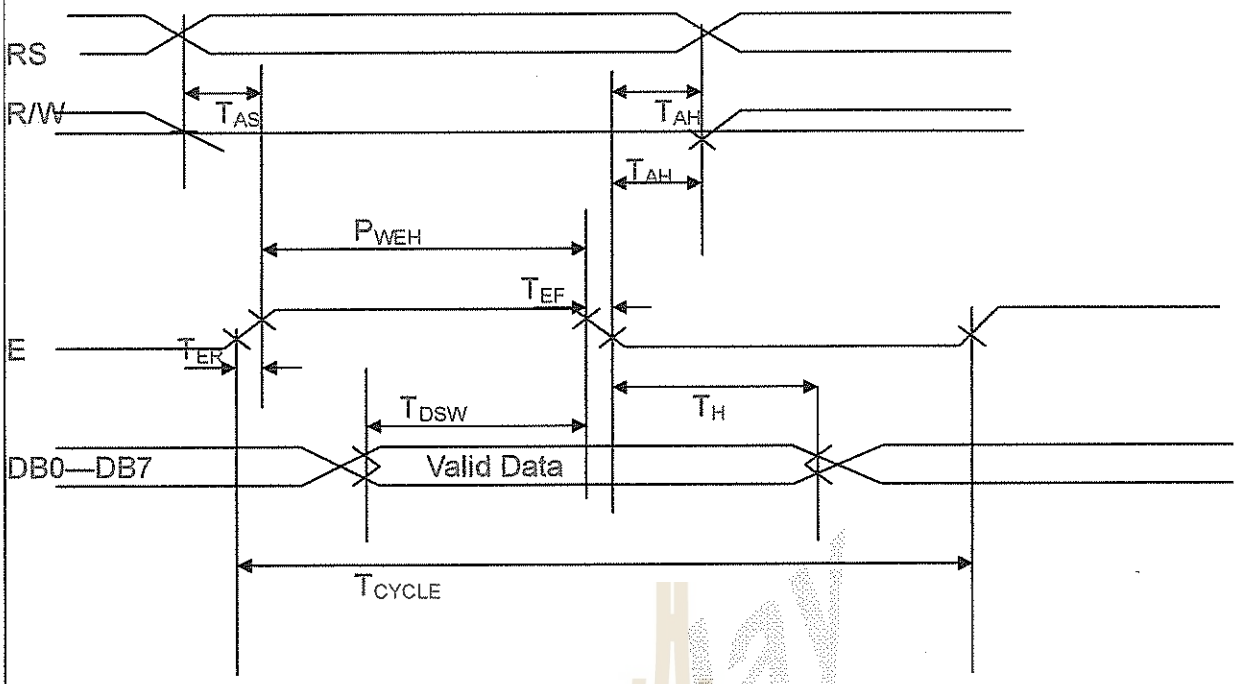
4.Timing Characteristics:

Write Operation and Read Operation

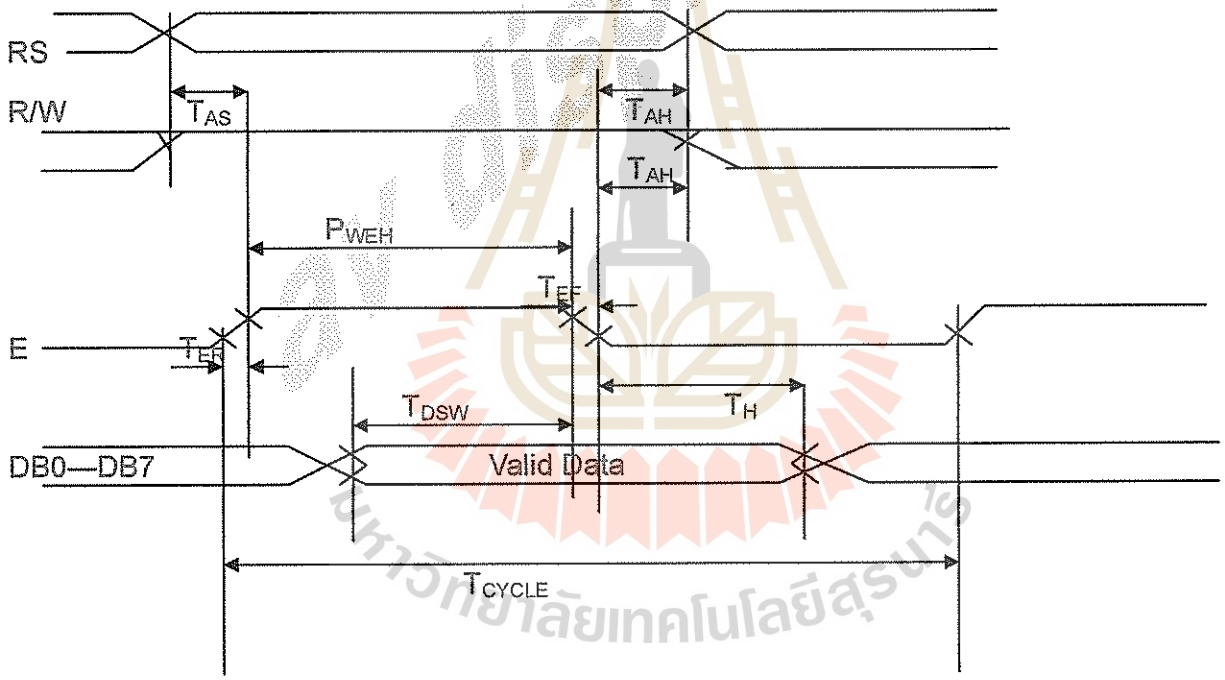
Item	Symbol	Min.	Typ.	Max.	Unit
Enable Cycle Time	T_{CYCLE}	500	--	--	nS
Enable Pulse Width	P_{WEH}	220	--	--	nS
Enable Rise & Fall Time	T_{ER}, T_{EF}	--	--	25	nS
Address Set-Up Time	T_{AS}	40	--	--	nS
Address Hold Time	T_{AH}	10	--	--	nS
Data Set-Up Time	T_{DSW}	60	--	--	nS
Data Hold Time	T_H	10	--	--	nS

Item	Symbol	Min.	Typ.	Max.	Unit
Enable Cycle Time	T_{CYCLE}	500	--	--	nS
Enable Pulse Width	P_{WEH}	220	--	--	nS
Enable Rise & Fall Time	T_{ER}, T_{EF}	-	--	25	nS
Address Set-Up Time	T_{AS}	40	--	--	nS
Address Hold Time	T_{AH}	10	--	--	nS
Data Set-Up Time	T_{DSW}	-	--	120	nS
Data Hold Time	T_H	20	--	--	nS

