โปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ

(Frequency Counter Program for Vehicles Detection)

โดย

นางสาวไพลิน โตกระโทก รหัส B5103836

นางสาวพิมพิสุทธิ์ พัฒนโสภณกุล รหัส B5115372

นางสาวภัสราภรณ์ วิจารณปัญญา รหัส B5127719

รายงานนี้เป็นส่วนหนึ่งของการศึกษารายวิชา 427499 โครงงานวิศวกรรมโทรคมนาคม

หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมโทรคมนาคม หลักสูตรปรับปรุง พ.ศ. 2546

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

ประจำภาคการศึกษาที่ 3 ปีการศึกษา 2554

โปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ

(Frequency Counter Program for Vehicles Detection)

คณะกรรมการสอบโครงงาน

(ผู้ช่วยศาสตราจารย์ ดร.รังสรรค์ ทองทา)
กรรมการ/อาจารย์ที่ปรึกษาโครงงาน

(ผู้ช่วยศาสตราจารย์ ดร.มนต์ทิพย์ภา อุฑารสกุล)
กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.ชุติมา พรหมมาก)
กรรมการ

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับรายงานโครงงานฉบับนี้ เป็นส่วนหนึ่งของการศึกษาระดับปริญญาตรี สาขาวิชาวิศวกรรมโทรคมนาคม รายวิชา 427499 โครงงานวิศวกรรมโทรคมนาคม ประจำปีการศึกษา 2554

หัวข้อโครงงาน โปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ
(Frequency Counter Program for Vehicles Detection)

จัดทำโดย 1. นางสาวไพลิน โตกระโทก B5103836
2. นางสาวพิมพิสุทธิ์ พัฒนโสภณกุล B5115372
3. นางสาวภัสราภรณ์ วิจารณปัญญา B5127719
อาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ ดร . รังสรรค์ ทองทา
สาขาวิชา วิศวกรรมโทรคมนาคม
ภาคการศึกษาที่ 3/2554

บทคัดย่อ

หลักการตรวจวัดยานพาหนะมีประโยชน์ในการจัดการกับปัญหาการจราจร ที่ปัจจุบันความต้องการในการใช้รถใช้ถนนมีมากขึ้น ซึ่งหลักของการตรวจวัดยานพาหนะนี้สามารถพัฒนาต่อยอดให้เป็นไฟจราจรอัจฉริยะเพื่อควบคุมระบบการจราจรให้มีความเป็นระเบียบและปลอดภัยมากขึ้น หลักการทำงานของการตรวจวัดยานพาหนะนี้ จะประกอบไปด้วยกลไกการทำงานที่สำคัญอยู่ 2 องค์ประกอบคือ โปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ (Frequency Counter Program for Vehicles Detection) ที่ใช้ในการตรวจนับความถี่ ซึ่งจะทำงานร่วมกับวงจรกำเนิดความถี่ (Oscillator Circuit) เมื่อมีค่าความถี่เข้ามาโปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะก็จะทำการตรวจนับความถี่เพื่อไปคำนวณหาอัตราการเปลี่ยนแปลงของความถี่ (Delta) แล้วตัดสินใจว่าในลูปเหนี่ยวนำนั้นมียานพาหนะอยู่หรือไม่

กิตติกรรมประกาศ

คุณความดีอันใดที่เกิดจากโครงงานฉบับนี้ ขอมอบแด่ครอบครัวผู้คอยห่วงใยให้โอกาส ให้กำลังใจ และให้การสนับสนุนทางการศึกษามาโดยตลอด

โครงงานเล่มนี้สามารถสำเร็จลุล่วงไปได้ด้วยดี เนื่องจากได้รับความกรุณาจากอาจารย์ที่ปรึกษา โครงงาน ผู้ช่วยศาสตราจารย์ ดร.รังสรรค์ ทองทา ผู้ที่เป็นเจ้าของแนวคิดเริ่มแรกของโครงงานโปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ (Frequency Counter Program for Vehicles Detection) ตลอดจนคอยให้คำชี้แนะและมุมมองต่างๆ ในการแก้ปัญหา

และนายปัญญา หันตุลา ที่ให้ความช่วยเหลือในการให้แนวคิด การดูแลเอาใจใส่ติดตามงาน ชี้แนะข้อพกพร่องที่ ถูกมองข้าม ตลอดจนฝึกฝนและสนับสนุนให้มีความสามารถในการทำโครงงานจนสามารถนำเสนอผลงานให้เป็นที่รู้จักและยอมรับได้

ขอขอบคุณผู้ที่เกี่ยวข้องอื่นๆดังนี้

- ขอขอบคุณอาจารย์ประจำสาขาวิศวกรรมโทรคมนาคมทุกท่าน ที่ อบรมสั่งสอนและให้ความรู้มาโดยตลอด
- ขอขอบคุณเพื่อนๆนักศึกษาสาขาวิศวกรรมโทรคมนาคมทุกคนที่เป็นกำลังใจและให้ความร่วมมือกันในการทำงานจนประสบความสำเร็จลุล่วงมาได้ด้วยดี

นางสาวไพลิน โตกระโทก
นางสาวพิมพิสุทธิ์ พัฒนโสภณกุล
นางสาวภัสราภรณ์ วิจารณปัญญา

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญภาพ

รูป **หน้า**

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

รูป **หน้า**

สารบัญภาพ (ต่อ)

รูป **หน้า**

สารบัญตาราง

ตาราง **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมา

ในปัจจุบันมีความต้องการในการใช้รถใช้ถนนกันมากขึ้น จึงก่อให้เกิดปัญหาในเรื่องของการจราจรที่คับคั่งซึ่งเสี่ยงต่อการเกิดอุบัติเหตุได้ง่าย การควบคุมระบบไฟจราจรนับเป็นอีกหนึ่งวิธีที่จะจัดการกับปัญหาดังกล่าว ซึ่งในการควบคุมระบบไฟจราจรจะต้องมีกลไกการทำงานที่สำคัญคือกระบวนการของ การตรวจวัดยานพาหนะ เครื่องตรวจวัดยานพาหนะนี้เมื่อทำสำเร็จขึ้นมาสามารถนำ ไปเชื่อมโยงข้อมูลเข้าในโปรแกรมสัญญาณจราจรอัจฉริยะเพื่อแก้ปัญญาการจราจรติดขัดได้ในรูปแบบต่างๆ ที่ผู้ใช้ต้องการอีกด้วย

1.2 หลักการและเหตุผล

โปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ (Frequency Counter Program for Vehicles Detection) จะใช้อัตราการเปลี่ยนแปลงความถี่ในการตรวจวัดยานพาหนะ โดยการทำงานของโปรแกรมจะทำงานร่วมกับวงจรกำเนิดความถี่ (Oscillator Circuit) เมื่ออัตราการเปลี่ยนแปลงความถี่มีค่ามากกว่าค่าของ Sensitivity ที่กำหนดไว้ โปรแกรมจะทำการตัดสินใจว่าพบรถยนต์เข้ามาอยู่ในลูปเหนี่ยวนำ ซึ่งหลักการนี้สามารถพัฒนาต่อยอดเพื่อเป็นแนวทางในการสร้างระบบควบคุมสัญญาณไฟจราจรได้

1.3 วัตถุประสงค์

1. เพื่อศึกษากลไกการทำงานของลูปที่ใช้เป็นตัวตรวจวัดและศึกษาการใช้งานร่วมกันระหว่าง ลูปเหนี่ยวนำ,วงจรวงจรกำเนิดความถี่และโปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ
2. เพื่อศึกษาเป็นแนวทางในการพัฒนาเครื่องมือวัดทางด้านการจราจร
3. เพื่อพัฒนาต่อยอดในการแก้ไขปัญหาจราจร
4. เพื่อศึกษาและออกแบบโปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ
5. เพื่อศึกษาโปรแกรมควบคุมและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์

1.4 ขอบเขตงาน

1. โปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะสามารถอ่านความถี่ที่ตรวจวัดได้

2. โปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ สามารถหาค่าอัตราการเปลี่ยนแปลงความถี่แล้วทำการตัดสินใจว่าในลูปเหนี่ยวนำมีรถอยู่หรือไม่

1.5 การเลือกใช้ซอฟต์แวร์

ซอฟต์แวร์ที่ใช้ในโครงงานนี้ ผู้จัดทำได้เลือกใช้ภาษาซีในการควบคุมสั่งงาน ไมโครคอนโทรลเลอร์ เนื่องจากภาษาซีเป็นภาษาโครงสร้างง่ายต่อการทำความเข้าใจ สามารถปรับปรุงพัฒนาต่อได้ นอกจากนั้นภาษาซียังเป็นภาษามาตรฐานไม่ขึ้นกับฮาร์ดแวร์ (ไมโครคอนโทรลเลอร์) มีความยืดหยุ่นในการนำไปใช้งานร่วมกับไมโครคอนโทรลเลอร์ตระกูลอื่นได้ง่าย

1.6 ขั้นตอนการดำเนินการ

1. ปรึกษาอาจารย์ที่ปรึกษาโครงงานเกี่ยวกับขอบเขตของโครงงานที่จะทำ
2. ศึกษาข้อมูลเกี่ยวกับอุปกรณ์บอร์ด CP-JRARM7 USB-LPC2148
3. สั่งซื้ออุปกรณ์ที่เกี่ยวข้องในการทำโครงงาน
4. ฝึกการใช้โปรแกรมในการเขียนคำสั่งควบคุมไมโครคอนโทรลเลอร์
5. เขียนโปรแกรมเพื่อสั่งการให้ไมโครคอนโทรเลอร์ทำงานตามวัตถุประสงค์
6. ทดลองใช้งานและแก้ไขสิ่งที่ผิดพลาด
7. จัดทำรูปเล่มรายงานของโครงงานเพื่อเสนออาจารย์ประจำสาขาวิชา

1.7 ผลที่คาดว่าจะได้รับ

1. ได้เรียนรู้และเข้าใจหลักการทำงานของโปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ
2. ได้เรียนรู้การเขียนโปรแกรมควบคุมและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์
3. สามารถนำความรู้ที่ได้ทางทฤษฎีมาประยุกต์ใช้ในทางปฏิบัติ
4. ได้เรียนรู้วิธีหาความรู้ได้ด้วยตัวเองเพื่อนำมาปฏิบัติและประยุกต์ใช้งานจริง
5. มีทักษะในการใช้งานอุปกรณ์วัดในห้องปฏิบัติการมากขึ้น
6. สามารถเรียนรู้การทำงานและการแลกเปลี่ยนความรู้กับบุคคลอื่น

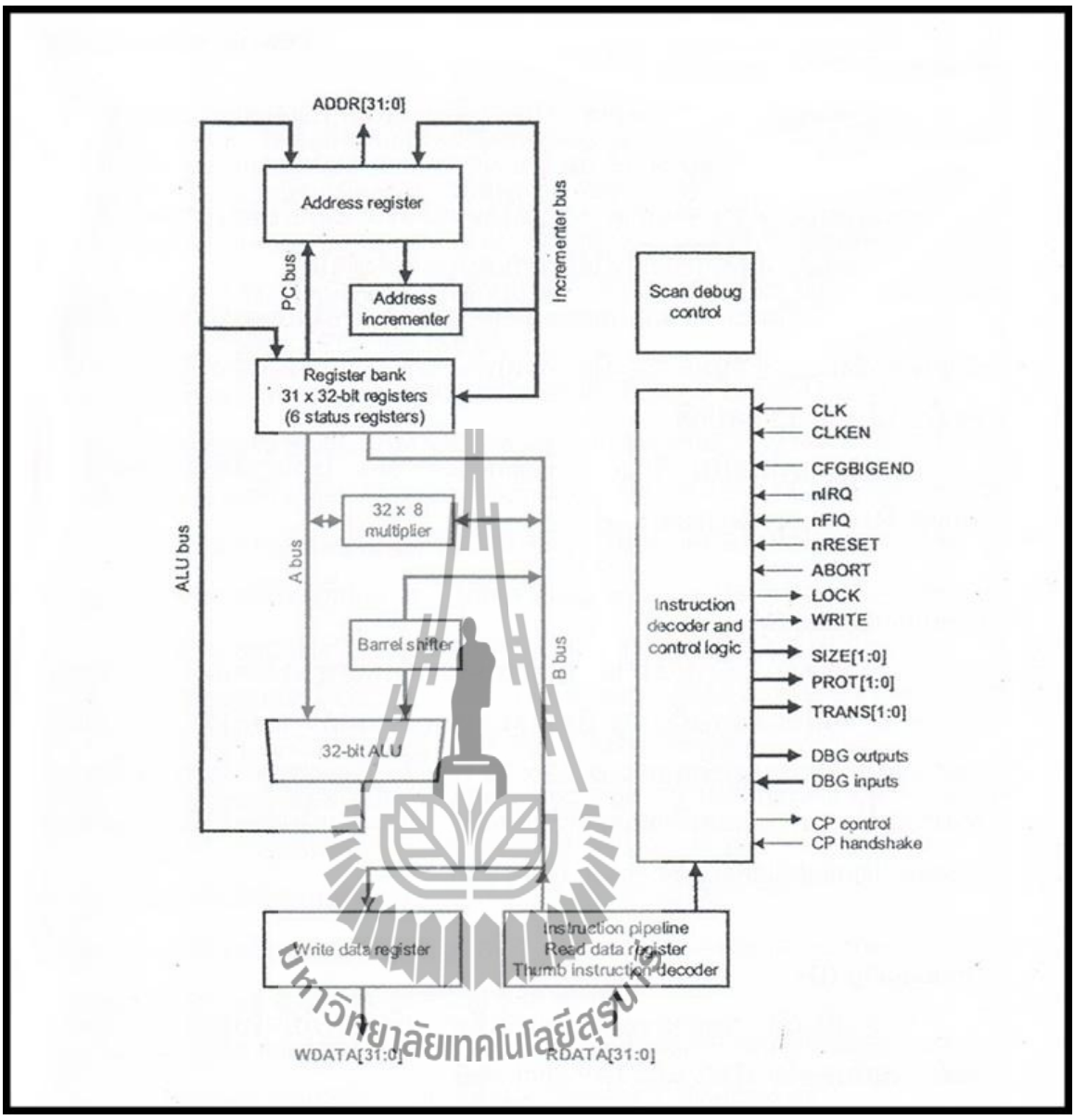
ระหว่างบริษัท Apple Computer , Acorn Computer Group และ VLSI Technology โดยบริษัท Apple และ VLSI Technology เป็นผู้ลงทุน โดยบริษัท Acorn เป็นผู้อนุญาตให้ใช้เทคโนโลยีและโอนทีมวิศวการผู้ออกแบบจำนวน 12 คน

ซีพียู ARM รุ่นแรกที่ออกจำหน่ายในปี 1991 เป็นซีพียูตระกูล ARM6 จัดว่าเป็นซีพียู RISC 32 บิตแบบฝังตัว (Embedded RISC) ตัวแรกของโลก นับเป็นการสร้างมาตรฐานใหม่ให้กับวงการซีพียู 32 บิต หลังจากนั้นก็มีบริษัทผู้ผลิตซีพียูไมโครโปรเซสเซอร์และไมโครคอนโทรลเลอร์ทั่วโลกอื่นๆเข้าร่วมผลิตซีพียูตระกูล ARM ในปัจจุบันมีซีพียู ARM ต่างๆ ดังนี้ ARM 7 , ARM 9 , ARM 9E , ARM 10 และ ARM 11

ในยุคแรกซีพียู ARM เป็นซีพียู RISC 32 บิต การทำงานจำเป็นต้องต่อกับหน่วยความจำและอุปกรณ์ภายนอกเมื่อมีบริษัทผู้ผลิตไมโครคอนโทรลเลอร์จำนวนมากได้นำลิขสิทธิ์ของซีพียู ARM ไปพัฒนาต่อได้ มีการเพิ่มหน่วยความจำภายในทั้ง ROM และ RAM และเพิ่มโมดูลอุปกรณ์เสริมต่างๆ เช่น วงจรสื่อสารแบบอนุกรม วงจรแปลงดิจิตอลเป็นแอนาลอก เป็นต้น ทำให้กลายเป็นไมโครคอนโทรลเลอร์แบบ 32 บิต ที่กินพลังงานต่ำสามารถทำงานได้โดยใช้ชิปไอซีเพียงตัวเดียวโดยไม่ต้องต่ออุปกรณ์เพิ่มเติมภายนอก ทำให้มีการนำไปใช้ในงานคอมพิวเตอร์ฝังตัว (Embedded Computer) มากขึ้น

2.1.2 สถาปัตยกรรมซีพียู ARM 7

สถาปัตยกรรมของ ARM7 เป็นซีพียูแบบ RISC ขนาด 32 บิต ภายในมีบัสขนาด 32 บิตตัวเดียวที่ใช้สำหรับรับส่งข้อมูลและคำสั่ง ชุดคำสั่งจะมีขนาด 32 บิตคงที่ ในขณะที่ข้อมูลสามารถเลือกได้ว่าจะมีขนาด 8 , 16 หรือ 32 บิต โดยแสดงแกนกลาง (Core) ของซีพียู ARM 7 ได้ดังรูป



รูปที่ 2.1 แกนกลางของซีพียู ARM 7

โครงสร้างของ ARM 7 จะเป็นแบบที่เรียบง่าย มีชุดคำสั่งไม่มากนัก ประหยัดพื้นที่สารกึ่งตัวนำที่ใช้สร้างอีกทั้งยังประหยัดพลังงานอีกด้วย

สถาปัตยกรรมของ ARM 7 จะเป็นแบบ Load-and–store ในการประมวลผลข้อมูลใดๆต้องกระทำผ่านทางรีจิสเตอร์ เริ่มต้นด้วยการโหลดค่าจากหน่วยความจำเก็บในรีจิสเตอร์ นำค่ามาประมวลผล เสร็จแล้วจะเขียนค่าเก็บในหน่วยความจำดังเดิม

รีจิสเตอร์ของ ARM 7 ที่ใช้งานได้สำหรับผู้ใช้มีทั้งหมด 16 ตัว คือ R0–R15 โดยทุกตัวมีขนาด 32 บิต โดย R0–R12 เป็นรีจิสเตอร์ทั่วไปที่ไม่ได้กำหนดหน้าที่การทำงานพิเศษ ส่วน R12 ทำหน้าที่เป็น Stack Pointer (SP) R14 ทำหน้าที่เป็น Link Register (LR) และ R15 ทำหน้าที่เป็น Program Counter (PC)

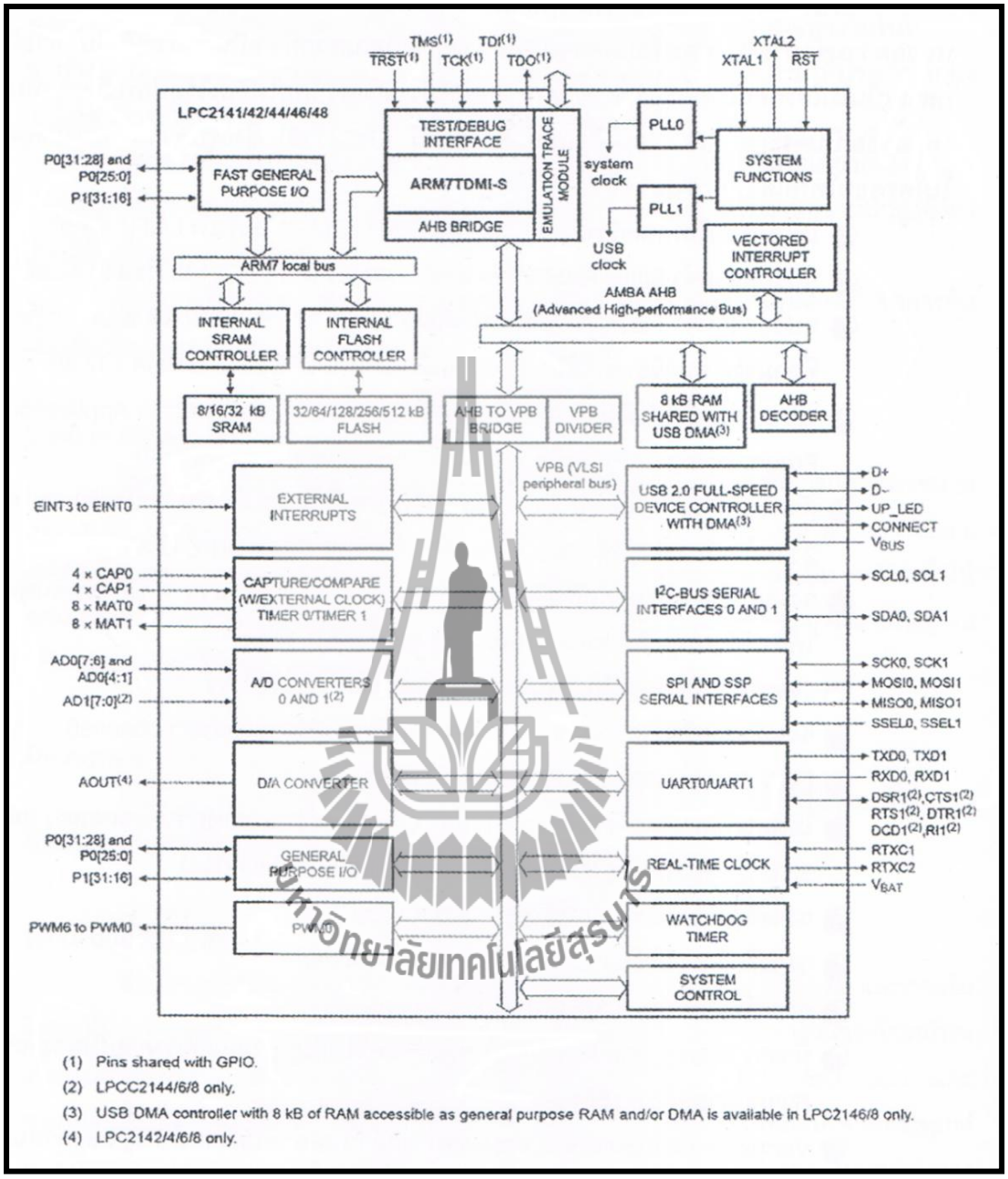
2.1.3 ไมโครคอนโทรลเลอร์ ARM 7 Phillips LPC2148

บริษัท Philips ได้ซื้อลิขสิทธิ์เพื่อผลิตซีพียูตระกูล ARM 7 โดยได้นำมาพัฒนาเพิ่มหน่วยความจำภายใน และเพิ่มโมดูลอุปกรณ์ต่างๆ จำนวนมากที่จำเป็นต่อการใช้งานทำให้กลายเป็นไมโครคอนโทรลเลอร์ที่สามารถทำงานได้ด้วยตนเอง โดยไม่ต้องใช้อุปกรณ์อื่นๆ เพิ่ม โดยตั้งชื่อเป็นตระกูล LPC ซึ่งในที่นี้ใช้เป็น LPC2148 ซึ่งสรุปความสามารถได้ดังนี้

- ไมโครคอนโทรลเลอร์ขนาด 16/32 บิต ARM 7 TDMI-S ในตัวถัง LQFP 64 ขา
- หน่วยความจำ Static RAM ขนาด 8 kb–40 kb โดย LPC2148 มีขนาด 40 kb
- หน่วยความจำ Flash Program Memory ขนาด 32 kb–512 kb อยู่ภายในชิปที่สามารถลบ / เขียนซ้ำได้ถึง 10,000 ครั้ง ถ้าเป็น LPC 2148 มีขนาด 512 kb
- โปรแกรมชิปได้ทันทีผ่าน In-System Programming (ISP) และ In-Application Programming (IAP) โดยใช้ซอฟแวร์ Boot-loader ที่อยู่ภายในชิป
- ตัวควบคุมอุปกรณ์ USB 2.0 Full-speed โดยมีแรมสำหรับ end is point ขนาด 2 kb ถ้าเป็น LPC2148 จะมี RAM 8 kb สำหรับการติดต่อแบบ DMA
- วงจรแปลงแอนาลอกเป็นดิจิตอลความละเอียด 10 บิต จำนวน 2 ชุด ที่รับอินพุตได้ถึง 14 อินพุต โดยมีเวลาในการแปลงค่าต่ำถึง 2.44 us
- วงจรแปลงดิจิตอลเป็นแอนาลอกความละเอียด 10 บิต 1ตัว
- วงจรไทเมอร์ขนาด 32 บิต 2 ชุด (มี 4 capture และ 4 compare channel)
- PWM (Pulse Width Modulation) 6 เอาต์พุต
- โมดูลนาฬิกาเวลาจริง (Real Time Clock) ที่สามารถต่อกับคริสตัลความถี่ 32 kHz และแบตเตอรี่ภายนอกได้ และวอชดอกซ์ (Watchdog)
- วงจรสื่อสารอนุกรม UART (16C550) จำนวน 2 ชุด

- วงจรสื่อสารอนุกรม I^2C ความเร็วสูง (400 kbits/s)
- วงจรสื่อสารอนุกรม SPI และ SSP
- มีวงจร Phase Lock Loop ภายในเพื่อคูณค่าให้สัญญาณนาฬิกาภายในทำงานที่ความถี่สูงสุดถึง 60 MHz
- Vectored Interrupt Controller ที่สามารถกำหนดลำดับความสำคัญและกำหนดแอดเดรสของเวกเตอร์ได้
- ใช้กับแหล่งจ่ายไฟชุดเดียวขนาด 3.0 V ถึง 3.6 V
- มี I/O pin เอนกประสงค์ที่สามารถใช้กับระดับแรงดัน 5 V ได้สูงสุด 45 ขา โดยสามารถจัดเป็นขาอินเตอร์รัปต์จากภายนอกได้สูงสุด 21 ขา
- มีโหมดประหยัดพลังงาน 2 โหมด ได้แก่ Idle และ Power–down

2.1.4 บล็อกไดอะแกรมของ LPC2148



รูปที่ 2.2 บล็อกไดอะแกรมของไมโครคอนโทรลเลอร์ LPC2148

จากรูปส่วนบนเป็นไมโครโปรเซสเซอร์ ARM 7 TDMI-S ซึ่งเป็นหัวใจหลักด้านซ้ายมือที่เป็นส่วนของ AMR7 Local Bus ที่ใช้ในการติดต่อกับหน่วยความจำแบบ Flash ที่ใช้เก็บโปรแกรม และหน่วยความจำ SRAM ที่ใช้เก็บข้อมูล ส่วนที่ใช้ในการติดต่อกับหน่วยความจำภายนอกมีการ

ติดต่อผ่านบัส AMBA AHB (Advanced High-performance Bus) ซึ่งใน LCP2148 ไม่สามารถต่อกับหน่วยความจำภายนอกได้

ในการติดต่อกับอุปกรณ์รอบข้างเช่น GPIO , I^2C , SPI , UART เป็นต้น จะติดต่อผ่านทางบัส VPB (VLSI peripheral BUS) ซึ่ง VPB บัสนี้ต่อกับ AHB บัสผ่าน AHB to VPB Bridge โดยสามารถปรับลดค่าความถี่ของ VPB บัสให้ทำงานช้ากว่าความถี่ของซีพียูได้เพื่อให้ทำงานร่วมกับอุปกรณ์เสริมต่างๆที่มีความเร็วต่ำกว่าได้

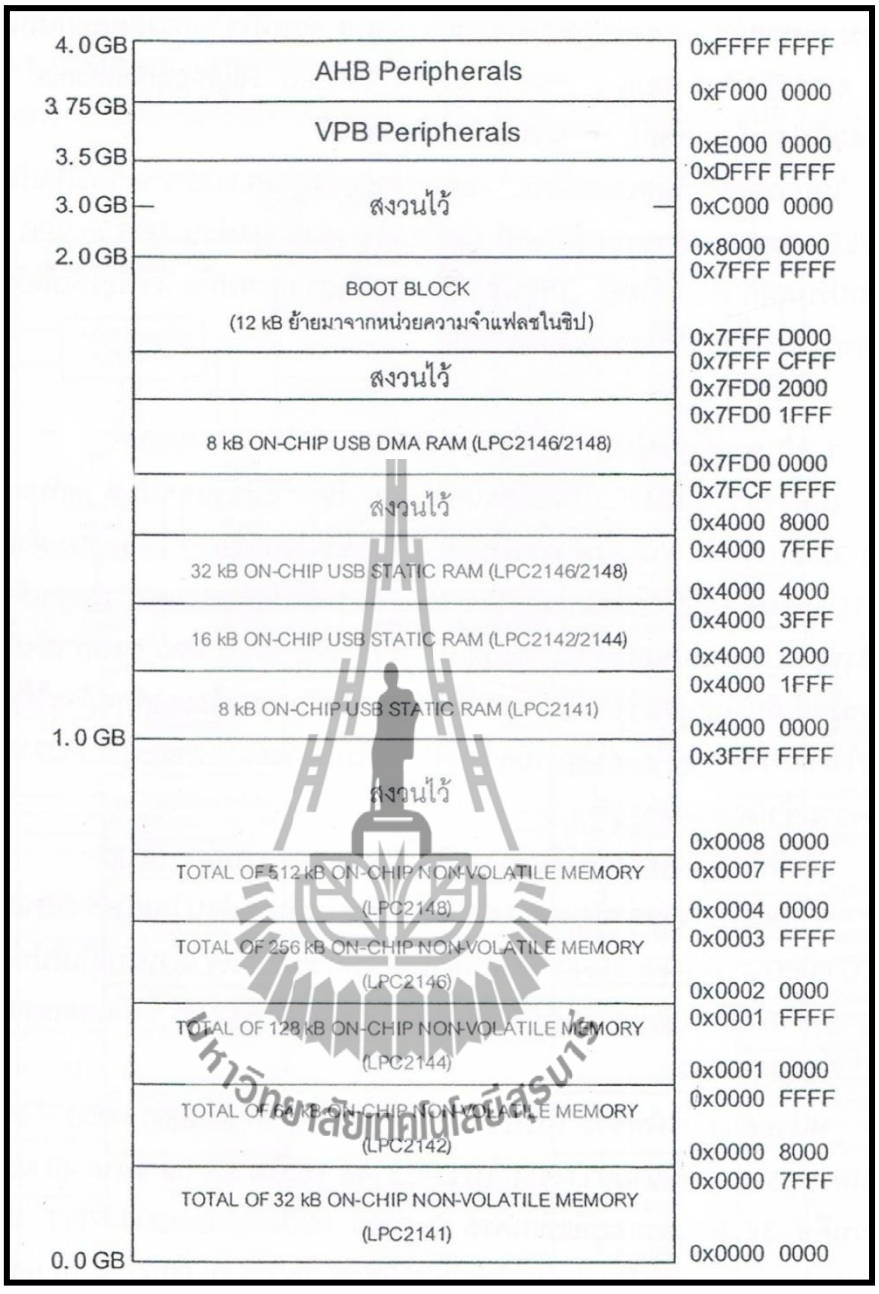
2.1.5 การจัดหน่วยความจำของ LPC2148

เนื่องจาก ARM7 เป็นซีพียูขนาด 32 บิต ที่มีขาแอดเดรสต่อกับหน่วยความจำจำนวน 32 เส้น ทำให้สามารถอ้างหน่วยความจำได้ถึง 4 GB (2^{32} = 4 GB) ตัวแกนหลักของ ARM7 DMI จะมีสถาปัตยกรรมแบบ Von Neumann ที่ใช้บัสขนาด 32 บิตชุดเดียวกันสำหรับตัวคำสั่งของโปรแกรมและข้อมูลโดยมีคำสั่ง load , store และ swap เท่านั้นที่ใช้ในการเรียกข้อมูลที่เก็บในหน่วยความจำ การติดต่อกับพอร์ตอินพุตหรือเอาต์พุตก็จะใช้คำสั่งเดียวกันกับการใช้คำสั่งจัดการเกี่ยวกับหน่วยความจำ ในไมโครคอนโทรลเลอร์ LPC2148 ได้จัดสรรหน่วยความจำโดยรวมดังรูปที่ 2.3

เมื่อซีพียูถูกรีเซ็ตจะเริ่มต้นทำงานที่หน่วยความจำ 0x000 0000

จากหน่วยความจำทั้งหมด 4 GB ได้แบ่งออกเป็น 4 ส่วน โดยใน 1 GB แรก (แอดเดรส 0x0000 0000 – 0x3FFF FFFF) จัดเป็นส่วนของหน่วยความจำสำหรับเก็บโปรแกรม ซึ่งกรณีของ LPC2148 ก็คือหน่วยความจำ Flash memory ขนาด 512 kB ซึ่งมีแอดเดรสจาก 0x0000 0000 – 0x0007 FFFF

หน่วยความจำในช่วง 1GB ถึง 2GB (แอดเดรส 0x4000 0000 – 0x7FFF FFFF) จัดเป็นส่วนของหน่วยความจำ RAM ใน LPC2148 จะเป็น SRAM ขนาด 40 kB โดยแย่งเป็น 2 ส่วน คือ 32 kB แรกอยู่ที่แอดเดรส 0x4000 0000 – 0x4000 7FFF และอีก 8 kB เป็นหน่วยความจำใช้สำหรับ USB โดยมีแอดเดรส 0x7FD0 0000 – 0x7FD0 1FFF ในกรณีที่ไม่ใช้การติดต่อกับ USB สามารถนำหน่วยความจำ RAM ส่วนนี้มาใช้เป็น RAM สำหรับใช้งานทั่วไปได้ หน่วยความจำส่วนที่ใกล้กับ 2 GB จะเป็นส่วนของ Boot Block ขนาด 12 ซึ่งเป็นโปรแกรมที่ทำงานเพื่อเขียนโปรแกรมลงในหน่วยความจำ Flash ของ LPC2148



