เครื่องควบคุมระบบป้องกันพนักงานรถจักรหมดสติของรถจักรดีเซลไฟฟ้า ของการรถไฟแห่งประเทศไทย

Vigilance for Diesel Electric Locomotive of State Railway of Thailand

นายปวิช เจียมจันทร์

นายอภิชัย ไชยประเทศ

นายภักดี งามสุทธิ

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีปทุม

ปีการศึกษา 2553

52EE207

หัวข้อโครงงาน	เครื่องควบคุมระบบป้องกันพนักงานรถจักรหมดสติของ
รถจักร	ดีเซลไฟฟ้าของการรถไฟแห่งประเทศไทย
โดย	นายปวิช เจียมจันทร์
	นายอภิชัย ไชยประเทศ
	นายภักดี งามสุทธิ
สาขา	วิศวกรรมไฟฟ้า
อาจารย์ที่ปรึกษา	อาจารย์เอกชัย ดีศิริ

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีปทุม อนุมัติให้นับโครงงานวิศวกรรมฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

…….…….…… หัวหน้าสาขาวิชา
วิศวกรรมไฟฟ้า
(ดร.นิมิต บุญภิรมย์)

……..……..……..……..…… อาจารย์ที่ปรึกษา
(อาจารย์เอกชัย ดีศิริ)

(วันที่…..….เดือน……..……พ.ศ.2554)

รหัสโครงงาน 52EE207

เครื่องควบคุมระบบป้องกันพนักงานรถจักรหมดสติของรถจักรดีเซลไฟฟ้าของการรถไฟฯ

Vigilance Safety Device for Diesel Electric Locomotive of State Railway of Thailand

บทคัดย่อ(Abstract)

โครงงานนี้ได้กล่าวถึงเครื่องควบคุมระบบป้องกันพนักงานรถจักรหมดสติของรถจักรดีเซลไฟฟ้าของการรถไฟแห่งประเทศไทย โดยประยุกต์ใช้ไมโครคอนโทรเลอร์ตระกูล MCS51 เป็นตัวตรวจจับสัญญาณอินพุตที่มาจากการรีเซทสวิตช์ต่างๆ ที่ห้องขับ และตรวจจับเวลาแล้วส่งสัญญาณไปควบคุมเอาต์พุตที่อุปกรณ์ส่งเสียงเตือน และห้ามล้อฉุกเฉิน เนื้อหาของโครงงานได้กล่าวถึงทฤษฎีของวงจรแบ่งแรงดัน ที่ใช้ในส่วนของอินพุตและเอาต์พุตที่มีแรงดันถึง 72Vdc วงจรสวิตช์ชิ่งเพาเวอร์ซัพพลาย ที่ใช้ในส่วนของชุดเพาเวอร์ซัพพลายเพื่อจ่ายไปให้กับชุดแปลงสัญญาณอินพุต ชุดควบคุม และชุดแปลงสัญญาณเอาต์พุต และวงจรไบอัสมอสเฟตแบบแบ่งแรงดัน ที่ใช้มอสเฟตขับโหลดที่เป็นแตรสัญญาณ กระดิ่งสัญญาณ และห้ามล้อฉุกเฉินแทนรีเลย์

กิตติกรรมประกาศ

การจัดทำโครงงานนี้สำเร็จผลลุล่วงไปได้ด้วยดีก็เนื่องจากความร่วมมือหลายฝ่ายด้วยกันทั้งได้รับคำแนะนำเกี่ยวกับการออกแบบจากอาจารย์ที่ปรึกษาโครงงานวิศวกรรม ท่านอาจารย์เอกชัย ดีศิริ และ นายประเวท สุวรรณเลิศ วิศวกรประจำฝ่ายแผงวงจรอิเล็กทรอนิกส์ นายอภิสิทธิ ศรีสาวงศ์ วิศวกรประจำฝ่ายค้นคว้า และวิจัย บริษัท ไทยโตชิบาอุตสาหกรรม จำกัด และได้รับความรู้ คำแนะนำจากอาจารย์ภาควิชาวิศวกรรมไฟฟ้าทุกท่านรวมถึงเจ้าหน้าของมหาวิทยาลัยศรีปทุม และผู้ให้กำเนิด บิดามารดา ที่ให้กำลังใจตลอดมา จึงใคร่ขอขอบคุณทุกๆ ท่านเป็นอย่างสูง ณ ที่นี้ด้วย สิ่งที่ดีและเป็นประโยชน์ของโครงงานนี้ ผู้จัดทำขอมอบให้กับผู้สนับสนุนโครงงานนี้ทุกๆ ท่าน สิ่งใดนี้ที่โครงงานมีความผิดพลาด ผู้จัดทำจะขอรับไว้แต่เพียงผู้เดียว

สารบัญ

	หน้า
บทคัดย่อ	ก
กิตติกรรมประกาศ	ข
สารบัญ	ค
สารบัญตาราง	จ
สารบัญภาพ	ฉ
บทที่ 1. บทนำ	1
1.1 ความสำคัญของปัญหา	1
1.2 วัตถุประสงค์ของโครงงาน	1
1.3 โครงสร้างของโครงงาน	2
1.4 ขอบเขตของโครงงาน	4
1.5 ประโยชน์ของโครงงาน	4
บทที่ 2. ทฤษฎีที่เกี่ยวข้อง	5
2.1 ระบบไมโครคอมพิวเตอร์	5
2.2 ภาษาซีสำหรับไมโครคอนโทรลเลอร์	14
2.3 ออปโต้คัปเปลอร์	28
2.4 วงจรแบ่งแรงดัน	30
2.5 สวิตชิ่งเพาเวอร์ซัพพลาย	32
2.6 เฟตชนิดออกไซด์ของโลหะ	37
บทที่ 3. การออกแบบ	42
3.1 การออกแบบชุดแปลงสัญญาณด้านอินพุต	42
3.2 การออกแบบชุดควบคุม	44
3.3 การคำนวณชุดแหล่งจ่าย	46
3.4 ชุดแสดงผล	47
3.5 ชุดแปลงสัญญาณเอาท์พุท	48

สารบัญ(ต่อ)

	หน้า
บทที่ 4. การทดลองและผลการทดลอง	54
4.1 ชุดแปลงสัญญาณอินพุตจาก	54
4.2 ชุดเพาเวอร์ซัพพลาย	55
4.3 ชุดควบคุม	57
4.4 ชุดแปลงสัญญาณเอาต์พุต	60
4.5 การทดลองติดตั้งที่รถจักร	62
บทที่ 5. สรุปและข้อเสนอแนะ	64
เอกสารอ้างอิง	65
ภาคผนวก	66

สารบัญตาราง

หน้า

ตารางที่ 2.1 หน้าที่ฟังก์ชันพิเศษต่างๆ ของขาพอร์ตที่ 3 7
ตารางที่ 2.2 ชนิดของตัวแปรต่างๆ 16
ตารางที่ 2.3 ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operator) 17
ตารางที่ 2.4 ตัวดำเนินการเปรียบเทียบและตรรกะลอจิก (Relational and Logic Operator) 18
ตารางที่ 2.5 ค่าความจริงของการแอนด์ (AND) เครื่องหมาย && 18
ตารางที่ 2.6 ค่าความจริงของการออ (OR) เครื่องหมาย || 19
ตารางที่ 2.7 ค่าความจริงของการน็อต (NOT) เครื่องหมาย ! 19
ตารางที่ 2.8 ตัวดำเนินการทางบิต (Bit Operator) 20
ตารางที่ 2.9 ลำดับการประมวลผลตัวดำเนินการ 21
ตารางที่ 4.1 ผลการทดลองในฟังก์ชั่น 8 วินาที 58
ตารางที่ 4.2 ผลการทดลองในฟังก์ชั่น 102 วินาที (เหยียบ Foot Switch ค้าง) 59

สารบัญภาพ

หน้า

ภาพที่ 1.1 Block diagram แสดงโครงสร้างการทำงาน 2
ภาพที่ 2.1 การจัดขาต่างๆ ของไอซีไมโครคอนโทรเลอร์ MCS - 51 6
ภาพที่ 2.2 ลักษณะวงจรของพอร์ตที่ 0, 1, 2, 3 8
ภาพที่ 2.3 วงจรขาพอร์ตที่ 0 9
ภาพที่ 2.4 วงจรขาพอร์ตที่ 1 9
ภาพที่ 2.5 วงจรขาพอร์ตที่ 2 10
ภาพที่ 2.6 วงจรขาพอร์ตที่ 3 10
ภาพที่ 2.7 การทำงานของพอร์ตลักษณะอินพุท 11
ภาพที่ 2.8 การทำงานของพอร์ตลักษณะเอาท์พุท 11
ภาพที่ 2.9 การทำงานของพอร์ตลักษณะเอาท์พุท 12
ภาพที่ 2.10 การจัดสรรพื้นที่ในหน่วยความจำข้อมูลภายใน 13
ภาพที่ 2.11 สัญลักษณ์ในการเขียน Flow Chart 22
ภาพที่ 2.12 สัญลักษณ์ของ Flow Chart ที่ใช้กับโปรแกรมควบคุมไมโครคอนโทรลเลอร์ 23
ภาพที่ 2.13 Flow Chart รูปแบบคำสั่ง if - else 25
ภาพที่ 2.14 Flow Chart รูปแบบคำสั่ง for 25
ภาพที่ 2.15 Flow Chart รูปแบบคำสั่ง while 26
ภาพที่ 2.16 Flow Chart รูปแบบคำสั่ง do 27
ภาพที่ 2.17 ฐานเวลาของไทเมอร์ 28
ภาพที่ 2.18 สัญลักษณ์อุปกรณ์เชื่อมต่อทางแสงชนิดต่างๆ 29
ภาพที่ 2.19 วงจรใช้งานออปโต้คัปเปลอร์เบื้องต้น 29
ภาพที่ 2.20 วงจรแบ่งแรงดันไฟฟ้าแบบไม่มีโหลด (Unloaded Voltage Divider) 30
ภาพที่ 2.21 วงจรแบ่งแรงดันแบบมีโหลด 31
ภาพที่ 2.22 องค์ประกอบพื้นฐานของสวิตชิ่งเพาเวอร์ซัพพลาย 32
ภาพที่ 2.23 วงจรสวิตชิ่งเรกูเลเตอร์แบบพื้นฐาน 33
ภาพที่ 2.24 ใช้ไดโอดและสวิตช์เดี่ยวทางเดียวที่ควบคุมโดย PWM 34
ภาพที่ 2.25 ใช้ทรานซิสเตอร์ที่ควบคุมโดย PWM 34
ภาพที่ 2.26 เป็นวงจรที่สร้างแรงดันเอาท์พุทที่เป็นบวก 37
ภาพที่ 2.27 เป็นวงจรที่ให้ค่า Vo เป็นลบ 37
ภาพที่ 2.28 โครงสร้างพื้นฐาน D - MOSFET 38