5.Write Operation:



Read Operation



Ⅱ.The Characteristics and The Reliability Test

1.Electro-Optic Characteristics:

Condition:TEMP=(21±3)℃ HUM=(70±5)%RH

V_{DD}: 5.0V F_{OSC}: 270KHZ

NO	Item		Symbol	Min.	Typ.	Max.	Unit	Remarks
1	Operating Voltage		Vop		5.0		V	
2	Current Consumption		Is		2.0		mA	
3	Response Time		Ton		150		ms	
			Toff		120		ms	
4	Contrast		CR	3				
5	Viewing Angle (CR≥3.0)	12H	θ 1		15		Deg.	
		6H	θ 2		45			
		3H	θ 3		50			
		9H	θ 4		50			
6	Threshold Voltage		Vth		1.14		V	
7	Backlight Current Consumption				20		mA	

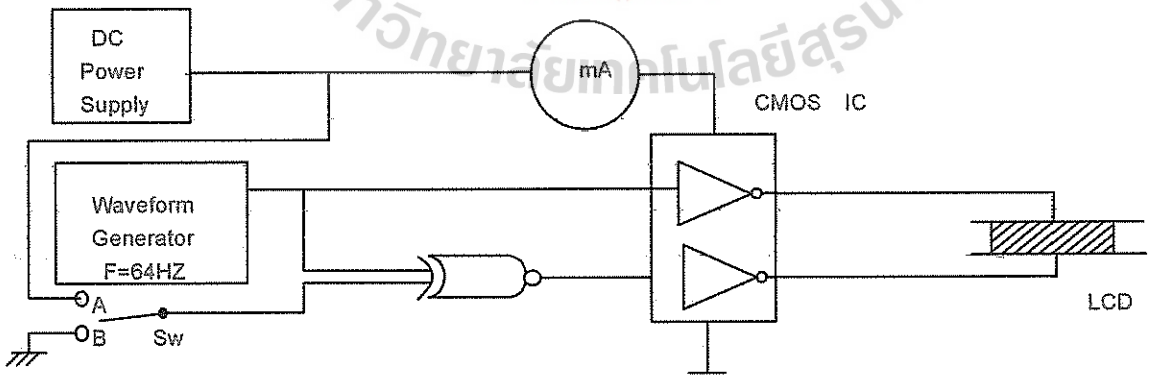
3.Reliability Test(No ITO heat)

No	Items	Test Condition		Test Result
1	High Temp Storage	Temp:70±2℃ Time:96h Restore:24h		Passed
2	Low Temp Storage	Temp:-20±3℃ Time:96h Restore:24h		Passed
3	High Temp Static drive	Temp:50±2℃ Vop:5V Time:96h Restore:24h		Passed
4	Low Temp Static drive	Temp:0±3℃ Vop:5V Time:96h Restore:24h		Passed
5	High Temp High Hum Storage	Temp:40±2℃ Hum:95%Rh Time:96h Restore:24h		Passed
6	Thermal Shock	Temp:(℃) 70 25 -20 30 5 30 5 5 Cycles Restore:24h		Passed

III.The LCD Measuring Method and Equipment

1. Current Consumption Measuring

(1) Equipment



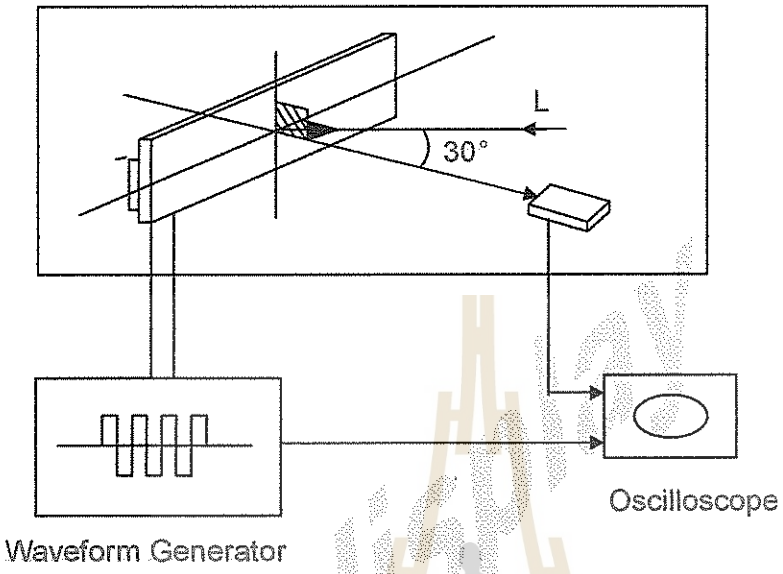
(2) Condition

Operating Frequency: 64HZ

Operating Voltage (RMS): Selected Voltage

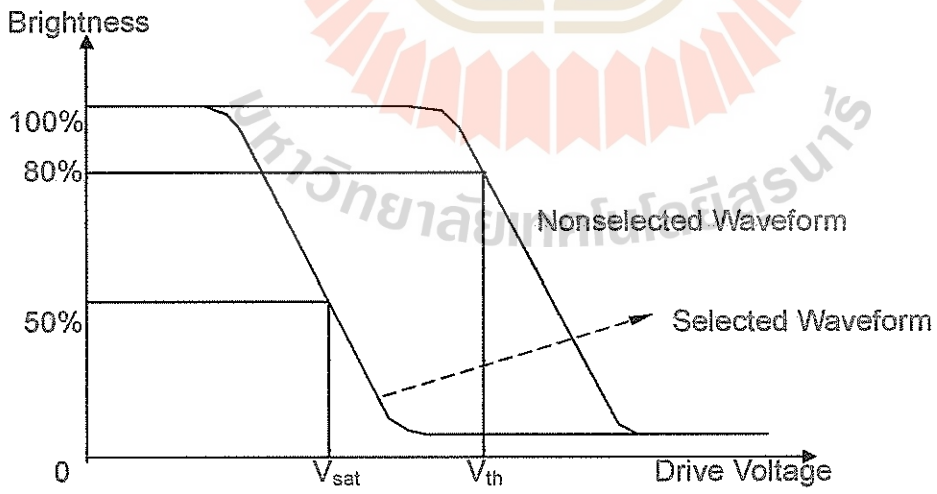
2. Threshold Voltage and Response Time Measuring