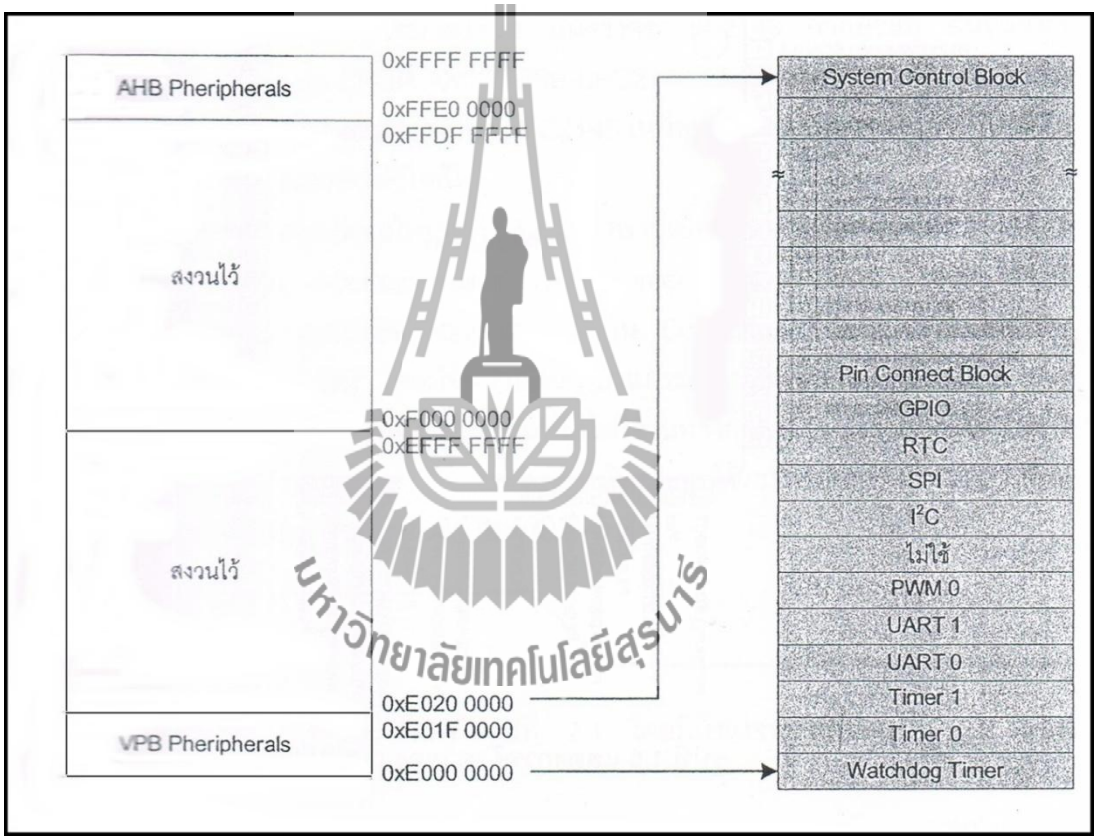
รูปที่ 2.3 การจัดหน่วยความจำของ LPC2148

หน่วยความจำช่วง 2 GB ถึง 3.5 GB (แอดเดรส 0x8000 0000 – 0xDFFF FFFF) สงวนไว้สำหรับต่อกับหน่วยความจำภายนอก ซึ่งกรณี LPC2148 ไม่ได้ใช้งาน

หน่วยความจำในช่วง 3.5 GB ถึง 4 GB จะเป็นพื้นที่สำหรับติดต่อกับอุปกรณ์อื่นที่อยู่ในชิป โดยแบ่งเป็นอุปกรณ์ที่ต่อกับ VPB บัสจะติดต่อกับหน่วยความจำช่วง 0xE000 0000 ถึง 0xEFFF FFFF

ถ้าเป็นอุปกรณ์ที่ติดต่อผ่านทาง AHB จะมีช่วงแอดเดรส 0xF000 0000 ถึง 0xFFFF FFFF ซึ่งในกรณีของ LPC2148 ไม่สามารถติดต่อกับหน่วยความจำภายนอก จึงไม่ได้ใช้หน่วยความจำในส่วนนี้

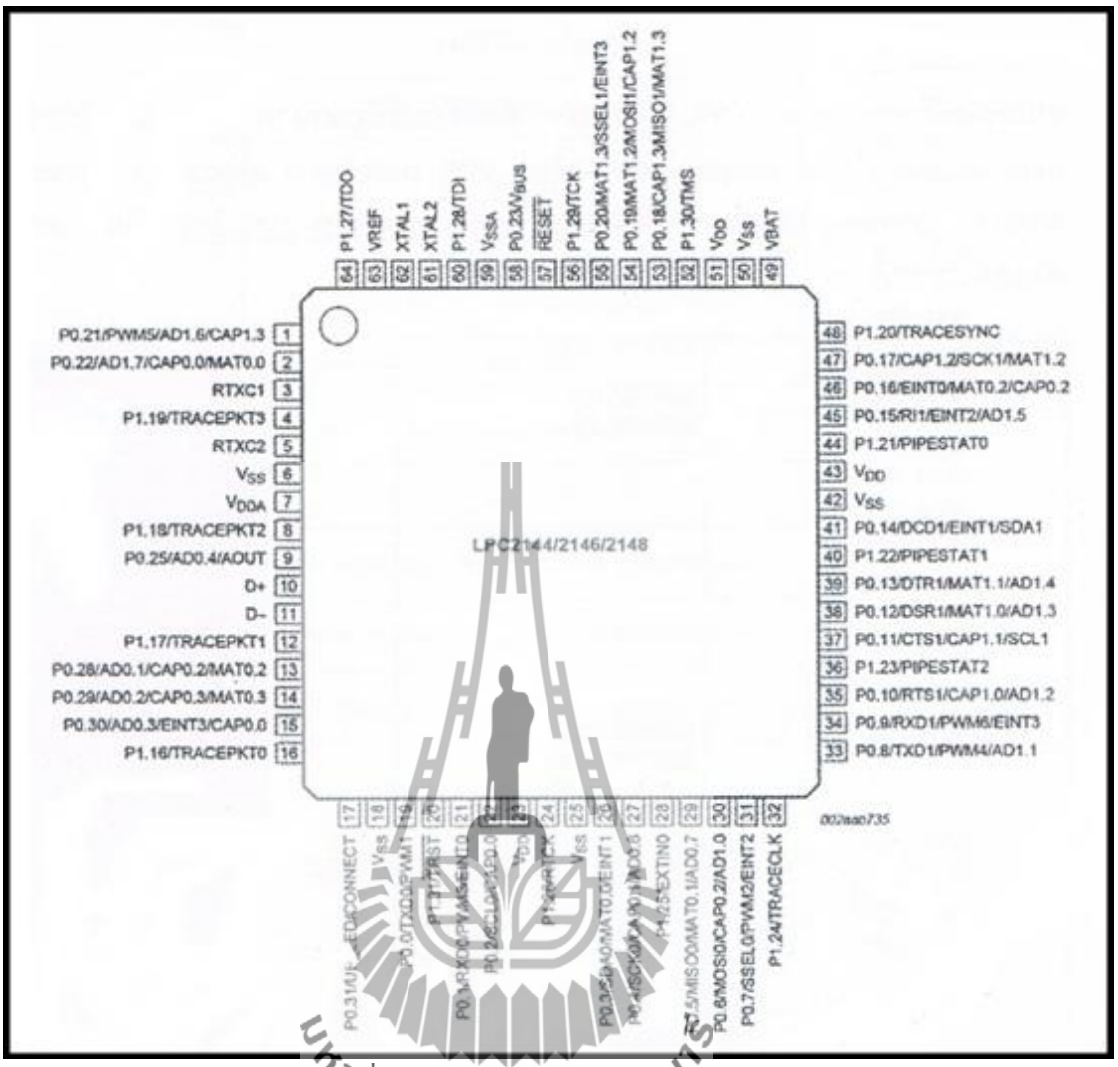
อุปกรณ์เสริมต่างๆ เช่น UART , RTC ฯลฯ ที่ติดต่อกับซีพียูผ่าน VPB bus มีชื่อเรียกว่า VPB peripheral#0–VPB papripheral#127 โดย VPB peripheral แต่ละตัวจะมีช่วงของหน่วยความจำอุปกรณ์ละ 16 kB ทำให้คิดเป็นหน่วยความจำรวม /MB โดยจะมีแอดเดรสต่างๆดังแสดงดังรูป



รูปที่ 2.4 ตำแหน่งแอดเดรสของอุปกรณ์เสริมที่ต่อกับ VPB บัส

2.1.6 การจัดขาของ LPC2148

ไมโครคอนโทรลเลอร์ LPC2148 อยู่ในตัวถังแบบ plastic low profile quad flat package (LQFP64) ที่มีขาต่อจำนวน 64 ขาโดยการจัดขาแสดงได้ในรูป

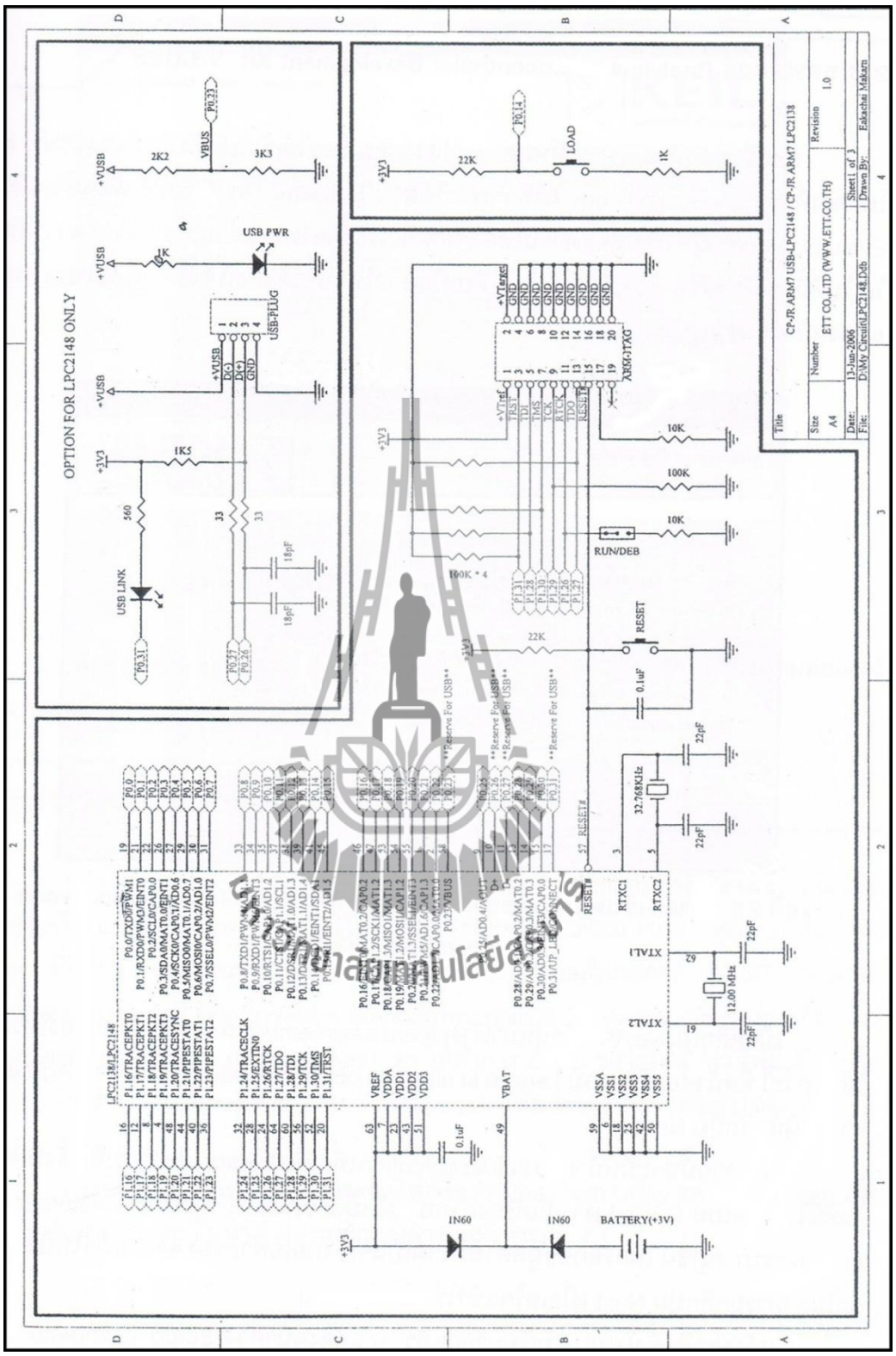


รูปที่ 2.5 แสดงการจัดขาของ LPC2148

หลังจากเกิดการรีเซ็ตขาพอร์ตทั้งหมดจะถูกกำหนดให้ทำหน้าที่เป็นอินพุต ขาแต่ละขาจะมีหน้าที่การทำงานหลายหน้าที่ดังตัวอย่างขาที่ 14 สามารถทำหน้าที่ได้ 4 หน้าที่ คือเป็น P0.29/AD0.2/CAP0.3/MAT0.3 ถ้าเป็นพอร์ตอินพุตเอาต์พุตจะเรียกว่าเป็นอินพุตเอาต์พุตอเนกประสงค์ (General Purpose Input Output : GPIO) ก็คือขา P0.29 ถ้าใช้วงจรแปลงแอนาลอกเป็นดิจิตอลขานี้ก็คือ AD 0.2 ซึ่งเป็นขาอินพุตสำหรับสัญญาณแอนาลอก ADC0 Channel 3 (Capture input for timer 0 , Channel3) หรือเป็น MAT 0.3 เป็นแมตช์เอาต์พุตสำหรับไทเมอร์ 0 แชนแนล 3 (Match output for timer0 , Channel)

2.1.7 แผงวงจร LPC2148 มีคุณสมบัติดังนี้

- ใช้ MCU ARM7 ของ PHILIPs เบอร์ LPC2148
- ใช้ X TAL 12.00MHz ความเร็วสูงสุดที่ 60 MHz
- USB FULL SPEED 2.0
- 14 PIN LCD PORT สำหรับต่อกับจอแสดงผล LCD แบบตัวอักษร
- 20 PIN BUS JTAG ARM
- ใช้กับแหล่งจ่ายไฟรุ่น 5VDC
- PROJECT BOARD รุ่น AD – 100 ขนาด 360 จุด
- VR 10K R ปรับค่า 4 ชุด สำหรับการทดลองวงจรแปลงแอนาลอกเป็นดิจิตอล
- TACT สวิตซ์ 4 ชุด สำหรับสร้างสัญญาณอินพุต
- LED DOT จำนวน 4 ชุด สำหรับแสดงผลลอจิก



รูปที่ 2.6 ผังวงจรของบอร์ด LPC2148

2.1.8 การกำหนดค่าเริ่มต้นให้กับไมโครคอนโทรลเลอร์ ARM7

ขั้นแรกก่อนที่ไมโครคอนโทรลเลอร์จะเริ่มทำงานต้องกำหนดค่าเริ่มต้นให้ก่อนโดยค่าที่ต้องกำหนดคือส่วนหน่วยความจำ RAM ที่ใช้เป็น Stack Pointer โดยเราจะใช้ชุดพัฒนาไมโครคอนโทรลเลอร์ Keil เป็นผู้ช่วยสร้างไฟล์ Startup.s ซึ่งเป็นภาษาแอสเซมบลีสำหรับกำหนดค่าของ Stack pointer ให้ภายในของไมโครคอนโทรลเลอร์ ARM7 มีวงจรเฟสลอกลูป (Phase Lock Loop:PLL) สำหรับคูณค่าความถี่ของสัญญา ณนาฬิกาจากภายนอกเพื่อให้ตัวไมโครคอนโทรลเลอร์สามารถทำงานที่ความถี่สูงๆ ได้โดยให้ผู้ใช้เป็นผู้กำหนดค่าตัวคูณความถี่ของสัญญาณนาฬิกาจากภายนอก เพื่อให้ตัวไมโครคอนโทรลเลอร์สามารถทำงานที่ความถี่สูงๆ ได้โดยให้ผู้ใช้เป็นผู้กำหนดค่าตัวคูณความถี่สำหรับ LPC2148 ภายในจะมีวงจรเฟสลอกลูปอยู่สองวงจรโดยตั้งชื่อเป็น PLL0 และ PLL1 โดย PLL0 ใช้ในการคูณค่าความถี่ของสัญญาณนาฬิกาจากภายนอกให้กับซีพียู ARM7 และ PLL1 ที่ทำหน้าที่คูณความถี่ให้กับวงจรส่วนของ USB โดยต้องคูณค่าให้ได้ความถี่ 48 MHz

2.1.9 การทดลองต่อ GPIO เป็นเอาต์พุต

ภายในไมโครคอนโทรลเลอร์ LPC2148 มีพอร์ตอเนกประสงค์ (General Purpose I/O) ขนาด 32 บิต ให้ใช้งาน 2 พอร์ต คือ Port0 และ Port1 โดยใน LPC2148 จะจัดให้ Port0 มีขาต่อใช้งานได้ 29 ขาคือ P0.0 ถึง P0.31 โดย ไม่มีขา P0.24, P0.26 และ P0.27 ให้ใช้งานและขา P0.31 ทำหน้าที่เป็นเอาต์พุตได้อย่างเดียว ส่วน Port1 จะมีต่อให้ใช้งานแค่ 16 ขาคือ P1.16 ถึง P1.31 โดยแต่ละขาของพอร์ต P0 และ P1 มีหน้าที่การทำงานได้หลายประเภท ซึ่งต้องกำหนดที่รีจิสเตอร์ PINSEL

2.1.10 การกำหนดหน้าที่การทำงานของ Port

หลังจากเกิดการรีเซ็ตไมโครคอนโทรลเลอร์ ARM7 วงจรภายในสั่งรีเ ซ็ตค่ารีจิสเตอร์ PINSEL ทุกตัวกำหนด ค่าให้ Port0 และ Port1 ทุกขาเป็น GPIO และให้ทุกขาเป็นอินพุต ขาแต่ละขาของ Port0 และ Port1 มีหน้าที่การทำงานได้หลายหน้าที่โดยกำหนดการทำงาน Port0 ที่รีจิสเตอร์ PINSEL0, PINSEL1 และกำหนดหน้าที่การทำงานของ Port1 ที่รีจิสเตอร์ PINSEL2 โดยรีจิสเตอร์แต่ละตัวจะมีแอดเดรสแสดงในตารางที่ 2.1

ตารางที่ 2.1 รีจิสเตอร์ PINSEL0,PINSEL1และ PINSEL2

ชื่อ	ความหมาย	การติดต่อ	แอดเดรส
PINSEL0	Pin function select register 0 กำหนดการทำงานของ P0.0-P0.15	อ่าน/เขียน	0xE002 C000
PINSEL1	Pin function select register 1 กำหนดการทำงานของ P0.16-P0.31	อ่าน/เขียน	0xE002 C004
PINSEL2	Pin function select register 2 กำหนดการทำงานของ P1.0-P1.31	อ่าน/เขียน	0xE002 C014

ค่าแต่ละบิตของรีจิสเตอร์ PINSEL0 แสดงได้ในตารางที่ 2.2 จากตารางจะพบว่าแต่ละขาจะมีหน้าที่ การ ทำงานได้ 3 หรือ 4 หน้าที่ ตัวอย่างเช่นขา P0.1 จะมีหน้าที่การทำงานได้ถึง 4 หน้าที่ คือ GPIO0.1,RxD0 (UART0) ,PWM3 และ EINT0 ถ้าต้องการกำหนดค่าให้ขา P0.0-P0.15 มีการทำงานเป็น GPIO ทำงานเป็น GPIO ทำได้โดยการเขียนค่า 0 ทุกบิตให้กับ PINSEL0 ซึ่งเขียนเป็นคำสั่งภาษาซีได้ดังนี้

```
PINSEL0 = 0x00000000; //Set P0.0-P0.15 to GPIO Function
```

ตารางที่ 2.2 ค่าประจำบิตของรีจิสเตอร์ PINSEL0

PINSEL0	ชื่อขา	การทำงาน เมื่อมีค่า 00	การทำงาน เมื่อมีค่า 01	การทำงาน เมื่อมีค่า10	การทำงาน เมื่อมีค่า11	ค่าหลัง รีเซ็ต
1:0	P0.0	GPIO Port 0.0	TxD0(UART0)	PWM1	Reserved	00
3:2	P0.1	GPIO Port 0.1	RxD0(UART0)	PWM3	EINT0	00
5:4	P0.2	GPIO Port 0.2	SCL0(I^2C)	Capture 0.0(TIMER0)	Reserved	00
7:6	P0.3	GPIO Port 0.3	SDA0(I^2C)	Match0.0(TIMER0)	EINT1	00
9:8	P0.4	GPIO Port 0.4	SCK0(SPI0)	Capture 0.1(TIMER0)	AD0.6	00
11:10	P0.5	GPIO Port 0.5	MISO0(SPI0)	Match0.1(TIMER0)	AD0.7	00

ตาราง 2.2 ค่าประจำบิตของรีจิสเตอร์ PINSEL0 (ต่อ)

PINSEL0	ชื่อขา	การทำงาน เมื่อมีค่า 00	การทำงาน เมื่อมีค่า 01	การทำงาน เมื่อมีค่า10	การทำงาน เมื่อมีค่า11	ค่าหลัง รีเซ็ต
13:12	P0.6	GPIO Port 0.6	MOSI0(SPI0)	Capture 0.1 (TIMER0)	AD1.0	00
15:14	P0.7	GPIO Port 0.7	SSEL0(SPI0)	PMW2	EINT2	00
17:16	P0.8	GPIO Port 0.8	TxD1(UART1)	PMW4	AD1.1	00
19:18	P0.9	GPIO Port 0.9	RxD1(UART1)	PMW6	EINT3	00
21:20	P0.10	GPIO Port 0.10	RTS1(UART1)	Capture 1.0 (TIMER1)	AD1.2	00
23:22	P0.11	GPIO Port 0.11	CTS1(UART1)	Capture 1.1 (TIMER1)	SCL1(I^2C1)	00
25:24	P0.12	GPIO Port 0.12	DSR1(UART1)	Match1.0 (TIMER1)	AD1.3	00
27:26	P0.13	GPIO Port 0.13	DTR1(UART1)	Match1.0 (TIMER1)	AD1.4	00
29:28	P0.14	GPIO Port 0.14	DCD1(UART1)	EINT1	SDA1(I^2C1)	00
31:30	P0.15	GPIO Port 0.15	RI1(UART1)	EINT2	AD1.5	00

ค่าบิตแต่ละบิตของรีจิสเตอร์ PINSEL1 แสดงได้ในตารางที่ 2.3 ตัวอย่างถ้าต้องการกำหนดค่าให้ขา P0.16-P0.31 มีการทำงานเป็น GPIO ทำได้โดยการเขียนค่า 0ทุกบิตให้กับ PINSEL1 ซึ่งเขียนเป็นคำสั่งภาษาซีได้ดังนี้

```
PINSEL1=0x00000000; //Set P0.16-P0.31 to GPIO Function
```

ตารางที่ 2.3 ค่าประจำบิตรีจิสเตอร์ PINSEL1

PINSEL0	ชื่อขา	การทำงาน เมื่อมีค่า 00	การทำงาน เมื่อมีค่า 01	การทำงาน เมื่อมีค่า10	การทำงาน เมื่อมีค่า11	ค่าหลัง รีเซ็ต
1:0	P0.16	GPIO Port 0.16	EINT0	Match0.2(TIMER0)	Capture0.2 (TIMER0)	00
3:2	P0.17	GPIO Port 0.17	Capture1.2 (TIMER1)	SCK1(SPI1)	Match1.2 (TIMER1)	00
5:4	P0.18	GPIO Port 0.18	Capture1.3 (TIMER1)	MISO1(SPI1)	Match1.3 (TIMER1)	00
7:6	P0.19	GPIO Port 0.19	Match1.2 (TIMER1)	MOSI1(SPI1)	Capture1.2 (TIMER1)	00
9:8	P0.20	GPIO Port 0.20	Match1.3 (TIMER1)	SSEL1(SPI1)	EINT3	00
11:10	P0.21	GPIO Port 0.21	PWM5	AD1.6	Capture1.3 (TIMER1)	00
13:12	P0.22	GPIO Port 0.22	AD1.7	Capture0.0 (TIMER0)	Match0.0 (TIMER0)	00
15:14	P0.23	GPIO Port 0.23	V_{BUS}	สงวนไว้	สงวนไว้	00
17:16	P0.24	สงวนไว้	สงวนไว้	สงวนไว้	สงวนไว้	00
19:18	P0.25	GPIO Port 0.25	AD0.4	AOUT	สงวนไว้	00
PINSEL0	ชื่อขา	การทำงาน เมื่อมีค่า 00	การทำงาน เมื่อมีค่า 01	การทำงาน เมื่อมีค่า10	การทำงาน เมื่อมีค่า11	ค่าหลัง รีเซ็ต
21:20	P0.26	สงวนไว้	สงวนไว้	สงวนไว้	สงวนไว้	00
23:22	P0.27	สงวนไว้	สงวนไว้	สงวนไว้	สงวนไว้	00
25:24	P0.28	GPIO Port 0.28	AD0.1	Capture0.2 (TIMER0)	Match0.2 (TIMER0)	00
27:26	P0.29	GPIO Port 0.29	AD0.2	Capture0.3 (TIMER0)	Match0.3 (TIMER0)	00

ตารางที่ 2.3 ค่าประจำบิตรีจิสเตอร์ PINSEL1 (ต่อ)

PINSEL0	ชื่อขา	การทำงาน เมื่อมีค่า 00	การทำงาน เมื่อมีค่า 01	การทำงาน เมื่อมีค่า10	การทำงาน เมื่อมีค่า11	ค่าหลัง รีเซ็ต
29:28	P0.30	GPIO Port 0.30	AD0.3	EINT3	Capture0.0 (TIMER0)	00
31:30	P0.31	GPIO Port 0.31	UP_LED	CONNECT		00

2.1.11 การกำหนดค่าควบคุมการทำงานของพอร์ตอเนกประสงค์ (GPIO)

หลังจากที่กำหนดค่าให้พอร์ตแต่ละตัวมีการทำงานเป็น GPIO แล้วการควบคุมการทำงานของ GPIO จะสั่งผ่านทางรีจิสเตอร์ 4 ตัวได้แก่ IOPIN , IOSET, IOCLR, IODIR โดยรีจิสเตอร์ แต่ละตัวจะมีแอดเดรสในไมโครคอนโทรลเลอร์ LPC2141/2/4/6/8 ได้เพิ่มความสามารถของพอร์ตอเนกประสงค์ให้ทำงานได้รวดเร็วขึ้นซึ่งจะเรียกได้ว่าเป็น Fast GPIO โดยได้กำหนดรีจิสเตอร์เพิ่มเติมขึ้นอีกหลายตัว แต่การทำงานที่รวดเร็วขึ้นนี้จะต้องเขียนโปรแกรมเป็นภาษาแอสเซมบลีเท่านั้น ถ้าเขียนเป็นภาษาซีจะสังเกตผลไม่ค่อยออกดังนั้นจึงได้เน้นเรื่องการใช้งาน GPIO เป็นโหมดปกติ ซึ่งจะมีข้อดีตรงที่สามารถนำโปรแกรมไปใช้งานกับไมโครคอนโทรลเลอร์ตระกูล LPC2000 รุ่นอื่นๆ ได้

ตารางที่ 2.4 ชื่อและแอดเดรสของรีจิสเตอร์ที่ใช้ควบคุม GPIO

ชื่อ ทั่วไป	ความหมาย	การ ติดต่อ	ชื่อและแอด เดรสของPort0	ชื่อและแอด เดรสของPort1
IOPIN	GPIO Port Pin value register ใช้อ่านสถานะปัจจุบันของพอร์ต	อ่าน	0xE002 8000 IOPIN0	0xE002 8010 IOPIN1
IOSET	GPIO Port Output set register ใช้กำหนดค่าให้ขาของพอร์ตมีค่าเป็น1ถ้าบิตใดเป็น 1 ขาของพอร์ตที่ตรงกันจะมีค่าเป็น 1	อ่าน/ เขียน	0xE002 8004 IOSET0	0xE002 8014 IOSET1

ตารางที่ 2.4 ชื่อและแอดเดรสของรีจิสเตอร์ที่ใช้ควบคุม GPIO (ต่อ)

ชื่อทั่วไป	ความหมาย	การติดต่อ	ชื่อและแอดเดรสของPort0	ชื่อและแอดเดรสของPort1
IODIR	GPIO Port Direction control register ใช้กำหนดว่าขาใดเป็นอินพุต ขาใดเป็นเอาต์พุต	อ่าน/เขียน	0xE002 8008 IODIR0	0xE002 8018 IODIR1
IOCLR	GPIO Port Output clear register ใช้กำหนดให้ขาของพอร์ตมีค่าเป็น 0	เขียน	0xE002 800C IOCLR0	0xE002 801C IOCLR1

การสั่งงานของ GPIO เริ่มต้นด้วยการกำหนดทิศทางของพอร์ตก่อนว่าจะให้เป็นอินพุตหรือเอาต์พุต โดยกำหนดค่าที่รีจิสเตอร์ IODIR โดยค่าบิตใดที่เป็น 0 คือให้ขาของบิตนั้นเป็นอินพุต ถ้าให้ค้าเป็น 1 ขาของบิตนั้นจะเป็นเอาต์พุต

*ในกรณีบอร์ดที่เราได้นำมาใช้งานนั้นมีการทำงานแบบ Active low

ตัวอย่างเช่น ต้องการให้ขา P0.22, P0.20, P0.19 และ P0.16 เป็นเอาต์พุต โดยขาของ P0 ที่เหลือเป็นอินพุต จะต้องกำหนดค่าให้รีจิสเตอร์ IODIR0 แต่ละบิตดังนี้

บิตที่	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ค่า	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1
	0				0				5				9			

บิตที่	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ค่า	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0				0				0				0			

ซึ่งสามารถเขียนเป็นส่วนของโปรแกรมภาษาซีได้ดังนี้

```
IODIR0 = 0x00590000;
```

หลังจากกำหนดให้พอร์ตเป็นเอาต์พุตแล้ว ถ้าต้องการสั่งให้พอร์ตที่เป็นเอาต์พุตนี้มีค่าเป็น 1 ต้องสั่งที่รีจิสเตอร์ IOSET ตัวอย่างเช่นต้องการสั่งให้ขา P0.16 และ P0.19 เป็น 1 จะต้องกำหนดค่าให้กับรีจิสเตอร์ IOSET0 ดังนี้

2.1.12 การทดลองต่อ GPIO เป็นอินพุต และการรับอินเตอร์รัปต์จากภายนอก

กลไกการอินเตอร์รัปต์ของไมโครคอนโทรลเลอร์ ARM7 ในไมโครคอนโทรลเลอร์ ARM7 มีโมดูล Vectored Interrupt Controller (VIC) เป็นตัวควบคุมกลไก ของการอินเตอร์รัปต์โดยสามารถรับอินเตอร์รัปต์จากอุปกรณ์ทั้งภายในและภายนอก และภายนอก ไมโครคอนโทรลเลอร์ ARM7 ได้ทั้งหมด 32 อินพุต

VIC จะนำอินพุตจากอินเตอร์รัปต์ทั้ง 32 แหล่งมาจัดแบ่งหมวดหมู่ได้เป็น 3 ประเภทคือ FIQ, vectored IRQ และ non-vectored IRQ ทำให้สามารถปรับลำดับความสำคัญของอินเตอร์รัปต์จาก อุปกรณ์ประกอบต่างๆ ได้ตามที่ต้องการ Fast Interrupt request (FIQ) จะลำดับความสำคัญสูงสุดโดยรีบสนองเร็วสุดVectored IRQs จะมีลำดับความสำคัญอยู่กึ่งกลาง โดยสามารถนำอินเตอร์รัปต์แค่ 16 หรือ 32 แหล่งมาจัดให้เป็น Vectored IRQs ได้ 16 ตัวโดย Slot0 จะมีความสำคัญสูงสุด Slot 15 มีความสำคัญต่ำสุด Non-vectored IRQ มีความสำคัญต่ำสุด

ตารางที่ 2.5 แหล่งกำหนดอินเตอร์รัปต์ทั้ง 32 ตัว

อุปกรณ์	แหล่งกำเนิดอินเตอร์รัปต์	VIC Channel #
WDT	Watchdog Interrupt(WDINT)	0
-	Reserved for software interrupts only	1
ARM Core	Embeded ICE, DbgCommRx	2
ARM Core	Embeded ICE, DbgCommRx	3
TIME0	Match 0-3 (MR0,MR1,MR2,MR3) Capture 0-3 (CR0,CR1,CR2,CR3)	4
TIME1	Match 0-3 (MR0,MR1,MR2,MR3) Capture 0-3 (CR0,CR1,CR2,CR3)	5

ตารางที่ 2.5 แหล่งกำหนดอินเตอร์รัปต์ทั้ง 32 ตัว (ต่อ)

อุปกรณ์	แหล่งกำเนิดอินเตอร์รัปต์	VIC Channel #
UART0	Rx Line Status(RLS) Transmit Holding Register Empty(THRE) Rx Data Available(RDA) Character Time-out Indicator(CTI)	6
UART1	Rx Line Status(RLS) Transmit Holding Register Empty(THRE) Rx Data Available(RDA) Character Time-out Indicator(CTI) Modem Status Interrupt(MSI)	7
PWM0	Match 0-6 (MR0,MR1,MR2,MR3,MR4,MR5,MR6)	8
I^2C0	SI (state change)	9
SPI0	SPI Interrupt Flag (SPIF) Mode Fault (Mode)	10
อุปกรณ์	แหล่งกำเนิดอินเตอร์รัปต์	VIC Channel #
SPI1(SSP)	SPI Interrupt Flag (SPIF) Mode Fault (Mode)	11
PLL	PLL Lock (PLOCK)	12
RTC	Counter Indrement (RTCCIF) Alarm (RTCALF)	13
System Control	External Interrupt 0 (EINT0)	14
System Control	External Interrupt 1 (EINT1)	15
System Control	External Interrupt 2 (EINT2)	16
System Control	External Interrupt 3 (EINT3)	17
ADC0	A/D Converter 0 end of conversion	18
I^2C1	SI (state change)	19