สารบัญภาพ(ต่อ)

หน้า

ภาพที่ 2.29 สัญลักษณ์ของ D - MOSFET 39
ภาพที่ 2.30 โครงสร้างและการทำงานของ E - MOSFET 39
ภาพที่ 2.31 วงจรไบอัส D - MOSFET แบบไบอัสด้วยแรงดันศูนย์ 40
ภาพที่ 2.32 วงจรไบอัส E - MOSFET 40
ภาพที่ 3.1 บล็อกไดอะแกรมแสดงส่วนต่างๆของระบบ 42
ภาพที่ 3.2 วงจรชุดแปลงสัญญาณอินพุต (Input Signal Converter Circuit) 43
ภาพที่ 3.3 วงจรชุดควบคุม (Controller Circuit) 44
ภาพที่ 3.4 วงจรชุดแหล่งจ่าย (Power Supply Circuit) 46
ภาพที่ 3.5 วงจรชุดแสดงผล (Display Circuit) 47
ภาพที่ 3.6 วงจรชุดแปลงสัญญาณเอาต์พุต (Output Signal Converter Circuit) 48
ภาพที่ 3.7 วงจรชุดแปลงสัญญาณเอาท์พุทของสัญญาณเสียงเตือนชุด 1 (Alert1) 49
ภาพที่ 3.8 Typical Output Characteristics ของ RDN100 49
ภาพที่ 3.9 แรงดันที่จุดต่างๆ ของวงจร 50
ภาพที่ 3.10 วงจรชุดแปลงสัญญาณเอาท์พุทของสัญญาณเสียงเตือนชุด 2 (Alert2) 51
ภาพที่ 3.11 แรงดันที่จุดต่างๆ ของวงจร 52
ภาพที่ 3.12 Vigilance Safety Circuit 53
ภาพที่ 3.13 PCB Assembly of Vigilance Safety Device 53
ภาพที่ 4.1 จุดที่วัดสัญญาณ Vout ของวงจรชุดแปลงสัญญาณอินพุท 54
ภาพที่ 4.2 ก. สัญญาณสภาวะที่ยังไม่กดสวิตช์ 55
ข. สัญญาณสภาวะที่กดสวิตช์
ภาพที่ 4.3 จุดที่วัดสัญญาณ Vout ของวงจรชุดเพาเวอร์ซัพพลาย 56
ภาพที่ 4.4 เป็นการวัดแรงดันอินพุทที่ใช้ในการทดลอง 56
ภาพที่ 4.5 สัญญาณแรงดันเอาท์พุท (Vout) ที่วัดได้ 56
ภาพที่ 4.6 การทดลองชุดควบคุม 58
ภาพที่ 4.7 ตำแหน่งการวัดเอาท์พุต 60
ภาพที่ 4.8 รถจักรดีเซลไฟฟ้าที่ใช้ทดสอบ 62
ภาพที่ 4.9 การติดตั้งที่ห้องเครื่องของรถจักรดีเซลไฟฟ้า 62
ภาพที่ 4.10 ทดสอบระบบ Vigilance ของรถจักรดีเซลไฟฟ้า 63
ภาพที่ 4.11 การแสดงผลขณะทำงาน 63

บทที่ 1
บทนำ

ด้วยในระบบป้องกันความปลอดภัยที่ติดตั้งใช้งานอยู่บนรถจักรดีเซล ของการรถไฟแห่งประเทศไทย มีระบบป้องกันต่างๆหลายระบบ เช่น ระบบป้องกันเครื่องยนต์ ระบบป้องกันอุปกรณ์ไฟฟ้าเสียหาย แต่มีอีกระบบหนึ่งที่เกี่ยวข้องกับความปลอดภัยเกี่ยวกับการเดินรถ นั่นคือระบบป้องกันพนักงานขับรถจักรหมดสติขณะทำขบวน ซึ่งมีความสำคัญมากแต่เนื่องจากรถจักรใช้งานมานานทำให้รถจักรบางคันระบบป้องกันนี้ชำรุดไม่สามารถทำงานได้ สำหรับโครงงานนี้เป็นการสร้างเครื่องควบคุมระบบป้องกันพนักงานขับรถจักรหมดสติ โดยใช้ไมโครคอนโทรลเลอร์ เพื่อทดแทนของเดิมที่ติดตั้งมากับรถจักร

1.1 ความเป็นมาของปัญหา

1. อุปกรณ์ที่ใช้ในการทำงาน ประกอบด้วย รีเลย์ และ รีเลย์หน่วงเวลา หลายตัวที่เริ่มชำรุด และ ปัจจุบัน ต้อง สั่งซื้อจากต่างประเทศซึ่งมีราคาแพง (โดยเฉพาะรถจักร ALSTHOM รุ่น 43 และ 44)

2. ระบบควบคุมติดตั้งอยู่บน แผงรีเลย์หลักของรถจักรซึ่ง มี รีเลย์ ของวงจรระบบอื่นติดตั้งอยู่ด้วยทำให้ยากแก่การตรวจสอบ

3. ผู้ที่ไม่หวังดี อาจต้องการไม่ให้ระบบทำงานได้โดยการถอดรีเลย์หรือเปลี่ยนรีเลย์ที่เกี่ยวข้องเพื่อไม่ให้ระบบทำงาน

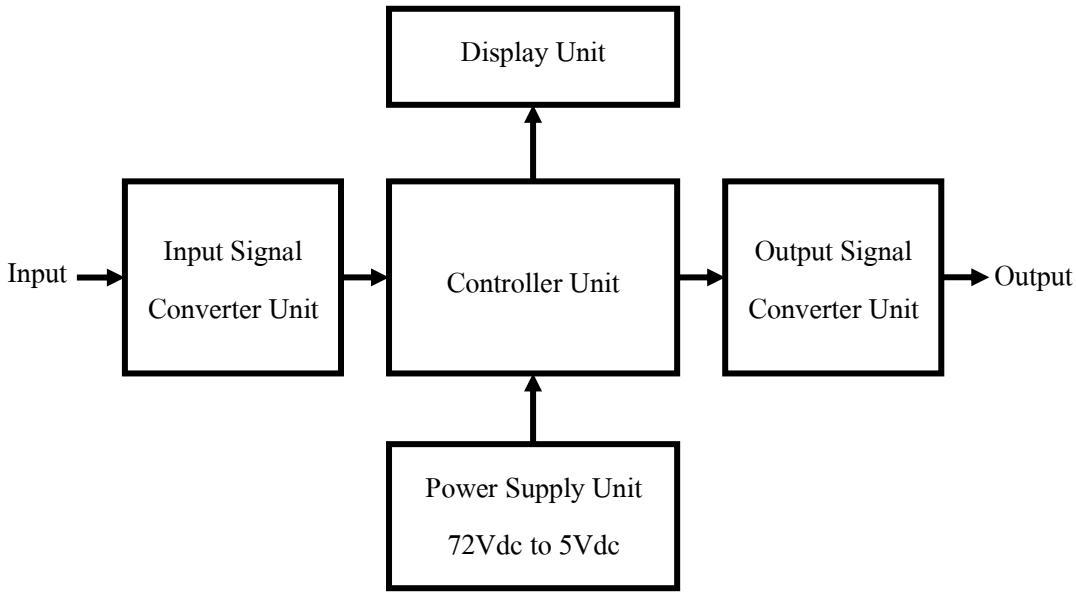
1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อสร้างเครื่องควบคุมระบบป้องกันพนักงานขับรถหมดสติขึ้นมาใหม่ เพื่อทดแทนของเดิมที่ติดตั้งมากับรถจักร

2. เพื่อประยุกต์การใช้ Microcontroller มาสร้างเครื่องควบคุมระบบป้องกันพนักงานขับรถหมดสติขึ้นมาใหม่เพื่อทดแทนของเดิมที่ติดตั้งมากับรถจักรที่ใช้กับการรถไฟแห่งประเทศไทย โดยยังคงไว้ซึ่งฟังก์ชั่นการใช้งานเหมือนเดิม

3. เพื่อศึกษาสมรรถนะของเครื่องควบคุมระบบป้องกันพนักงานขับรถหมดสติของรถจักรที่สร้างขึ้นใหม่ และเปรียบเทียบกับเครื่องตรวจสอบฟังก์ชั่นตัวเดิม