(1) Equipment

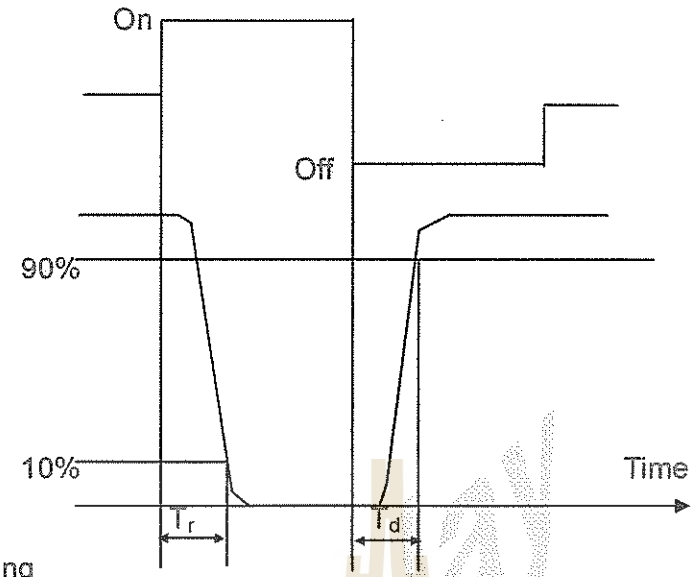


(2) Definition

A. Threshold Voltage (V_{th})

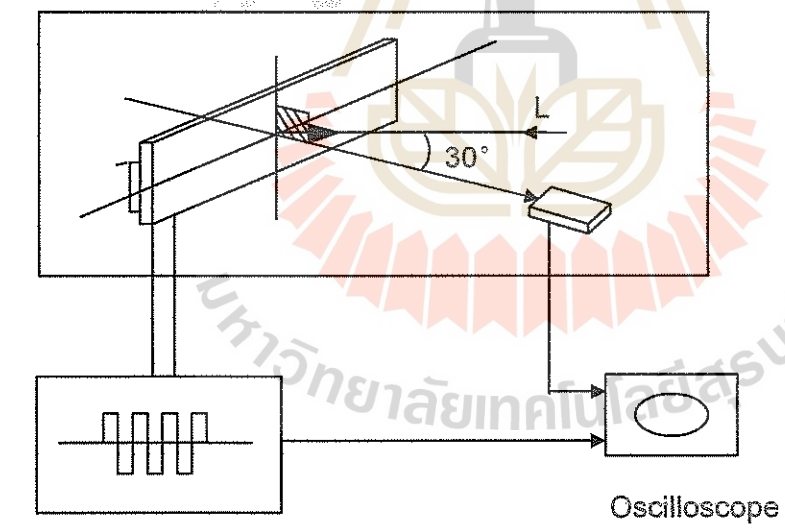


B. Response Time



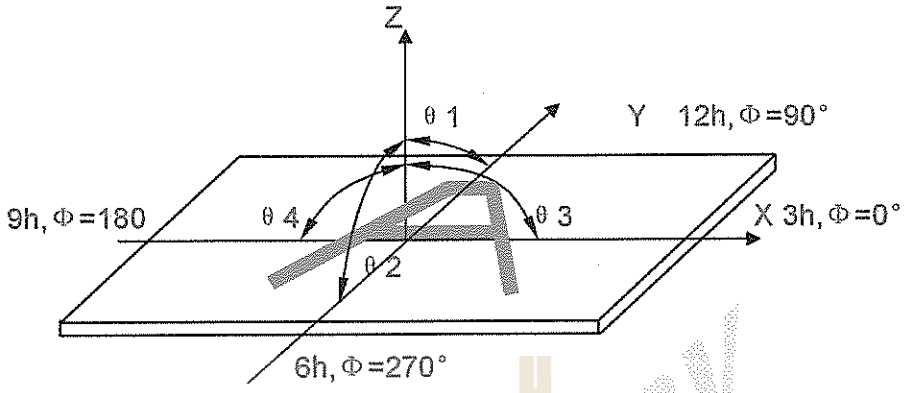
3. Contrast Measuring

(1) Equipment



(2)Definition:

A.Viewing Angle:



B. Contrast Ratio (Positive)

$$CR= \frac{\text{Brightness of non-selected wave-form}}{\text{Brightness of selected wave-form}}$$

Ⅳ.Standard Specifications for Product Quality

1. Manner of Test:

1.1.The Test Must Be Under 40w Flourescent Light, And The Distance Of View Must Be At 30cm.

1.2.The Test Direction Is Based On Around 15° - 45° Of Vertical Line.