ตารางที่ 2.5 แหล่งกำหนดอินเตอร์รัปต์ทั้ง 32 ตัว (ต่อ)

อุปกรณ์	แหล่งกำเนิดอินเตอร์รัปต์	VIC Channel #
BOD	Brown our detect	20
ADC1	A/D Converter 1 end of conversion	21
USB	USB Interrupt, DMA Interrupt	22
	Reserved	23-31

รีจิสเตอร์ที่เกี่ยวข้องกับอินเตอร์รัปต์มีทั้งหมด 43 ตัว ดังแสดงในตารางที่ 2.6 โดยในหัวข้อนี้เป็นเลือกเฉพาะรีจิสเตอร์ที่เกี่ยวกับอินเตอร์รัปต์ในแบบ Vector IRQ ซึ่งก็คือ รีจิสเตอร์ VICVectAddr0-15 , VICVectCntl0-15 และ VICIntEnable

ตารางที่ 2.6 รีจิสเตอร์ที่เกี่ยวกับอินเตอร์รัปต์

ชื่อ	ความหมาย	การติดต่อ	ค่าหลังรีเซ็ต	แอดเดรส
VICIRQStatus	IRQ Status Request อ่านค่า สถานะว่าอนุญาตให้มี interrupt request จากตัวใดและถูกจัดเป็น IRQ	อ่าน	0	0xFFFF F000
VICRawIntr	Raw Interrupt Status Register ใช้อ่านสถานะของอินเตอร์รัปต์จากทั้ง 32แหล่งหรือจากซอฟแวร์โดยไม่สน ว่าถูกเปิดใช้งานหรือถูกจัดประเภท หรือไม่	อ่าน	0	0xFFFF F008
VICIntSelect	Interrupt Select Register ใช้ใน การระบุว่าอินเตอร์รัปต์ทั้ง 32 แหล่ง แต่ละตัวเป็น FIQ หรือ IRQ	อ่าน/เขียน	0	0xFFFF F00C

ตารางที่ 2.6 รีจิสเตอร์ที่เกี่ยวกับอินเตอร์รัปต์ (ต่อ)

ชื่อ	ความหมาย	การติดต่อ	ค่าหลังรีเซ็ต	แอดเดรส
VICIntEnable	Interrupt Enable Register ควบคุมว่าให้อินเตอร์รัปต์จาก 32 แหล่งตัวไหนทำงานและให้เป็น FIQ หรือ IRQ	อ่าน/เขียน	0	0xFFFF F010
VICIntEnClr	Interrupt Enable Clear Register ใช้ล้าง ค่าบิตหนึ่งบิตหรือมากกว่าหนึ่งบิตของ รีจิสเตอร์ Interrupt Enable Register	เขียน	0	0xFFFF F014
VICSoftlnt	Software Interrupt Register ค่าของรีจิสเตอร์นี้จะถูกนำไป ORกับอินเตอร์รัปต์จากอุปกรณ์ต่างๆทั้ง 32 แหล่ง	อ่าน/เขียน	0	0xFFFF F018
VICSoftlnt Clear	Software Interrupt Clear Register ใช้ล้างค่าบิตหนึ่งบิตหรือมากกว่าหนึ่งบิต ของ รีจิสเตอร์ Software Interrupt Register	เขียน	0	0xFFFF F01C
VICProtection	Protection enable register ใช้จำกัดการเข้าถึง VIC register	อ่าน/เขียน	0	0xFFFF F020
VICVectAddr0	Vector address 0 register รีจิสเตอร์ VICVectAddr0-15จะเก็บแต่ละตัวจะ ควบคุม IRQ Slot แต่ละตัวโดย Slot0 มีความสำคัญสูงสุดSlot15มีความ สำคัญต่ำสุด	อ่าน/เขียน	0	0xFFFF F100
VICVectAddr1	Vector address 1 register	อ่าน/เขียน	0	0xFFFF F104
VICVectAddr2	Vector address 2 register	อ่าน/เขียน	0	0xFFFF F108
VICVectAddr3	Vector address 3 register	อ่าน/เขียน	0	0xFFFF F10C

ตารางที่ 2.6 รีจิสเตอร์ที่เกี่ยวกับอินเตอร์รัปต์ (ต่อ)

ชื่อ	ความหมาย	การติดต่อ	ค่าหลังรีเซ็ต	แอดเดรส
VICVectAddr4	Vector address 4 register	อ่าน/เขียน	0	0xFFFF F110
VICVectAddr5	Vector address 5 register	อ่าน/เขียน	0	0xFFFF F114
VICVectAddr6	Vector address 6 register	อ่าน/เขียน	0	0xFFFF F118
VICVectAddr7	Vector address 7 register	อ่าน/เขียน	0	0xFFFF F11C
VICVectAddr8	Vector address 8 register	อ่าน/เขียน	0	0xFFFF F120
VICVectAddr9	Vector address 9 register	อ่าน/เขียน	0	0xFFFF F124
VICVectAddr10	Vector address 10 register	อ่าน/เขียน	0	0xFFFF F128
VICVectAddr11	Vector address 11 register	อ่าน/เขียน	0	0xFFFF F12C
VICVectAddr12	Vector address 12 register	อ่าน/เขียน	0	0xFFFF F130
VICVectAddr13	Vector address 13 register	อ่าน/เขียน	0	0xFFFF F134
VICVectAddr14	Vector address 14 register	อ่าน/เขียน	0	0xFFFF F138
VICVectAddr15	Vector address 15 register	อ่าน/เขียน	0	0xFFFF F13C
VICVecCntl0	Vector control 0 รีจิสเตอร์ Vector Control Register0-15 แต่ละตัวจะควบคุม IRQ Slot แต่ละตัว โดย Slot 0 มีความสำคัญ สูงสุดและ Slot 15 มีความสำคัญต่ำสุด	อ่าน/เขียน	0	0xFFFF F200
VICVecCntl1	Vector control 1 register	อ่าน/เขียน	0	0xFFFF F204
VICVecCntl2	Vector control 2 register	อ่าน/เขียน	0	0xFFFF F208
VICVecCntl3	Vector control 3 register	อ่าน/เขียน	0	0xFFFF F20C
VICVecCntl4	Vector control 4 register	อ่าน/เขียน	0	0xFFFF F210
VICVecCntl5	Vector control 5 register	อ่าน/เขียน	0	0xFFFF F214
VICVecCntl6	Vector control 6 register	อ่าน/เขียน	0	0xFFFF F218
VICVecCntl7	Vector control 7 register	อ่าน/เขียน	0	0xFFFF F21C
VICVecCntl8	Vector control 8 register	อ่าน/เขียน	0	0xFFFF F220
VICVecCntl9	Vector control 9 register	อ่าน/เขียน	0	0xFFFF F224

ตารางที่ 2.6 รีจิสเตอร์ที่เกี่ยวกับอินเตอร์รัปต์ (ต่อ)

VICVecCntl10	Vector control 10 register	อ่าน/เขียน	0	0xFFFF F228
VICVecCntl11	Vector control 11 register	อ่าน/เขียน	0	0xFFFF F22C
VICVecCntl12	Vector control 12 register	อ่าน/เขียน	0	0xFFFF F230
VICVecCntl13	Vector control 13 register	อ่าน/เขียน	0	0xFFFF F234
VICVecCntl14	Vector control 14 register	อ่าน/เขียน	0	0xFFFF F238
VICVecCntl15	Vector control 15 register	อ่าน/เขียน	0	0xFFFF F23C

รีจิสเตอร์ VICVectAddr0-15 เป็นรีจิสเตอร์ที่เก็บค่าแอดเดรสของโปรแกรมที่ตอบสนองต่อ อินเตอร์รัปต์โดย VICVectAddr0 จะเป็นของอินเตอร์รัปต์ Slot 0 ที่มีความสำคัญสูงสุด ตามลำดับไป จนถึง VICVectAddr15 จะเป็น Slot15 ที่มีความสำคัญต่ำสุด

VICVecCntl 0-15 ใช้ในการควบคุมว่าที่ Slot หมายเลขที่ตรงกับ VICVecCntl นี้จะรับการอินเตอร์ รัปต์จากแหล่งใด จากทั้งหมด 32 ที่ตามตารางที่ 2.6 โดยการกำหนดค่าในบิต 4:0 โดยบิตที่ 5 ถ้าเป็น 1 หมายถึงอนุญาตให้อินเตอร์รัปต์จากอุปกรณ์ที่กำหนดใน Slot นี้

VIClntEnable ใช้เปิดอินเตอร์รัปต์จากแหล่งต่างๆทั้ง 32 แหล่งตามตารางที่ 2.6 โดยบิตใดมีค่าเป็น 1 หมายถึงอนุญาตให้อินเตอร์รัปต์ได้ ถ้าบิตใดเป็น 0 ไม่อนุญาตให้อุปกรณ์นั้นๆอินเตอร์รัปต์

ตัวอย่าง ต้องการให้ไมโครคอนโทรลเลอร์รับอินเตอร์รัปต์จากภายนอกจากขา EINT1 โดยกำหนด ให้เป็นอินเตอร์รัปต์แบบ IRQ มีความสำคัญสูงสุด

จากความต้องการให้อินเตอร์รัปต์ที่มีความสำคัญสูงสุดจึงต้องกำหนดให้เป็น Slot 0 โดยกำหนดค่าของรีจิสเตอร์ VICVectAddr0 ให้ชี้ไปยังแอดเดรสของโปรแกรมย่อยตอบสนองการอินเตอร์รัปต์ โดยเขียนเป็นคำสั่งภาษาซีได้ดังนี้

VICVectAddr0 = (unsigned)isr_eint1;

เมื่อ isr_eint1 เป็นชื่อของโปรแกรมย่อยตอบสนองการอินเตอร์รัปต์

จากตารางที่ 2.6 พบว่า EINT1 จะมีหมายเลข VIC channel #15 จะต้องกำหนดค่าบิต 4:0 ของรีจิสเตอร์ VICVecCntl0 ให้มีค่าเท่ากับ 15 และกำหนดให้บิตที่ 5 ของรีจิสเตอร์ VICVectCntl0 ให้เป็น 1 เพื่ออนุญาตให้ทำการอินเตอร์รัปต์ได้ โดยเขียนคำสั่งภาษาซีได้ดังนี้

VICVecCntl0 = 0x0000002F;

หรือเขียนอีกวิธีหนึ่งได้ดังนี้

VICVectCntl0 = 0x20|15;

ในการกำหนดค่าให้กับรีจิสเตอร์ VIClntEnable จะต้องสั่งให้บิตที่ 15 ของรีจิสเตอร์นี้มีค่าเป็น 1 ซึ่ง จะไม่เขียนค่าให้กับรีจิสเตอร์โดยตรง แต่จะใช้การ OR บิตเพื่อให้บิตที่ 15 เปลี่ยนค่าตัวเดียวโดยบิต อื่นๆไม่เปลี่ยนแปลงซึ่งเขียนเป็นคำสั่งภาษาซีได้ดังนี้

VIClntEnable | = 0x00008000;

หรือเขียนอีกวิธีหนึ่งได้ดังนี้

VIClntEnable | = 1<<15

ตารางที่ 2.7 ค่าประจำบิตของรีจิสเตอร์ EXTINT

ค่าบิตของ EXTINT	หน้าที่	ความหมาย	ค่าหลังรีเซ็ต
0	EINT0	เมื่อขา EINT0 ทำงานที่ระดับลอจิกหรือมีขอบของสัญญาณตามที่ กำหนดเกิดขึ้นบิตนี้จะมีค่าเป็น 1 เมื่อเขียนค่า 1ให้มัน	0
		เป็นการล้างค่ายกเว้นกรณีที่ทำงานตามระดับลอจิกต้องรอให้ค่าลอจิกหยุดการทำงานก่อน ขาที่เป็น EINT0 คือ P0.1 และ P0.16	
1	EINT1	เมื่อขา EINT1 ทำงานที่ระดับลอจิกหรือมีขอบของสัญญาณตามที่ กำหนดเกิดขึ้นบิตนี้จะมีค่าเป็น 1 เมื่อเขียนค่า 1ให้มัน เป็นการล้างค่ายกเว้นกรณีที่ทำงานตามระดับลอจิกต้องรอให้ค่าลอจิกหยุดการทำงานก่อน ขาที่เป็น EINT1 คือ P0.3 และ P0.14	0

ตารางที่ 2.7 ค่าประจำบิตของรีจิสเตอร์ EXTINT (ต่อ)

ค่าบิตของ EXTINT	หน้าที่	ความหมาย	ค่าหลังรีเซ็ต
2	EINT2	เมื่อขา EINT2 ทำงานที่ระดับลอจิกหรือมีขอบของสัญญาณตามที่ กำหนดเกิดขึ้นบิตนี้จะมีค่าเป็น 1 เมื่อเขียนค่า 1ให้มันเป็นการล้างค่ายกเว้นกรณีที่ทำงานตามระดับลอจิกต้องรอให้ค่าลอจิกหยุดการทำงานก่อน ขาที่เป็น EINT2 คือ P0.7 และ P0.15	0
3	EINT3	เมื่อขา EINT3 ทำงานที่ระดับลอจิกหรือมีขอบของสัญญาณตามที่ กำหนดเกิดขึ้นบิตนี้จะมีค่าเป็น 1 เมื่อเขียนค่า 1ให้มันเป็นการล้างค่ายกเว้นกรณีที่ทำงานตามระดับลอจิกต้องรอให้ค่าลอจิกหยุดการทำงานก่อน ขาที่เป็น EINT3 คือ P0.9 , P0.20 และ P0.30	0
7:4	สงวนไว้	ห้ามเขียนค่า 1 ให้กับบิตเหล่านี้ค่าที่อ่านได้ไม่มีความหมาย	ไม่กำหนด

ในการกำหนดว่าสัญญาณที่อินเตอร์รัปต์นี้ทำงานที่ระดับสัญญาณหรือที่ขอบของสัญญาณกำหนดได้ที่รีจิสเตอร์ EXTMODE ซึ่งมีค่าดังตารางที่ 2.8

ตารางที่ 2.8 ค่าประจำบิตของรีจิสเตอร์ EXTMODE

ค่าบิตของ EXTINT	หน้าที่	ความหมาย	ค่าหลังรีเซ็ต
0	EXTMODE0	โหมดการทำงานของ EINT0 เมื่อมีค่าเป็น 0 ทำงานที่ระดับสัญญาณ เมื่อมีค่าเป็น 1 ทำงานที่ขอบของสัญญาณ	0
1	EXTMODE1	โหมดการทำงานของ EINT1 เมื่อมีค่าเป็น 0 ทำงานที่ระดับสัญญาณ เมื่อมีค่าเป็น 1 ทำงานที่ขอบของสัญญาณ	0
2	EXTMODE 2	โหมดการทำงานของ EINT2 เมื่อมีค่าเป็น 0 ทำงานที่ระดับสัญญาณ เมื่อมีค่าเป็น 1 ทำงานที่ขอบของสัญญาณ	0
3	EXTMODE 3	โหมดการทำงานของ EINT3 เมื่อมีค่าเป็น 0 ทำงานที่ระดับสัญญาณ เมื่อมีค่าเป็น 1 ทำงานที่ขอบของสัญญาณ	0
7:4	สงวนไว้	ห้ามเขียนค่า 1 ให้กับบิตเหล่านี้ค่าที่อ่านได้ ไม่มีความหมาย	ไม่กำหนด

2.2 การตรวจวัดยานพาหนะ

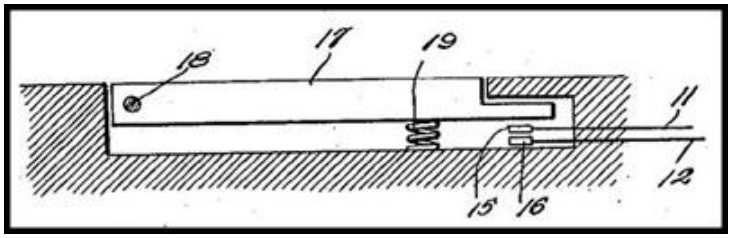
อุปกรณ์ตรวจวัดการจราจร (Traffic flow sensors) คืออุปกรณ์ที่ใช้ในการตรวจวัดการมีอยู่ (Presence) หรือการผ่าน (Passage) ของยวดยาน ณ บริเวณจุดสำรวจให้ข้อมูลที่ใช้ในการจัดการจราจรและข้อมูลที่ใช้ในการวางแผน

2.2.1 พัฒนาการของเทคโนโลยีตรวจวัดการจราจร

ค.ศ. 1928 Horn-activated traffic signal (developed by Charles Adler,Jr.) เป็นเครื่องมือวัดปริมาณรถโดยอาศัยการติดตั้ง Micro Phone

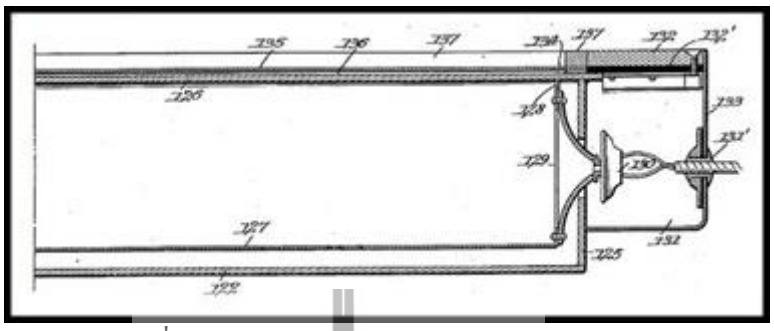
รูปที่ 2.7 การตรวจวัดยานพาหนะโดยใช้ Micro Phone

ค.ศ. 1928 In-roadway pressure sensitive sensor (Developed by Henry A. Haugh) เป็นเครื่องมือวัดปริมาณรถโดยอาศัยกระแสไฟฟ้าที่ได้จากแรงกดทับของแผ่นเหล็กที่ติดตั้งไว้ใต้ผิวถนน



รูปที่ 2.8 การตรวจวัดยานพาหนะโดยการกดทับของแผ่นเหล็ก

ค.ศ. 1930 In-roadway pressure sensitive sensor (developed by Charles Adler,Jr.) เป็นเครื่องมือวัดปริมาณรถโดยอาศัยเสียงซึ่งได้มาจากการติดตั้งเครื่องวัดเสียงที่ติดตั้งไว้ใต้ผิวถนน



รูปที่ 2.9 การตรวจวัดยานพาหนะโดยอาศัยเสียง

ซึ่ง In-roadway pressure sensitive sensor ได้รับความนิยมมากแต่ก็ยังมีปัญหาซึ่งสรุปได้ดังนี้

1. อุปกรณ์มีราคาแพง
2. ต้องติดตั้งอุปกรณ์ใหม่หากมีการปรับปรุงผิวถนน
3. ความเสียหายเนื่องจากรถกวาดหิมะ

รูปที่ 2.10 ปัญหาของการตรวจวัดยานพาหนะ

2.2.2 เทคโนโลยีตรวจวัดการจราจรในปัจจุบัน

1. อุปกรณ์ตรวจวัดการจราจรที่ติดตั้งอยู่บนหรือใต้ผิวทาง (In-roadway sensors)

1. Pneumatic road tube เริ่มใช้ตั้งแต่ปี 1920s เป็นเครื่องตรวจวัดการจราจรที่ติดตั้งโดยอาศัยท่อยางเพื่อตรวจวัดความถี่

หลักการทำงาน น้ำหนักรถกดทับท่อลมเป็นเครื่องตรวจวัดการจราจรที่ติดตั้งบนผิว ทางวางขวางช่องทางเดินรถเคลื่อนที่อาศัยน้ำหนักรถกดทับท่อลมซึ่งทำด้วยยางแล้วเกิดแรงดันกระจายไปในแนวท่อยางโดยปลายด้านหนึ่งของท่อยางปิดตันและอีกด้านต่อเข้ากับอุปกรณ์ระบบไฟฟ้าซึ่งทำงา นด้วยแรงดันที่เปลี่ยนแปลงในเรือท่อทำให้ทราบจำนวนการกดทับและวิ่งผ่านท่อลมซึ่งสามารถจัดเก็บข้อมูลได้ดังนี้

- ปริมาณจราจร
- ความเร็วรถ
- แยกประเภทรถโดยใช้จำนวนเพลาและระยะห่างระหว่างเพลา,เวลาห่างระหว่างรถ (Gap)

การติดตั้ง Pneumatic road tube

- เลือกตำแหน่งที่ต้องการติดตั้ง
- ติดตั้ง End plug เพื่อกันน้ำหรือฝุ่นเข้า
- ยึดปลายด้านหนึ่งของท่อลมให้แน่นกับพื้นถนน
- ติดท่อลมกับผิวถนนด้วยเทปกาวแอสฟัลท์
- วัดระยะเพื่อติดตั้งท่อลมอีกเส้น (ถ้ามี)
- ติดปลายท่ออีกด้านเข้ากับเครื่องนับและถ้าทำการติดตั้ง Two-pneumatic road tubes connected together จะช่วยลดข้อผิดพลาดจากการวัดระยะห่างระหว่างท่อลมขณะติดตั้ง

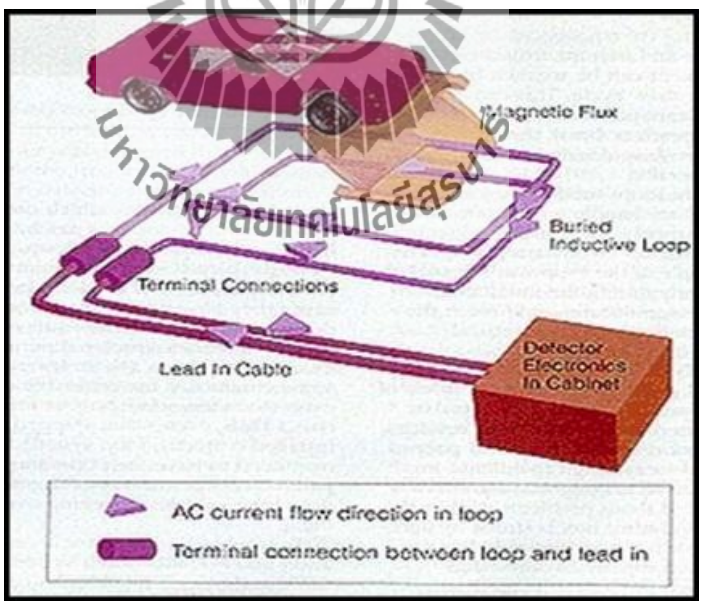
ข้อดี-ข้อเสีย

- **ข้อดี**
  - ใช้งานง่าย
  - ราคาถูก
- **ข้อเสีย**
  - ท่อลมชำรุดได้ง่ายถ้ามีปริมาณการจราจรสูง
  - การติดตั้งเหมาะสมกับการติดตั้งแบบชั่วคราวและสถานที่ไม่ซับซ้อน
  - การติดตั้งต้องปิดการจราจรถึงจะทำการติดตั้งได้

รูปที่ 2.11 การตรวจวัดยานพาหนะโดยใช้ท่อลม

2. Inductive Loop เริ่มใช้ตั้งแต่ปี 1960s เป็นเครื่องมือสำหรับจัดก รจราจรที่ตรวจวัดการเปลี่ยนแปลงของสนามแม่เหล็กไฟฟ้า

หลักการทำงาน เปลี่ยนแปลงค่าความต้านทานการเหนี่ยวนำของกระแสไฟฟ้าในขดลวดเหนี่ยวนำ



รูปที่ 2.12 การตรวจวัดยานพาหนะโดยใช้ Inductive Loop

ซึ่งประเภทของ Inductive ลูปเหนี่ยวนำ มีดังนี้

- Saw cut ลูป เป็นการกัดพื้นผิวของถนนแล้วทำการฝังสายลูป ซึ่งมีวิธีการติดตั้ง Saw-cut ลูป ดังนี้
  1. วาง layout ของ ลูป บนผิวถนน
  2. กัดผิวถนน
  3. ติดตั้งสาย ลูป
  4. ปิด ลูป ด้วย Sealant
- Trenched-in ลูป เป็นการวางสายลูป ก่อนการทำถนน
- Pre-formed ลูป เป็นการทำสายลูป สำเร็จรูปแล้วนำมาวางบนพื้นถนนแล้วนำเทปกาวติดเพื่อยึดสาย ลูป ไว้บนถนนซึ่งมีข้อเสียคือจะไม่มั่นคงแต่จะมีข้อดีตรงที่สามารถเคลื่อนย้ายง่ายเมื่อต้องการนำไปติดตั้ง

รูปที่ 2.13 Inductive Loop บนถนน

<u>หมายเหตุ</u> ตัวอย่าง Layout ของ ลูปเหนี่ยวนำs ต่างๆเช่นรูปวงกลม , รูปสี่เหลี่ยม, รูปหกเหลี่ยมเป็นต้น

เครื่องตรวจวัดสามารถจัดเก็บข้อมูลได้ดังนี้

- ปริมาณจราจร
- ความเร็วรถ
- การมีรถบริเวณนั้น (Presence)

- ระยะเวลาครอบครอง (Occupancy)
- ระยะห่างของรถแต่ละคัน (Gap)
- Headway
- แยกประเภทรถโดยใช้ vehicle signature

ข้อดี-ข้อเสีย

- **ข้อดี**
  - เป็นเทคโนโลยีที่มีการพัฒนามานานมีคนใช้เยอะ
  - ให้ข้อมูลการนับรถที่ถูกต้องที่สุด
  - ไม่ไวต่อสภาพอากาศที่เปลี่ยนแปลงใช้งานง่าย
- **ข้อเสีย**
  - การติดตั้งจำเป็นต้องมีการกัดผิวทาง
  - การติดตั้งที่ไม่ดีทำให้ผิวทางมีอายุการใช้งานสั้นลง
  - ต้องมีการปิดการจราจรขณะติดตั้งและซ่อมแซม
  - ในแต่ละจุดต้องใช้ ลูป หลายตัวในการตรวจวัดการจราจร
  - น้ำหนักกดทับและอุณหภูมิส่งผลต่อ ลูป ที่ฝังไว้

3. Magnetic sensors

หลักการทำงาน ใช้การเปลี่ยนแปลงของสนามแม่เหล็กโลก ซึ่งประเภทของ Magnetic sensors มีดังนี้

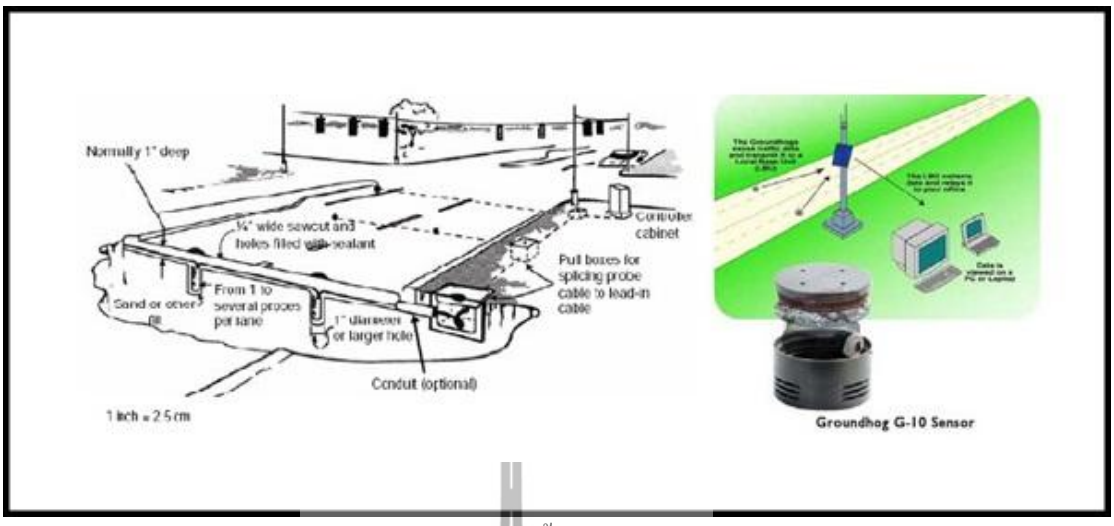
- Two-axis fluxgate magnetometer
- Induction or search coil magnetometer

เครื่องตรวจวัดสามารถจัดเก็บข้อมูลได้ดังนี้

- ปริมาณจราจร
- ความเร็วรถ
- การมีรถบริเวณนั้น (Presence)
- ระยะเวลาครอบครอง (Occupancy)
- แยกประเภทรถโดยใช้ vehicle magnetic signature

การติดตั้ง Magnetic sensors

เจาะรูปบนผิวถนนความลึกประมาณ 5 cm แล้วนำ Portable magnetic sensor วางไว้แล้วนำแผ่นยางปิดบริเวณดังกล่าวโดยใช้ตะปูตอกเพื่อยึดแผ่นยางไม่ได้แผ่นยางเปิดวิธีนี้ง่ายต่อการติดตั้ง



รูปที่ 2.14 การติดตั้ง Magnetic sensors

ข้อดี-ข้อเสีย

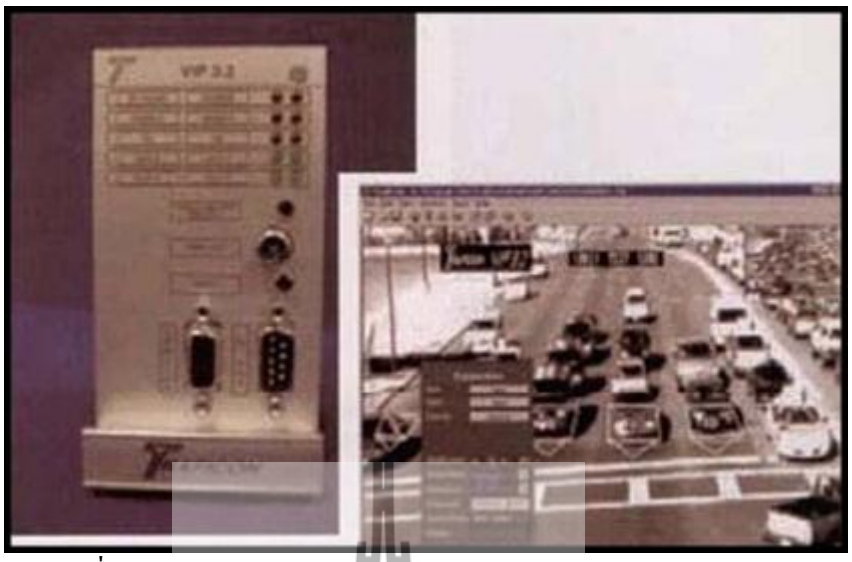
- **ข้อดี**
  - สามารถใช้ในบริเวณที่ไม่สามารถติดตั้งลูปได้
  - ไม่ไวต่อสภาพอากาศ
  - ผลจากแรงกดทับจากปริมาณจราจรมีน้อยกว่า ลูป
- **ข้อเสีย**
  - การติดตั้งจำเป็นต้องมีการกัดผิวทางหรือเจาะเป็นช่องใต้ผิวทาง
  - ไม่สามารถตรวจวัดรถที่จอดนิ่งอยู่กับที่ได้หากไม่มีการวาง Layout ของอุปกรณ์ตรวจวัดแบบพิเศษและใช้ Software เสริม

2. อุปกรณ์ตรวจวัดการจราจรทีติดตั้งอยู่เหนือผิวทาง (Over-roadway sensors)

1. Video Image Processors เริ่มใช้ตั้งแต่ปี 1970s

หลักการทำงาน ใช้การวิเคราะห์ภาพจาก Video frames เครื่องตรวจวัดสามารถให้ข้อมูลได้ดังนี้

- ปริมาณจราจร
- ความเร็วรถ
- การมีรถบริเวณนั้น (Presence)
- ระยะเวลาครอบครอง (Occupancy)
- ระยะห่างของรถแต่ละคัน



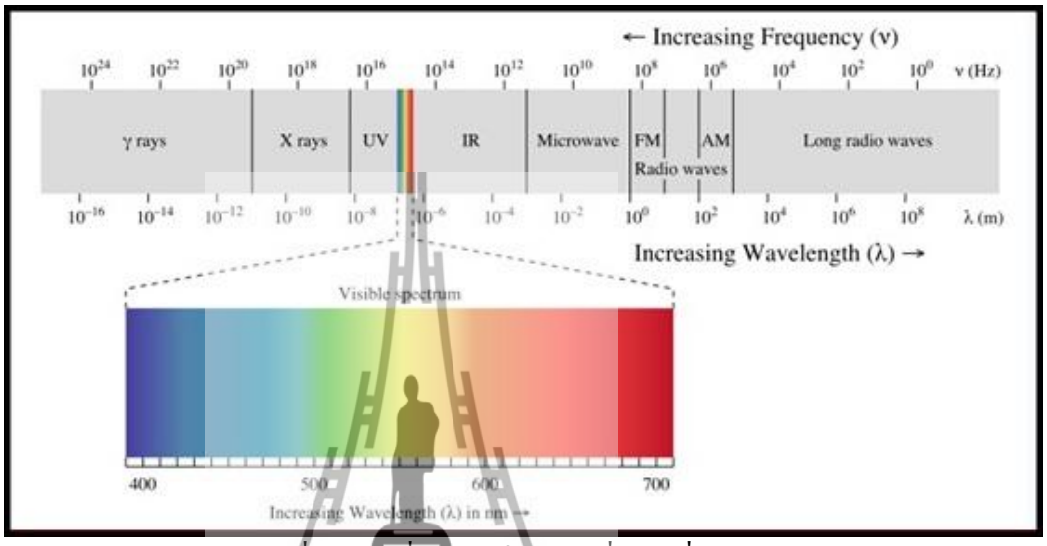
รูปที่ 2.15 การตรวจวัดยานพาหนะโดยใช้ Video Image Processors