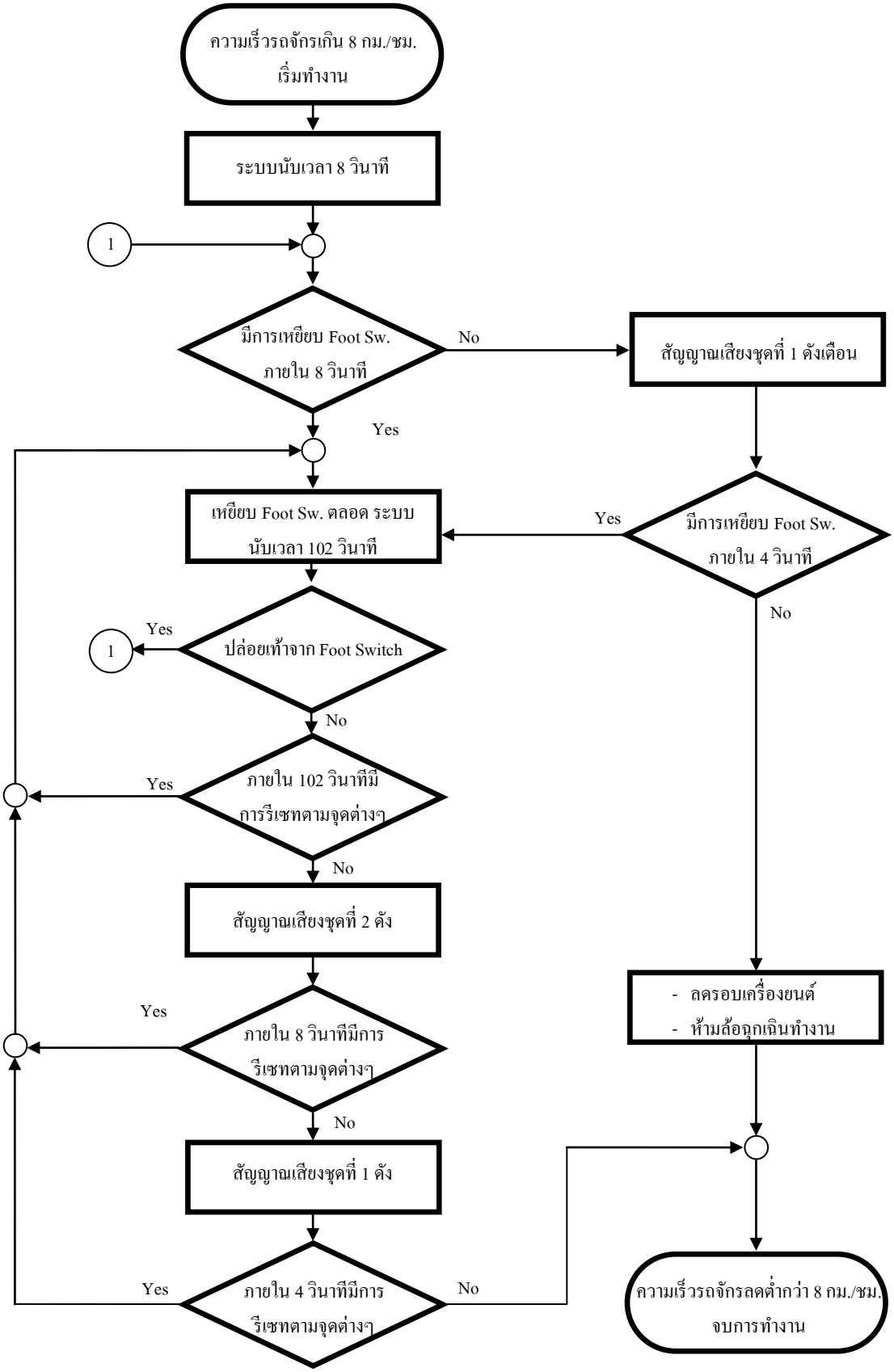
1.3 โครงสร้างของโครงงาน



ภาพที่1.1 Block diagram แสดงโครงสร้างการทำงาน

1. Input Signal Converter Unit: เป็นชุดแปลงสัญญาณอินพุทจากสัญญาณดิจิตอล 72V เป็นสัญญาณดิจิตอล 5V ซึ่งได้มาการตอบสนองตามจุดรีเซทต่างๆ ประกอบด้วย สวิตช์เท้าเหยียบ ปุ่มกดรีเซทหน้าห้องขับ คันเร่งรอบเครื่องยนต์ ปุ่มโรยทราย
2. Controller Unit: ชุดควบคุมการทำงานของโดยใช้ไมโครคอนโทรลเลอร์ เป็นตัวควบคุมและประมวลผล
3. Display Unit: ชุดแสดงผลการนับเวลาโดยใช้ 7 Segment แสดงสถานะ การนับ
4. Power Supply: ชุดวงจร DC to DC converter แปลงไฟกรแสตรง 72V เป็นไฟ 5V เพื่อไปเลี้ยงชุดวงจรควบคุม (Controller Unit)
5. Output Signal Converter Unit: เป็นชุดแปลงสัญญาณเอาท์พุทจากสัญญาณดิจิตอล 5V เป็นสัญญาณดิจิตอล 72V เพื่อไปจ่ายให้โหลดต่างๆของรถจักรดีเซลไฟฟ้าซึ่งได้แก่ อุปกรณ์ส่งเสียงเตือนชุดที่ 1 และชุดที่ 2 และรีเลย์ควบคุมห้ามล้อฉุกเฉิน

1.3.1 โฟล์วชาร์ทแสดงการทำงาน



1.4 ขอบเขตของโครงงาน

1. สร้างตัวแปลงสัญญาณดิจิตอลจาก 72V เป็นสัญญาณดิจิตอล 5V สำหรับรับค่าอินพุท
2. สร้างตัวแปลงสัญญาณดิจิตอลจาก 5V เป็นสัญญาณดิจิตอล 72V สำหรับส่งค่าเอาท์พุท
3. สร้างชุด Power Supply แปลงไฟจาก 72V เป็น 5V เพื่อเลี้ยงวงจรชุดควบคุม, ชุดแปลงอินพุท, ชุดแปลงสัญญาณเอาท์พุท และชุดแสดงผล
4. สร้างชุดควบคุม ที่สามารถรับอินพุทได้ 5 อินพุท และควบคุมเอาท์พุท 3 เอาท์พุท และจอแสดงสถานะ
5. สร้างโปรแกรมเพื่อใช้งานกับไมโครคอนโทรลเลอร์

1.5 ประโยชน์ของโครงงาน

1. ทดแทนอุปกรณ์เดิมที่ใช้งานมานานซึ่งเริ่มชำรุด
2. สามารถประยุกต์ใช้งานกับรถจักรรุ่นอื่นในการรถไฟแห่งประเทศไทยได้
3. ลดค่าใช้จ่ายในการซื้ออุปกรณ์จากต่างประเทศ
4. ป้องกันผู้ใช้งานฝืนระบบโดยการสลับรีเลย์หรือยกเลิกการใช้งานโดยการถอดรีเลย์ออก
5. ป้องกันผู้ที่ไม่มีหน้าที่เกี่ยวข้องทำการปรับแต่งค่าเวลาต่างๆ

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ระบบไมโครคอมพิวเตอร์ [1] [2]

ปัจจุบันไมโครคอนโทรเลอร์ ได้เข้ามามีบทบาทมากกับการควบคุมในงานอุตสาหกรรม การพัฒนาอุปกรณ์สารกึ่งตัวนำอย่างต่อเนื่อง ผนวกกับเทคโนโลยีการผลิตที่วิวัฒนาการมาเป็นหน่วยไมครอนนำไปสู่ การสร้าง Chip ที่มีขนาดเล็กและมาก ด้วยคุณภาพไมโครคอนโทรเลอร์ 8051 เป็น Chip ที่นิยมมากตัวหนึ่ง ที่ใช้ในงานควบคุมเนื่องจากความสามารถที่สูงและง่ายต่อการใช้งาน ทั้งนี้ต้องขึ้นอยู่กับผู้ใช้งานที่มีวิธีการ อย่างไร ในการดึงความสามารถของ Chip มาให้ใช้เต็มที่ เพื่อให้เกิดประสิทธิภาพสูงสุด

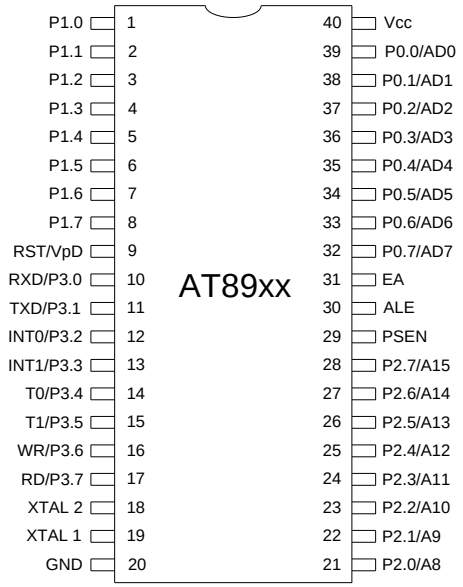
2.1.1 โครงสร้างไมโครคอนโทรลเลอร์ตระกูล 8051

ไอซีตระกูล 8051 มีด้วยกันหลากหลายเบอร์ ขึ้นอยู่กับโครงสร้างภายในของไอซีแต่ละเบอร์ บางเบอร์มีหน่วยความจำโปรแกรม 4 Kbyte (Fash) บางตัวมีขนาด 8 Kbyte (Fash) ทั้งนี้ก็ต้องศึกษา รายละเอียดจากคู่มือของตัวไอซีแต่ละเบอร์ด้วย แต่คุณสมบัติที่สำคัญของไมโครคอนโทรลเลอร์ ที่นิยมนำมาใช้งาน เช่น เบอร์AT89C51 (บริษัท Atmel) จะมีคุณสมบัติดังนี้

- หน่วยประมวลผลการ (CPU) มีขนาด 8 บิต
- มีหน่วยความจำโปรแกรม (ROM) แบบ flash ขนาด 4 Kbyte
- มีหน่วยความจำข้อมูล (RAM) ขนาด 128 byte
- มีพอร์ตในการอินพุท/เอาท์พุท จำนวน 4 พอร์ต (Porto - 3)
- มีตัวฟังก์ชัน Timer/Counter จำนวน 2 ตัว Timer0, Timer1
- สามารถอินเตอร์รัปต์ได้ 5 - 6 แหล่ง
- มีวงจรกำเนิดสัญญาณนาฬิกาภายในตัวไอซี
- มีพอร์ตอนุกรมที่สามารถสื่อสารรับส่งข้อมูลแบบ Full Duplex
- สามารถขยายหน่วยความจำโปรแกรมภายนอกได้ 64 Kbyte
- สามารถขยายหน่วยความจำข้อมูลภายในอกได้ 64 Kbyte

2.1.2 การจัดขาของไมโครคอนโทรลเลอร์ตระกูล 8051

ไมโครคอนโทรลเลอร์ตระกูล 8051 โครงสร้างของตัวไอซีนั้นเป็นแบบ DIP ขนาด 40 ขา โดยขาต่างๆ จะทำหน้าที่เป็นขาควบคุม, ขาพอร์ตอินพุทเอาท์พุท, ขารีเซต ดังภาพที่ 2.1 ต่อไปนี้



ภาพที่ 2.1 การจัดขาต่างๆ ของไอซีไมโครคอนโทรเลอร์ MCS - 51

สามารถอธิบายหน้าที่ของขาต่างๆ ในไมโครคอนโทรลเลอร์ได้ ดังนี้คือ

ขา RST (Reset)

เป็นขาที่ทำหน้าที่ในการกำหนดให้ไมโครคอนโทรลเลอร์เข้าสู่ภาวะเริ่มต้นการทำงานใหม่โดยปกติที่ขารีเซตจะมีสภาวะทางลอจิกเป็น 0 และถ้าป้อนสภาวะทางลอจิก 1 เป็นเวลาไม่น้อยกว่า 2 แมชชินไซเกิล ก็จะทำให้ไมโครคอนโทรลเลอร์เกิดสภาวะรีเซตขึ้น

ขา $\overline{\text{PSEN}}$ (Program Store Enable)

เป็นขาที่ทำหน้าที่ส่งสัญญาณ เมื่อไมโครคอนโทรลเลอร์ต้องการจะอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก

ขา EA (External Access)

เป็นขาที่ทำหน้าที่เลือกใช้งานหน่วยความจำโปรแกรมว่าจะเลือกใช้งานหน่วยความจำโปรแกรมภายนอก หรือหน่วยความจำโปรแกรมภายในไมโครคอนโทรลเลอร์ โดยการกำหนดสภาวะทางลอจิกให้ขา EA = “0” จะเป็นการเลือกใช้งานหน่วยความจำโปรแกรมภายนอก และเมื่อกำหนดให้ขา EA = “1” จะเป็นการเลือกใช้งานหน่วยความจำโปรแกรมภายใน

ขา ALE (Address Latch Enable)