2. Definition Of Defects

2.1 Major Defects

A:Non-Display

B:Segment Missing

C:Over Current

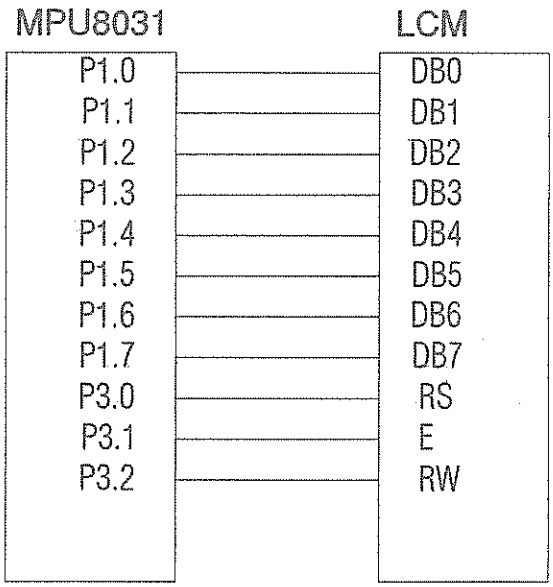
D:Segment Short

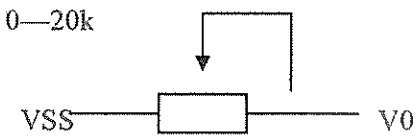
E:Sealant Dishardexn

F:Wrong Polarizer Direction

2.2 Interface Circuit and Drive Programe on LCM of character series.

A. Interface circuit:





B.Drive programme for testing LCM of character series.

```
ORG     0000H
AJMP    MAIN

ORG     0300H
DB      58H,58H,58H,58H,58H,58H,58H,58H,
DB      58H,58H,58H,58H,58H,58H,58H,58H,
DB      58H,58H,58H,58H,58H,58H,58H,58H,
DB      58H,58H,58H,58H,58H,58H,58H,58H,
DB      58H,58H,58H,58H,58H,58H,58H,58H,
DB      58H,58H,58H,58H,58H,58H,58H,58H,
DB      58H,58H,58H,58H,58H,58H,58H,58H,
DB      58H,58H,58H,58H,58H,58H,58H,58H,
DB      58H,58H,58H,58H,58H,58H,58H,58H,
DB      58H,58H,58H,58H,58H,58H,58H,58H,
DB      58H,58H,58H,58H,58H,58H,58H,58H,
```

```
  ORG      0350H
  DB       2AH,59H,55H,53H,55H,4EH,47H,2AH,
  DB       45H,4CH,45H,43H,2EH,4CH,54H,44H,
  DB       2AH,44H,4FH,54H,2AH,4DH,41H,54H,
  DB       52H,49H,58H,2AH,4CH,43H,44H,2AH,
  DB       4BH,65H,5AH,6FH,6EH,48H,75H,69H,
  DB       2AH,59H,55H,53H,55H,4EH,47H,2AH,
  DB       45H,4CH,45H,43H,2EH,4CH,54H,44H,
  DB       2AH,44H,4FH,54H,2AH,4DH,41H,54H,
  DB       52H,49H,58H,2AH,4CH,43H,44H,2AH,
  DB       4BH,65H,5AH,6FH,6EH,48H,75H,69H,
  DB       2AH,2AH,2AH,2AH,2AH,2AH,2AH,2AH,
  DB       44H,4FH,54H,20H,4DH,41H,54H,52H,
  DB       49H,58H,20H,4CH,49H,51H,55H,49H,
  DB       44H,20H,43H,52H,59H,53H,54H,41H,
  DB       4CH,20H,44H,49H,53H,50H,4CH,41H,
  DB       59H,20H,4DH,4FH,55H,44H,4CH,45H,
  DB       2AH,2AH,2AH,2AH,2AH,2AH,2AH,2AH,
  DB       2AH,2AH,2AH,2AH,2AH,2AH,2AH,2AH,
  DB       54H,4DH,0B0H,44H,4DH,43H,34H,30H,
  DB       32H,2AH,2AH,2AH,2AH,2AH,2AH,2AH,
  DB       2AH,2AH,2AH,2AH,2AH,2AH,2AH,2AH,
  DB       2AH,2AH,2AH,2AH,2AH,2AH,2AH,2AH,
MAIN :
  MOV      SP, #60H        ;Initial for the first display
  MOV      P1, #38H        ;set function
  LCALL    WINST
  MOV      P1, #0EH        ;set display on/off control
  LCALL    WINST
  MOV      P1, #06H        ;set Entry mode
  LCALL    WINST
  MOV      P1, #01H        ;clear display,write code 20h into all DDRAM
  LCALL    WINST
  LCALL    DELAY1
```

```
  MOV     DPTR, #0300H
  MOV     R0, #28H          ;Set Pointer
  MOV     R2, #00H
  MOV     A, #00H
  MOV     P1, #80H          ;set DDRAM address 0000h
  LCALL   WINST
LOOP1:
  MOVC    A, @A+DPTR
  MOV     P1, A
  LCALL   WDATA
  INC     R2
  MOV     A, R2
  DJNZ    R0, LOOP1

  MOV     DPTR, #0328H
  MOV     R0, #28H
  MOV     R2, #00H
  MOV     A, #00H
  MOV     P1, #0C0H
  LCALL   WINST
LOOP2:
  MOVC    A, @A+DPTR
  MOV     P1, A
  LCALL   WDATA
  INC     R2
  MOV     A, R2
  DJNZ    R0, LOOP2     ;The first display is over
  LCALL   DELAY2        ;paused   about 5ms


  MOV     SP, #60H      ;initial for the second display
  MOV     P1, #38H
```

```
  LCALL   WINST
  MOV     P1, #0EH
  LCALL   WINST
  MOV     P1, #06H
  LCALL   WINST
  MOV     P1, #01H
  LCALL   WINST
  LCALL   DELAY1

  MOV     DPTR, #0350H     ;ready for the first line display
  MOV     R0, #28H
  MOV     R2, #00H
  MOV     A, #00H
  MOV     P1, #80H
  LCALL   WINST
LOOP3:
  MOVC    A, @A+DPTR
  MOV     P1, A
  LCALL   WDATA
  INC     R2
  MOV     A, R2
  DJNZ    R0, LOOP3        ;THE first line display is over

  MOV     DPTR, #0378H     ;ready for the second line display
  MOV     R0, #28H
  MOV     R2, #00H
  MOV     A, #00H
  MOV     P1, #0C0H
  LCALL   WINST
LOOP4:
  MOVC    A, @A+DPTR
  MOV     P1, A
  LCALL   WDATA
  INC     R2
```

```
  MOV     A, R2
  DJNZ    R0, LOOP4        ;main program is end upto here
LOOP5:
   LCALL   DELAY2
  AJMP    MAIN

WINST:
   CLR     P3.0             ;write to instruction register
  CLR     P3.2
  SETB    P3.1
  LCALL   DELAY1
  CLR     P3.1
  LCALL   DELAY1
  RET

WDATA:
   CLR     P3.2             ;write to data register
  SETB    P3.0
  SETB    P3.1
  LCALL   DELAY1
  CLR     P3.1
  LCALL   DELAY1
  RET

DELAY1:
        MOV     50H, #08H        ;delay   1648 us
ADDR1:  PUSH    50H
ADDR2:  PUSH    50H
ADDR3:  PUSH    50H
ADDR4:  DJNZ    50H, ADDR4
        POP     50H
        DJNZ    50H, ADDR3
        POP     50H
        DJNZ    50H, ADDR2
```

```
  POP     50H
  DJNZ    50H, ADDR1
  RET

DELAY2:
  MOV     R0, #0CCH
  MOV     R2, #66H
ADDR5:
  LCALL   DELAY1          ;delay ccH X 1648us
  DJNZ    R0, ADDR5
ADDR6:
  LCALL   DELAY1          ;delay 66H X 1648us    total 5.05ms
  DJNZ    R2, ADDR6
  RET

  END
```