ข้อดี-ข้อเสีย

- **ข้อดี**
  - สามารถตรวจวัดได้หลายช่องจราจรพร้อมๆกัน
  - ง่ายในการเพิ่มพื้นที่ตรวจวัด
  - ให้ข้อมูลจราจรหลายอย่าง
  - สามารถตรวจวัดได้ในบริเวณกว้างเมื่อมีการเชื่อมข้อมูลระหว่างกล้อง
- **ข้อเสีย**
  - ต้องมีการปิดช่องจราจรหากติดตั้งกึ่งกลางถนน
  - ต้องมีการทำความสะอาดเลนส์ตามระยะเวลา
  - สภาพอากาศและความสั่นสะเทือนมีผลต่อประสิทธิภาพของกล้อง
  - ต้องมีไฟส่องสว่างที่เพียงพอในเวลากลางคืน
  - ต้องติดตั้งกล้องจากมุมสูง
  - จะคุ้มค่าก็ต่อเมื่อมีการตรวจวัดหลายพื้นที่พร้อมกันในกล้องเดียวหรือต้องการข้อมูลบางอย่างเป็นพิเศษ

2. Microwave radar sensors

คลื่นไมโครเวฟ (Microwaves) เป็นคลื่นแม่เหล็กไฟฟ้าที่มีความถี่อยู่ในช่วง 300 MHz คลื่นไมโครเวฟได้ถูกนำมาประยุกต์ใช้ในอุปกรณ์ต่างๆเช่นเตาไมโครเว ฟโทรศัพท์มือถือรวมถึงเครื่องมือในการตรวจวัดการจราจร



รูปที่ 2.16 คลื่นแม่เหล็กไฟฟ้าที่ความถี่ต่างๆ

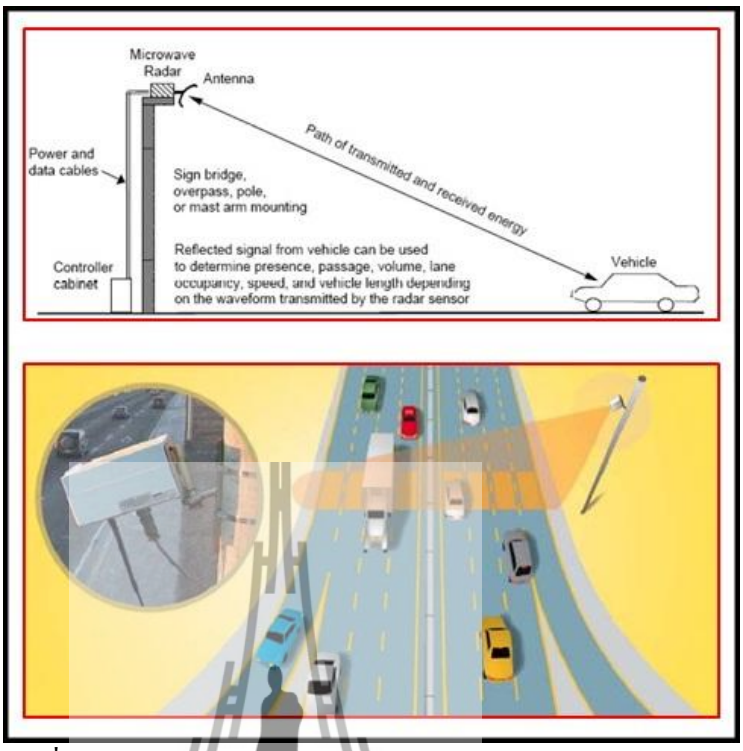
เครื่องมือตรวจวัดการจราจรแบบ Microwave Radar Sensors มี 2 ประเภทคือ

1. แบบ Doppler Microwave

2. แบบ Frequency – modulated continuous wave

หลักการทำงาน เครื่องตรวจวัดจะส่งสัญญาณและสะท้อนกลับของคลื่น Microwave ออกมาเพื่อตรวจวัดรถยนต์ที่วิ่งผ่านแนวสัญญาณเครื่องตรวจวัดประมวลผลออกมาเป็นข้อมูลดังนี้

- ปริมาณจราจร
- ความเร็วรถ
- การมีรถบริเวณนั้น (Presence)
- ระยะเวลาครอบครอง (Occupancy)
- ระยะห่างของรถแต่ละคัน (Gap)
- Headway
- ประเภทรถยนต์จำแนกตามความยาวรถแต่ละคัน



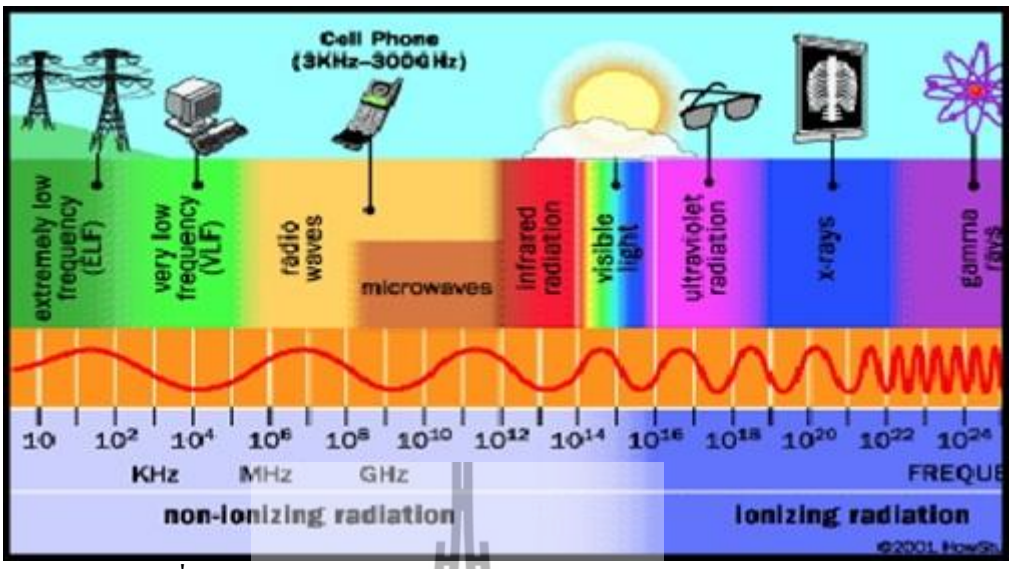
รูปที่ 2.17 การตรวจวัดยานพาหนะโดย Microwave radar sensors

ข้อดี-ข้อเสีย

- **ข้อดี**
  - ไม่ไวต่อสภาพอากาศสำหรับการใช้งานตรวจวัดในช่วงสั้น
  - สามารถวัดความเร็วได้โดยตรง
  - สามารถใช้ตรวจวัดการจราจรได้หลายช่องทางในเวลาเดียวกัน
- **ข้อเสีย**
  - Continuous Wave Doppler Sensors ไม่สามารถตรวจวัดรถที่จอดนิ่งอยู่กับที่

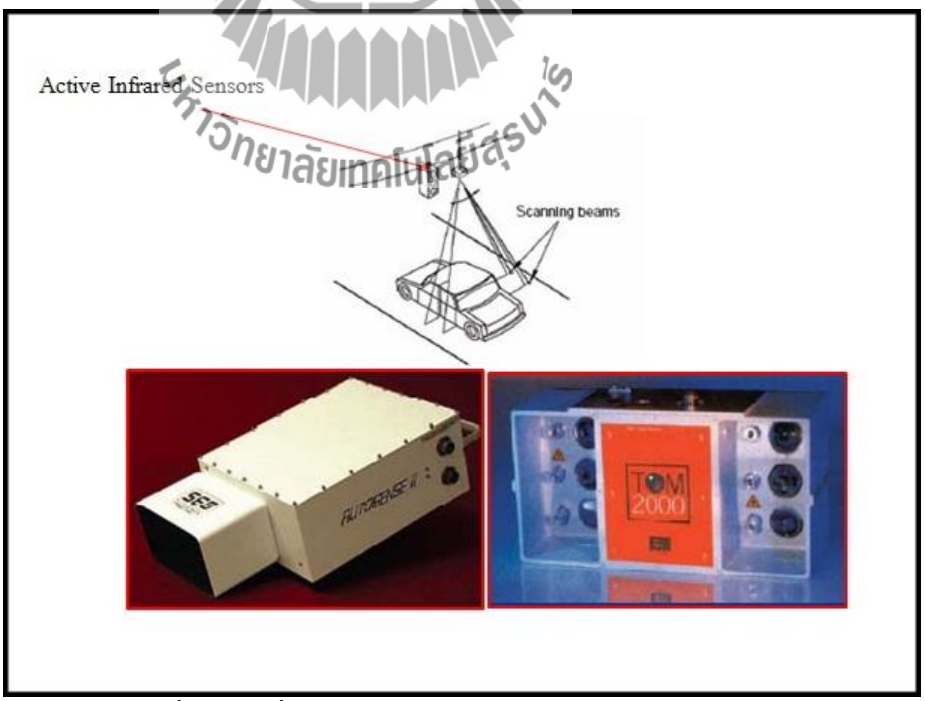
3. Active infrared sensors (laser sensors)

คลื่นอินฟราเรด (Infrared) เป็นคลื่นแม่เหล็กไฟฟ้าที่มีความถี่อยู่ในช่วง 1011 Hz–1014 Hz ซึ่งมีความถี่สูงกว่าคลื่นไมโครเวฟเครื่องตรวจวัดการจราจรแบบ Active Infrared Sensors



รูปที่ 2.18 การตรวจวัดยานพาหนะโดยใช้ Active Infrared Sensors

เครื่องตรวจวัดการจราจรแบบ Active Infrared Sensors มีลักษณะการทำงานคล้ายๆกับการทำงานของเครื่องตรวจวัดการจราจรแบบ Microwave Radar Sensors คือมีการปล่อยคลื่นอินฟราเรดออกมาเพื่อตรวจวัดรถยนต์ที่วิ่งผ่านแนวสัญญาณดังรูป



รูปที่ 2.19 เครื่องตัววัดจราจร แบบ Active Infrared Sensors

หลักการทำงาน

เป็นอุปกรณ์ลำแสง 2 ชุดกวาดผ่านช่องจราจรที่จะทำการตรวจวัดแล้วอุปกรณ์ตรวจวัดพลังงานแสงที่เกิดการกระจายเมื่อยานพาหนะเคลื่อนที่ผ่านแล้วแปรสัญญาณส่งผ่านอุปกรณ์ประมวลผลซึ่งเครื่องตรวจวัดสามารถจัดเก็บข้อมูลได้ดังนี้

- ปริมาณการจราจร
- ความเร็วรถ
- การมีรถบริเวณนั้น (Presence)
- ประเภทรถยนต์จำแนกตามความยาวรถแต่ละคัน

ข้อดี-ข้อเสีย

• ข้อดี

- สัญญาณที่ส่งออกมามีหลายลำทำให้สามารถตรวจวัดตำแหน่งรถความเร็วและประเภทของรถได้อย่างแม่นยำสูง
- สามารถใช้ตรวจวัดการจราจรได้หลายช่องทางในเวลาเดียวกัน (Gap)

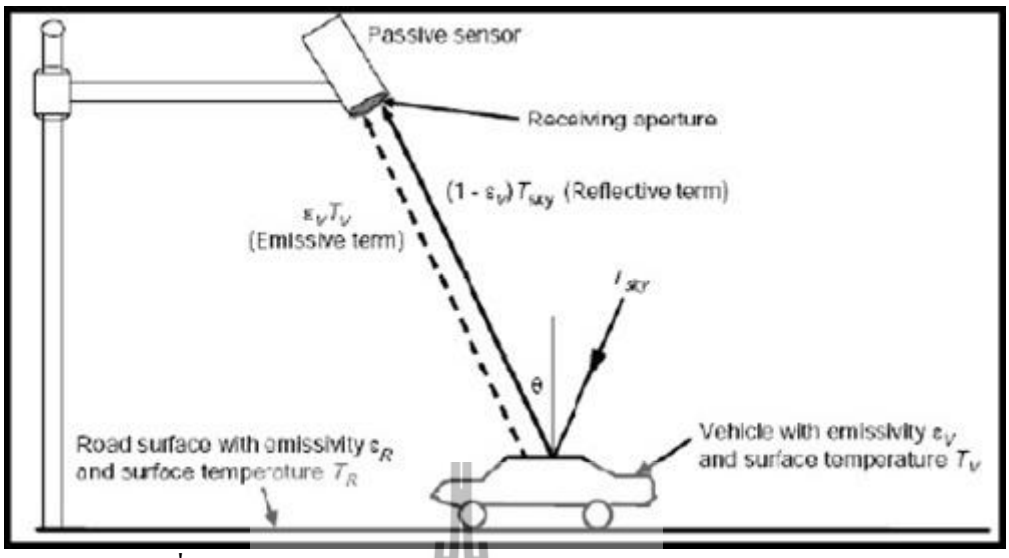
• ข้อเสีย

- ประสิทธิภาพในการทำงานลดหากมีหมอกหรือมีหิมะตก
- ต้องมีระยะเวลาในการทำความสะอาดเลนส์เป็นประจำ
- ต้องมีการปิดช่องการจราจรเพื่อทำการติดตั้ง

4. Passive infrared sensors

หลักการทำงาน

เป็นเครื่องมือวัดการจราจรตรวจวัดพลังงานที่สะท้อนจากวัตถุเข้าสู่ตัวกล้องที่เกิดจากวัตถุที่เคลื่อนที่ผ่าน หรือวัตถุที่อยู่ในบริเวณพื้นที่ตรวจวัด เช่นผิวจราจรรถยนต์และพลังงานที่เกิดจากบรรยากาศบริเวณนั้นสะท้อนผ่านวัตถุที่เคลื่อนที่เครื่องมือจะตรวจวัดการเปลี่ยนแปลงขอ ง Infrared Spectrum แปรข้อมูลส่งผ่านเครื่องประมวลผล



รูปที่ 2.20 การตรวจวัดยานพาหนะโดยใช้ Passive infrared sensors

ซึ่งประเภทของ Passive infrared sensors มีดังนี้

- Non-imaging
- Imaging

เครื่องตรวจวัดสามารถให้ข้อมูลได้ดังนี้

- ปริมาณจราจร
- ความเร็วรถ
- การมีรถบริเวณนั้น (Presence)
- แยกประเภทรถโดยใช้ความยาว
- ความยาวของแถวรถที่ติดอยู่บนท้องถนน

ข้อดี-ข้อเสีย

- **ข้อดี**
  - ระบบตรวจวัดแบบหลายพื้นที่ใช้วัดความเร็ว
- **ข้อเสีย**
  - ประสิทธิภาพลดลงช่วงฝนตกหิมะตกหมอกลง
  - บางรุ่นไม่สามารถตรวจวัดการมีรถในบริเวณที่กำหนดได้

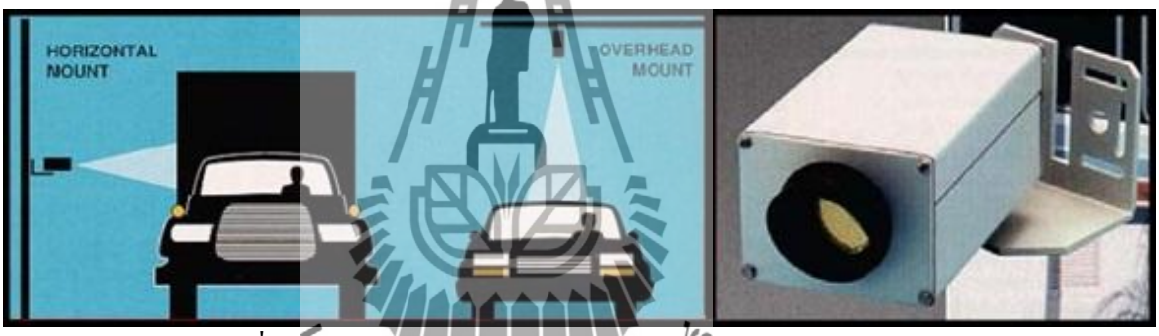
5. Ultrasonic sensors

เริ่มใช้ตั้งแต่ปี 1950s เป็นเครื่องมือตรวจวัดการจราจรที่ใช้การส่งสัญญาณเสียงในระดับความถี่ช่วง 25-50 kHz ใช้ตรวจวัดการสะท้อนกลับของคลื่นเสียง

หลักการทำงาน

เป็นการส่งสัญญาณเสียงในระดับความถี่ดังที่กล่าวมาโดยวัดระยะเวลาในการส่งสัญญาณจนถึงระยะเวลานี้เสียงสะท้อนกลับมาซึ่งจะสามารถแสดงผลเป็นระยะห่างระหว่างรถกับสัญญาณเครื่องตรวจวัดสามารถให้ข้อมูลได้ดังนี้

- ปริมาณจราจร
- ความเร็วรถ
- การมีรถบริเวณนั้น (Presence)
- แยกประเภทรถโดยใช้ความยาว



รูปที่ 2.21 การตรวจวัดยานพาหนะโดยใช้ Ultrasonic sensors

ข้อดี-ข้อเสีย

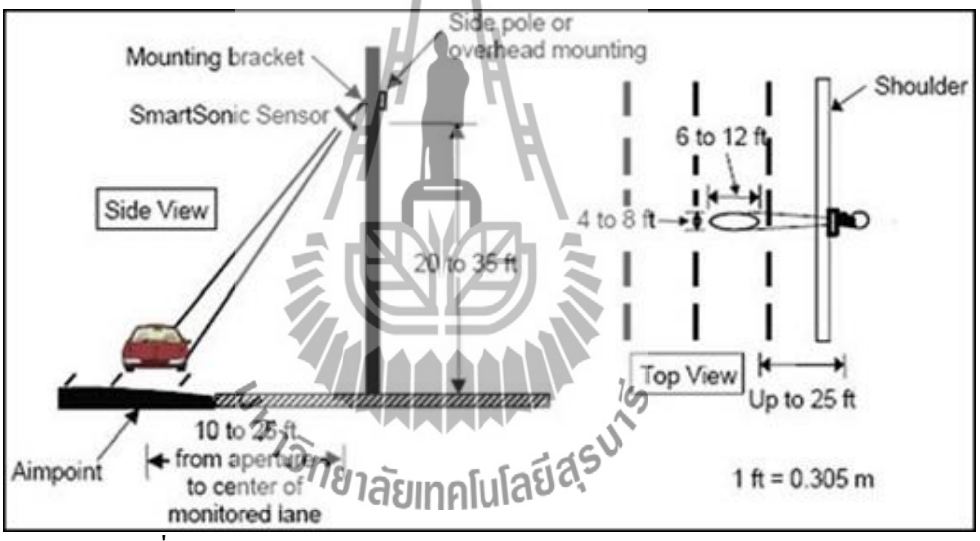
- **ข้อดี**
  - ตรวจวัดได้หลายช่องจราจรพร้อมๆกัน
  - สามารถตรวจวัดรถที่มีความสูงเกินกำหนดได้
  - ใช้งานแพร่หลายในประเทศญี่ปุ่น
- **ข้อเสีย**
  - อุณหภูมิและลมมีผลต่อประสิทธิภาพ
  - คลื่นช่วงกว้างส่งผลต่อความถูกต้องในการวัด Occupancy บน Freeway

6. Passive acoustic sensors เป็นเครื่องมือตรวจวัดการจราจรที่ใช้พลังงานเสียง

หลักการทำงาน

ทำงานโดยไม่ต้องมีการส่งสัญญาณเสียงออกไปแต่จะต้องอาศัยการตรวจวัดเสียงที่เกิดขึ้นจากยานพาหนะที่เคลื่อนที่ผ่านเช่นเสียงเครื่องยนต์เสียงล้อรถเมื่อมีเสียงเกิดขึ้นในพื้นที่การตรวจวัด ซึ่งจะสามารถจำกัดเป็นช่องทางจราจรได้โดยเสียงที่เกิดนอกเขตพื้นที่ตรวจวัดจะไม่ส่งผลต่ออุปกรณ์การวัดจากระดับเสียงที่ตรวจวัดได้จะถูกแปรสัญญาณโดยเครื่องประมวลผลได้ข้อมูลดังนี้

- ปริมาณจราจร
- ความเร็วรถ
- การมีรถบริเวณนั้น (Presence)
- การผ่านของรถบริเวณนั้น (Passage)



รูปที่ 2.22 การตรวจวัดยานพาหนะโดยใช้ Passive acoustic sensors

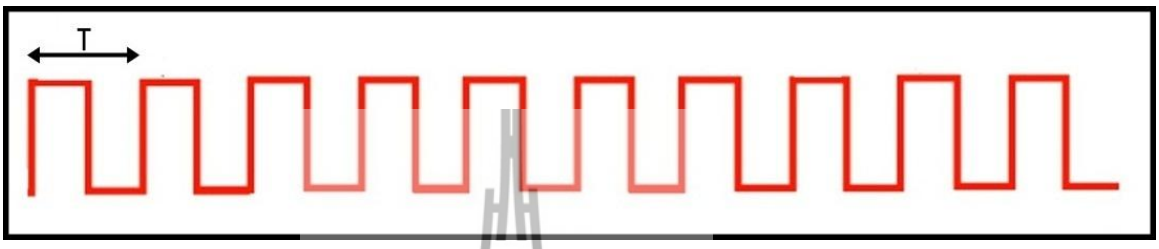
ข้อดี-ข้อเสีย

- **ข้อดี**
  - เป็นเทคโนโลยีแบบ passive
  - ไม่ไวต่อสภาพฝน
  - บางรุ่นสามารถใช้ตรวจวัดได้หลายช่องจราจรพร้อมๆกัน
- **ข้อเสีย**
  - ความถูกต้องของการนับรถลดลงที่อุณหภูมิต่ำ
  - บางรุ่นไม่เหมาะกับสภาพการจราจรติดขัด

2.3 วิธีการนับความถี่

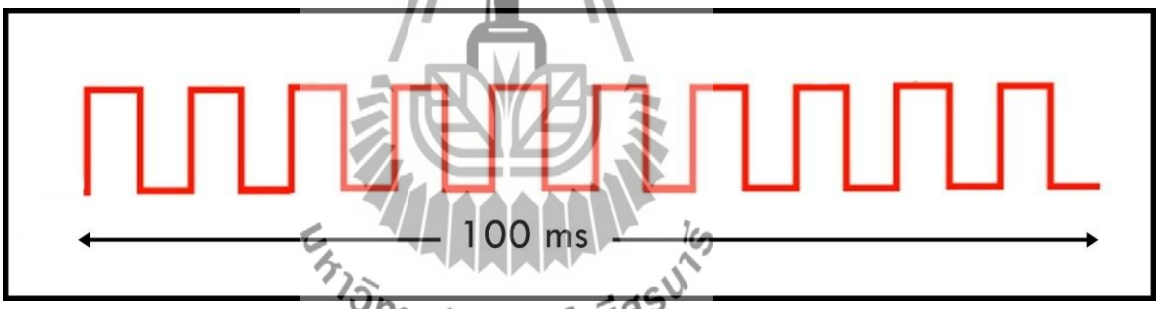
วิธีการนับความถี่มี 2 วิธี คือ

2.3.1 นับสัญญาณที่เป็นรายคาบ โดยจะนับสัญญาณที่คาบเวลา (T) แล้วนำมาหาค่าความถี่จาก สมการ $T = \frac{1}{f}$



รูปที่ 2.23 วิธีการนับสัญญาณที่เป็นรายคาบ

2.3.2 นับสัญญาณในช่วงเวลา จะนับสัญญาณความถี่ในช่วงเวลาที่ได้กำหนด



รูปที่ 2.24 วิธีการนับสัญญาณในช่วงเวลา

ในการนับความถี่ของโปรแกรม นับความถี่สำหรับตรวจวัดรถยนต์ นั้น เราจะนับความถี่แบบนับในช่วงเวลา ซึ่งความถี่ที่เข้ามาจะเป็นสัญญาณ Pulse โดยจะนับความถี่ในช่วงเวลา 1 วินาที (sec) และนับว่าสัญญาณนั้นมีกี่ลูกคลื่น แล้วค่าที่ได้นำไปคำนวณหาค่าความถี่ต่อไป ซึ่งในไมโครคอนโทรลเลอร์นั้น จะใช้ Interrupt ในการนับความถี่ สัญญาณพัลส์ที่เข้ามาจะมีทั้ง Logic1 และ Logic0

บทที่ 2
หลักการและทฤษฎีที่เกี่ยวข้อง

2.1 ไมโครคอนโทรลเลอร์ ARM 7 (CP–JR ARM7 USB-LPC2148)

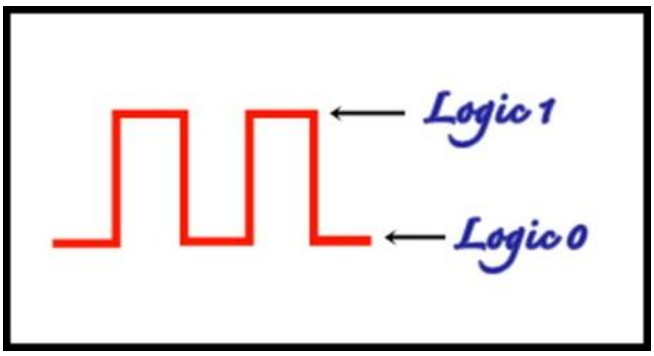
2.1.1 ประวัติความเป็นมาของซีพียู ARM

บริษัท Acorn Computers Limited ในเมืองเคมบริดจ์ ประเทศอังกฤษเป็นประเทศแรกที่ออกแบบซีพียู ARM ในช่วงเดือนตุลาคม ค.ศ. 1983 ถึงเมษายน ค.ศ. 1985 โดยสร้างต้นแบบของซีพียู ARM ในวันที่ 26 เมษายน ค.ศ.1985 จากนั้นได้ถูกส่งไปผลิตที่บริษัท VLSI Technology Inc. ประเทศสหรัฐอเมริกา

บริษัท Acorn เป็นบริษัทชั้นนำผู้ผลิตเครื่องไมโครคอมพิวเตอร์ BBC Microcomputer ที่ได้รับความนิยมแพร่หลายอย่างมากในประเทศอังกฤษตั้งแต่เริ่มวางจำหน่ายในเดือนมกราคม ค.ศ. 1982 เครื่อง BBC Microcomputer เป็นเครื่องที่ทำงานโดยใช้ซีพียู 8 บิต เบอร์ 6502 (ซีพียู 6502 ยังได้รับการนำไปใช้ในเครื่องไมโครคอมพิวเตอร์ APPLE ที่ได้รับความนิยมอย่างมากในประเทศสหรัฐอเมริกาและแพร่หลายไปทั่วโลก)

หลังจากที่ประสบความสำเร็จกับเครื่อง BBC microcomputer แล้วทางบริษัท Acorn ได้คิดพัฒนาซีพียูสำหรับเครื่องไมโครคอมพิวเตอร์ยุคถัดไป โดยช่วงที่กำลังพัฒนาปี ค.ศ. 1983 นั้นซีพียู 16 บิตที่มีใช้งานเป็นแบบ CISC (Complex Instruction Set Computer) ที่มีความซับซ้อนมาก ทำงานช้ากว่าหน่วยความจำที่มีในยุคนั้น ทางวิศวกรข องบริษัทจึงคิดออกแบบซีพียูเพื่อใช้งานทางการค้าเอง ซึ่งถ้าออกแบบเองทั้งหมดจะต้องใช้เวลาและแรงงานมาก ทางทีมวิศวกรผู้ออกแบบจึงนำซีพียู Berkeley RISC I ซึ่งเป็นซีพียูแบบ RISC (Reduced Instruction Set Computer) ขนาด 32 บิต ที่ถูกออกแบบโดยนักศึกษาปริญญาโทของมหาวิทยาลัย Berkeley ที่เป็นซีพียูแบบง่ายๆไม่ซับซ้อน ใช้งานด้านการศึกษามาพัฒนาต่อให้เป็นซีพียู 32 บิต เอนกประสงค์เพื่อใช้งานทางการค้า ซีพียู ARM มีข้อดีในเรื่องสถาปัตยกรรมที่ไม่ซับซ้อน ประหยัดพื้นที่ในการผลิตชิปซีพียูกินพลังงานน้อยโดยที่ยังคงมีสมรรถนะที่สูง

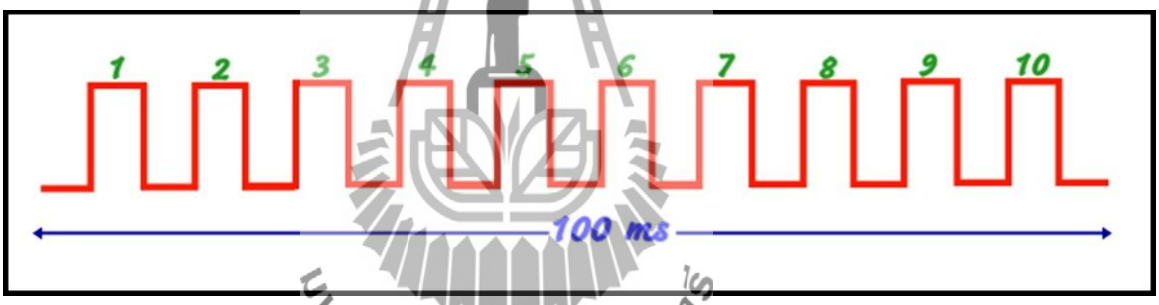
หลังจากที่ออกแบบและทดสอบจนพอใจแล้วได้มีการก่อตั้งบริษัท Advanced RISC Machines Ltd. ขึ้นในเดือนพฤศจิกายน ปี ค.ศ. 1990 ในประเทศอังกฤษ โดยเป็นบริษัทร่วมทุน



รูปที่ 2.25 สัญญาณพัลส์ที่เข้ามา Logic1 และ Logic0

โดย Interrupt นั้นจะเริ่มทำงานนับความถี่ที่สัญญาณพัลส์เข้ามาที่มีค่าเท่ากับ Logic 1ซึ่งจะนับในช่วงเวลาที่เราทำการ Delay ไว้ 100 ms ว่าในช่วงเวลานี้นั้น Logic1 เท่ากับเท่าไหร่ จากนั้นนำค่าที่ได้มานั้นว่าคูณ 10 เข้าไปเพื่อหาความถี่ในหน่วย Hz เช่น

ตัวอย่าง สัญญาณ Pulse ที่เข้ามาในเวลา 100 ms



รูปที่ 2.26 สัญญาณ Pulse ที่จับได้ในช่วงเวลา 100 ms

จากภาพด้านบนจะเห็นว่าสัญญาณ Pulse ที่เข้ามานั้น ในช่วงเวลา 100 ms สัญญาณ Pulse ที่เข้ามาเป็น Logic 1 ที่ Interrupt นับได้มีค่าเท่ากับ 10

เพราะฉะนั้น ค่าความถี่ที่ได้จากช่วงเวลานี้คือ 10 x 10 = 100 Hz

บทที่ 3
การออกแบบโครงงาน

3.1 การออกแบบการทำงานของการตรวจวัดยานพาหนะ

หลักการทำงานของการตรวจวัดยานพาหนะ จะประกอบไปด้วยการทำงานหลักอยู่ 2 องค์ประกอบที่ทำงานร่วมกัน คือส่วนของวงจรกรองความถี่ที่ทำหน้าที่จ่ายความถี่ที่เกิดขึ้นจากการเหนี่ยวนำของลูปเหนี่ยวนำที่ถูกฝังอยู่ใต้พื้นถนนมาให้กับไมโครคอนโทรลเลอร์ ซึ่งไมโครคอนโทรลเลอร์นี้ถูกป้อนคำสั่งให้ทำการนับความถี่ที่ตรวจวัดได้ และคำนวณหาค่าอัตราการเปลี่ยนแปลงความถี่ก่อนที่จะทำการตัดสินใจว่า ในลูปเหนี่ยวนำมียานพาหนะอยู่หรือไม่

รูปที่ 3.1 วงจรกำเนิดความถี่สร้างความถี่ให้ไมโครคอนโทรลเลอร์

3.2 การออกแบบโปรแกรมนับความถี่สำหรับการตรวจวัดยานพาหนะ

โปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ ถูกออกแบบมาเพื่อทำการตรวจนับความถี่ที่สามารถตรวจวัดได้ และทำการคำนวณหาอัตราการเปลี่ยนแปลงความถี่ เพื่อใช้ในการตัดสินใจว่าบนลูปเหนี่ยวนำมีรถอยู่หรือไม่ ซึ่งโครงงานนี้เป็นส่วนประกอบที่สำคัญในกระบวนการตรวจวัดยานพาหนะ โปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ นี้จะประกอบไปด้วย ไมโครคอนโทรลเลอร์บอร์ด ARM7 ที่จะเป็นตัวควบคุมการทำงานหลัก โดยจะต้องมีซอฟต์แวร์ที่ใช้ในการเขียนคำสั่งควบคุม ผู้จัดทำเลือกใช้ภาษาซีโดยใช้โปรแกรม Keil uVision3 เนื่องจากภาษาซีเป็นภาษาโครงสร้างง่ายต่อการทำความเข้าใจและสามารถปรับปรุงพัฒนาต่อได้ง่าย

นอกจากนั้นภาษาซียังเป็นภาษามาตรฐาน ไม่ขึ้นกับฮาร์ดแวร์ (ไมโครคอนโทรลเลอร์) มีความยืดหยุ่นและสามารถใช้งานกับไมโครคอนโทรลเลอร์ตระกูลอื่นได้ง่าย โดยหลักการทำงานหลักของโปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะสามารถลำดับการทำงานเป็นขั้นตอนต่างๆ ดังรูปที่ 3.2 Flow Chart การทำงานหลักของโปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ

3.3 อธิบาย Flow Chart การทำงานหลักของโปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ

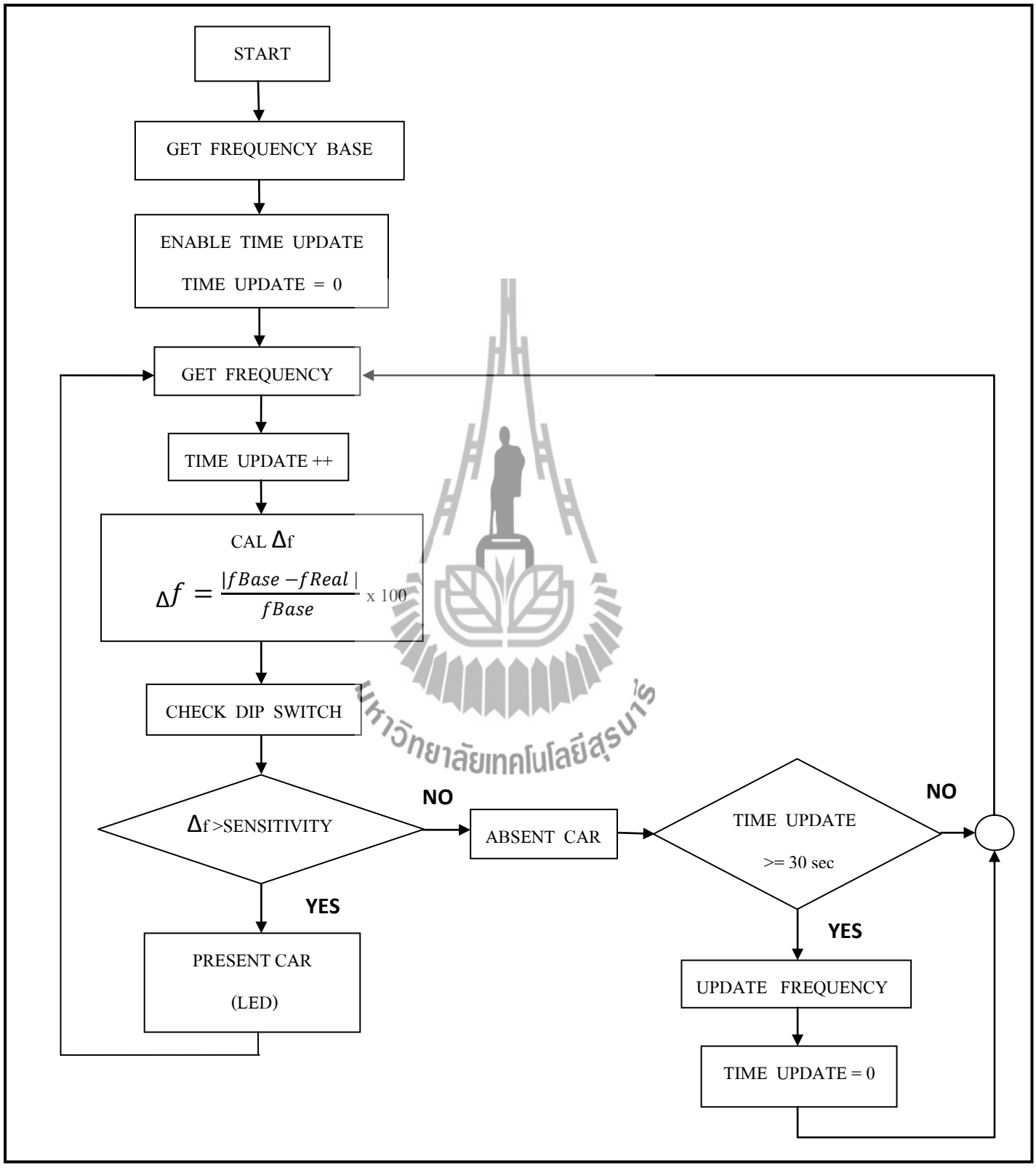
การทำงานหลักของ โปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ นี้ จะเริ่มเก็บความถี่อ้างอิง (Frequency Base) หลังจากที่ทำการเปิดเครื่องเริ่มต้นการทำงาน ค่าของความถี่ที่สามารถนับได้ก็จะนำมาคำนวณเปรียบเทียบหาอัตราการเปลี่ยนแปลงความถี่ว่ามีค่ามากกว่าค่า Sensitivity ที่กำหนดไว้หรือไม่ ถ้าใช่หมายความว่ามีรถยนต์เข้ามาอยู่ในลูปเหนี่ยวนำแล้ว จากนั้นก็วนกลับไปอ่านค่าความถี่ที่นับได้เหมือนเดิม เมื่ออัตราการเปลี่ยนแปลงความถี่มีค่าน้อยกว่าค่า Sensitivity ที่กำหนดไว้โปรแกรมจะทำการตัดสินใจว่าในลูปเหนี่ยวนำไม่มีรถเข้ามาอยู่และทำการเช็คเงื่อนไข Time update >= 30 วินาทีหรือไม่ ถ้าใช่จะทำการอัพเดตความถี่อ้างอิง กล่าวคือ โปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ นี้จะทำการอัพเดตความถี่อ้างอิงทุกๆ 30 วินาที เมื่อทำการอัพเดตเสร็จแล้วจะกำหนดค่าของ Time update = 0 เพื่อเคลียร์การนับเวลาทั้งหมดแล้ววนกลับไปอ่านความถี่ที่ตรวจนับได้ ถ้า Time update < 30 วินาที จะไม่อัพเดตความถี่อ้างอิงแต่จะกลับไปวนลูปอ่านค่าความถี่ที่ตรวจนับได้อีกครั้ง

3.4 อธิบาย Flow Chart การทำงานของฟังก์ชัน Update Frequency

โปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ จะต้องทำการหาค่าอัตราการเปลี่ยนแปลงความถี่อยู่ตลอดเวลา ดังนั้นจึงต้องมีการอัพเดตความถี่อ้างอิง ทุกๆ 30 วินาที ซึ่งความถี่ที่จะนำมาเป็นความถี่อ้างอิงตัวใหม่ บางครั้งอาจจะมีความแตกต่างไปจากความถี่อ้างอิงเดิมมากอาจเนื่องมาจากสภาพแวดล้อมหรือสภาพภูมิอากาศที่แปรเปลี่ยน หรือความถี่ที่อ่านได้นั้นอาจจะเกิดมาจากสัญญาณรบกวน ทำให้ค่าความถี่ที่จะใช้เป็นความถี่อ้างอิงมีค่าที่ผิดเพี้ยนไป ฟังก์ชัน Update Frequency นี้ จะมีการตั้งเงื่อนไขในการอัพเดตความถี่อ้างอิงว่า ถ้าความถี่ที่อ่านได้มีอัตราการเปลี่ยนแปลงไปจากค่าความถี่อ้างอิงน้อยกว่า 0.3 ($\Delta f_{Base} < 0.3$) แสดงว่าความถี่ที่นับ

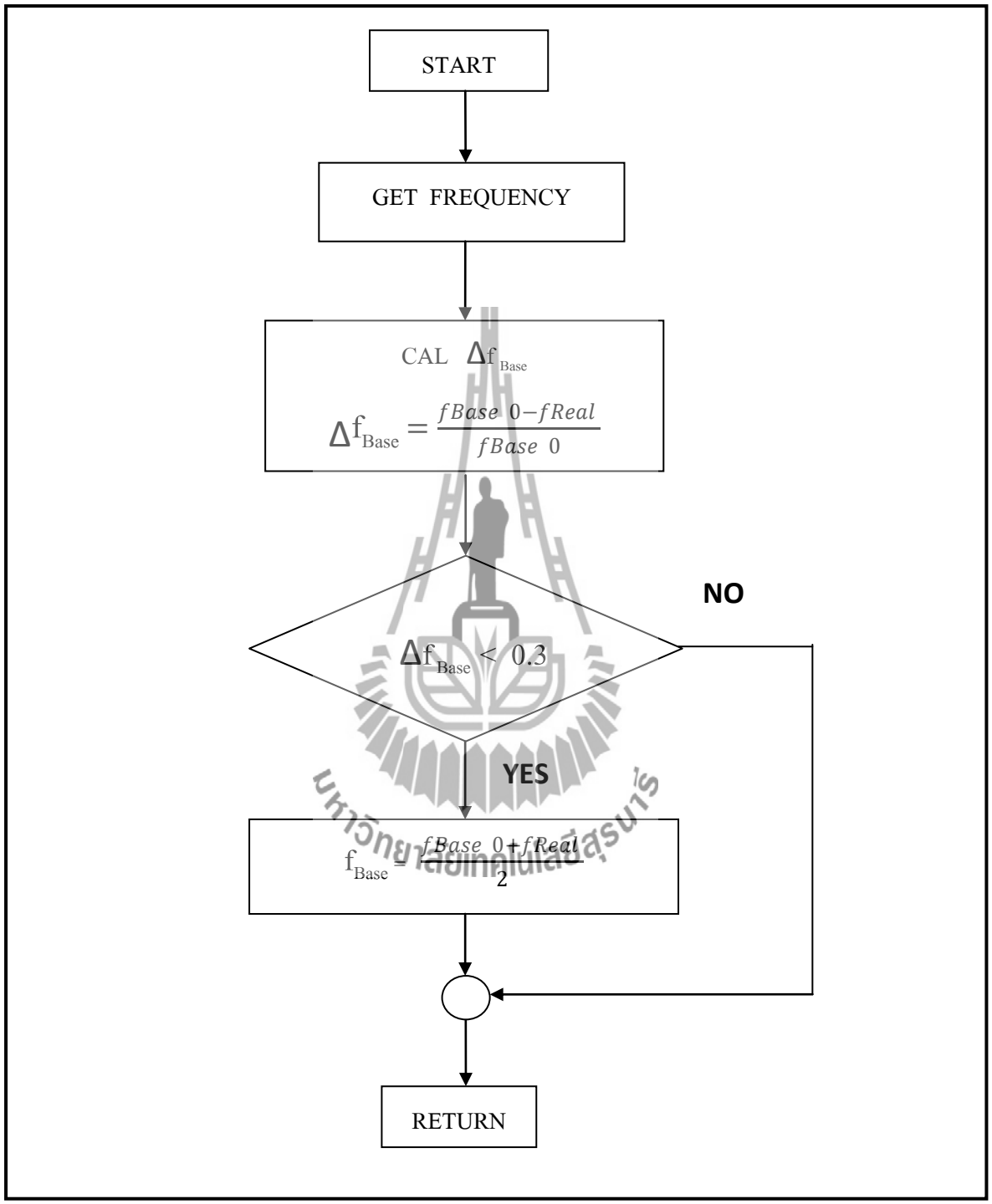
ได้สามารถใช้เป็นความถี่อ้างอิงได้ และเพื่อป้องกันปัญหาของการเกิดสัญญาณรบกวน จึงนำความถี่อ้างอิงเดิมกับความถี่ที่ผ่านเงื่อนไขเข้ามา มาทำการหาค่าเฉลี่ยได้เป็นความถี่อ้างอิงตัวใหม่ที่จะนำไปใช้งาน ถ้าความถี่ที่นับได้มีค่ามากกว่า 0.3 ($\Delta f_{Base} > 0.3$) จะไม่ทำการอัพเดตความถี่และให้ส่งค่ากลับกลับไปยังตำแหน่งที่เรียกใช้

3.5 Flow Chart การทำงานหลักโปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ



รูปที่ 3.2 Flow Chart แสดงการทำงานของโปรแกรมหลัก

3.6 Flow Chart Function Update Frequency



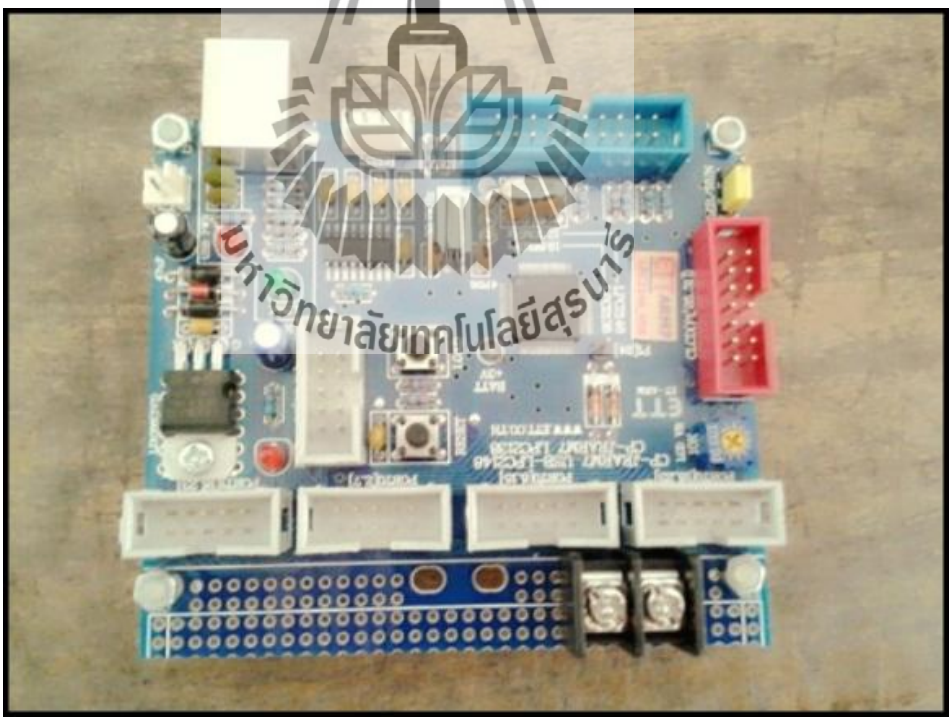
รูปที่ 3.3 Flow Chart แสดงการทำงานของ Function Update Frequency

บทที่ 4

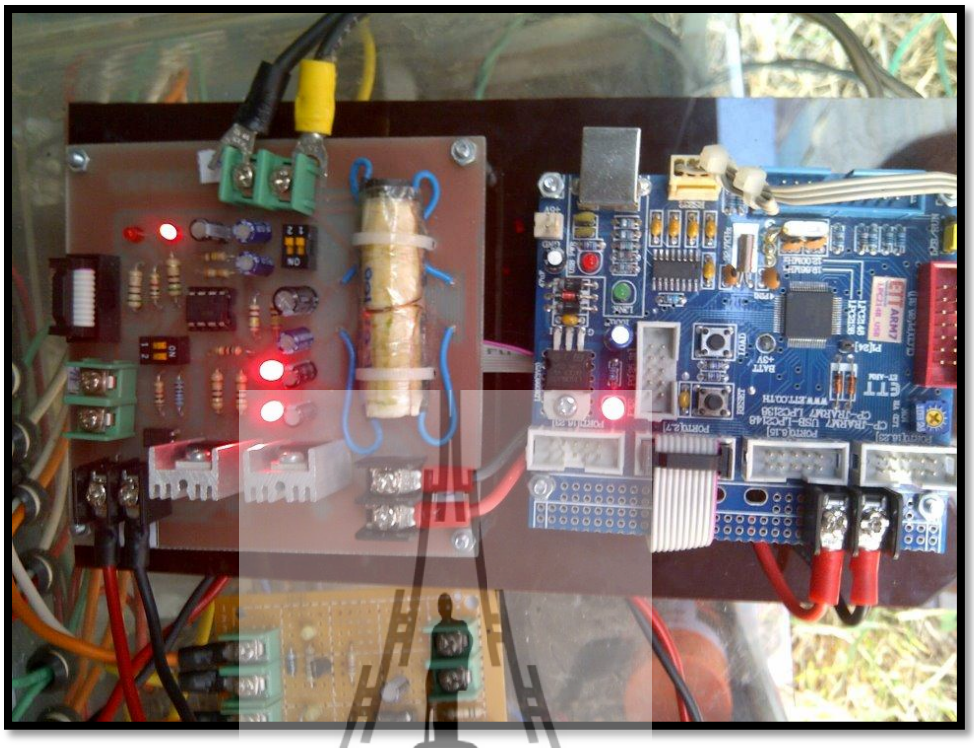
การทดสอบ

หลักการทำงานของตัวตรวจวัดยานพาหนะคือ การสร้างความถี่ที่มีค่าคงที่เมื่อไม่มียานพาหนะ แต่เมื่อมียานพาหนะเข้ามาที่ ลูปเหนี่ยวนำ ก็จะทำให้ความถี่เปลี่ยนไป ซึ่งตัวเหนี่ยวนำความถี่จะเป็นเซ็นเซอร์ในการตรวจวัดยานพาหน ะโดยอัตราการเปลี่ยนแปลงความถี่ที่เกิดขึ้นจะนำมาตัดสินใจว่ามียานพาหนะผ่านมาที่ลูปเหนี่ยวนำหรือไม่ ซึ่งการทดสอบจะทำการทดสอบกับยานพาหนะประเภทรถยนต์ส่วนบุคคลและยานพาหนะขนาดเล็กอัตราการเปลี่ยนแปลงที่เกิดขึ้นจากรถยนต์ส่วนบุคคลจะมีค่ามากกว่า เพราะขนาดของยานพาหนะมีขนาดใหญ่กว่า

4.1 เงื่อนไขในการทดสอบ



รูปที่ 4.1 บอร์ด CP-JR ARM7 USB-LPC2148



รูปที่ 4.2 การทำงานร่วมกันระหว่างบอร์ดกำเนิดความถี่และบอร์ดไมโครคอนโทรลเลอร์

ในการทดสอบจะนำบอร์ด LPC2148 มาต่อใช้งานร่วมกับบอร์ด วงจรกำเนิดความถี่สำหรับการตรวจวัดยานพาหนะ โดยบอร์ด วงจรกำเนิดความถี่สำหรับการตรวจวัดยานพาหนะ ทำหน้าที่สร้างสัญญาณความถี่เข้าสู่ บอร์ด LPC2148 เมื่อมียานพาหนะผ่านเข้ามาที่ลูปเหนี่ยวนำ บอร์ดวงจรกำเนิดความถี่สำหรับการตรวจวัดยานพาหนะจะมี LED ติดสว่างหนึ่งดวงเพื่อเป็นสัญญาณบอกว่ามียานพาหนะผ่านเข้ามาที่ลูปเหนี่ยวนำ และรอให้บอร์ด LPC2148 ทำหน้าที่ตัดสินใจว่าจะอัพเดตความถี่ที่เวลาใดต่อไป

รูปที่ 4.3 แยกการจราจรที่ใช้ทดสอบ (1)

รูปที่ 4.4 แยกการจราจรที่ใช้ทดสอบ (2)

รูปที่ 4.5 ตู้ควบคุมสัญญาณไฟจราจรที่ติดตั้งตรงบริเวณแยกการจราจร

ตัวตรวจวัดยานพาหนะที่ได้ออกแบบนั้นจะใช้หลักการของขดลวดเหนี่ยวนำ โดยจะติดตั้งตัวเหนี่ยวนำความถี่ลูปเหนี่ยวนำไว้ที่ใต้พื้นถนนตรงจุดที่เราต้องการจะตรวจสอบยานพาหนะ โดยตัวเหนี่ยวนำความถี่จะมีลักษณะเป็นวงกลมเส้นผ่านศูนย์กลาง 1.5 เมตร และห่างจากตู้ควบคุมสัญญาณไฟจราจรประมาณ 30 เมตร

รูปที่ 4.6 ลูปเหนี่ยวนำที่ติดตั้งที่พื้นถนน

4.2 ตัวตรวจวัดยานพาหนะ

ตัวตรวจวัดยานพาหนะที่ออกแบบนั้นจะสามารถเลือกความถี่ได้ 4 ความถี่ โดยการเลือก Dip Switch เพื่อให้ได้ความถี่ตามต้องการและความถี่ที่สามารถเลือกใช้งานมีทั้งหมดดังนี้

ตารางที่ 4.1 ความถี่ของตัวตรวจวัดยานพาหนะ

Dip Switch		Frequency (kHz)
1	2	
OFF	OFF	48
ON	OFF	34
OFF	ON	26
ON	ON	23

รูปที่ 4.7 Dip Switch สำหรับเลือกความถี่ของตัวตรวจวัดยานพาหนะ

ในการตรวจวัดยานพาหนะความไวเป็นพารามิเตอร์ที่สำคัญอย่างยิ่งที่จะบอกถึงประสิทธิภาพในการตรวจวัดยานพาหนะ เพราะถ้ามีความไวในการทดสอบมากจะทำให้สามารถตรวจวัดยานพาหนะขนาดเล็กได้ซึ่งในงาน ทดสอบนี้ได้ออกแบบความไวในการตรวจสอบยานพาหนะไว้ 4 ระดับสามารถเลือกความไวในการตรวจสอบโดยการเลือกจาก Dip Switch ซึ่งจะสามารถเลือกได้ดังนี้

ตารางที่ 4.2 ความไวสำหรับตรวจวัดยานพาหนะ

Dip Switch		Sensitivity
1	2	Channel 1
OFF	OFF	Disable
OFF	ON	2%
ON	OFF	1%
ON	ON	0.5%

รูปที่ 4.8 Dip Switch สำหรับเลือกความไวของตัวตรวจวัดยานพาหนะ

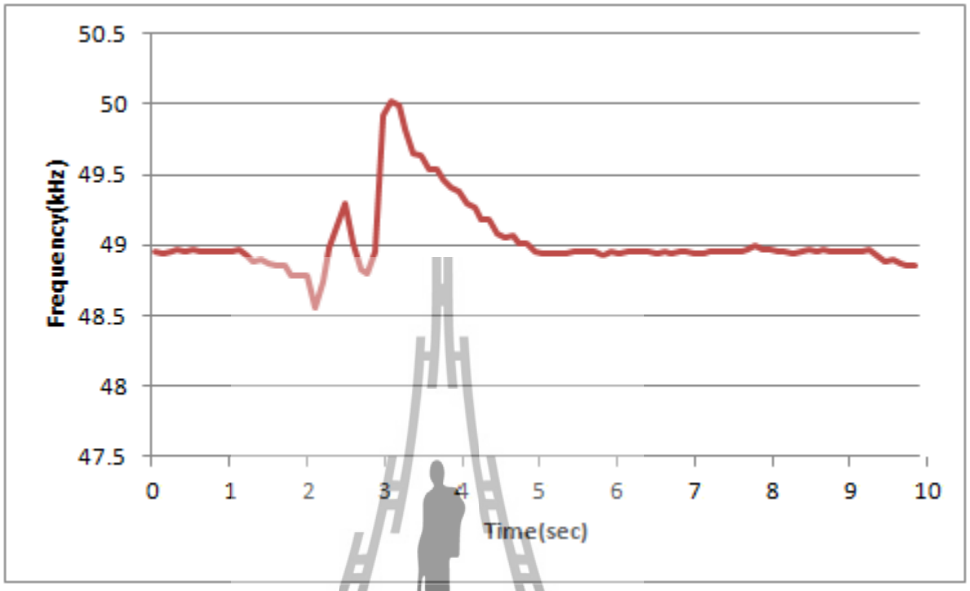
ความไว 0.5% หมายถึง การตรวจสอบอัตราการเปลี่ยนแปลงความถี่ของตัวตรวจวัดยานพาหนะ โดยความไวที่ 0.5%ถือว่าเป็นความไวที่สูงที่สุดเพราะว่าสามารถตรวจวัดยานพาหน ะขนาดเล็กได้

รูปที่ 4.9 ลักษณะยานพาหนะขนาดเล็กอยู่บนลูปเหนี่ยวนำ

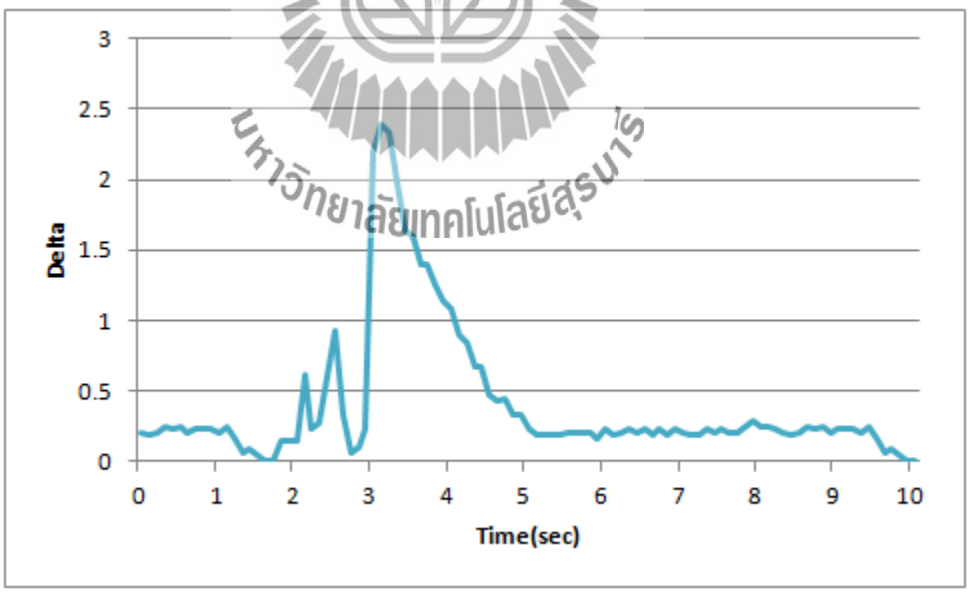
รูปที่ 4.10 ลักษณะยานพาหนะประเภทรถยนต์ส่วนบุคคลอยู่บนลูปเหนี่ยวนำ

4.3 ผลการทดสอบโปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ

4.3.1 ทดสอบตรวจวัดยานพาหนะประเภทรถยนต์ส่วนบุคคลที่ความถี่ 48 kHz

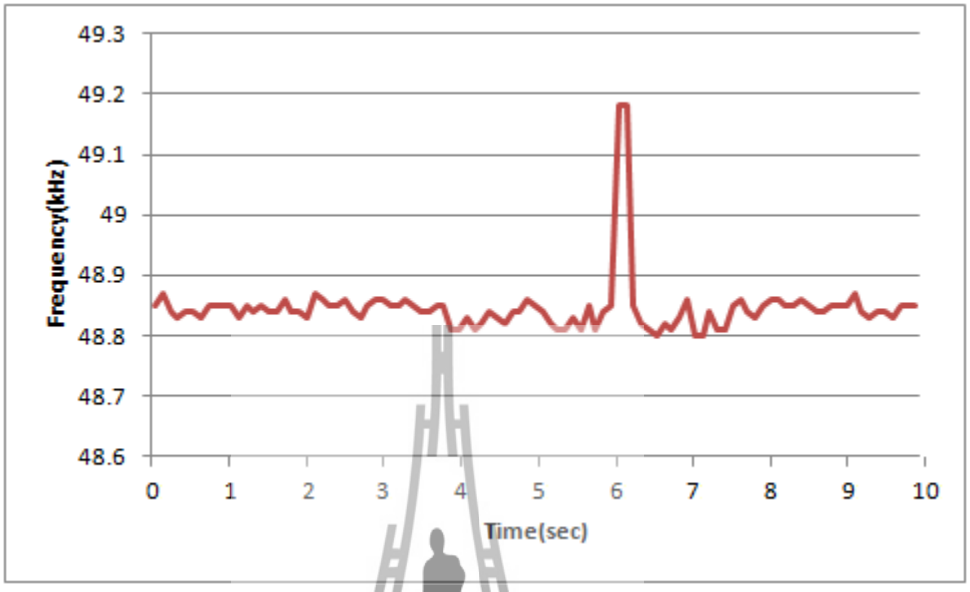


รูปที่ 4.11 ความถี่ 48 kHz ที่เกิดการเปลี่ยนแปลงเมื่อมีรถยนต์ส่วนบุคคลอยู่บนลูปเหนี่ยวนำ

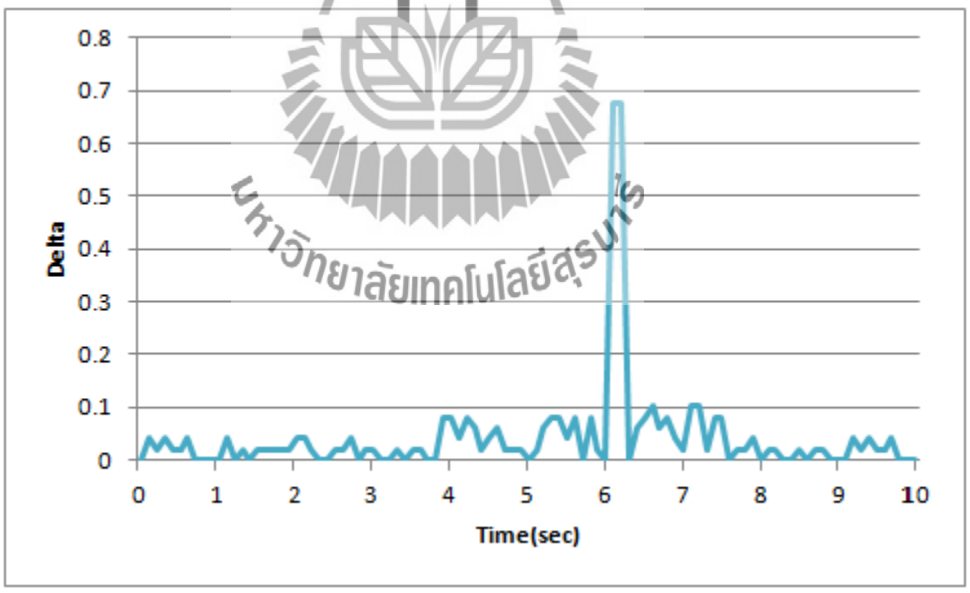


รูปที่ 4.12 อัตราการเปลี่ยนแปลงความถี่ที่ 48 kHz เมื่อมีรถยนต์ส่วนบุคคลอยู่บนลูปเหนี่ยวนำ

4.3.2 ทดสอบตรวจวัดยานพาหนะประเภทรถจักรยานยนต์ที่ความถี่ 48 kHz

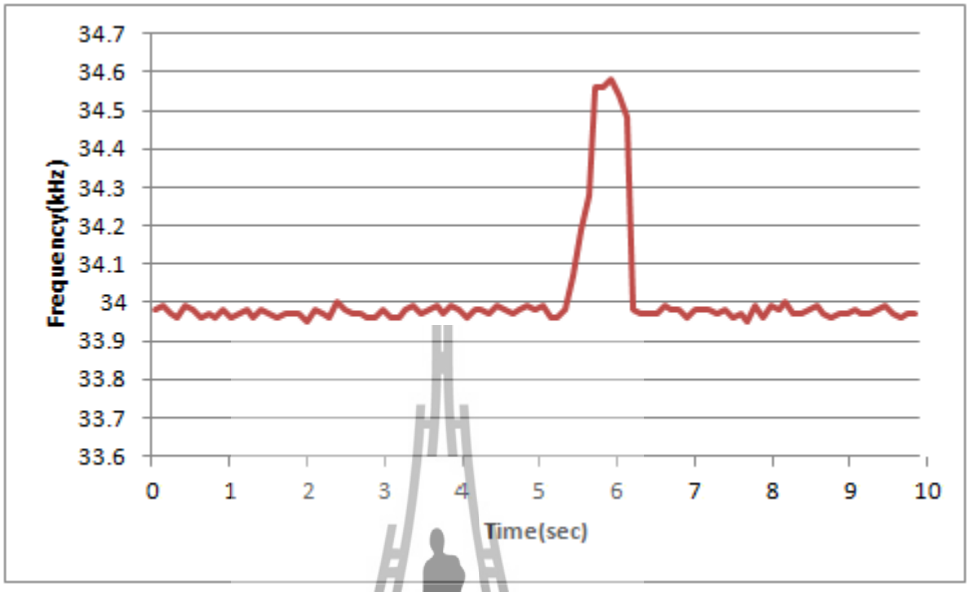


รูปที่ 4.13 ความถี่ 48 kHz ที่เกิดการเปลี่ยนแปลงเมื่อมีรถจักรยานยนต์อยู่บนลูปเหนี่ยวนำ

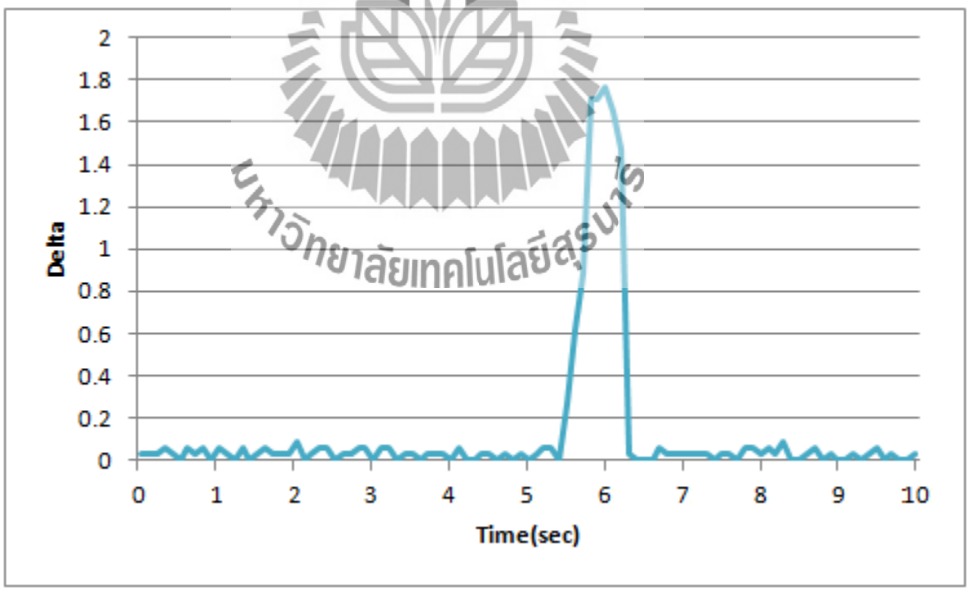


รูปที่ 4.14 อัตราการเปลี่ยนแปลงความถี่ที่ 48 kHz เมื่อมีรถจักรยานยนต์อยู่บนลูปเหนี่ยวนำ