เป็นขาที่ทำหน้าที่ควบคุมการ Latch ตำแหน่งของข้อมูล (Address Bus, A0 - A7) เมื่อทำการ ติดต่อกับหน่วยความจำโปรแกรมภายนอก

ขา VCC/VSS

เป็นขาที่ทำหน้าที่ต่อแหล่งจ่ายไฟเลี้ยงไอซี + 5V และเป็นขากราวนด์ตามลำดับ

ขา XTAL1, XTAL2

เป็นขาที่ทำหน้าที่ต่อเชื่อมกับตัวผลิตสัญญาณนาฬิกาให้กับตัวไอซี โดยปกติจะต่อเชื่อมกับ คริสตอลขนาด 11.059 MHz และมีคาปาซิสเตอร์ 33pF ป้องกันความถี่สูง

ขา Port 0 (P0.0 - P0.7)

เป็นขาที่ทำหน้าที่อินพุทและเอาท์พุทกับอุปกรณ์ภายนอก และยังทำหน้าที่เป็นขา Address Bus (A0 - A7) และData Bus (D0 - D7) เพื่อการเชื่อมต่อกับหน่วยความจำภายนอก

ขา Port 2 (P2.0 - P2.7)

เป็นขาที่ทำหน้าที่อินพุทและเอาท์พุทกับอุปกรณ์ภายนอก และยังทำหน้าที่ Address Bus

ขา Port 1 (P1.0 - P1.7)

เป็นขาที่ทำหน้าที่อินพุทและเอาท์พุทกับอุปกรณ์ภายนอก โดยสามารถอ้างอิงทีละบิตได้

ขา Port 3 (P3.0 - P3.7)

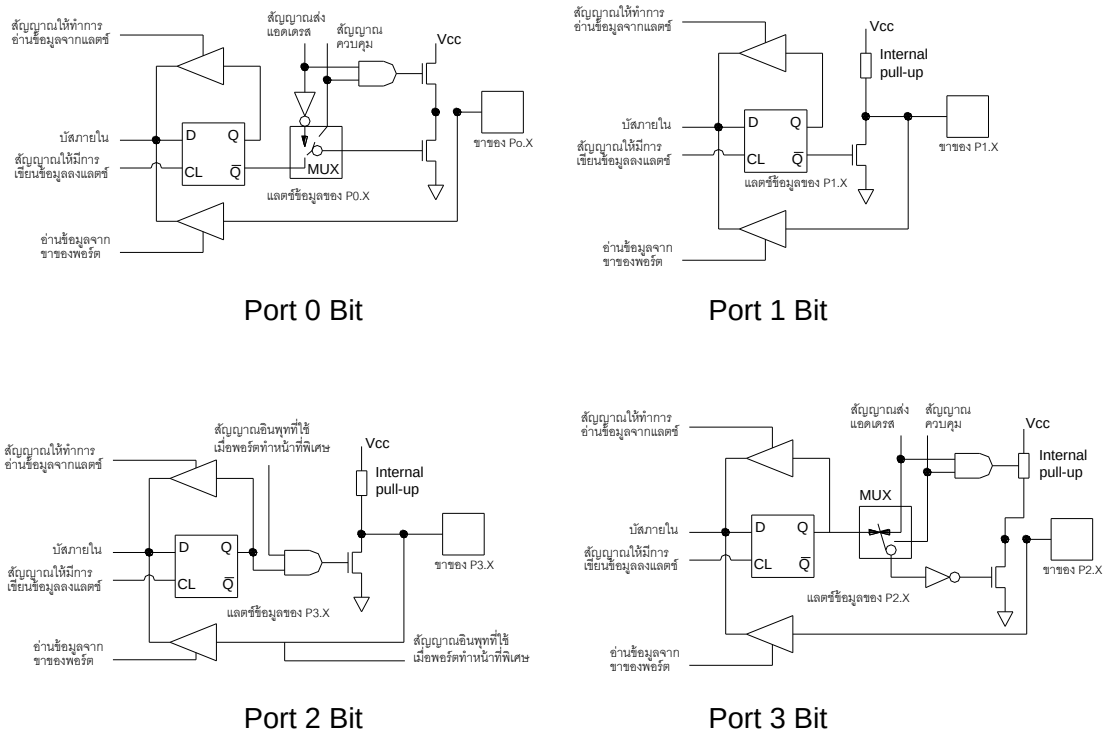
เป็นขาที่ทำหน้าที่อินพุทและเอาท์พุทกับอุปกรณ์ภายนอก และยังทำหน้าที่ฟังก์ชันพิเศษในการติดต่อควบคุมต่างๆ ดังตารางที่ 1.2 แสดงหน้าที่ฟังก์ชันพิเศษต่างๆ ของขาพอร์ต

ตารางที่ 2.1 หน้าที่ฟังก์ชันพิเศษต่างๆ ของขาพอร์ตที่ 3

ขาพอร์ต 3	**ชื่อ**	**หน้าที่การทำงาน**
P3.0	RXD	รับข้อมูลแบบอนุกรม
P3.1	TXD	ส่งข้อมูลแบบอนุกรม
P3.2	INTO	อินเตอร์รัปต์ภายนอกหมายเลข 0
P3.3	INT1	อินเตอร์รัปต์ภายนอกหมายเลข 1
P3.4	T0	Timer/counter ตัวที่ 0
P3.5	T1	Timer/counter ตัวที่ 1
P3.6	WR	สัญญาณในการเขียนข้อมูลหน่วยความจำภายนอก
P3.7	RD	สัญญาณในการอ่านข้อมูลหน่วยความจำภายนอก

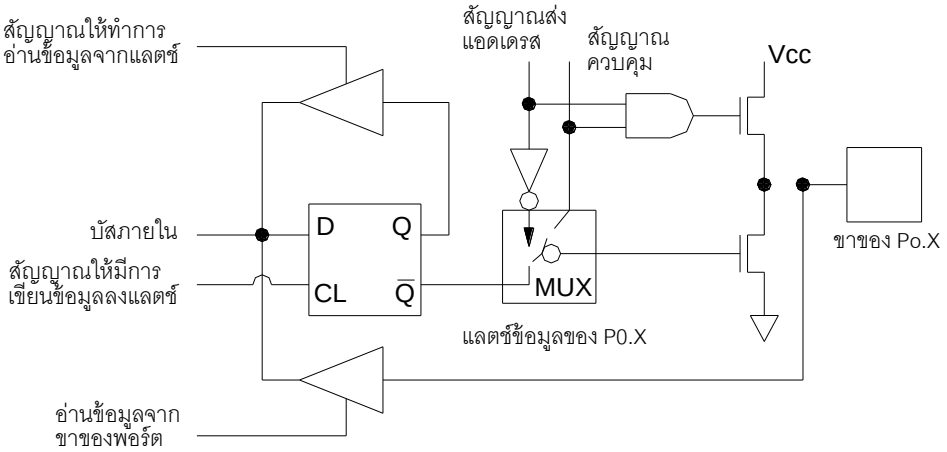
2.1.3 โครงสร้างและการทำงานของพอร์ต

ไมโครคอนโทรลเลอร์ MCS 51 จะมีพอร์ตที่ใช้งานอยู่ 4 พอร์ต คือ พอร์ตที่ 0 - 3 แต่ละพอร์ต มีขนาด 8 บิต ซึ่งแต่ละพอร์ตเป็นพอร์ตแบบติดต่อได้สองทิศทาง คือ สามารถเป็นได้ทั้งอินพุทและ เอาท์พุท จากภาพที่ 2.2 ลักษณะโครงสร้างของพอร์ตต่างๆ จะเห็นได้ว่าแต่ละพอร์ตจะมีวงจรแลตช์ (Latch) และวงจรขับบัฟเฟอร์อินพุท สังเกตจากภาพที่ 2.2



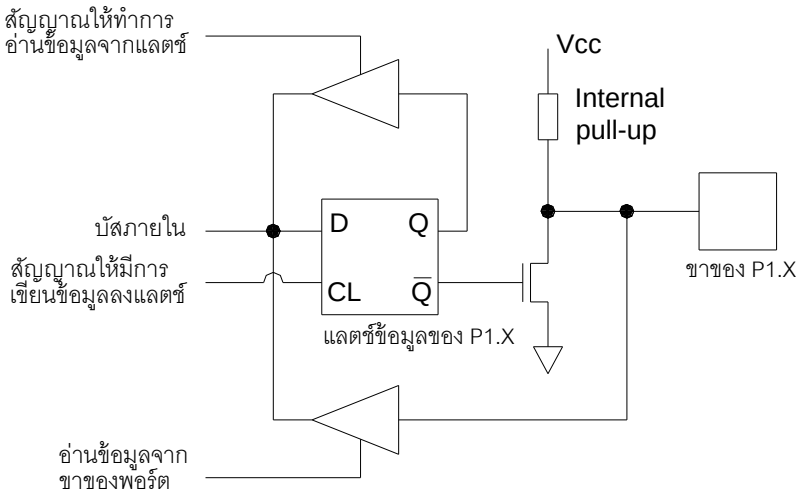
ภาพที่ 2.2 แสดงลักษณะวงจรของพอร์ตที่ 0, 1, 2, 3

พอร์ตในไมโครคอนโทรลเลอร์ **MCS 51** ทั้ง 4 พอร์ตจะมีวงจรมัลติเพล็กซ์สำหรับกำหนด การทำงานพอร์ตว่าจะเป็นอินพุท/เอาท์พุท หรือใช้ในการติดต่อกับหน่วยความจำโปรแกรมภายนอก ซึ่งพอร์ตที่ 0 เป็นขาชี้ตำแหน่ง (A7 - A0) และขารับส่งข้อมูล (D7 - D0) พอร์ตที่ 0 สังเกตจะเห็นว่า จะไม่มีวงจรพูลอัป (Pull - up) ภายใน ดังนั้นหากจะนำพอร์ตที่ 0 ไปใช้งานจะต้องต่อต้านทานพูลอัป เข้าไปที่ขาพอร์ตที่ 0 ด้วย