3.Inspection Item and Standards

Item	The Standard Of Quality Inspection	Checking Manner	Quality Ratio
Frame	Smooth and even surface,no crack,no scratch,no rusty,and not be wrenched out of shape.the range between convex and concave is:d≤0.35mm,and the frame must be connected to the ground.	Checking With Eyes And Using Vernier Caliper, Multimeter	100%
LCD	The major defects would be reject.no scratch and no dusty on the LCD glass surface.d≤0.15mm n≤2 diameter of bubble:d≤0.5 n≤2 damaged size of polarizer:d≤0.15mm, n≤2.	Check It When Displaying	100%
The Relative Position of LCD and Frame	The sealant mouth of the LCD must be at the same side with the frame's.	Checking With Eyes	100%
The Relative Position of PCB Paneland Frame	The frame installing direction must be correct.the twisted angle of the pin is from 45° to 60° ,the pin is vertical to PCB panel and it must be in the middle position of the installing holes.	Checking With Eyes	100%
Function Test	1. The major defects must be reject. 2. Test flow chart (see attached chart) 3. Background changes evenly and no disorderly displaying phenomenon. 4. Display no shortage.	Check It When Displaying	100%

Note:D～Diameter N～Quantity Unit:mm

Ⅴ. Instruction System and Description of Details

1.Instruction System

Only two SPLC780A OR KS0066 registers,the Instruction Register (IR) and the Data Register (DR) can be directly controlled by the MPU.Prior to internal operation start,control information is temporarily stored in these registers,to allow interface form SPLC780A OR KS0066 internal operation to various types of MPUs which operate in different speeds or to allow interface to peripheral control ICS.SPLC780A OR KS0066 internal operation is determined by signals sent from the MPU.These signals include register selection signal(RS),read/write signals (R/W) and data bus signals (DB0—DB7),and are called instructions,here.Table 1 shows the instructions and their execution time.Details are explained in subsequent sections.

Instructions are of 4 types, those that,

(1) Designate SPLC780A OR KS0066 functions such as display format,data length,etc.

(2) Give internal RAM addresses.

(3) Perform data transfer with internal RAM.

(4) Others.

In normal use,category (3) instructions are used most frequently.However,automatic incrementing by +1 (or decrementing by –1) of SPLC780A OR KS0066 internal RAM addresses after each data write lessens the MPU program load.The display shift is especially able to perform concurrently with display data write ,enabling the user to develop systems in minimum time with maximum programming efficiency.For an explanation of the shift function in its relation to display ,.When an instruction is executing during internal operation,no instruction other than the busy flag/address read instruction will be executed.

Because the busy flag is set to “1” while an instruction is being executed,check to make sure it is on “1” before sending an instruction from the MPU.

Note 1

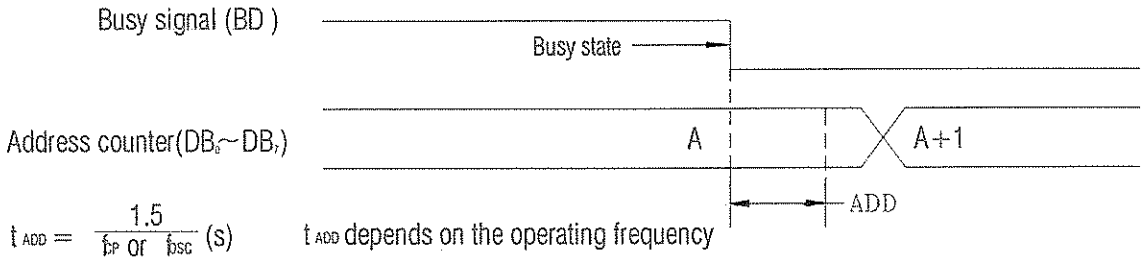
Make sure the SPLC780A OR KS0066 is not in the busy state (BF=0) before sending the instruction from the MPU to the SPLC780A OR KS0066. If the instruction is sent without checking the busy flag the time between first and next instructions is much longer than the instruction time.

See Table 1 for a list of each instruction execution time.

Note 2

After executing instruction of writing data to CG/DD RAM or reading data from CG/DD RAM,RAM address counter is automatically incremented by 1 (or decremented

by 1). In this case, this shift is executed after Busy flag is set to "Low".T_{ADD} is stipulated the time from the fall edge of busy flag to the end of address counter's renewal.



$t_{ADD} = \frac{1.5}{f_{CP} \text{ or } f_{OSC}}$ (s) t_{ADD} depends on the operating frequency

Table 1 Instructions

Instruction	Code										Description	Execution time (when Fose is 250 KHz) Note 1	Execution time (when Fose is 160 KHz) Note 2
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0			
Clear display	0	0	0	0	0	0	0	0	0	1	Clears all display and returns theCursor to home position (Address 0).	82us~1.64ms	120us~4.9ms
Return home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position (Address 0).Also returns the display being shifted to the original position. DDRAM contents remain unchanged.	40us~1.6ms	120us~4.8ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and specifies or not to shift the display.These operations are performed during data write and read	40us	120us
Display ON/OFF control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of all display (D), cursor ON/OFF (C),and blink of cursor position character (B).	40us	120us
Cursor and display shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shites the display without changing DD RAM contents.	40us	120us
Function set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL) number of display lines (L) and character font (F).	40us	120us
Set CG RAM address	0	0	0	1	ACG						Sets the CG RAM address.CG RAM data is sent and received After this setting .	40us	120us
Set DD RAM address	0	0	1	ADD							Sets the DD address. DD RAM data is sent and received After this setting .	40us	120us
Read busy flag & address	0	1	BF	AC							Reads Busy flag (BF) indicating internal operation is being performed and reads address counter contents.	1us	1us
Write data to CG o r DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	40us	120us
Read data to CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM	40us	120us
	I/D=1: Increment (+1)I/D=0: Decrement (-1) S =1: Accompanies display shift S/C=1: Display shift S/C=0:Cursor move R/L=0: Shift to right R/L=0: Shift to left DL =1: 8 bits DL =0: 4 bits N =1: 2 lines N =0: 1 lines F =1: 5x10 dots F =0: 5x7 dots BF =1: Internally operating BF =0: Can accept instruction										DD RAM: Display data RAM CG RAM: Character generator RAM ACG: CG RAM address ADD: DD RAM address Corresponds to cursor address AC: Address counter used For both of DD and CG RAM address	Execution time changes when Frequency changes. (Example) When fosc is 270k Hz: 250 / 40us x 270 = 37us	

* No effect

Notes 1: Applied to models driven by 1/8 duty or 1/11 duty.

2: Applied to models driven by 1/16 duty.