4.3.3 ทดสอบตรวจวัดยานพาหนะประเภทรถยนต์ส่วนบุคคลที่ความถี่ 34 kHz

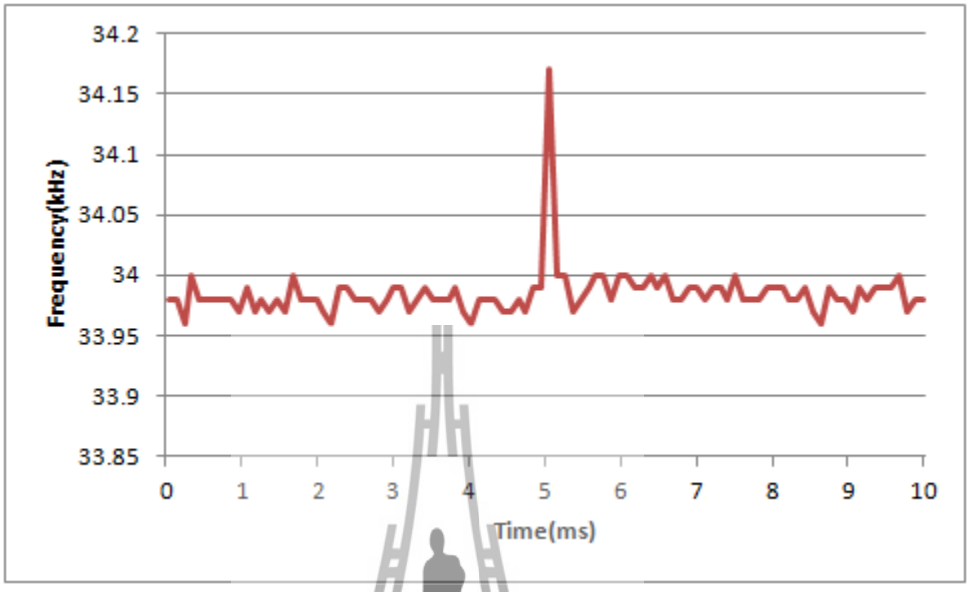


รูปที่ 4.15 ความถี่ 34 kHz ที่เกิดการเปลี่ยนแปลงเมื่อมีรถยนต์ส่วนบุคคลอยู่บนลูปเหนี่ยวนำ

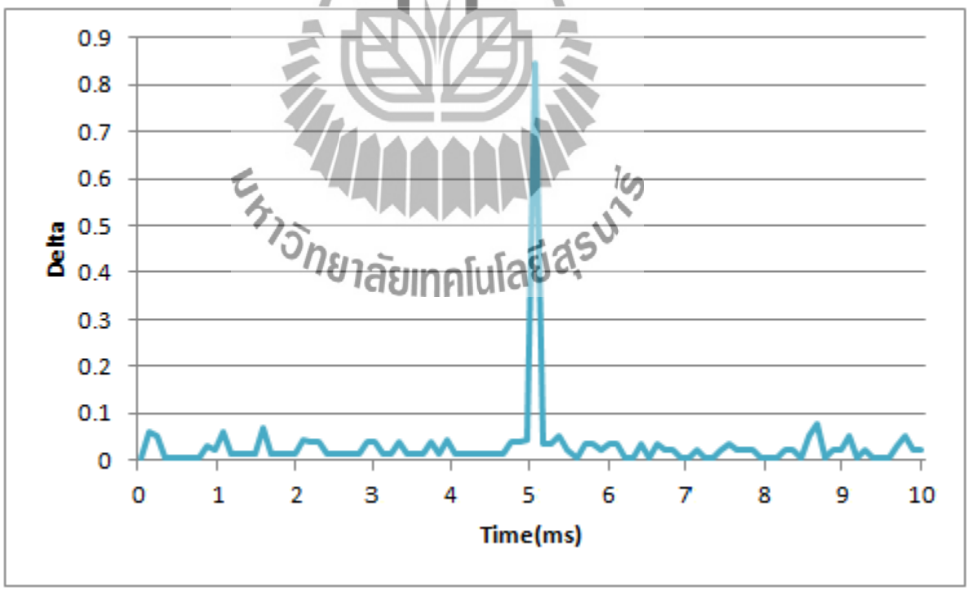


รูปที่ 4.16 อัตราการเปลี่ยนแปลงความถี่ที่ 34 kHz เมื่อมีรถยนต์ส่วนบุคคลอยู่บนลูปเหนี่ยวนำ

4.3.4 ทดสอบตรวจวัดยานพาหนะประเภทรถจักรยานยนต์ที่ความถี่ 34 kHz

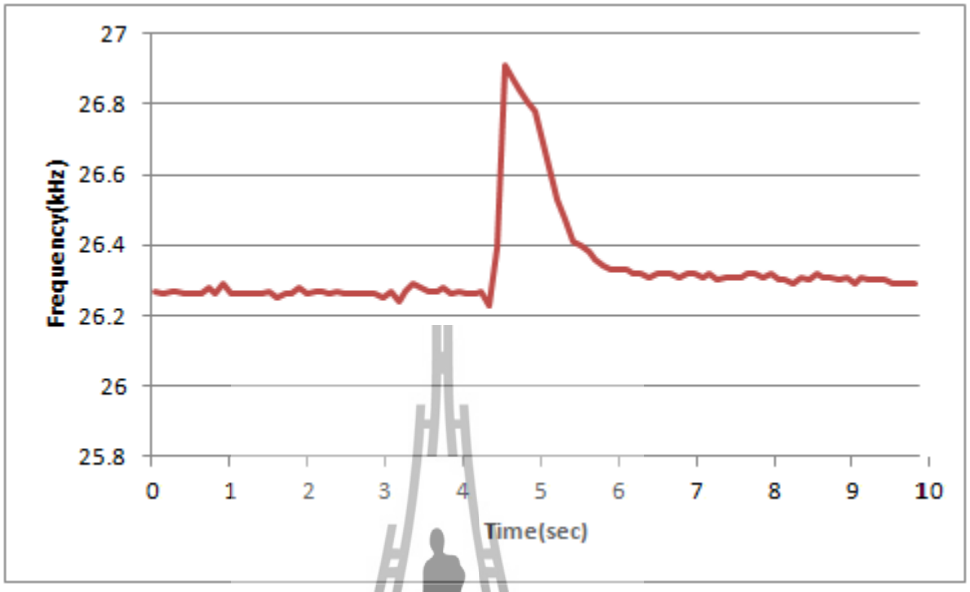


รูปที่ 4.17 ความถี่ 34 kHz ที่เกิดการเปลี่ยนแปลงเมื่อมีรถจักรยานยนต์อยู่บนลูปเหนี่ยวนำ

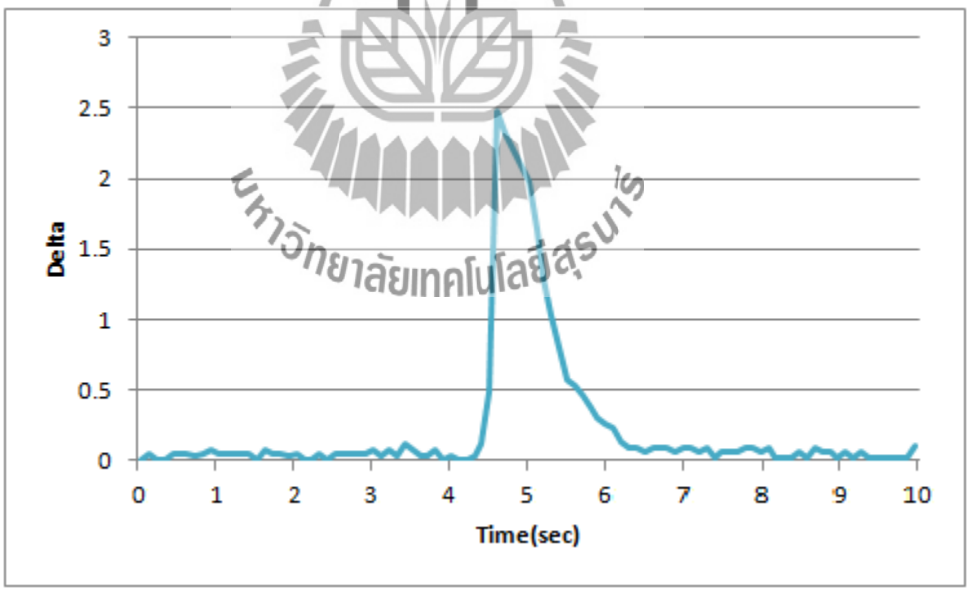


รูปที่ 4.18 อัตราการเปลี่ยนแปลงความถี่ที่ 34 kHz เมื่อมีรถจักรยานยนต์อยู่บนลูปเหนี่ยวนำ

4.3.5 ทดสอบตรวจวัดยานพาหนะประเภทรถยนต์ส่วนบุคคลที่ความถี่ 26 kHz

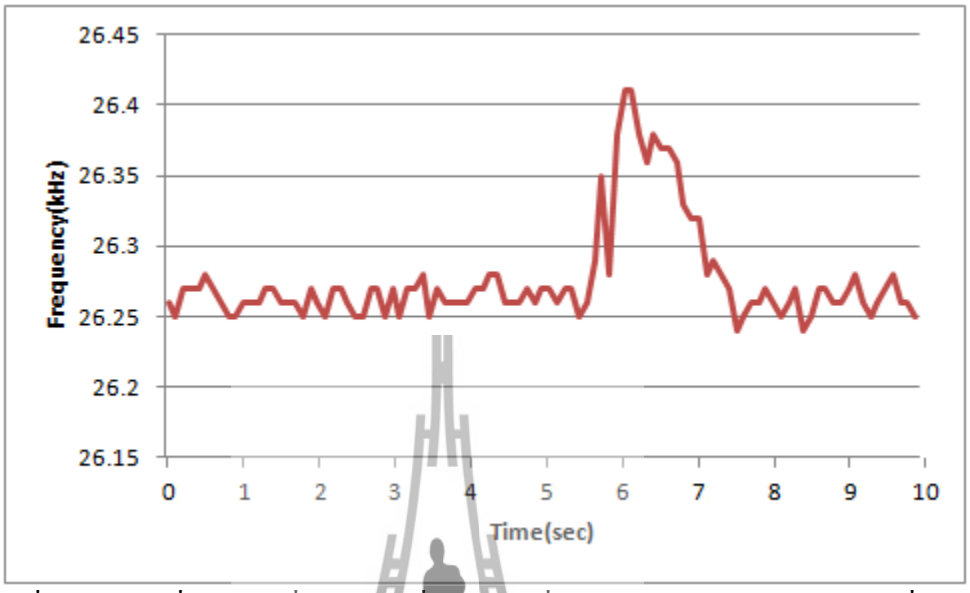


รูปที่ 4.19 ความถี่ 26 kHz ที่เกิดการเปลี่ยนแปลงเมื่อมีรถยนต์ส่วนบุคคลอยู่บนลูปเหนี่ยวนำ

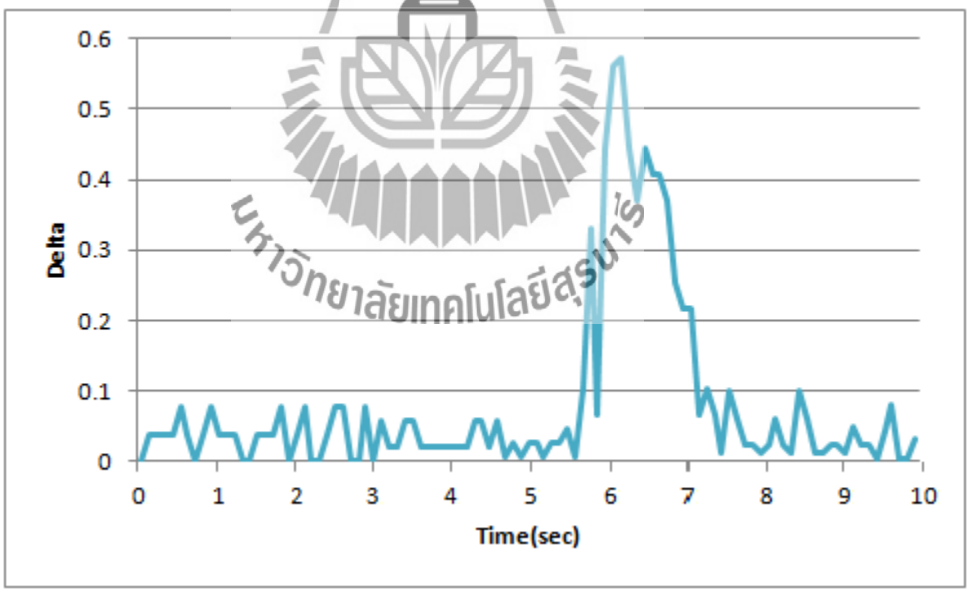


รูปที่ 4.20 อัตราการเปลี่ยนแปลงความถี่ที่ 26 kHz เมื่อมีรถยนต์ส่วนบุคคลอยู่บนลูปเหนี่ยวนำ

4.3.6 ทดสอบตรวจวัดยานพาหนะประเภทรถจักรยานยนต์ที่ความถี่ 26 kHz

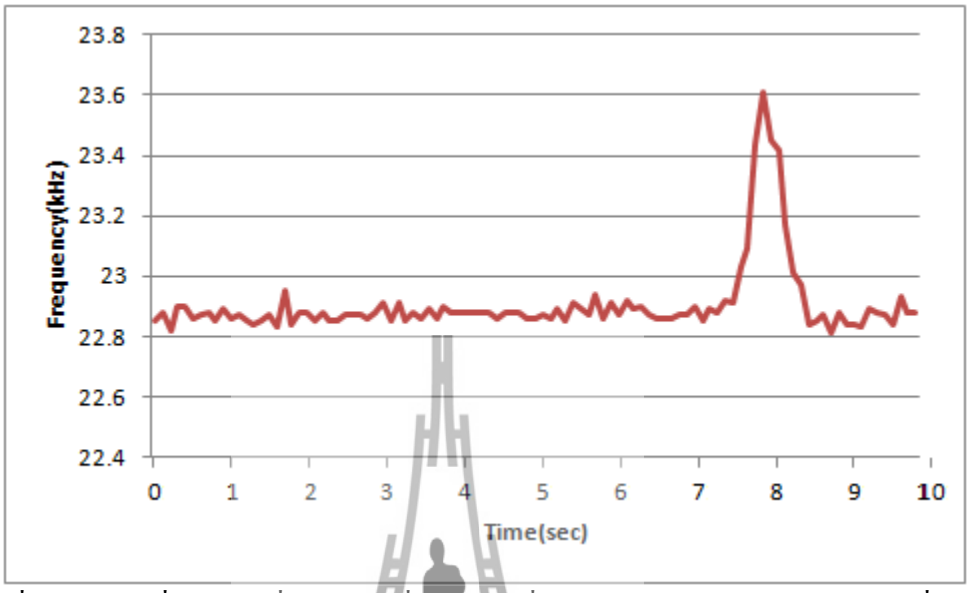


รูปที่ 4.21 ความถี่ 26 kHz ที่เกิดการเปลี่ยนแปลงเมื่อมีรถจักรยานยนต์อยู่บนลูปเหนี่ยวนำ

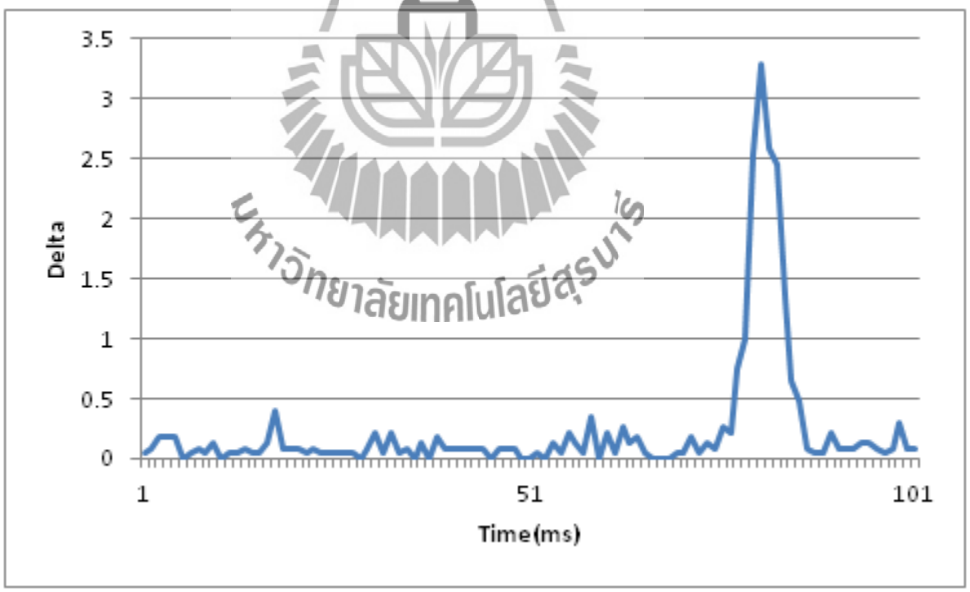


รูปที่ 4.22 อัตราการเปลี่ยนแปลงความถี่ที่ 26 kHz เมื่อมีรถจักรยานยนต์อยู่บนลูปเหนี่ยวนำ

4.3.7 ทดสอบตรวจวัดยานพาหนะประเภทรถยนต์ส่วนบุคคลที่ความถี่ 22 kHz

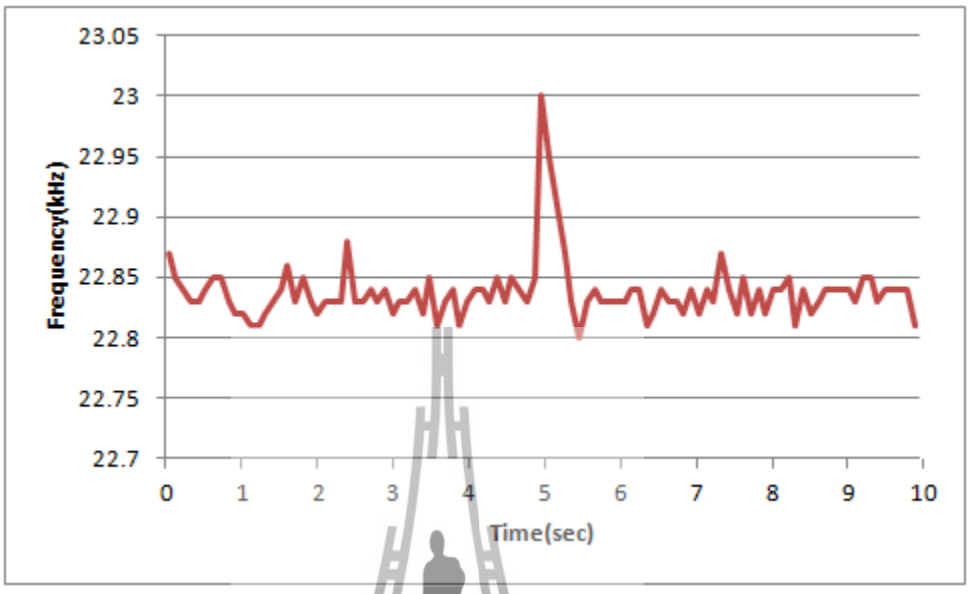


รูปที่ 4.23 ความถี่ 22 kHz ที่เกิดการเปลี่ยนแปลงเมื่อมีรถยนต์ส่วนบุคคลอยู่บนลูปเหนี่ยวนำ

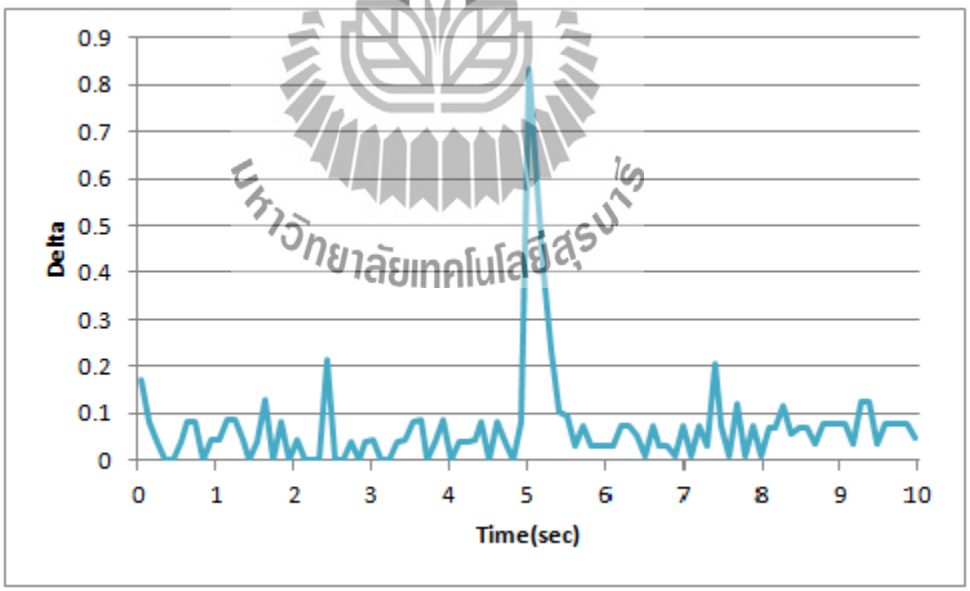


รูปที่ 4.24 อัตราการเปลี่ยนแปลงความถี่ที่ 22 kHz เมื่อมีรถยนต์ส่วนบุคคลอยู่บนลูปเหนี่ยวนำ

4.3.8 ทดสอบตรวจวัดยานพาหนะประเภทรถจักรยานยนต์ที่ความถี่ 22 kHz



รูปที่ 4.25 ความถี่ 22 kHz ที่เกิดการเปลี่ยนแปลงเมื่อมีรถจักรยานยนต์อยู่บนลูปเหนี่ยวนำ



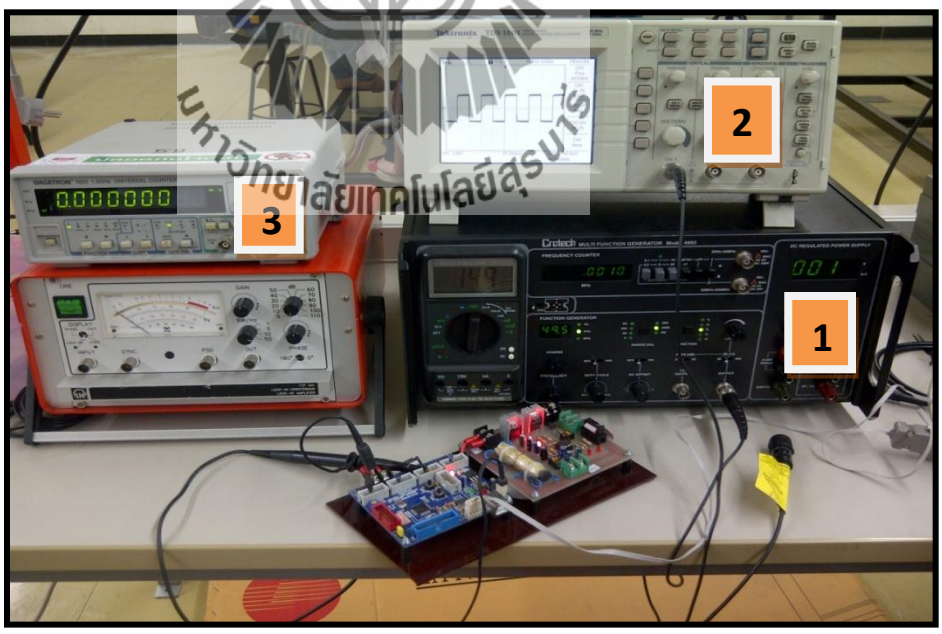
รูปที่ 4.26 อัตราการเปลี่ยนแปลงความถี่ที่ 22 kHz เมื่อมีรถจักรยานยนต์อยู่บนลูปเหนี่ยวนำ

จากการทดสอบทั้ง 4 ความถี่กับยานพาหนะขนาดเล็กและยานพาหนะประเภทรถยนต์ส่วนบุคคล พบว่าเมื่อยานพาหนะเข้ามาอยู่บนจุดที่ติดตั้งตัวเหนี่ยวนำความถี่จะทำให้เกิดความถี่เปลี่ยนแปลงไปจากเดิม เมื่อคำนวณหาอัตราการเปลี่ยนแปลงความถี่พบว่า ยานพาหนะประเภทรถยนต์ส่วนบุคคลจะมีอัตราการเปลี่ยนแปลงเฉลี่ยอยู่บนประมาณ 1.5-3.5 % และยานพาหนะขนาดเล็กจะมีอัตราการเปลี่ยนแปลงเฉลี่ยอยู่บนประมาณ 0.5–1 % ดังนั้นเมื่อเราพิจารณาความไวในการตรวจวัดยานาพาหนะเราสามารถตรวจวัดยานพาหนะขนาดเล็กได้

4.4 การทดสอบโปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะกับเครื่องมือวัดในห้องปฏิบัติการ

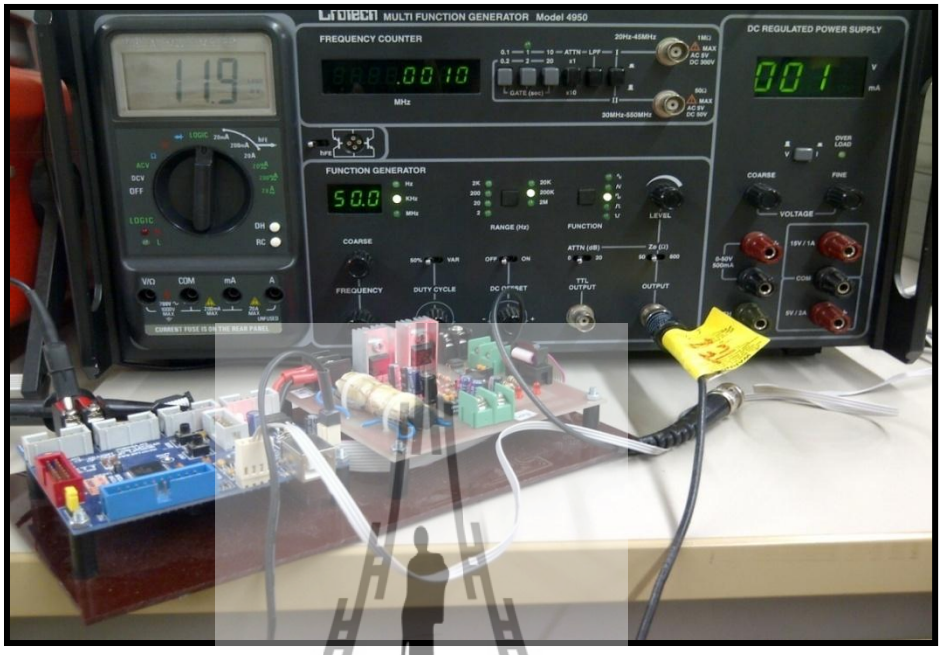
การทดสอบโปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะกับเครื่องมือวัดในห้องปฏิบัติการประกอบด้วยเครื่องมือต่างๆ ดังนี้

1. เครื่อง Multi Function Generator ใช้สำหรับสร้างความถี่ที่ต้องการทดสอบ
2. เครื่อง Oscilloscope ใช้วัดระดับของสัญญาณและความถี่ที่สามารถจับได้
3. เครื่อง Universal Counter ใช้สำหรับตรวจนับความถี่



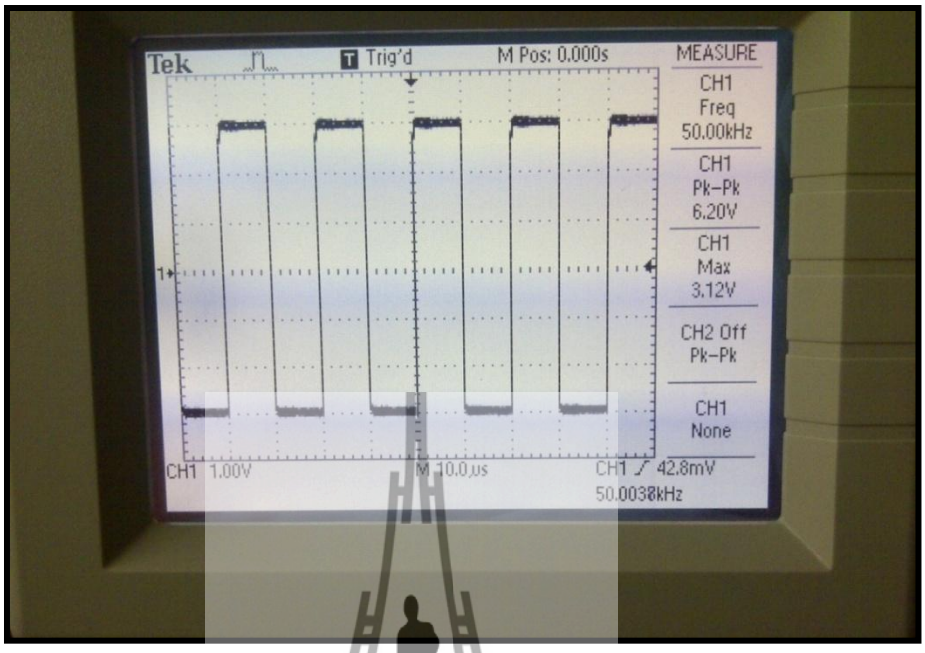
รูปที่ 4.27 อุปกรณ์การทดสอบโปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะในห้องปฏิบัติการ

4.4.1 การทดสอบโปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะที่ความถี่ 50 kHz

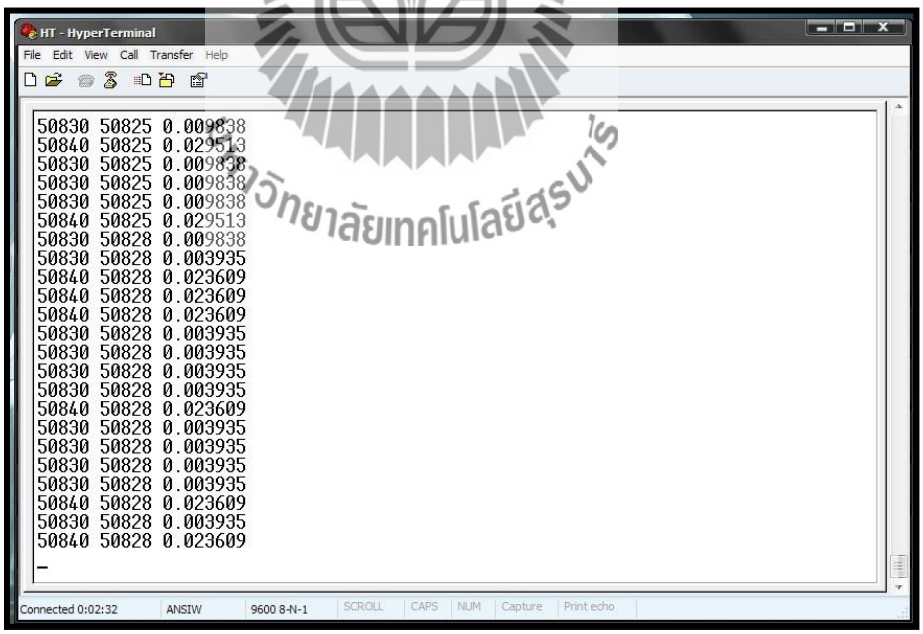


รูปที่ 4.28 เครื่อง Multi Function Generator สร้างความถี่ที่ 50 kHz

รูปที่ 4.29 เครื่อง Universal Counter อ่านความถี่ได้ที่ 50 kHz



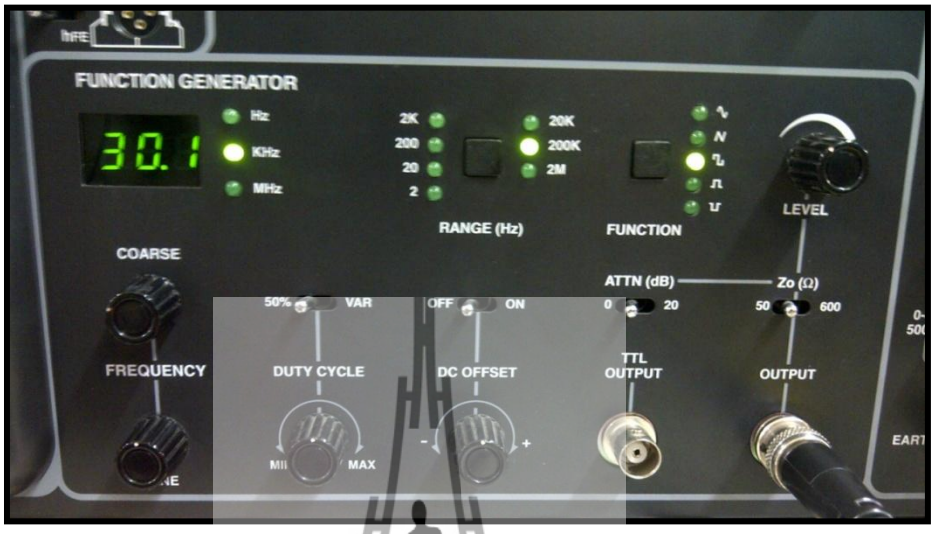
รูปที่ 4.30 Oscilloscope วัดความถี่ได้ 50 kHz