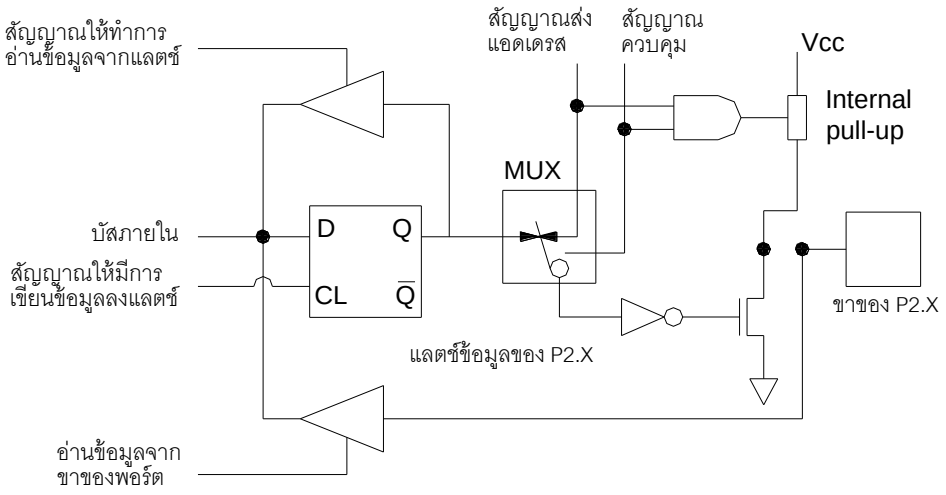
ภาพที่ 2.3 วงจรขาพอร์ตที่ 0

พอร์ตที่ 1 จะมีวงจรพูลอัปภายในแต่ละบิตและมีลักษณะคล้ายกับพอร์ตที่ 0 จะต่างกันตรงพอร์ตที่ 1 จะไม่มีวงจรมัลติเพล็กซ์ เนื่องจากพอร์ตที่ 1 นี้จะไม่ได้ใช้ติดต่อกับหน่วย ความจำโปรแกรมภายนอก เหมือนกับ พอร์ตที่ 0



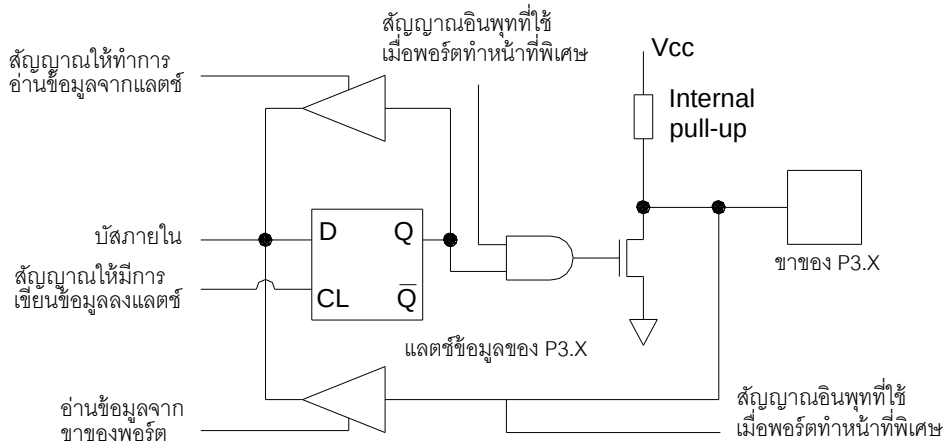
ภาพที่ 2.4 วงจรขาพอร์ตที่ 1

พอร์ตที่ 2 จะมีลักษณะคล้ายพอร์ตที่ 0 แต่จะต่างกันตรงพอร์ตที่ 2 จะมีวงจรพูลอัปภายใน ภายในโครงสร้างพอร์ตจะมีวงจรมัลติเพล็กซ์ เพราะว่าพอร์ตที่ 2 ทำหน้าที่เป็นอินพุท/เอาท์พุท และใช้พอร์ตที่ 2 ในการติดต่อกับหน่วยความจำโปรแกรมภายนอก โดยเป็นตัวชี้ตำแหน่ง (A15 - A8)



ภาพที่ 2.5 วงจรขาพอร์ตที่ 2

พอร์ตที่ 3 จะมีลักษณะคล้ายกับพอร์ตที่ 1 แต่จะมีวงจรบัฟเฟอร์และวงจรอินพุท/เอาท์พุท เนื่องจากพอร์ตที่ 3 ทำงานในฟังก์ชันพิเศษแบบต่างๆ

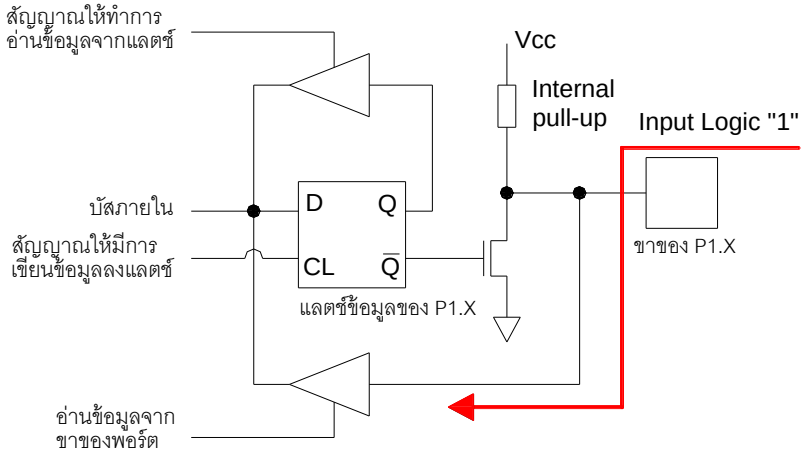


ภาพที่ 2.6 วงจรขาพอร์ตที่ 3

2.1.4 การใช้งานพอร์ตในลักษณะอินพุท/เอาท์พุท

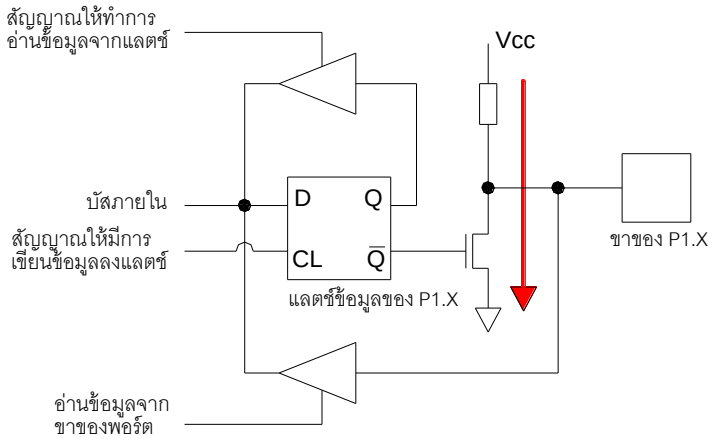
การใช้งานพอร์ตต่างๆ ในไมโครคอนโทรลเลอร์ MCS 51 เป็นพอร์ตที่สามารถติดต่อแบบสองทิศทาง คือ เป็นได้ทั้งอินพุทและเอาท์พุท ดังนั้นจึงต้องเข้าใจการกำหนดลักษณะ การทำงานให้แก่พอร์ตของไมโครโทรลเลอร์ MCS 51

การกำหนดการทำงานของพอร์ตเป็นอินพุท ต้องเริ่มจากเขียนข้อมูลให้พอร์ตที่ต้องการให้ทำงาน เป็นลักษณะอินพุท มีสภาวะลอจิก “1” ทุกบิต เพื่อหยุดการทำงานของตัวเฟตที่อยู่ภายในโครงสร้าง ของพอร์ต ดังนั้นเมื่อตัวเฟตหยุดทำงานก็จะทำให้ขาสัญญาณ ของพอร์ตถูกต่อเชื่อมกับวงจรพูลอัปภายใน โดยตรง ส่งผลให้ขาพอร์ตมีสภาพวะลอจิก “1” สามารถรับลอจิก “0” คำสั่ง (ภาษาแอสเซมบลี) ที่ใช้ในการรับค่าจากพอร์ตที่เป็นอินพุท

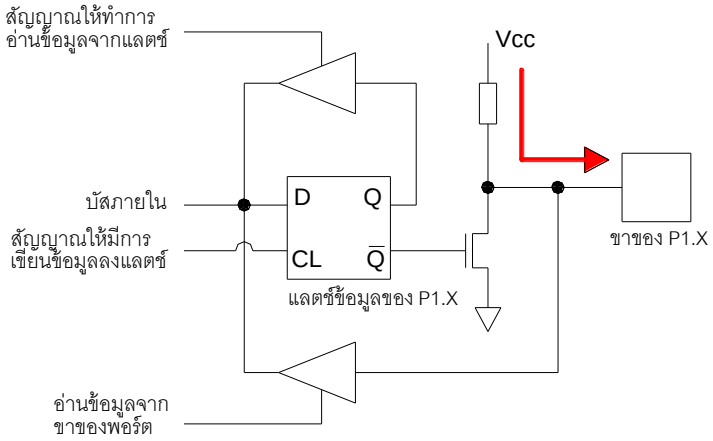


ภาพที่ 2.7 การทำงานของพอร์ตลักษณะอินพุท

การกำหนดให้พอร์ตทำงานเป็นเอาท์พุท โดยปกติแล้วไมโครคอนโทรลเลอร์จะกำหนดให้พอร์ต ทำงานลักษณะเอาท์พุทอยู่แล้ว เช่น เมื่อจะกำหนดสภาวะลอจิก “0” ให้ปรากฏที่ขาพอร์ตที่กำหนดก็แค่ เขียน ข้อมูลลอจิก “0” ให้วงจรแลตช์ ส่งสัญญาณไปขับตัวเฟตภายในโครงสร้างที่ทำงาน ที่ขาพอร์ตที่กำหนด ก็จะปรากฏสภาวะลอจิก “0” ขึ้น สังเกตจากภาพที่ 2.8



ภาพที่ 2.8 การทำงานของพอร์ตลักษณะเอาท์พุท



ภาพที่ 2.9 การทำงานของพอร์ตลักษณะเอาท์พุท

ในการใช้พอร์ตเป็นเอาท์พุทแต่ละขาของพอร์ตสามารถจ่ายกระแสหรือเรียกว่ากระแสซอร์ซ (Source Current) ได้สูงสุด 10mA และถ้าใช้ทุกขารวมกันของพอร์ตใดพอร์ตหนึ่ง (1 พอร์ต มี 8 บิต) จะจ่ายกระแสได้สูงสุด 26mA แต่ละพอร์ตที่ 0 จะได้ 15mA และถ้าใช้งานพอร์ตที่ 1 - 3 รวมกันเป็น พอร์ตเอาท์พุทจะสามารถจ่ายกระแสได้รวมกันสูงสุด 71mA เห็นได้ว่าถ้าจะนำพอร์ตของไมโครคอนโทรลเลอร์ MCS51 ไปใช้งานเป็นเอาท์พุทเพื่อไม่ให้เกิดปัญหาในการจ่ายกระแสให้อุปกรณ์ภายนอกที่นำมาต่อเชื่อมกับ พอร์ต ควรจะต้องมีวงจรบัฟเฟอร์เพื่อช่วยขับกระแสต่อกับพอร์ตที่เป็นเอาท์พุทด้วย คำสั่ง (แอสเซมบลี) ในการส่งสัญญาณนอกที่พอร์ตที่ทำงานเป็นเอาท์พุท เช่น MOV P2, A มีความหมายคือ นำข้อมูลที่อยู่ในรีจิสเตอร์ A หรือแอกคิวมิวเลเตอร์ส่งข้อมูลออกไปให้พอร์ต P2