2. Description of details

(1) Clear display

	RS	R/W	DB7	----------					DB0
Code	0	0	0	0	0	0	0	0	1

Writes space code “20” (hexadecimal) (character pattern for character code “20” must be blank pattern) into all DD RAM address.Set DD RAM address 0 in address counter.Returns display to its original status if it was shifted. In other words.the display disappears and the cursor or blink go to the left edge of the display (the first if 2 lines are displayer).Set I/D= 1 (Increment Mode) of Entry Mode.S of Entry Mode doesn’t change.

(2) Return home

	RS	R/W	DB7	----------					DB0
Code	0	0	0	0	0	0	0	0	1	*

*No effect

Sets the DD RAM address 0 in address counter. Returns display to its original status if it was shifted. DD RAM contents do not change. The cursor or blink go to the left edge of the display (the first line if 2 lines are displayed).

(3) Entry mode set

	RS	R/W	DB7	----------					DB0
Code	0	0	0	0	0	0	0	1	I/D	S

I/D: Increments (I/D = 1) or decrement s (I/D) the DD RAM address by 1 when a character code is written into or read from the DD RAM .The cursor blink moves to the right when incremented by 1 and to the left when decremented by 1. The same applies to writing and reading of CG RAM.

S: Shifts the entire display either to the right or to the left when S is 1; to the left when I/D = 1 and to the right when I/D = 0. Thus it looks as if the cursor stands still and the display moves.The display does not shift when reading from the DD RAM when writing into or reading out from the CG RAM does it shift when S =0.

(4) Display ON/OFF control

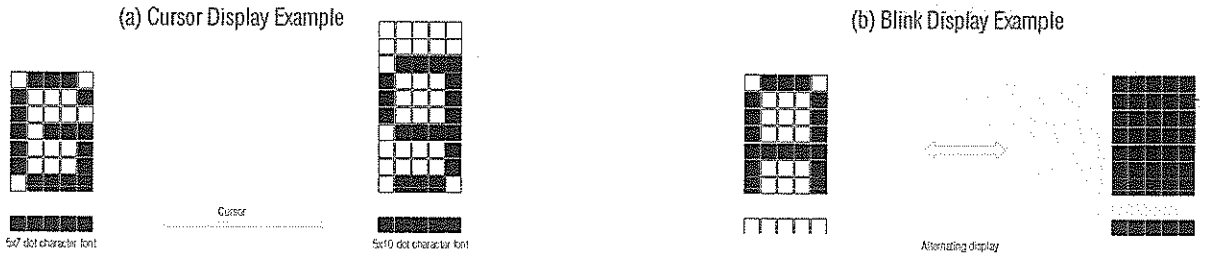
	RS	R/W	DB7	----------					DB0
Code	0	0	0	0	0	0	1	D	C	B

D: The display is ON when D = 1 and OFF when D = 0. when off due to D = 0,display Data remains in the DD RAM.It can be displayed immediately by setting D = 1.

C: The cursor displays when C = 1 and does not display when C = 0. Even if the Cursor disappears,the function of I/D,etc.does not change during display data write.

The cursor is displayed using 5 dots in the 8th line when the 5x7 dot character font Is selected and 5 dots in the 11th line when the 5x10 dot character font is selected.

B: The character indicated by the cursor blink when B = 1.The blink is displayed by Switching between all blank dots and display characters at 409.6 ms interval when fcp or fosc =250Khz. The cursor and the blink can be set to display simultaneously. (The blink frequency changes according to the reciprocal of fcp or fose. 409.6x250/ 270 = 379.2ms when fcp = 270kHz).



(5) Cursor or display shift

	RS	R/W	DB7	----------						DB0
Code	0	0	0	0	0	1	S/c	R/l	*	*

*No effect

Shifts Cursor position or display to the right or left without writing or reading display data.This function is used to correct or search for the display .In a 2-lines display, the cursor moves to the 2nd line when its passes the 40th digit of the 1st line.Notice that the 1st and 2nd line display will shift at the same time.When the displayed data is shifted repeatedly each line only moves horizontally.The 2nd line display does not shift into the 1st line position

S/C R/L

0 0 Shifts the cursor position to the left.(AC is decremented by one.)

0 1 Shifts the cursor position to the right. (AC is decremented by one.)

1 0 Shifts the entire display to the left. The cursor follows the display shift.

2 1 Shifts the entire display to the right.The cursor follows the display shift.

Address counter (AC) contents do not change if the only action performed is shift

(6) Function set

	RS	R/W	DB7	----------						DB0
Code	0	0	0	0	1	DL	N	F	*	*

*No effect

DL: Sets interface data length.Data is sent or received in 8 bit lengths (DB7~DB0) when DL = 1and in 4 bit lengths (DB7~DB4) when DL = 0.when the 4 bit length is selected Data must be sent or received twice.

N: Sets number of display lines

F: Sets character font.

(Note) Perform the function at the head of the program before executing all instruction (expect "Busy flag/address read").From this point,the function set instruction cannot be executed unless the interface data length is changed.

N	F	No.of display lines	Character font	Duty factor	Remarks
0	0	1	5x7 dots	1/8	
0	1	1	5x10 dots	1/11	
1	*	2	5x7 dots	1/16	Cannot display 2 lines with 5x10 dot character font.

*No effect

(7) Set CG RAM address

	RS	R/W	DB7	----------						DB0
Code	0	0	0	1	A	A	A	A	A	A

←Higher Order Bits Lower Order Bits→

Sets the CG RAM address into the address counter in binary AAAAAA.Data is then Written or read from the MPU for the CG RAM

(8) Set DD RAM address

	RS	R/W	DB7	----------						DB0
Code	0	0	1	A	A	A	A	A	A	A

←Higher Order Bits Lower Order Bits→

Sets the DD RAM address into the address counter in binary AAAAAAA.Data is then Written or read from the MPU for the DD RAM.

However, When N = 0 (1-line display),AAAAAAA is "00" ~ "4F" (hexadecimal).
When N = 1 (2-line display),AAAAAAA is "00" ~ "27" (hexadecimal) for the first line,and "40" ~ "67" (hexadecimal) for the second line.

(9) Read busy flag & address

	RS	R/W	DB7	----------						DB0
Code	0	1	BF	A	A	A	A	A	A	A

←Higher Order Bits Lower Order Bits→

Reads the busy flag (BF) that indicates the system is now internally operating by a previously received instruction.BF=1 indicates that internal operation is in progress. The next instruction will not be accepted until BF is set to "0".check the BF status before the next write operation.At the same time,the value of the address counter expressed in binary AAAAAAA is read out. The address counter is used by both CG and DD RAM address ,and its value is determined by the previous instruction.