รูปที่ 4.31 โปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ

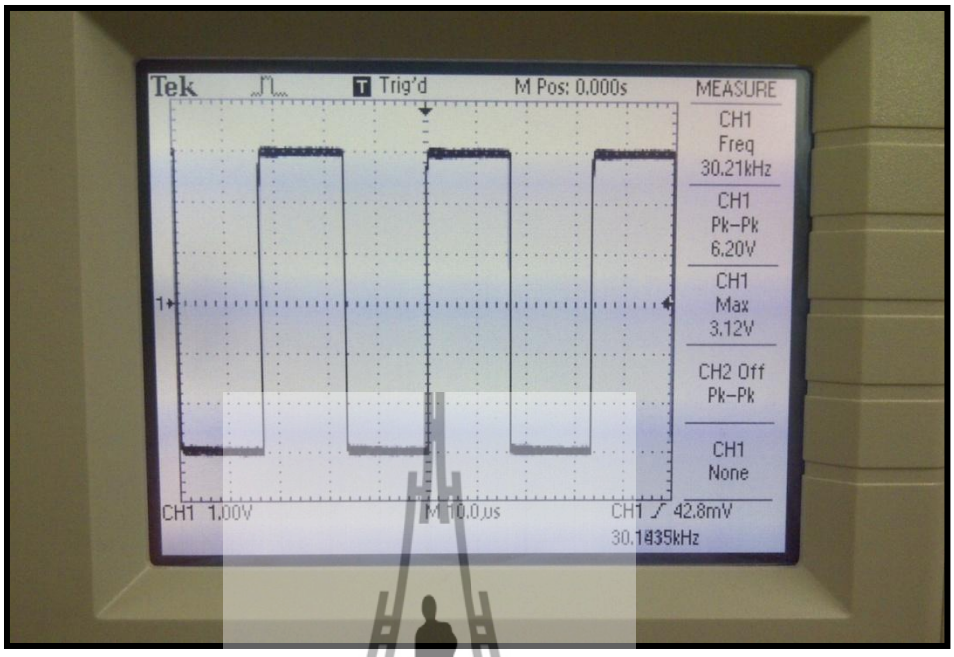
อ่านความถี่ที่สร้างมาจากเครื่อง Multi Function Generator ที่ 50 kHz

4.4.2 การทดสอบโปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะที่ความถี่ 30 kHz

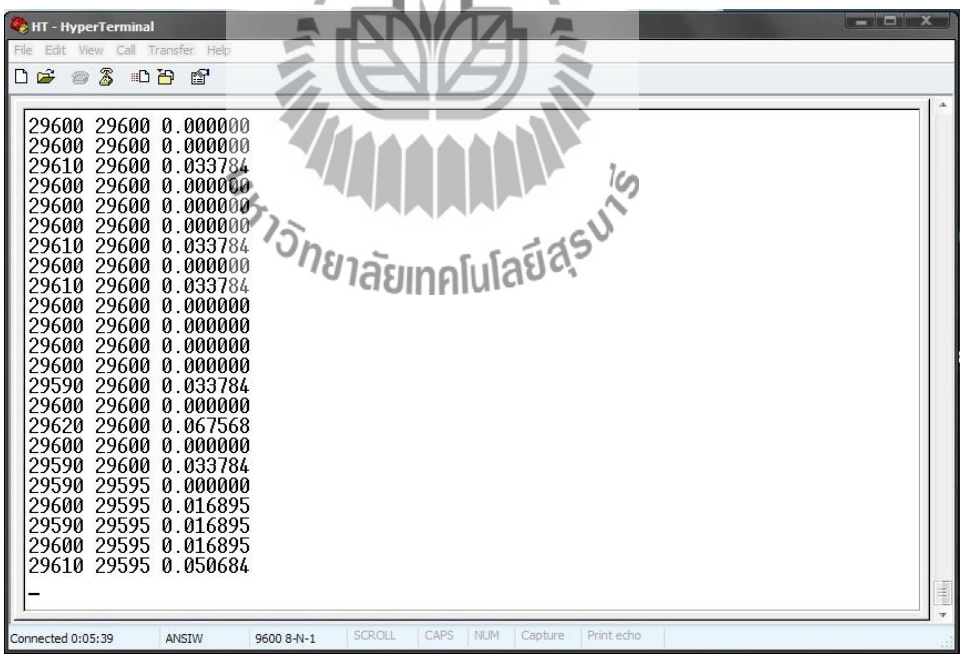


รูปที่ 4.32 เครื่อง Multi Function Generator สร้างความถี่ที่ 30 kHz

รูปที่ 4.33 เครื่อง Universal Counter อ่านความถี่ได้ที่ 30 kHz



รูปที่ 4.34 Oscilloscope วัดความถี่ได้ 30 kHz



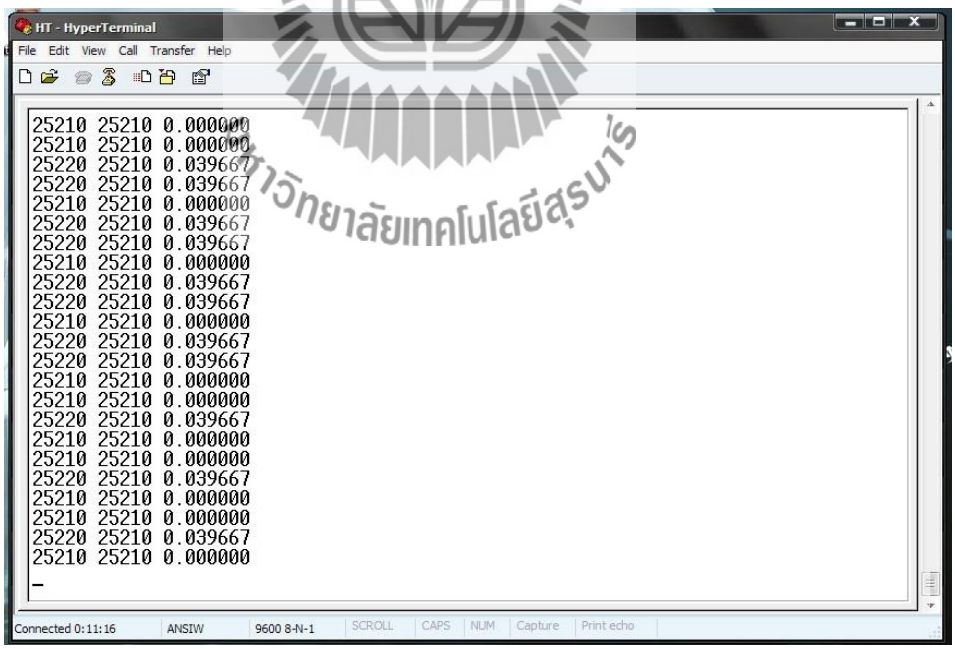
รูปที่ 4.35 โปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ

อ่านความถี่ที่สร้างมาจากเครื่อง Multi Function Generator ที่ 30 kHz

4.5 การทดสอบโปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ กับความถี่ที่สร้างมาจากวงจรกำเนิดความถี่

4.5.1 ทดสอบที่ความถี่ 25 kHz

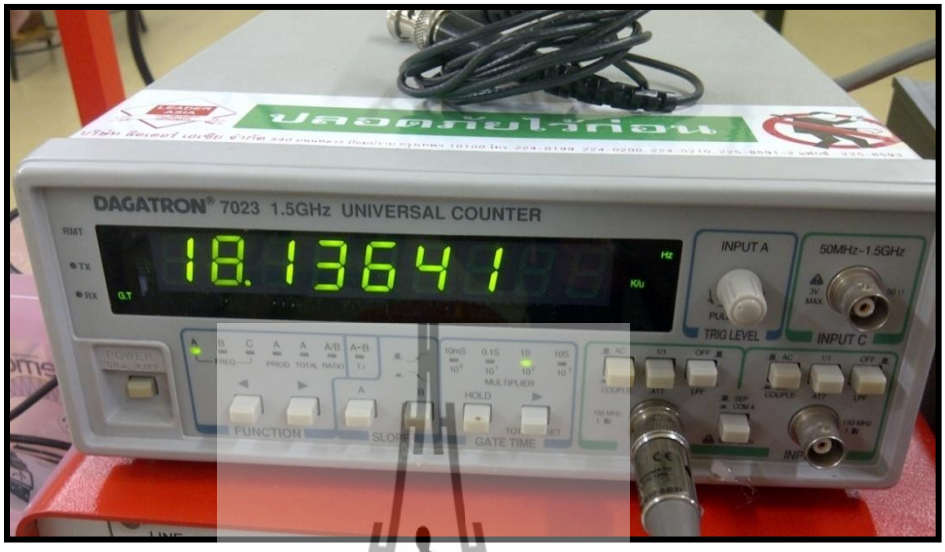
รูปที่ 4.36 เครื่อง Universal Counter อ่านความถี่ได้ 25 kHz



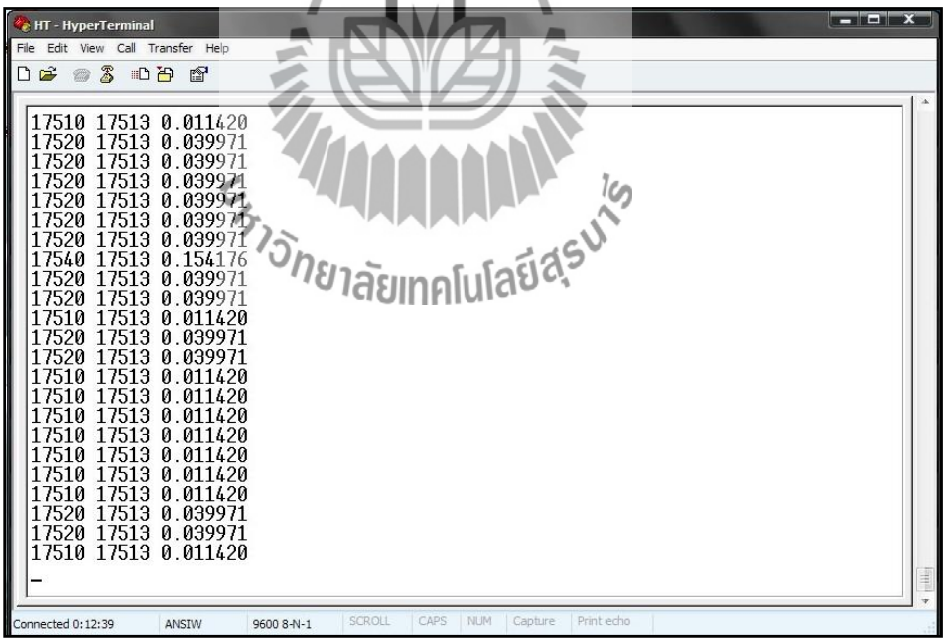
รูปที่ 4.37 โปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ

อ่านความถี่ที่สร้างมาจากเครื่อง Multi Function Generator ที่ 25 kHz

4.5.2 ทดสอบที่ความถี่ 18 kHz



รูปที่ 4.38 เครื่อง Universal Counter อ่านความถี่ได้ 18 kHz

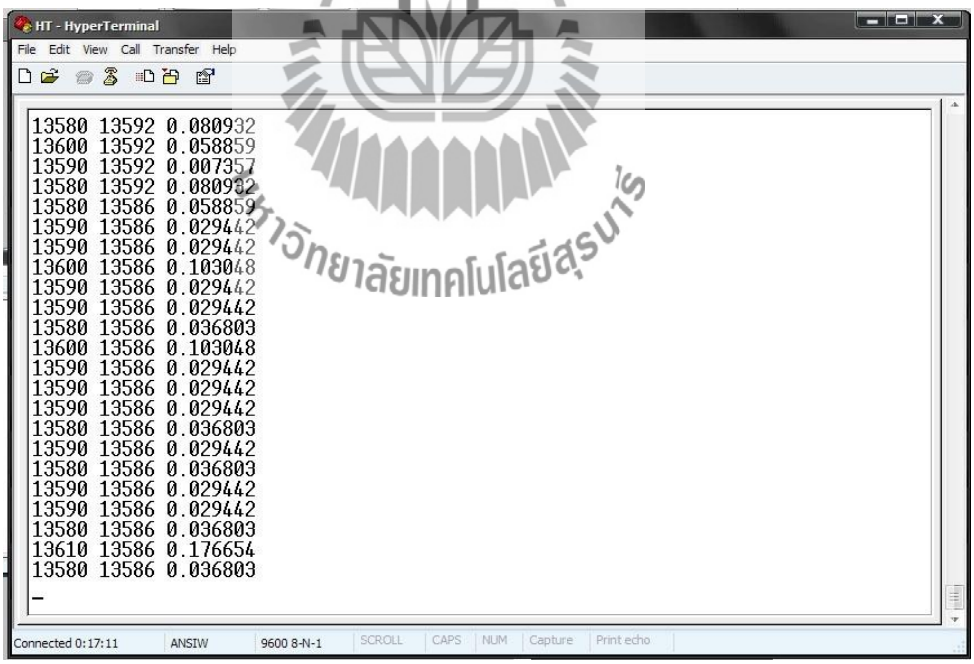


รูปที่ 4.39 โปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ

อ่านความถี่ที่สร้างมาจากเครื่อง Multi Function Generator ที่ 18 kHz

4.5.3 ทดสอบที่ความถี่ 14 kHz

รูปที่ 4.40 เครื่อง Universal Counter อ่านความถี่ได้ 14 kHz

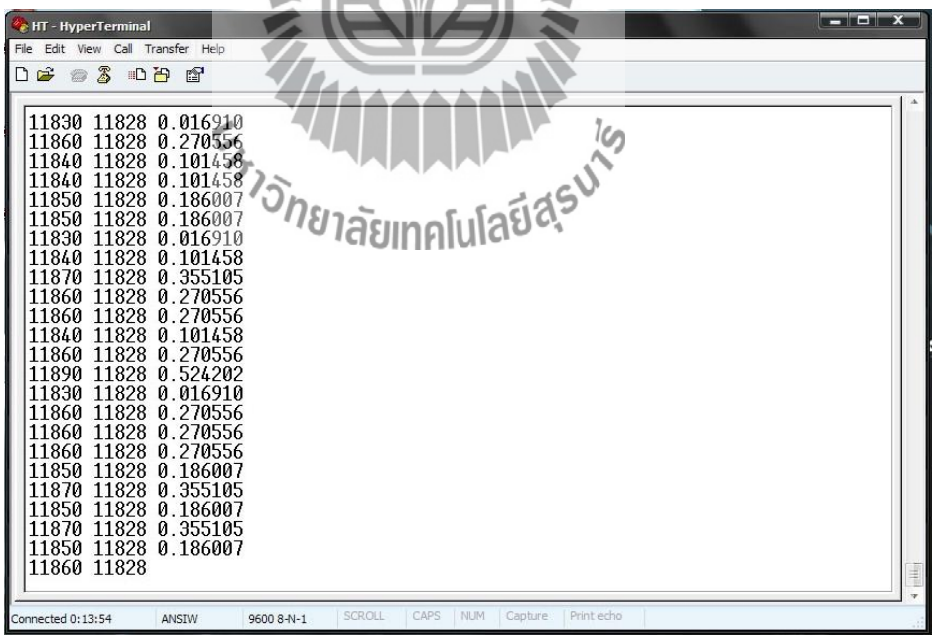


รูปที่ 4.41 โปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ

อ่านความถี่ที่สร้างมาจากเครื่อง Multi Function Generator ที่ 14 kHz

4.5.4 ทดสอบที่ความถี่ 12 kHz

รูปที่ 4.42 เครื่อง Universal Counter อ่านความถี่ได้ 12 kHz



รูปที่ 4.43 โปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ

อ่านความถี่ที่สร้างมาจากเครื่อง Multi Function Generator ที่ 12 kHz

จากการทดสอบ โปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ กับเครื่องมือวัดในห้องปฏิบัติการ ทำให้สามารถพิสูจน์ได้ว่า โปรแกรมนับความถี่สามารถอ่านความถี่ได้ตรงกับค่าของความถี่ที่ถูกสร้างออกมาจากเครื่องกำเนิดสัญญาณ (Multi Function Generator) ซึ่งค่าที่อ่านได้อาจจะมีความคลาดเคลื่อนเล็กน้อยเนื่องจากสภาพความสมบูรณ์ของอุปกรณ์การวัดในห้องปฏิบัติการไม่สมบูรณ์ 100 เปอร์เซ็นต์

บทที่ 5

สรุปผลการทดลองและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

โปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ เป็นส่วนหนึ่งของการตรวจวัดยานพาหนะ โดยใช้ลูปเหนี่ยวนำ ซึ่งจะนำความถี่ที่สามารถวัดได้ในการทดสอบมาคำนวณหาค่าอัตราการเปลี่ยนแปลงความถี่ (Delta) เพื่อใช้ในการตรวจสอบว่ามียานพาหนะผ่านเข้ามาที่ลูปเหนี่ยวนำแล้วหรือไม่ ซึ่งโปรแกรมมีการอัพเดตความถี่อ้างอิง (Frequency Base) ทุกๆ 30 วินาที เนื่องจากความถี่มีการเปลี่ยนแปลงตลอดเวลาตามสภาพแวดล้อมและสภาพภูมิอากาศเราจึงต้องมีการอัพเดตความถี่เพื่อให้ค่าความถี่มีความคลาดเคลื่อนน้อยที่สุดดังนั้นจากการทดสอบการใช้งานของโปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะสามารถสรุปได้ดังนี้

1. เราสามารถใช้ไมโครคอนโทรลเลอร์วัดความถี่ได้
2. โปรแกรมสามารถใช้ความถี่ที่วัดได้เป็นตัวตรวจจับว่าเป็นยานพาหนะได้จริง
3. โปรแกรมสามารถทำงานได้ทุกความถี่และสามารถปรับค่า Sensitivity ได้ในกรณีที่ผู้ใช้ต้องการให้โปรแกรมมีความไวในการตรวจจับยานพาหนะตามต้องการ

5.2 ปัญหาและอุปสรรค

1. ยานพาหนะที่เข้ามาในลูปเหนี่ยวนำถ้าเป็นรถจักยานยนต์จะสามารถวัดความถี่ได้น้อยมาก จึงทำให้ค่า Delta ที่คำนวณออกมามีค่าน้อยมากโปรแกรมอาจตัดสินใจว่าไม่มียานพาหนะอยู่ในลูปเหนี่ยวนำเลย
2. ยานพาหนะที่ขับผ่านนั้นต้องขับเข้ามาในลูปเหนี่ยวนำเท่านั้นถึงจะตรวจวัดความถี่ได้ดีเมื่อยานพาหนะบางคันขับออกด้านข้างลูป ค่าความถี่ที่วัดได้จะมีการเปลี่ยนแปลงที่น้อยถ้าเทียบกับขับเข้ามาในลูป เช่นเดียวกับรถจักยานยนต์

5.3 สิ่งที่ได้รับจากการทำโครงงาน

1. ได้รับความรู้เกี่ยวกับการใช้งานโปรงแกรม Keil uVision3
2. ได้รับความรู้เกี่ยวกับการตรวจสอบยานพาหนะ (Detected)
3. ได้รับความรู้เกี่ยวการใช้งานของบอร์ด Microcontroller Board ARM7
4. ได้เรียนรู้ระบบการคิด การรู้จักคิดวิเคราะห์ และการประยุกต์ใช้งาน
5. ได้เรียนรู้การทำงานร่วมกับผู้อื่น
6. สามารถนำความรู้ที่ได้จากการศึกษาทฤษฎีนำมาปฏิบัติได้จริง
7. สามารถนำความรู้และประสบการณ์ที่ได้จากการทำโครงงานไปประยุกต์ใช้ในการทำงานได้

เอกสารอ้างอิง

(1) นคร ภักดีชาติ , อรรถพล บุญยะโภคาม ,โอภาส ศิริครรชิตถาวร และ ชัยวัฒน์ ลิ้มพรวิไล (ม.ป.ป.). คู่มือทดลองไมโครคอนโทรลเลอร์ 32 บิตตระกูล ARM7 เบื้องต้นฉบับ LPC2148 บริษัท อิโนเวตีฟ เอ็กเพอริเมนต์ จำกัด

(2) บริษัทอีทีที จำกัด .คู่มือการใช้งานบอร์ด CP-JR ARM7 USB-LPC2148EXP [On line] จาก: http://www.ett.co.th/product/ARM/images/CP_JR_ARM7_LPC2148/MAN_CP_JR_ARM7_LPC2148.pdf

ประวัติผู้จัดทำ

นางสาวไพลิน โตกระโทก

เกิดเมื่อ วันที่ 24 มิถุนายน พ.ศ. 2532

ภูมิลำเนาอยู่ที่ ตำบลโชคชัย อำเภอโชคชัย จังหวัดนครราชสีมา

สำเร็จการศึกษาจากระดับมัธยมปลายจากโรงเรียนโชคชัยสามัคคี

อำเภอโชคชัย จังหวัดนครราชสีมา เมื่อปี พ.ศ. 2550

ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิศวกรรมโทรคมนาคม

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

นางสาวพิมพิสุทธิ์ พัฒนโสภณกุล

เกิดเมื่อวันที่ 4 มกราคม พ.ศ. 2534

ภูมิลำเนาอยู่ที่ ตำบลหมากแข้ง อำเภอเมือง จังหวัดอุดรธานี

สำเร็จการศึกษาระดับมัธยมปลายจากโรงเรียนอุดรพิทยานุกูล

อำเภอเมือง จังหวัดอุดรธานี เมื่อปี พ.ศ. 2550

ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิศวกรรมโทรคมนาคม

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

นางสาวภัสราภรณ์ วิจารณปัญญา

เกิดเมื่อวันที่ 24 มีนาคม พ.ศ. 2533

ภูมิลำเนาอยู่ที่ ตำบลมวกเหล็ก อำเภอมวกเหล็ก จังหวัดสระบุรี

สำเร็จการศึกษาระดับมัธยมปลายจากโรงเรียนมวกเหล็กวิทยา

อำเภอมวกเหล็ก จังหวัดสระบุรี เมื่อปี พ.ศ. 2550

ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิศวกรรมโทรคมนาคม

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

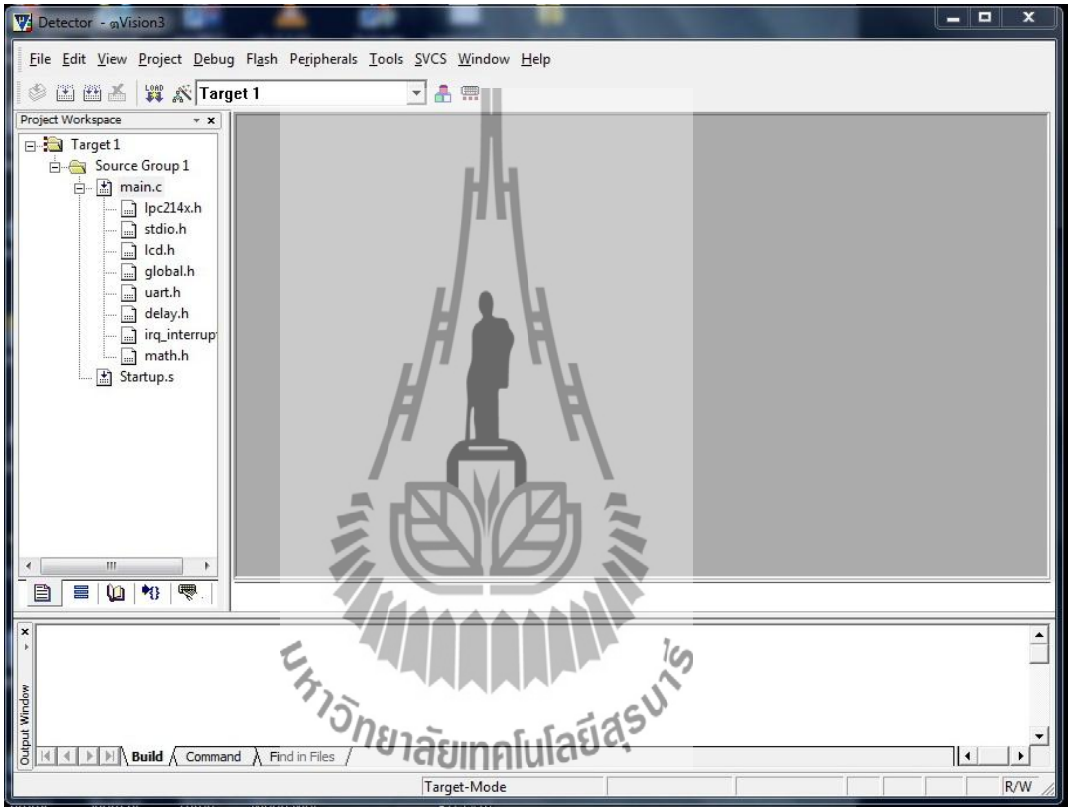
ภาคผนวก

(ก)

การใช้งานโปรแกรม

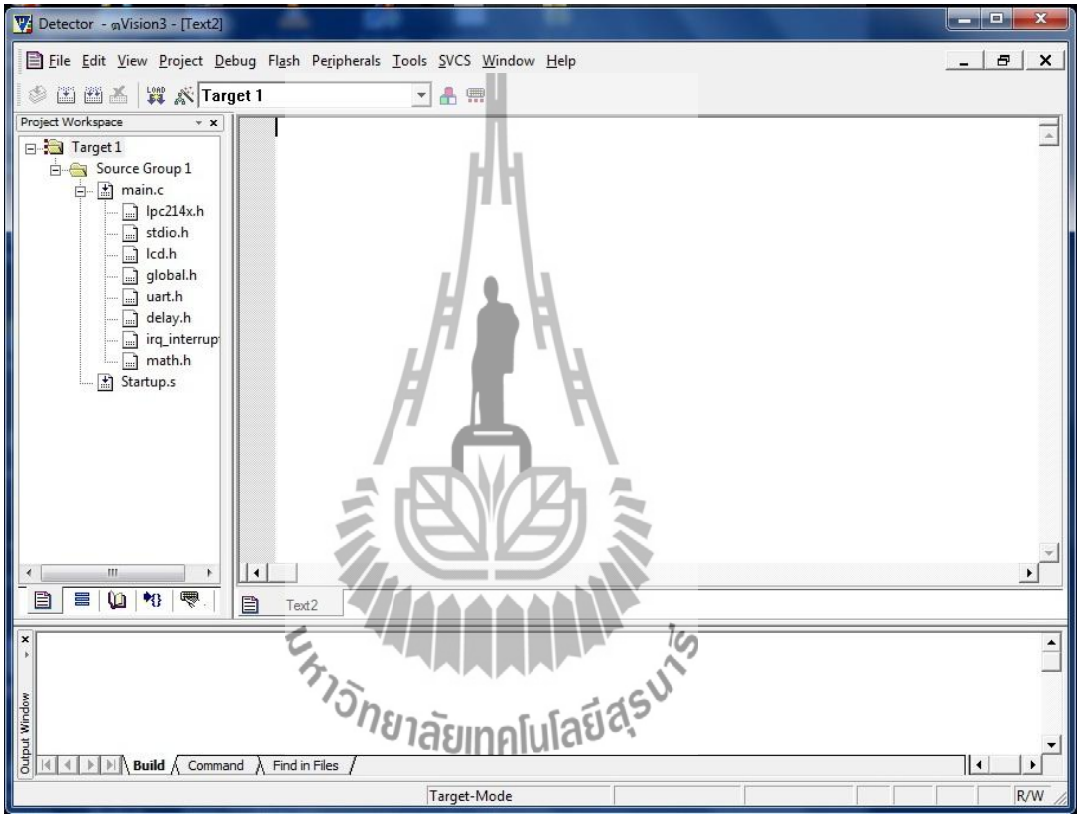
1. การใช้งานโปรแกรม Keil uVision 3

เมื่อเปิดโปรแกรม Keil uVision3 ซึ่งเป็นโปรแกรม Text Editor โดย Keil-CARM ใช้สำหรับการเขียนโปรแกรมที่เป็น Source Code ภาษาซี โดยจะมีลักษณะดังรูป



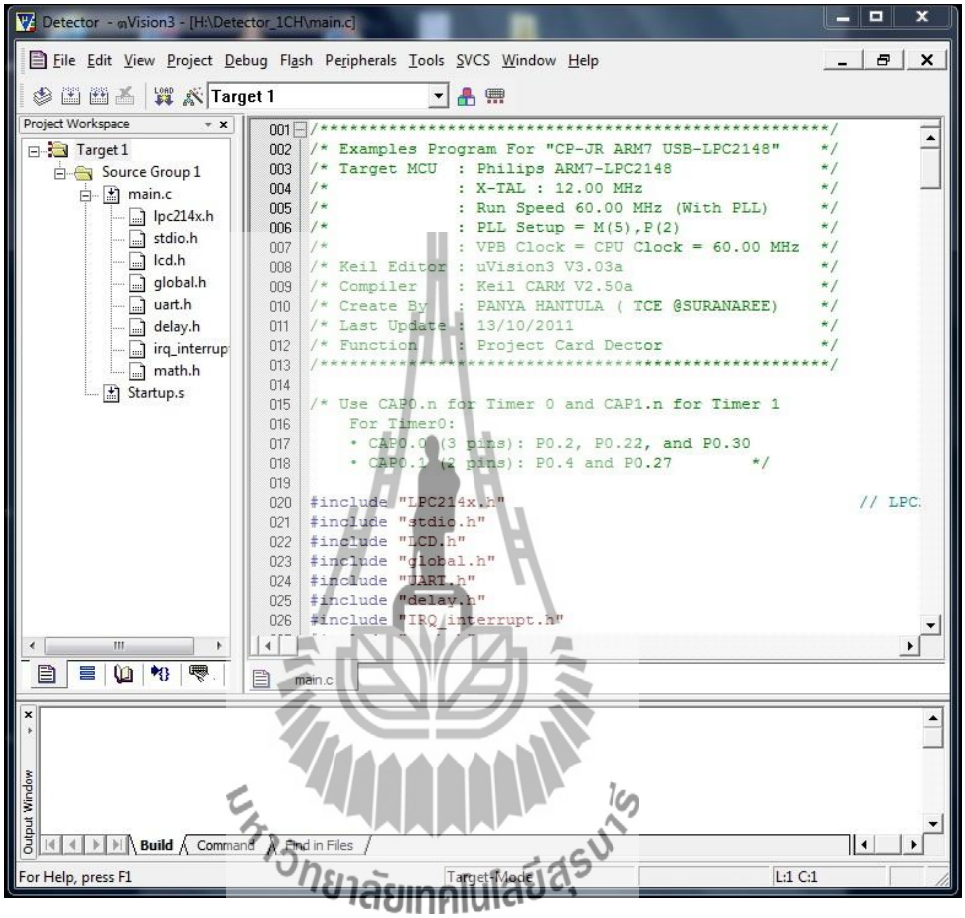
รูปที่ 1. หน้าต่างเริ่มต้นของโปรแกรม Keil uVision 3

หน้าต่างเริ่มต้นของตัวโปรแกรม โดยจะเริ่มต้นเขียน Source Code ภาษาซีโดยให้เลือกคลิกเมาส์ที่เมนูคำสั่ง File→ New… ซึ่งจะได้พื้นที่ในการเขียน Text File ขึ้นมา โดยครั้งแรกจะกำหนดชื่อตามค่า Defaultเป็น “Text2”ดังรูป



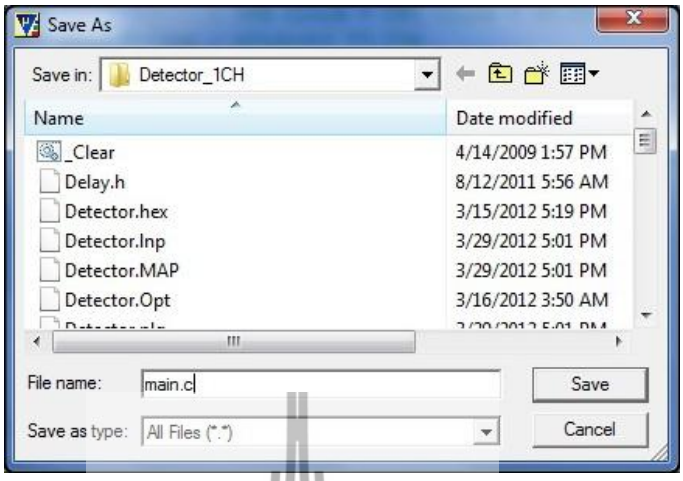
รูปที่ 2. เริ่มต้นของการเขียน Source Code

ขั้นตอนของการพิมพ์ Source Code ภาษาซีตามข้อกำหนดของ Keil-CARMในพื้นที่เขียนโปรแกรมตามที่ต้องการ

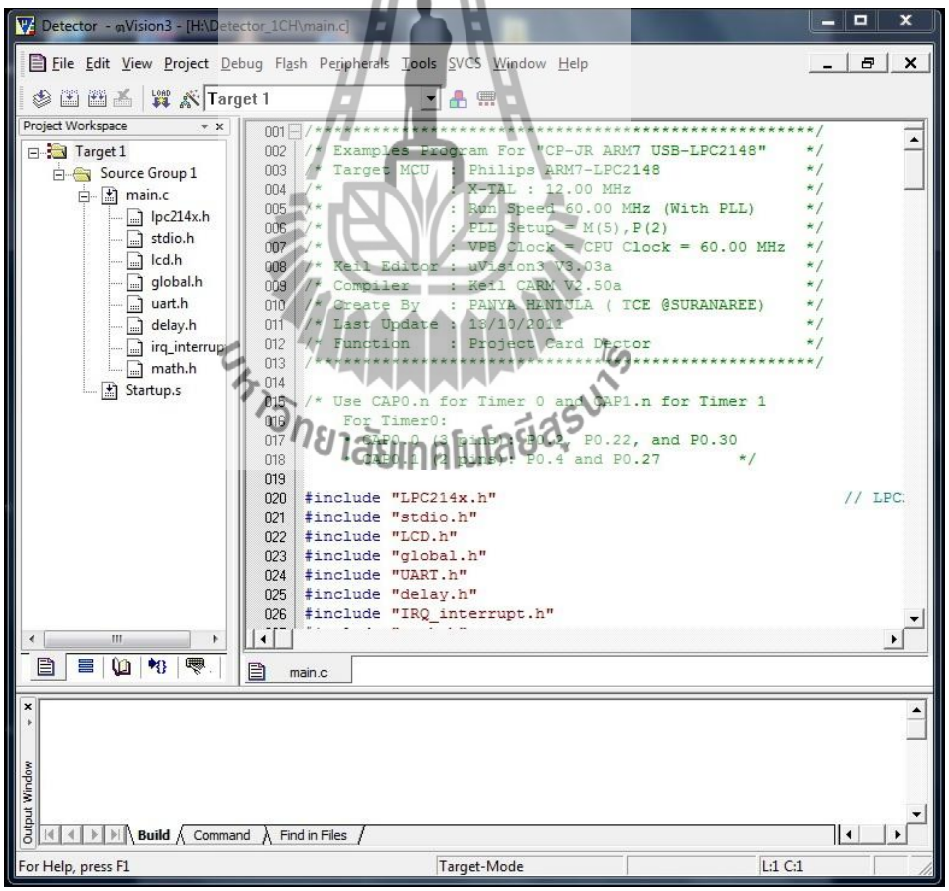


รูปที่ 3. หน้าต่างสำหรับพิมพ์ Source Code ภาษาซีลงในโปรแกรม Keil uVision3

หลังจากพิมพ์คำสั่งภาษาซีเสร็จเรียบร้อยแล้วตามที่ต้องการให้ทำการ Save ไฟล์ดังกล่าว โดยต้องกำหนดนามสกุลเป็น ".C" ในที่นี้ขอแนะนำให้สั่ง Save โดยใช้คำสั่ง File Save As… แล้วกำหนดชื่อและนามสกุลของไฟล์เป็น "main.c" ดังรูป