2.1.5 การจัดหน่วยความจำของไมโครคอนโทรลเลอร์ MCS - 51

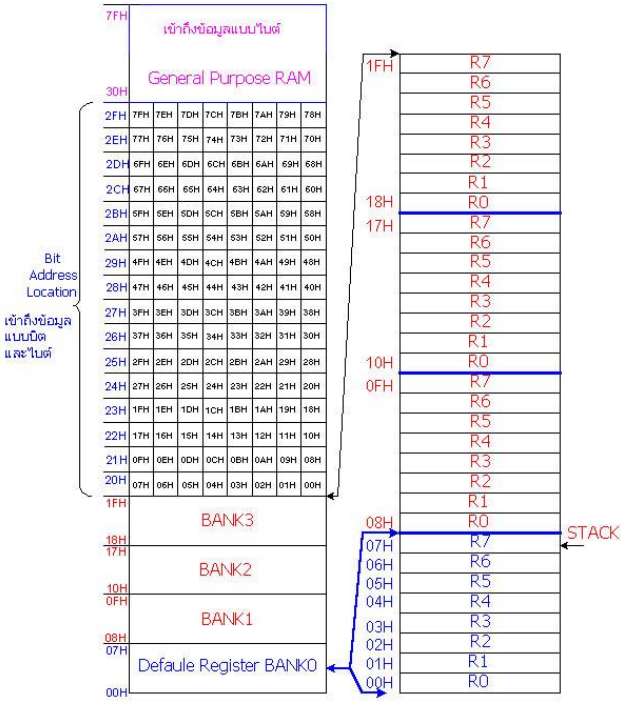
ในไมโครคอนโทรลเลอร์ MCS - 51 มีหน่วยความจำอยู่ 2 แบบตามลักษณะของข้อมูลที่เก็บคือ

1. หน่วยความจำข้อมูล (Data Memory) ก็คือหน่วยความจำส่วนที่เรียกว่า แรม (RAM) ซึ่งสามารถอ่านเขียนข้อมูลได้ตลอดเวลา แต่ไม่สามารถรันโปรแกรมได้บนหน่วยความจำส่วนนี้ได้

2. หน่วยความจำโปรแกรม (Program Memory) ก็คือหน่วยความจำที่เรียกว่า รอม (ROM) ซึ่งสามารถอ่านได้อย่างเดียว ซึ่งจะเก็บโปรแกรมที่จะทำให้ไมโครคอนโทรลเลอร์ทำงานหรือที่เรียกว่า โปรแกรมมอนิเตอร์ (Monitor Program)

หน่วยความจำโปรแกรม ในไมโครคอนโทรลเลอร์ MCS - 51 จะแบ่งออกเป็น 2 แบบ คือ หน่วยความจำโปรแกรมภายนอกและหน่วยความจำโปรแกรมภายใน ตัวไมโครคอนโทรลเลอร์ที่

นำมา ใช้งานขนาดของหน่วยความจำโปรแกรมภายในจะมีขนาดแตกต่างกันไป ตามเบอร์ของไมโครคอนโทรลเลอร์



ภาพที่ 2.10 การจัดสรรพื้นที่ในหน่วยความจำข้อมูลภายใน

จากภาพที่ 2.9 การจัดสรรพื้นที่ในหน่วยความจำข้อมูลภายใน โดยแบ่งออกเป็น 4 กลุ่มหลักๆ คือ

1. ที่ตำแหน่ง Address 00H - 1FH ขนาด 32 ไบต์ เป็นพื้นที่ของรีจิสเตอร์ R0 - R7 ซึ่งรีจิสเตอร์นี้ จะมีด้วยกัน 4 ชุด หรือเรียกว่า 4 แบงก์ ดังนั้นจึงมีรีจิสเตอร์ไว้ใช้งานถึง 32 ตัว
2. ที่ตำแหน่ง Address 20H - 2FH ขนาด 16 ไบต์ เป็นพื้นที่ใช้งานทั่วไป สามารถเข้าถึง ระดับบิตได้
3. ที่ตำแหน่ง Address 30H - 7FH หน่วยความจำข้อมูลที่ใช้งานทั่วไป
4. ที่ตำแหน่ง Address 80H - FFH เป็นพื้นที่ของรีจิสเตอร์พิเศษ (Special Function Register : SFR) ที่ทำหน้าที่ควบคุมการทำงานของไมโครคอนโทรลเลอร์ เช่น รีจิสเตอร์ A รีจิสเตอร์ B, PC, PSW, DPTR, SP รีจิสเตอร์ที่เกี่ยวกับพอร์ต P0, P1, P2, P3 มีรีจิสเตอร์ที่เกี่ยวกับพอร์ตอนุกรม SCON, SBUFF และรีจิสเตอร์ที่เกี่ยวกับไทเมอร์ T0, T1 เป็นต้น

2.2 ภาษาซีสำหรับไมโครคอนโทรลเลอร์ [2]

ภาษาซีสำหรับไมโครคอนโทรลเลอร์ ในที่นี้จะกล่าวถึงโครงสร้างภาษาซี ชนิดของตัวแปรในภาษาซี ตัวดำเนินการต่าง การสร้างฟังก์ชั่นย่อย และประกาศฟังก์ชั่นย่อยของภาษาซี ในส่วนของการคอมไพเลอร์ในที่นี้จะใช้คอมไพเลอร์ C51 ของ Keil Software ซึ่งมีลักษณ์ในการใช้งานที่เป็นไปตามมาตรฐาน ANSI - C

2.2.1 โครงสร้างของภาษาซี

โครงสร้างของภาษาซี จะมีส่วนประกอบหลักตามมาตรฐาน ANSI - C จะประกอบด้วยส่วนพรีโปรเซสเซอร์ (Preprocessor) ส่วนแสดงโปรโตไทป์ (Prototype) ส่วนโปรแกรมหลักหรือฟังก์ชั่นหลัก Main ( ) ส่วนประกาศตัวแปร (Variable Declare) ในภาษาซีโปรแกรมจะเขียนเป็นชุด แต่ละชุดโปรแกรมจะเรียกว่าฟังก์ชั่น ซึ่งก็คือโปรแกรมย่อยที่เรียกกันในโปรแกรมภาษาอื่นๆ นั่นเอง โดยฟังก์ชั่นเหล่านี้จะมีชื่ออย่างไรก็ได้ ในการเขียนภาษาซีนั้นจะมีฟังก์ชั่นหลัก ซึ่งต้องเรียกมาใช้เสมอเมื่อมีการเขียนโปรแกรมนั่นก็คือ Main ( ) สังเกตได้จากแบบโครงสร้างของโปรแกรมภาษาซีดังต่อไปนี้

```
#include<reg51.h>          /*ฟรีโปรเซสเซสเซอร์ เป็นส่วนนำข้อมูลอื่นๆ มาใช้ร่วมกับ
โปรแกรมที่จะเขียน*/
void func0 (void);         /*โปรโตไทป์ เป็นส่วนประกาศฟังก์ชั่นย่อยหรือโปรแกรมย่อย
                           แบบไม่มีการคืนค่า*/
int func1 (int i, int j);  /*โปรโตไทป์ เป็นส่วนประกาศฟังก์ชั่นย่อยหรือโปรแกรมย่อย
                           แบบมีการคืนค่า*/
main ()                    /*ฟังก์ชั่นหลักหรือโปรแกรมหลัก*/
        {                  /*แสดงการเริ่มต้นโปรแกมหลัก*/
        int ans;           /*ประกาศตัวแปร*/
        P1 = 0x0aa;
        func0 ();          /*เป็นการเรียกฟังก์ชั่นย่อย*/
        ans = func1(10, 5); /*เป็นการเรียกฟังก์ชั่นย่อยที่มีการส่งค่า*/
        P1 = ans;
        }                  /*แสดงการสิ้นสุดโปรแกมหลัก*/
void func0(void)           /*ส่วนฟังก์ชั่นย่อยหรือโปรแกรมย่อยแบบไม่มีการคืนค่า*/
```

```
{
P1 = 0x44;
}
int func1(int i, int j)    /*ส่วนฟังก์ชั่นย่อยหรือโปรแกรมย่อยแบบมีการคืนค่า*/
{
return (i + j);
}
```

2.2.2 ลักษณะของข้อมูลและการเขียนหมายเหตุในภาษาซี

ข้อมูลที่ใช้ในการเขียนพัฒนาโปรแกรมภาษาซี คือ สิ่งที่โปรแกรมจะนำไปประมวลผลเพื่อให้ได้ผลลัพธ์ตามที่ต้องการ โดยข้อมูลนั้นจะแบ่งออกเป็น 2 แบบคือ

1. ข้อมูลที่ใช้คำนวณ ได้แก่ค่าคงที่ที่เป็นตัวเลขจำนวนเต็ม ทศนิยม ตัวเลขฐานสิบ ตัวเลขฐานแปด ตัวเลขฐานสิบหก และค่าคงที่ที่เป็นตัวอักขระ

2. ข้อมูลที่ไม่ได้ใช้คำนวณ ได้แก่ ตัวอักษรหรือตัวเลขที่เขียนอยู่ในเครื่องหมาย (“ ”) เช่น “ABC51”

ในภาษาซีมักนิยมเขียนหมายเหตุเพื่ออธิบายการทำงานในส่วนต่างๆ ของโปรแกรม เพื่อให้เข้าใจเมื่อผู้ที่มาศึกษาโปรแกรมสามารถอ่านและทำความเข้าใจโปรแกรมได้ง่ายขึ้น การเขียนหมายเหตุจะใช้เครื่องหมาย (/*) หรือเครื่องหมาย (//) ดังรูปแบบดังนี้

/*……ข้อความที่ต้องการอธิบาย……*/

หรือ

//……ข้อความที่ต้องการอธิบาย……

2.2.3 การประกาศตัวแปรและค่าคงที่ของภาษาซี

การประกาศตัวแปรที่ใช้ในภาษาซีจะต้องมีการประกาศชนิดของตัวแปรให้สอดคล้องกับข้อมูลที่จะเก็บในตัวแปรนั้น โดยจะต้องประกาศชนิดของตัวแปรก่อนที่จะนำตัวแปรนั้นๆไปใช้งาน การประกาศตัวแปรในภาษาซีมีรูปแบบดังนี้

type Variable name;

type คือ ชนิดของตัวแปร ซึ่งสามารประกาศเป็นชนิดต่างๆ ได้ดังตารางที่ 2.2

Variable_name คือ ชื่อของแปรที่ต้องการประกาศ ถ้ามีมากกว่าหนึ่งตัวสามารถใช้ (,) แยกตัวประกอบได้ เช่น int x, y, z เป็นการประกาศตัวแปร x, y, z เป็นตัวแปรชนิดจำนวนเต็มเหมือนกัน