Address contents are the same as in terms (7) and (8).

(10) Write data to CG or DD RAM

	RS	R/W	DB7							DB0
Code	1	0	D	D	D	D	D	D	D	D

←Higher Order Bits Lower Order Bits→

Writes binary 8 bit data DDDDDDDD to the CG or the DD RAM.Whether the CG or DD RAM is to be written into is determined by the previous specification of CG RAM or DD RAM address setting .After write ,the address is automatically incremented or decremented by 1 according to entry mode.The entry mode also determines display shift.

(11) Read data from CG or DD RAM

	RS	R/W	DB7							DB0
Code	1	1	D	D	D	D	D	D	D	D

←Higher Order Bits Lower Order Bits→

Reads binary 8 bits data DDDDDDDD from the CG or DD RAM. The previous designation determines whether the CG or DD RAM is to be read. Before entering the read instruction ,you must execute either the CG RAM or DD RAM address set instruction.If you don't,the first read data will be invalidated. When serially executing the "read" instruction,the next address data is normally read from the second read. The "address set" instruction need not be executed just before the "read" instruction when shifting the cursor by cursor shift instruction (when reading out DD RAM).The cursor shift instruction operation is the same as that of the DD RAM 's address set instruction.

After a read, the entry mode automatically increases or decreases the address by 1. However, display shift is not executed no matter what the entry mode is.

(Note) The address counter (AC) is automatically incremented or decremented by 1 after"write"instructions to either CG RAM or DD RAM. RAM data selected by the AC cannot than be read out even if "read" instructions are executed. The conditions for correct data read out are : execute either the address set instruction or cursor shift instruction (only with DD RAM), just before reading out execute the "read" instruction from the second time the "read" instruction is serial.

3. Precaution on programming

(1) Instruction of function set

Perform the function at the head of program that accesses SPLC780A OR KS0066 before executing all instructions,and not change the data of the instruction Register in the program. The data of function register can be changed by the program as follow;

a. • Changing of DL (Data Length)

- when DL is changed from 8-bit length mode.
- when DL is changed from 4-bit length mode.

b. • Changing of N (Column Number)

- Perform the instruction of function set after executing instruction of display clear or display off.

In this case, sequence of AC and DD RAM must be changed.Thus, rewrite the address set register after that.

c. • Changing of F (Font)

- There is no problem in this case, but for dual-line display,the font mode of 5x11 cannot be selected (this mode is forbidden by hardware).

When N of F is changed, power supply voltage for LCD must be changed.If not Changed,crosstalk will appear,or contrast will be poor.

(2) Busy flag check

SPLC780A OR KS0066 is produced in the CMOS process,therefore internal executing time is long.

Standard time is 40us~1.6ms. (This varies by instruction).

When the high speed MPU controls it, check the busy flag before performing Instruction or reading data.

While internal operation is active, Enable signal is not accepted. (Enable signal at Reading status register for checking busy flag is accepted) Busy flag signal is output through DB7, as shown in Table 3, when RS = "0",R/W = "1 ",and Enable="1"

(3) Iuput of unidentified instruction code

Undefined instruction code of SPLC780A OR KS0066 is only as follows;

RS	R/W	DB7~DB
0	0	0~

(Others are included to defined instruction)

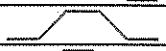
When the undefined instruction code is loaded to SPLC780A OR KS0066, it accepts

the code,but Does not change the internal states (RAM and other status of Flags).

Busy state,

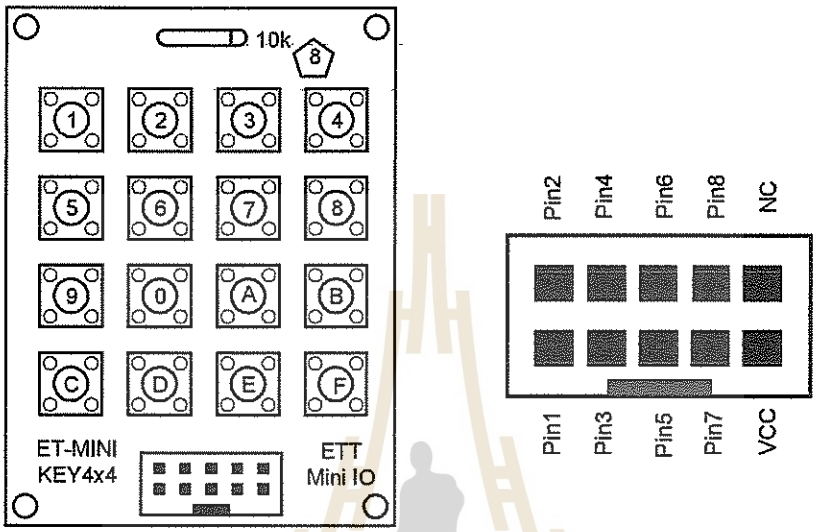
However continues for maximum 40us by the acceptance of the code.

Table 2 The relation between the operation and the combination of RS,R/W

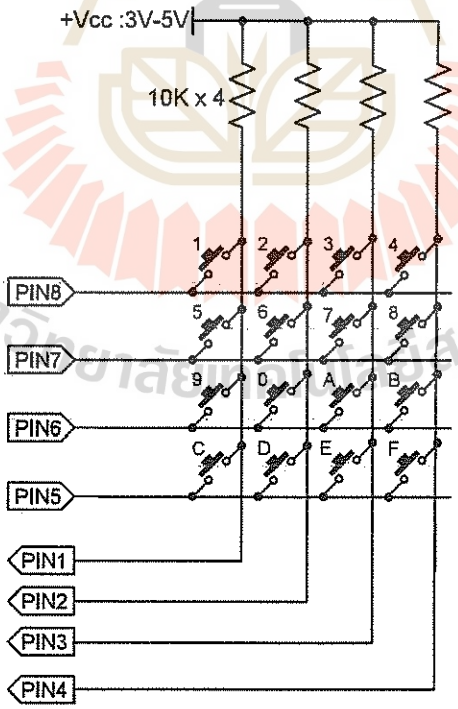
RS	R/W	E	Operation
0	0		Write instruction code
0	1		Read busy flag and address counter
1	0		Write data
1	1		Read data

When performing data and instruction code by 4 bit, transfer RS, R/W every time.