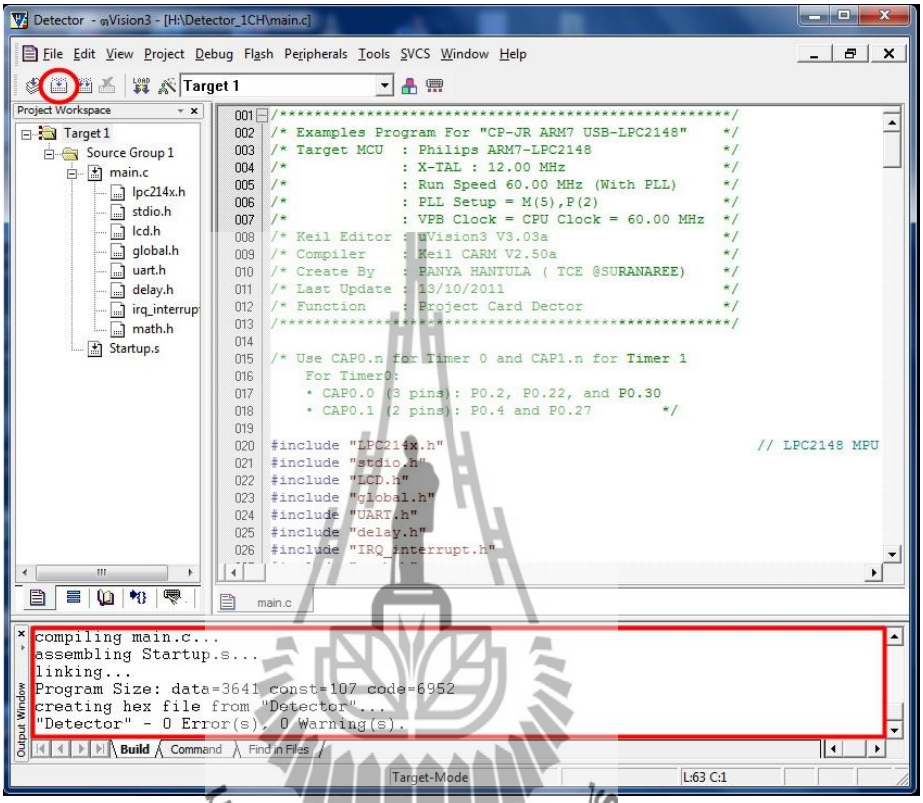


รูปที่ 4. การ Save ไฟล์โดยกำหนดนามสกุลเป็น ".C"



รูปที่ 5. หน้าต่างของโปรแกรม Keil uVision 3 เมื่อทำการ Save

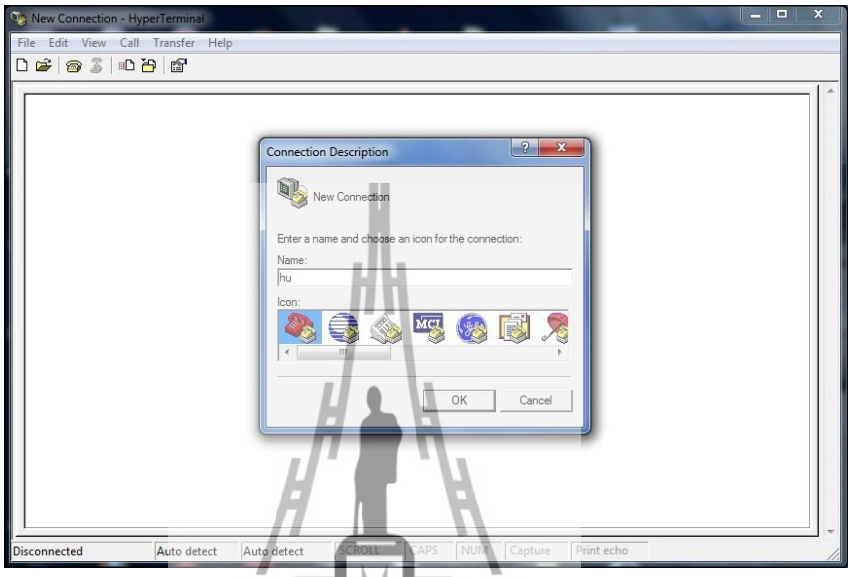
เมื่อกดปุ่ม Build Terget ตรงที่เป็นวงกลมสีแดงนั้นโปรแกรมจะ Run เอาผลออกมาจะเห็นว่าตัวโปรแกรมได้ผลที่ถูกต้องและไม่เกิดการผิดพลาดใดๆ (0 Error และ 0 warming) ดังภาพ



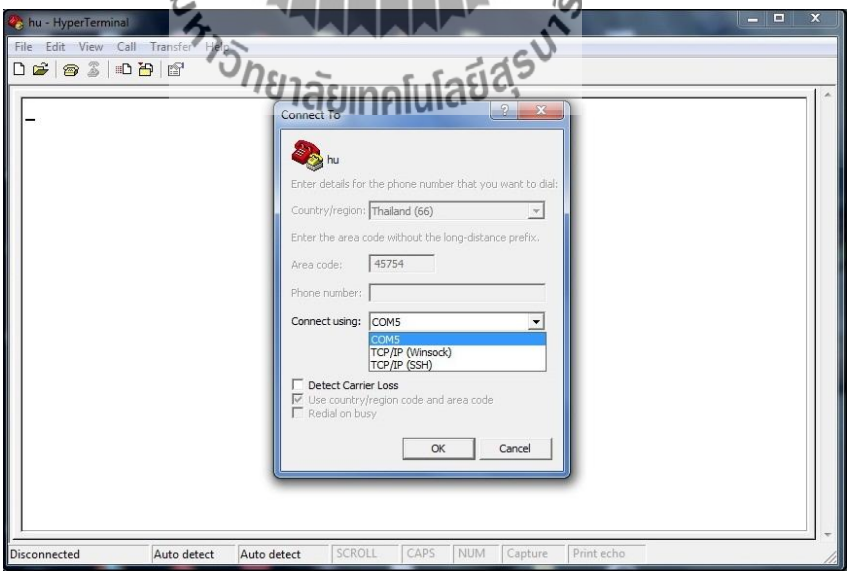
รูปที่ 6. ผลการรันโปรแกรม

2. การใช้งานโปรแกรม Hyper Terminal

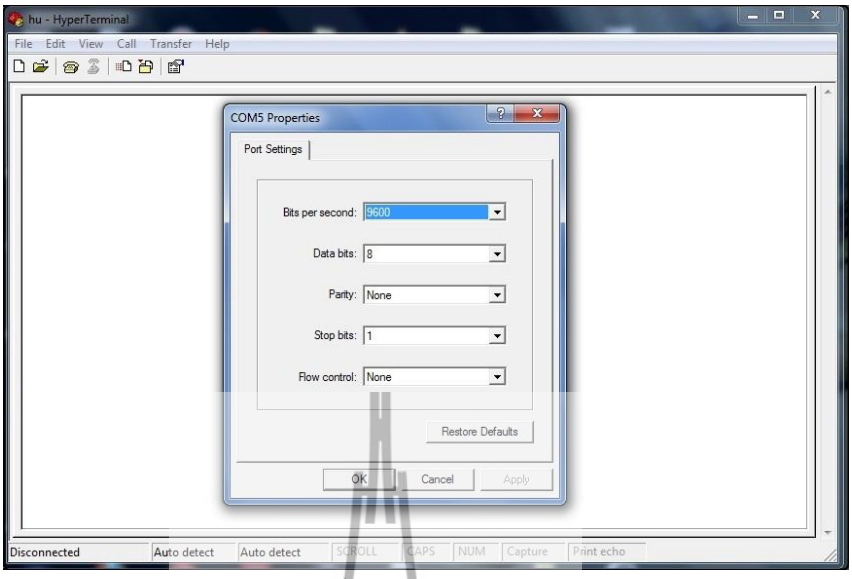
สั่งให้โปรแกรม ทำงานซึ่งจะปรากฏหน้าต่างดังรูปที่ 7 แล้วพิมพ์ชื่อลงในแถบกรอกข้อความ แล้วใช้เมาส์คลิก OK จะปรากฏในหน้ารูปที่ 8



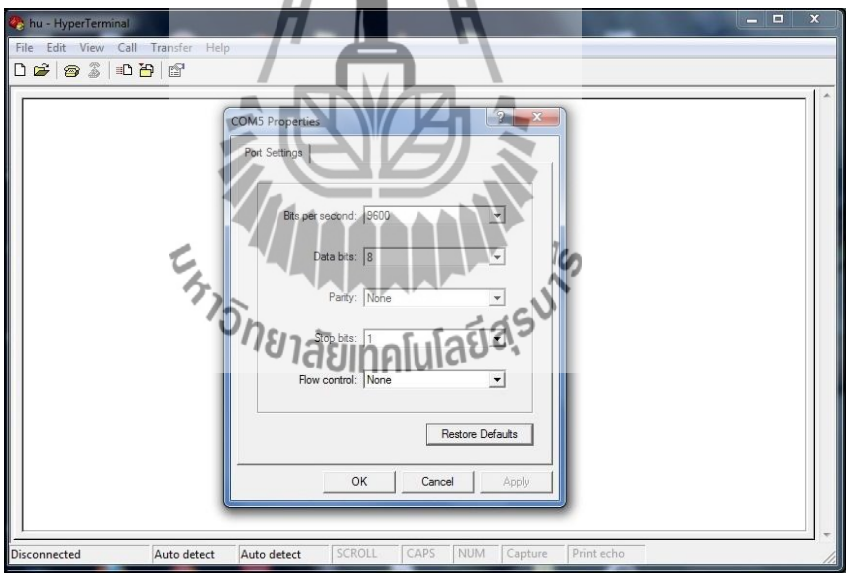
รูปที่ 7. หน้าต่างเริ่มต้นการทำงานของโปรแกรม Hyper Terminal



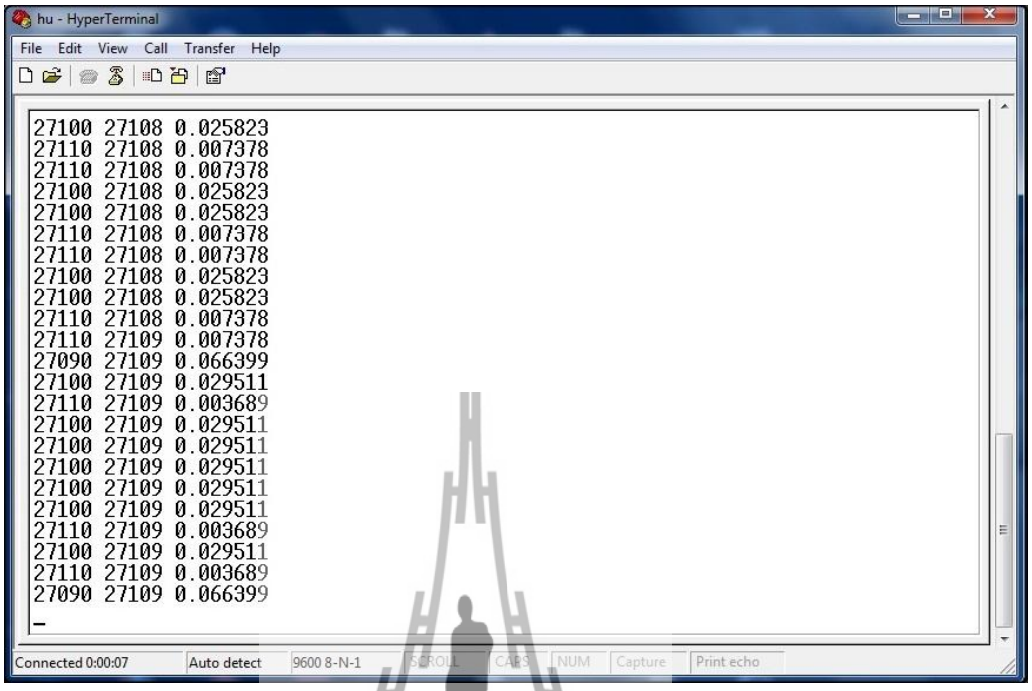
รูปที่ 8. การเลือก Comport ใช้งาน



รูปที่ 9. ทำการตั้งค่าที่ COM5 ให้เป็นค่าเริ่มต้น



รูปที่ 10. หน้าต่างแสดงภายหลังการตั้งค่าเริ่มต้นของ COM5

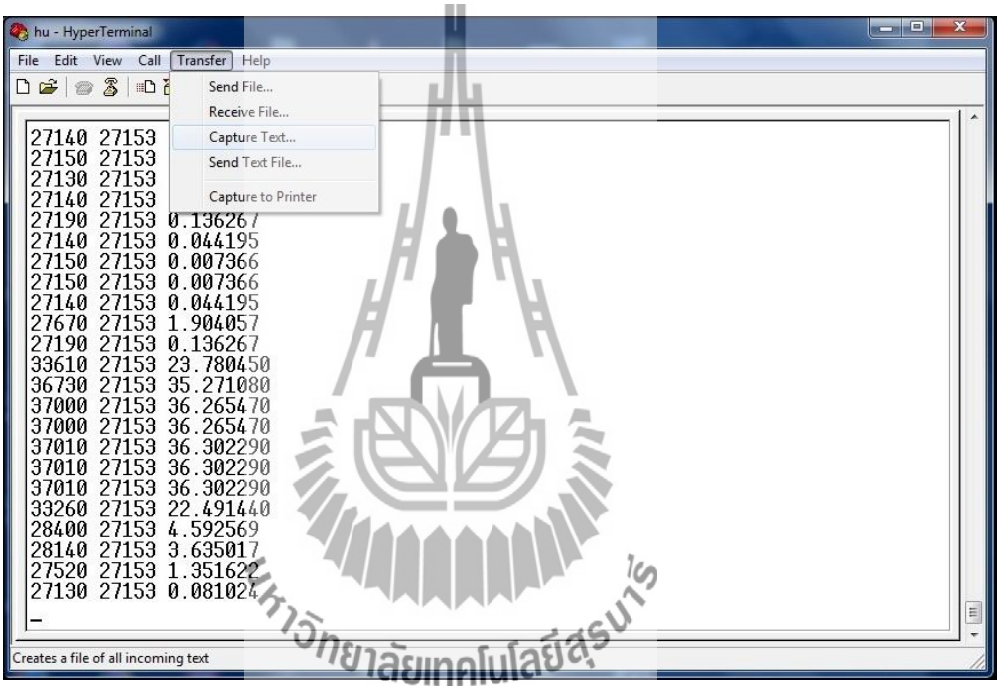


รูปที่ 11. โปรแกรม Hyper Terminal ทำการรันค่าความถี่

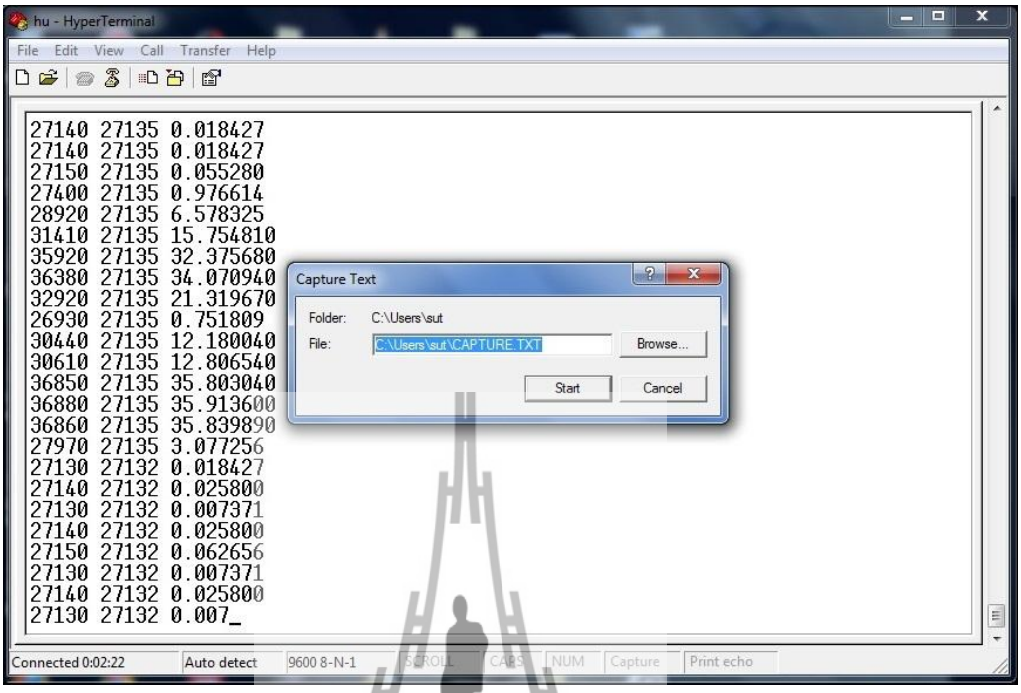
3. การ Capture ของโปรแกรม Hyper Terminal

เมื่อเราทำการรันโปรแกรมเพื่อเก็บความถี่จากโปรแกรม Hyper Terminal โดยที่ค่าความถี่และค่า Delta ที่ได้จากโปรแกรม Hyper Terminal ค่าจะรันไปเรื่อยๆ เพราะฉะนั้นเราจะเก็บค่า ผลการทดลอง ณ ช่วงเวลาหนึ่งเพื่อ นำค่า มาวิเคราะห์ผลการทดลอง โดย ทำการ Capture หน้าต่างของโปรแกรม Hyper Terminal ไว้ ซึ่งวิธีการ Capture ของโปรแกรม Hyper Terminal มีดังนี้

หลังจากรูปที่ 11 เมื่อโปรแกรม Hyper Terminal อ่านค่าความถี่



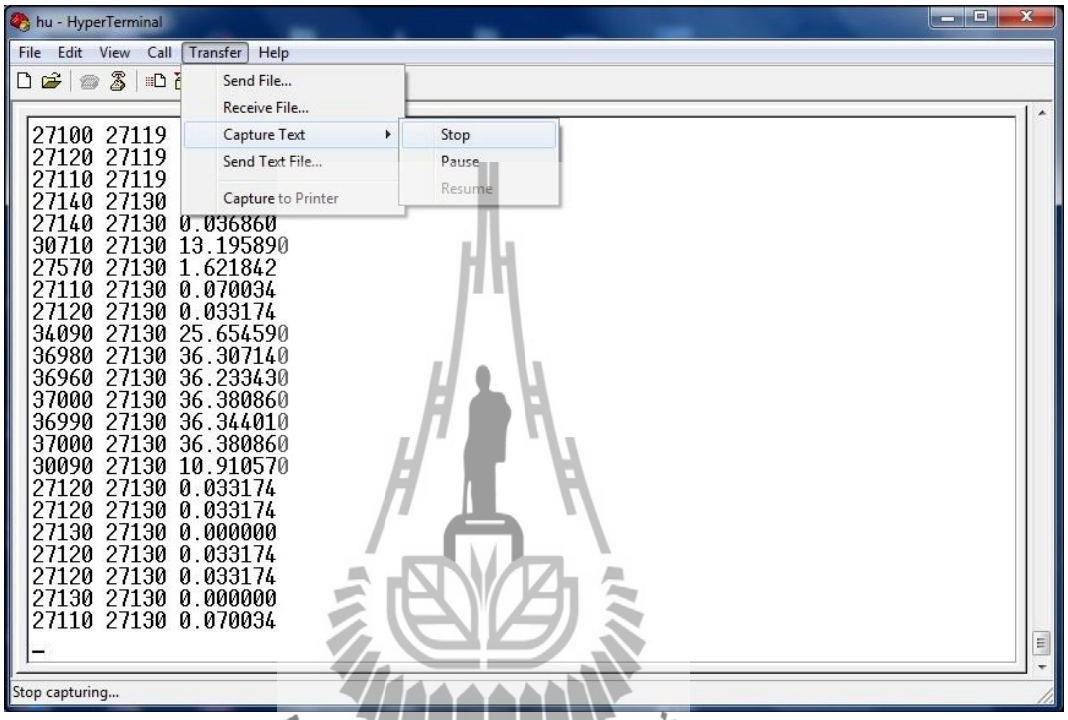
รูปที่ 12. เริ่มทำการ Capture ค่าความถี่



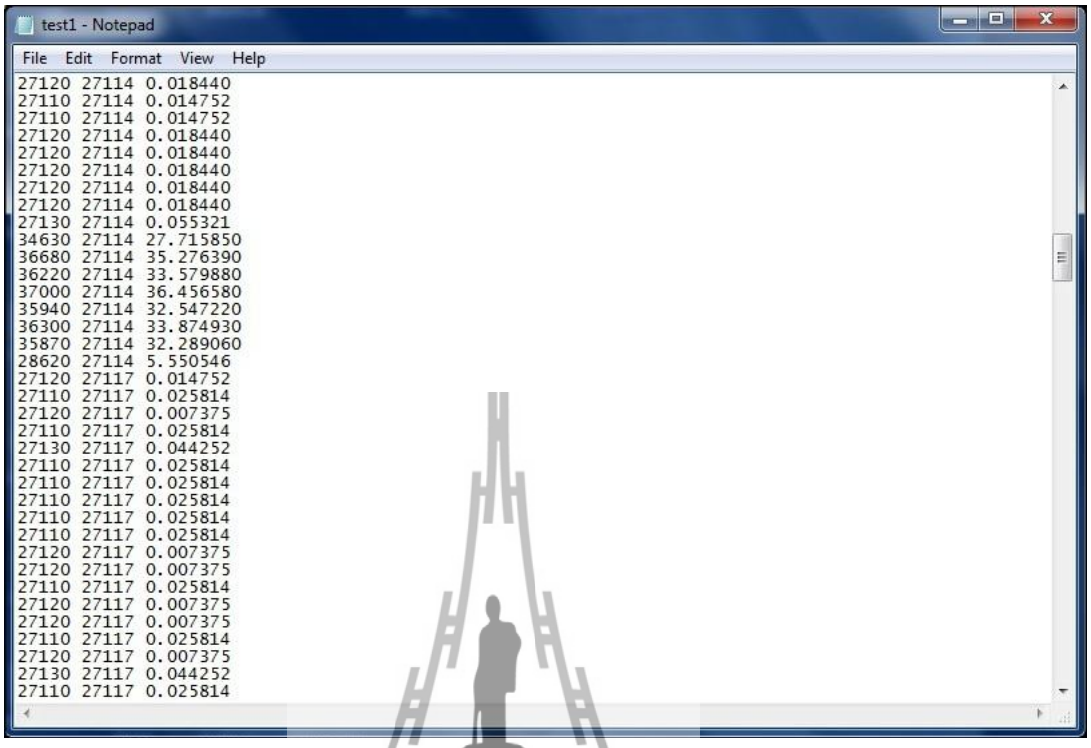
รูปที่ 13. ทำการเลือก Folder ที่เราต้องการจะเก็บงาน

รูปที่ 14. การตั้งชื่องาน

โปรแกรม Hyper Terminal จะทำการ Capture ค่าความถี่ไว้ เมื่อเราต้องการหยุดการ Capture ให้เราไปเลือกที่ Transfer อีกครั้งแล้วคลิกที่ Capture → Stop โปรแกรม Hyper Terminal ก็จะหยุดการ Capture ค่าความถี่



รูปที่ 15. หน้าต่างแสดงการหยุด Capture ค่าความถี่



รูปที่ 16. หน้าต่างแสดงค่าความถี่ที่ถูก Capture ไว้

ภาคผนวก

(ข)

โค้ดการทำงานของโปรแกรมนับความถี่สำหรับตรวจวัดยานพาหนะ

```
#include "LPC214x.h"                                    // LPC2148 MPU Register
#include "stdio.h"
#include "LCD.h"
#include "global.h"
#include "UART.h"
#include "delay.h"
#include "IRQ_interrupt.h"
#include "math.h"
#define Loop_OFF                IOSET0 |= 0x00000004;          // P0.2 is OFF
#define Loop_ON                 IOCLR0 |= 0x00000004;          // P0.2 is ON
#define LED1_OFF                IOSET0 |= 0x00000040;          // P0.6  OFF
#define LED1_ON                 IOCLR0 |= 0x00000040;          // P0.6  ON
#define LED2_OFF                IOSET0 |= 0x00000080;          // P0.7  OFF
#define LED2_ON                 IOCLR0 |= 0x00000080;          // P0.7  ON
#define TIMER0_IRQ 4
#define TIMER_IR_MR(n)                  (1 << (n))
#define TIMER_IR_CR(n)                  (1 << (n+4))
#define TIMER_TCR_CNTR_ENABLE           (1 << 0)
#define TIMER_TCR_CNTR_RESET            (1 << 1)
#define TIMER_CTCR_TIMER_MODE           (0)
#define TIMER_CTCR_CNTR_MODE_RISING     (1)
#define TIMER_CTCR_CNTR_MODE_FALLING    (2)
```

```
#define TIMER_CTCR_CNTR_MODE_BOTH       (3)
#define TIMER_CTCR_COUNT_CAP(n)         ((n) << 2)
#define TIMER_CCR_CAP_RE(n)             (1 << (n*3))
#define TIMER_CCR_CAP_FE(n)             (2 << (n*3))
#define TIMER_CCR_CAP_BE(n)             (3 << (n*3))
#define TIMER_CCR_CAP_I(n)              (4 << (n*3))
#define CHANNEL                         0
#define SW1     4
#define SW2     5
void delay_ms(long);
void delay_us(long);
void capture_freq (void);
void Freq_Update(void);
void Cal_Delta (void);
void Check_Present (void);
void Check_Sensitivity (void);
void isr_rtc (void) __irq;
void T0isr (void)           __irq;
void init_timer0();
float freq,Freq_Base,Freq_Real,Delta,Sensitivity_Loop;
int i,Timer_Update=0,Detec_Timer=0,Present=0;
int F_Update=1,Enable_Loop=1,buff_freq_base[600];
unsigned int Pulse;
```

```
//---------------------------------  Main Function ------------------------------------------//
int main(void)
{
        init();                                 // Initial
        uart0_init(9600);
        PINSEL0 &= 0x0000000F;          // P0.2 - P0.13 is GPIO
        IODIR0  |= 0x0000F0E8;          // P0.2 - P0.7 , P0.12-P0.13 is Output
        IOCLR0  |= 0x0000C0E8;          // P0.3 , P0.5 , P0.6 , P0.7, P0.14 , P0.15 is LOW
        IOCLR0  |= 0x00003000;          // P0.12 - P0.13 is Low
        PINSEL0 |= (2 << 4);            // Use P0.2 as capture channel 0
        PREINT = 0x00000392;            // Set RTC Prescaler for PCLK 30 MHz
        PREFRAC = 0x00004380;
        CIIR = 0x00000001;              // Enable seconds counter interrupt
        ALSEC = 0x00000003;             // Set alarm register for 3 seconds(match when xx:xx:03)
        AMR = 0x000000FE;               // Enable seconds alarm
        CCR  = 0;
        CCR |= 0x00000010;
        CCR |= 0x00000002;              // Reset   Clock
        CCR &= 0xFFFFFFFD;              // Release Reset
        CCR |= 0x00000001;              // Start RTC Clock
        VICVectAddr13 = (unsigned)isr_rtc;
        VICVectCntl13 = 0x20 | 13;      // Interrupt RTC
        VICIntEnable |= (1<<13);        // Enable RTC Interrupt
```

```
        Loop_OFF;
        LED1_OFF;
        LED2_OFF;
        Pulse = 0;                                          // Clear Counter
          Check_Sensitivity();
                printf("Frequency Counter (0.1sec)\n");    // Call Printf Function
                for(i=0;i<5;i++)                            // Read Frequecny
                capture_freq();
                Freq_Base = freq;                           // Update Base Frequency
                F_Update=0;
        while(1)
        {               capture_freq();
                        Check_Sensitivity();
                        Cal_Delta();
                        Check_Present();
        }
}
//================================
void Check_Sensitivity (void)
{
        if((!(input_0(SW1))) & (!(input_0(SW2))))
        {       Sensitivity_Loop = 0.5;
                Enable_Loop = 1;        }
        else if(!(input_0(SW1)))
```

```
        {       Sensitivity_Loop = 1;
                Enable_Loop = 1;        }
        else if (!(input_0(SW2)))
        {       Sensitivity_Loop = 2;
                Enable_Loop = 1;        }
        else
        {       Sensitivity_Loop = 0;
                Freq_Update();
                Enable_Loop = 0;        }
        return;
}
//==================== Cal_Delta ==============
void Check_Present (void)
{       // Loop 1
        if ((Enable_Loop)&&(Sensitivity_Loop > 0))
        {       if(Delta >= Sensitivity_Loop)                   // Delta 1
                {       Loop_ON;                                // Present Loop 1
                        LED1_ON;
                        Present = 1;                            // Start Detec_Timer 1
                }
                else
                {       Loop_OFF;                               // Absent Loop 1
                        LED1_OFF;
                        Detec_Timer = 0;
```

```
                        if (Timer_Update >= 30)          // 30 Sec Update Frequency Base
                        {       F_Update = 1;             // Start Update Frequency Loop 1
                                Freq_Update();            // Update Frequency
                                F_Update = 0;             // Not Update Frequency Loop 1
                                Timer_Update = 0;         }
                }
        }
        return;
}
//==================== Cal_Delta ================
void Cal_Delta (void)
{       if(Enable_Loop)
        {       Freq_Real = freq;
                Delta = (abs(Freq_Base - Freq_Real)/Freq_Base) * 100;
        }
        return;
}
//==================== Freq_Update =============
void Freq_Update(void)
{
        float temp_Freq,Delta_freq;
        LED2_ON;
        //printf("---------------------------------\r\n");
        capture_freq();
```

```
            temp_Freq = Freq_Base;
            Freq_Real = freq;
            Delta_freq = (abs(temp_Freq - Freq_Real)/temp_Freq) * 100;
            if (F_Update && Enable_Loop && (Delta_freq < 0.3))
                        Freq_Base = (freq+temp_Freq)/2;
            LED2_OFF;
            return;
}
//---------------------------- Interrupt service routine for UART1 ------------------------------//
void isr_rtc (void) __irq
{           if(ILR & 0x01)                                  // Check Interrupt block generate
   {        if (Present)
                        Detec_Timer++;
                        Timer_Update++;
                        ILR = 0x01;    // Clear interrupt flag
   }
   if(ILR & 0x02)
   {  ILR = 0x02;  }          // Clear interrupt flag
   VICVectAddr = 0;       // Acknowledge Interrupt

}
void capture_freq (void)
{
                        VICIntEnClr |= (1<<13);                         // Enable RTC Interrupt
                        // Initial External Interrupt-0(GPIO0.14)
```

```
                if(Enable_Loop)
                {
                        init_timer0();
                        Pulse = 0;
                        delay_ms(100);
                        freq = Pulse * 10;
                        Pulse = 0;
                }
                VICIntEnable |= (1<<13);                // Enable RTC Interrupt
                return;
}
void T0isr (void)           __irq
{
        Pulse++;
        //count = T0CR0;
        T0CR0 = 0;
        T0IR |= 0x00000001;
        VICVectAddr = 0x00000000;
        return;
}
//=======================
void init_timer0(void)
{
        T0TCR = TIMER_TCR_CNTR_RESET;           // Reset TC and PC registers
```

```
        T0PR = 0;                                                   // Set prescaler to 0
        T0CTCR = TIMER_CTCR_CNTR_MODE_RISING |
TIMER_CTCR_COUNT_CAP(CHANNEL);
T0CCR =  TIMER_CCR_CAP_I(CHANNEL) | TIMER_CCR_CAP_RE(CHANNEL);
        T0TCR = TIMER_TCR_CNTR_ENABLE; // Enable TC and PC registers
        VICVectCntl0 = (1 << 5) | TIMER0_IRQ; // Use slot 0 for TIMER0 and enable it
        VICVectAddr0 = (unsigned) T0isr; // Set the vector address for slot 0
        VICIntEnable = (1 << TIMER0_IRQ); // Enable the TIMER0 interrupt
}
//---------------------------------- Function delay ----------------------------------------//
void delay_ms(long ms)  // delay 1 ms per count @ CCLK 60 MHz
{       long i,j;
        for (i = 0; i < ms; i++ )
        for (j = 0; j < 6665; j++ );                                // 1 ms
}
//---------------------------------- Function delay ----------------------------------------//
void delay_us(long ms)  // delay 1 ms per count @ CCLK 60 MHz
{       long i,j;
        for (i = 0; i < ms; i++ )
        for (j = 0; j < 9; j++ );                                   // 1 ms
}
```