ตารางที่ 2.2 ชนิดของตัวแปรแบบต่างๆ

ชนิดของตัวแปร	ขนาด (บิต)	ช่วงของค่าตัวเลข
bit	1	0 ถึง 1
char	8	(- 128) ถึง (+127)
unsigned char	8	0 ถึง 255
int	16	(- 32768) ถึง (+32767)
unsigned int	16	0 ถึง 65535
long	32	(- 2147483648) ถึง (+2147483647)
unsigned long	32	0 ถึง 4294967295
float	32	3.4e - 38 ถึง 3.4e+38
double	64	1.7e - 308 ถึง 1.7e+308

2.2.4 ตัวแปรชุด (Array Variable)

ในโปรแกรมภาษาซีถ้าต้องการสร้างตัวแปรที่เป็นชนิดเดียวกันหลายๆ ตัว โดยที่ไม่ต้องมีปัญหาในการตั้งชื่อตัวแปร ภาษาซีมีการตั้งชื่อตัวแปรที่ตั้งชื่อเพียงชื่อเดียว แล้วใช้ตัวเลขเป็นตัวกำหนดว่าตัวแปรตัวนี้เป็นตัวแปรตัวที่เท่าไร ลักษณะการสร้างตัวแปรชนิดนี้จะเรียกกันว่า ตัวแปรชุดหรือตัวแปรอะเรย์ โดยมีรูปแบบดังนี้

Type Array - name [n]

Type คือ ชนิดของตัวแปรชุดอะเรย์

Array - name คือ ชื่อของตัวแปรชุดอะเรย์

n คือ ตัวเลขจำนวนเต็มที่แสดงถึงขนาดของตัวแปรชุดอาจจะไม่ต้องกำหนดก็ได้ภาษาซีทำการจัดเตรียมหน่วยความจำไว้เท่ากับจำนวนจริงเอง

2.2.5 ตัวดำเนินการในภาษาซี (Operator)

ตัวดำเนินการต่างๆ กับตัวแปรและค่าคงที่ในภาษาซี จะแบ่งออกเป็น 3 กลุ่ม คือ ตัวดำเนินการทางคณิตศาสตร์ ตัวดำเนินการทางการเปรียบเทียบและตรรกะ และตัวดำเนินการทางบิต โดยมีรูปแบบเครื่องหมายดังต่อไปนี้

- ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators) ได้แก่ ตัวดำเนินในการคำนวณ บวก ลบ คูณ หารค่าต่างๆ ดังต่อไปนี้

ตารางที่ 2.3 ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operator)

ตัวดำเนินการ	ความหมาย	ตัวอย่าง
+	การบวก	a + b
-	การลบ	a - b
*	การคูณ	a * b
/	การหาร	a / b
%	การหารแบบเอาเศษ	a % b เช่น 5%2 = 1 เก็บเศษเป็นคำตอบ
++	การเพิ่มค่าขึ้นอีกหนึ่งค่า	a++ จะเหมือนกับ a = a+1
- -	การลดค่าขึ้นอีกหนึ่งค่า	a - - จะเหมือนกับ a = a - 1
+ =	บวกขึ้นอีก ด้วยค่าทางขวามือ	x += 10 จะเหมือนกับ x = x+10
- =	ลบค่าลงอีก ด้วยค่าทางขวามือ	x - = 12 จะเหมือนกับ x = x+12
* =	การคูณด้วยค่าทางขวามือ	x *= 2 จะเหมือนกับ x = x*2
/ =	การหารด้วยค่าทางขวามือ	x /= 11 จะเหมือนกับ x = x/11

- ตัวดำเนินการเปรียบเทียบและตรรกะลอจิก (Relational and Logical Operators) ได้แก่ ตัวดำเนินในการที่ใช้ในการเปรียบเทียบตัดสินใจ ดังต่อไปนี้ ซึ่งผลลัพธ์จากการตรวจสอบเงื่อนไขจะได้เป็น “1” แสดงเงื่อนไขว่าเป็นจริง (True) และจะได้ค่า “0” แสดงว่าเงื่อนไขตรวจสอบเป็นเท็จ (False) ตัวดำเนินการที่ใช้ในการเปรียบเทียบต่างๆ มีดังนี้

ตารางที่ 2.4 ตัวดำเนินการเปรียบเทียบและตรรกะลอจิก (Relational and Logical Operators)

ตัวดำเนินการ	ความหมาย	ตัวอย่าง
>	มากกว่า	a > b
≥	มากกว่าหรือเท่ากับ	a ≥ b
<	น้อยกว่า	a < b
≤	น้อยกว่าหรือเท่ากับ	a ≤ b
= =	เท่ากับ	a = = b (a เท่ากับ b)
!=	ไม่เท่ากับ	a != b (a ไม่เท่ากับ b)
!	NOT กลับค่าทางลอจิก	!(a > b) เป็นการกลับค่าทางลอจิก
&&	AND การแอนด์	(a > b) && (b ≤ c)
||	OR การออร์	(a > b) || (b ≤ c)

เครื่องหมายตัวดำเนินการทางตรรกะในการเปรียบเทียบเงื่อนไข โดยถ้าเงื่อนไขที่เป็นจริงค่าจะได้เป็น “1” (True) และถ้าเงื่อนไขเป็นเท็จค่าจะได้เป็น “0” (False)

ตารางที่ 2.5 ค่าความจริงของการแอนด์ (AND) เครื่องหมาย &&

เงื่อนไข A	เงื่อนไข B	(เงื่อนไข A) && (เงื่อนไข B)
0	0	0
0	1	0
1	0	0
1	1	1

ตารางที่ 2.6 ค่าความจริงของการออ (OR) เครื่องหมาย ||

เงื่อนไข A	**เงื่อนไข B**	**(เงื่อนไข A) || (เงื่อนไข B)**
0	0	0
0	1	1
1	0	1
1	1	1

ตารางที่ 2.7 ค่าความจริงของการน็อต (NOT) เครื่องหมาย !

ผลลัพธ์เงื่อนไข	**!(ผลลัพธ์เงื่อนไข)**
0	1
1	0

ตัวอย่างการใช้ตัวดำเนินการเปรียบเทียบและตรรกะลอจิก

เช่น กำหนดให้ a = 5, b = 10, c = 2

(a != c) หมายความว่า เปรียบเทียบเงื่อนไข a ไม่เท่ากับ c จริง ผลลัพธ์ที่ได้เป็นจริงได้ค่า “1”

!(a != c) หมายความว่า เปรียบเทียบเงื่อนไข a ไม่เท่ากับ c จริง ผลลัพธ์ที่ได้จะกลับสภาวะเป็นเท็จ “0” เนื่องจากเครื่องหมายนอต (!)

!((a != c) && (a < b)) หมายความว่า เป็นการกลับผลลัพธ์ของการแอนด์กับระหว่างเงื่อนไข a น้อยกว่า c และเงื่อนไข a น้อยกว่าหรือเท่ากับ b ถ้าผลลัพธ์จากการแอนด์กันเป็นจริง (True) “1” แต่จะถูกกลับสภาวะให้เป็นตรงข้ามด้วยตัว ! ดังนั้นจะมีผลลัพธ์เป็นเท็จ (False) “0”

- ตัวดำเนินการทางบิต (Bit Operators) ได้แก่ ตัวดำเนินการที่สามารถเข้าถึงข้อมูลระดับบิตได้ มีดังต่อไปนี้

ตารางที่ 2.8 ตัวดำเนินการทางบิต (Bit Operators)

ตัวดำเนินการ	ความหมาย	ตัวอย่าง
~	กลับค่าบิต	~ a หรือ a = ~a (กลับค่าบิต a)
>>	เลื่อนบิตไปทางขวา	a >> 1 (เลื่อนบิตของ a ไปทางขวา 1 บิต)
<<	เลื่อนบิตไปทางซ้าย	a >> 1 (เลื่อนบิตของ a ไปทางซ้าย 1 บิต)
>≥	เลื่อนบิตไปทางขวาแล้วให้เท่ากับ	a >≥ 1 (เลื่อนบิตของ a ไปทางขวา 1 บิตเก็บไว้ที่ a)
<≤	เลื่อนบิตไปทางซ้ายแล้วให้เท่ากับ	a <≤ 1 (เลื่อนบิตของ a ไปทางซ้าย 1 บิตเก็บไว้ที่ a)
^	การเอ็กคูซีฟออร์ของบิต	a ^ b (เอ็กคูซีฟออร์ของบิตของ a กับ b)
^ =	การเอ็กคูซีฟออร์ของบิตแล้วให้เท่ากับ	a ^= 0x23 (เอ็กคูซีฟออร์ของบิตของ a กับ 23H เก็บค่าไว้ที่ a)
&	การแอนด์ของบิต	a && b (a แอนด์ของบิตกับ b)
&=	การแอนด์ของบิตแล้วให้เท่ากับ	a && = 0xFF (a แอนด์ของบิตกับค่า FFH เก็บค่าไว้ที่ A)
|	การออร์ของบิต	a | b (a ออร์ของบิตกับ b)
|=	การออร์ของบิตแล้วให้เท่ากับ	a |= 0xFF (a ออร์ของบิตกับค่า FFH เก็บค่าไว้ที่ a)

2.2.6 ลำดับการประมวลผลของตัวดำเนินการ

ตัวดำเนินการต่างๆ กับตัวแปรและค่าคงที่ในภาษาซี จะแบ่งออกเป็น 3 กลุ่มคือตัวดำเนินการทางคณิตศาสตร์ ตัวดำเนินการทางการเปรียบเทียบและตรรกะลอจิก และตัวดำเนินการทางบิต ในภาษาซีจะมีลำดับการประมวลผลเป็นลำดับ ดังตารางดังต่อไปนี้

ตารางที่ 2.9 ลำดับการประมวลผลของตัวดำเนินการ

ตัวดำเนินการ	**ลำดับการประมวลผล**
เครื่องหมายทางคณิตศาสตร์	1
!	2
> , ≥ , < , ≤	3
= = , !=	4
&&	5
||	6

2.2.7 การประกาศฟังก์ชัน และเรียกใช้ฟังก์ชัน

ในภาษซี มีการเขียนโปรแกรมเป็นชุดข้อมูลที่เรียกว่าฟังก์ชัน ซึ่งฟังก์ชันจะมี 2 ประเภท คือ ประเภทที่ 1 คือฟังก์ชันหลัก main() หรือโปรแกรมหลัก ที่ต้องเรียกใช้เสมอเมื่อมีการเขียนโปรแกรมภาษาซี และประเภทที่ 2 เป็นฟังก์ชันย่อยหรือโปรแกรมย่อย (Subprogram) ที่สามารถสร้างขึ้นเองได้ เพื่อพัฒนาโปรแกรมที่เขียนให้เข้าใจง่ายและรวดเร็วขึ้น การประกาศฟังก์ชันหลัก มีรูปแบบดังนี้

```
main ()
{
Function0 ();
Statement 1;
Statement 2;
}
```