เป็นชุด โดยจะมี ใช้ต่อกับไมโครคอนโทรลเลอร์ทั้งหมด โดย จะใช้ต่อเป็น เพื่อให้ อ่านค่ารหัสของคีย์ที่ถูกกด ส่วน ใช้ต่อเป็น เพื่อให้ ส่งลอจิก มาทำการสแกนคีย์ในแต่ละแถว การทำงานนั้น จะอ่านสถานะทางลอจิกของคีย์แต่ละหลักเข้ามาทาง ซึ่งถ้าไม่มีการกดคีย์จะอ่านลอจิกได้ ถ้ามีการกดคีย์ลอจิกที่อ่านได้ในหลักนั้นจะเป็น แต่ก่อนที่จะอ่านค่าลอจิกแต่ละหลัก จะต้องให้ลอจิก แก่แถวของคีย์แต่ละแถว ในการอ่านลอจิกเข้ามาแต่ละครั้งเสมอ ตำแหน่ง และวงจร แสดงดังรูปด้านล่าง

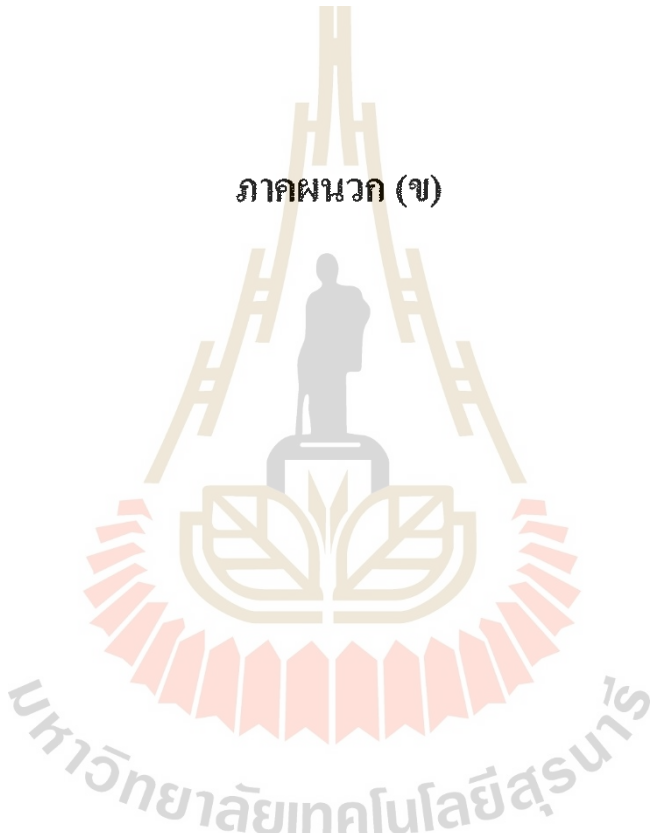


รูป โครงสร้าง และตำแหน่งขา ใช้งาน

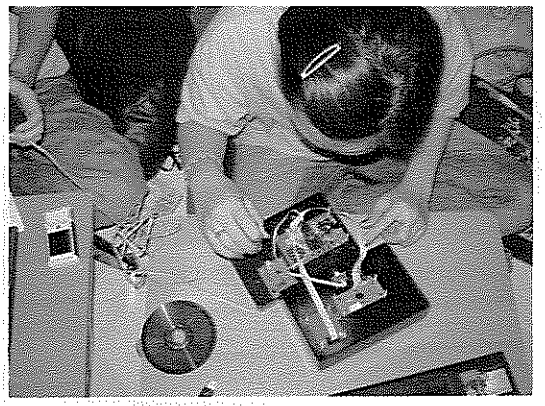


รูป วงจร

ภาคผนวก (ข)



รูปในการปฏิบัติโครงงาน

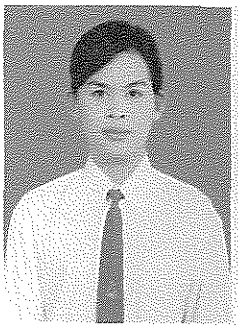


นำวงจรต่างๆต่อลงในกล่อง

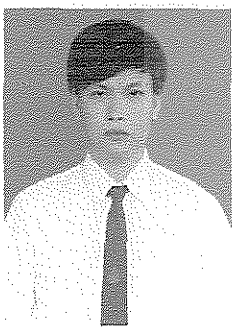
่อเชื่อมวงจรต่างๆในกล่องโดยการบัดกรี

ทดลองจริงกับถังรูปทรงต่างๆ

ประวัติผู้จัดทำ



นายธนวัฒน์ ควรรติกุล เกิดเมื่อวันที่ 24 ธันวาคม พ.ศ. 2528 ภูมิลำเนาเดิมอยู่บ้านเลขที่ 43 ถนนมหาชัยดำริห์ ตำบลตลาด อำเภอเมืองมหาสารคาม จังหวัดมหาสารคาม สำเร็จการศึกษาระดับชั้นมัธยมศึกษาตอนปลายจากโรงเรียนสารคามพิทยาคม อำเภอเมืองมหาสารคาม จังหวัดมหาสารคาม ปัจจุบันกำลังศึกษาอยู่ชั้นปีที่ 5 สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี จังหวัดนครราชสีมา



นายบูรนิติ มาศอมรพันธุ์ เกิดเมื่อวันที่ 19 มกราคม พ.ศ. 2528 ภูมิลำเนาเดิมอยู่บ้านเลขที่ 264/108 หมู่ที่ 13 ถนนเทพารักษ์ ตำบลในเมือง อำเภอเมืองขอนแก่น จังหวัดขอนแก่น สำเร็จการศึกษาระดับชั้นมัธยมศึกษาตอนปลายจากโรงเรียนขอนแก่นวิทยายน อำเภอเมืองขอนแก่น จังหวัดขอนแก่น ปัจจุบันกำลังศึกษาอยู่ชั้นปีที่ 5 สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี จังหวัดนครราชสีมา

นางสาวรัชดาพร มานุวงศ์ เกิดเมื่อวันที่ 19 มีนาคม พ.ศ. 2528 ภูมิลำเนาเดิมอยู่บ้านเลขที่ 14/26 ถนนสระหลวง ตำบลในเมือง อำเภอเมืองพิจิตร จังหวัดพิจิตร สำเร็จการศึกษาระดับชั้นมัธยมศึกษาตอนปลายจากโรงเรียนพิจิตรพิทยาคม อำเภอเมืองพิจิตร จังหวัดพิจิตร ปัจจุบันกำลังศึกษาอยู่ชั้นปีที่ 5 สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี จังหวัดนครราชสีมา