2.2.8 ตัวแปรที่ใช้ในฟังก์ชันของภาษา

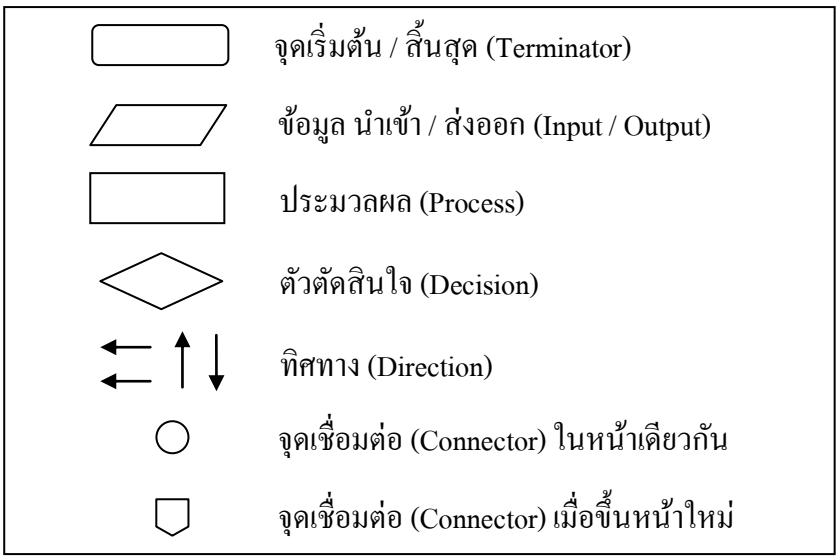
ตัวแปรที่ใช้ในฟังก์ชันของภาษาซีนั้นจะมีลักษณะของตัวแปรที่นำมาใช้อยู่ 2 แบบคือ แบบแรกเป็นแบบตัวแปรที่ถูกกำหนดขึ้นภายในลูปของฟังก์ชันที่สร้างขึ้น หรือเรียกว่าตัวแปร Local Variable และแบบที่ 2 เป็นตัวแปรที่ถูกกำหนดขึ้นภายนอกฟังก์ชันใดๆ ตัวแปรแบบนี้สามารถใช้ร่วมงานกันได้ทุกฟังก์ชันในโปรแกรมเดียวกันหรือเรียกว่า Global Variable

2.2.9 การเขียน Flow Chart

งานพัฒนาโปรแกรมคอมพิวเตอร์ คือ กระบวนการแก้ปัญหาโจทย์อย่างเป็นขั้นตอน สรุปจะมีขั้นตอนอยู่ 5 ขั้นตอน คือ

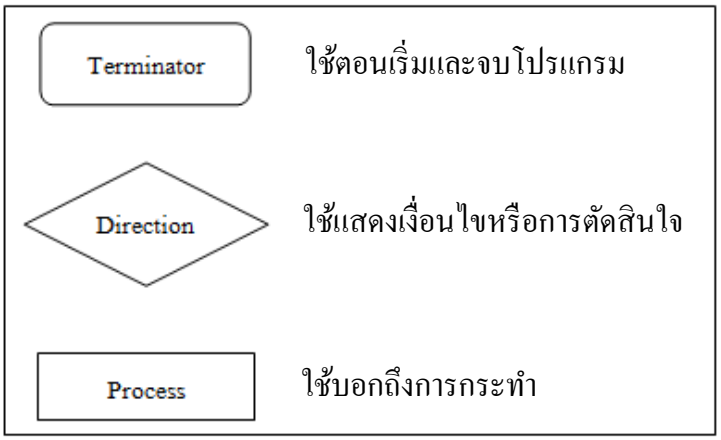
1. เริ่มต้นต้องรู้ว่าปัญหาที่โจทย์ต้องการคืออะไร
2. วางแผนวิเคราะห์การแก้ปัญหาอย่างไร
3. ออกแบบลำดับขั้นตอนการแก้ปัญหาเป็นลำดับ (อัลกอริธึม)
4. เขียนโปรแกรมภาษาคอมพิวเตอร์จากลำดับขั้นตอนที่ออกแบบไว้
5. ทดลองทำงานโปรแกรมภาษาคอมพิวเตอร์

จากขั้นตอนการพัฒนาโปรแกรมทั้ง 5 ขั้นตอนนี้ ขั้นตอนที่ 3 เป็นการออกแบบลำดับขั้นตอนการแก้ปัญหา (อัลกอริธึม) เป็นส่วนสำคัญ ในการเขียนอัลกอริธึมจะมีเครื่องมือที่นิยมใช้คือ Flow Chart การเขียน Flow Chart จะเขียนเป็นรูปภาพผังงาน มีลูกศรชี้ทิศทาง ทำให้ง่ายต่อการเข้าใจ การเขียน Flow Chart (แผนผังงาน) จะใช้สัญลักษณ์ดังต่อไปนี้คือ



ภาพที่ 2.11 สัญลักษณ์ในการเขียน Flow Chart

ในการเขียนสัญลักษณ์ของ Flow Chart ที่ใช้กับโปรแกรมควบคุมไมโครคอนโทรลเลอร์ จะมีเพียง 3 สัญลักษณ์เท่านั้นที่พบเห็นบ่อยๆ จึงง่ายต่อการใช้งานและจดจำ ดังต่อไปนี้



ภาพที่ 2.12 สัญลักษณ์ของ Flow Chart ที่ใช้กับโปรแกรมควบคุมไมโครคอนโทรลเลอร์

- คำสั่ง START - END ในการเขียนโปรแกรมของ PDL ควรจะเริ่มต้นด้วยคำว่า START เพื่อเป็นการบอกถึงการเริ่มต้นในส่วนของโปรแกรม และคำว่า END เมื่อสิ้นสุดโปรแกรม
- คำสั่ง Sequencing เป็นคำสั่งอธิบายการทำงานโปรแกรม โดยปกติมักจะเขียนคำสั้นๆ เป็นภาษาอังกฤษเมื่อต้องการบรรยายส่วนของโปรแกรมนั้นๆ ในหนังสือเล่มนี้ ขอใช้ภาษาไทยในการอธิบายทำงานโปรแกรม เพื่อสามารถเข้าใจโปรแกรมง่ายขึ้น
- คำสั่ง IF THEN - ELSE - ENDIF เป็นคำสั่งที่อธิบายการควบคุมในโปรมแกรมที่มีเงื่อนไงการตัดสินใจ

ตัวอย่าง

IF switch = 0 (กดสวิตช์) THEN

Turn on LED1 หลอดไฟ LED1 ติดสว่าง

ELSE

Turn off LED1 หลอดไฟ LED1 ดับ

ENDIF

- คำสั่ง DO - ENDDO เป็นคำสั่งควบคุมการทำงานโปรแกรมแบบวนซ้ำในรูปแบบ PDL

ตัวอย่าง

Turn on LED	หลอดไฟ LED ติดสว่าง
DO 5 time	5 ครั้ง
Set clock to 1	เซตสัญญาณนาฬิกา 1
Set clock to 0	เซตสัญญาณนาฬิกา
ENDDO	

2.2.10 คำสั่งควบคุมการทำงานในภาษาซี

ในการเขียนโปรแกรมภาษาซีสำหรับไมโครคอนโทรลเลอร์ MCS - 51 ส่วนใหญ่แล้วก็จะยึดรูปแบบของมาตรฐาน ANSI - C ซึ่งการทำงานของโปรแกรมนั้นมักจะทำงานเรียงลำดับจากบนลงไปล่าน ตั้งแต่ต้นโปรแกรมจนกระทั่งจบโปรแกรมแต่ในภาษาซีจะมีคำสั่งที่สามารถทำให้การทำงานเปลี่ยนแปลงได้ เช่น กระโดดไปทำงานตำแหน่งที่ต้องการ มีการตรวจสอบเงื่อนไขก่อนทำงานหรือวนการทำงานซ้ำๆ กันตลอด ซึ่งคำสั่งควบคุมการทำงานในภาษาซีแบ่งออกเป็น 3 ชนิด ดังต่อไปนี้คือ

1. คำสั่งให้ไปทำงานโดยไม่มีเงื่อนไข ได้แก่ คำสั่ง goto
2. คำสั่งให้ไปทำงานโดยมีเงื่อนไข ได้แก่ คำสั่ง if, switch
3. คำสั่งให้ทำงานวนซ้ำ ได้แก่ คำสั่ง for, while, do - while

คำสั่งให้ไปทำงานโดยไม่มีเงื่อนไข goto เป็นคำสั่งที่ให้กระโดดไปทำงานยังตำแหน่งอื่นๆ หรือบรรทัดใดๆ ของโปรแกรม โดยไม่มีเงื่อนไข มีรูปแบบดังต่อไปนี้

รูปแบบคำสั่ง goto

goto Statement - Labal;

โดยที่

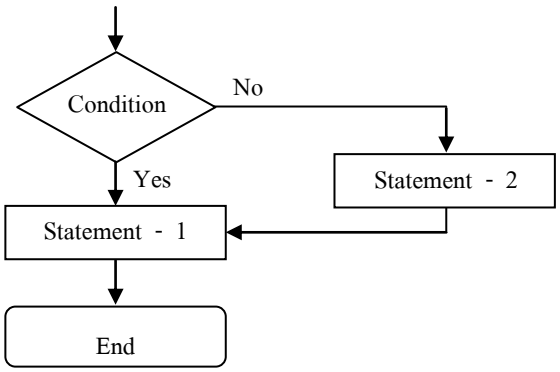
Statement - Label หมายถึง ชื่อที่ใช้ในการอ้างตำแหน่งหรือบรรทัดที่จะให้ไปทำงานตามคำสั่ง goto

goto loop; หมายความว่า ให้กระโดดไปยังตำแหน่งหรือบรรทัดที่มี label ชื่อ loop อยู่

คำสั่งให้ทำงานโดยมีเงื่อนไข if - else เป็นคำสั่งที่มีการตรวจสอบเงื่อนไขเพื่อให้ไปทำงานยังคำสั่งใดคำสั่งหนึ่ง ซึ่งเงื่อนไขในการตัดสินใจมีโอกาสเป็นไปได้ 2 แบบคือ เงื่อนไขเป็นจริงมีค่าไม่เท่ากับศูนย์ และเงื่อนไขเป็นเท็จมีค่าเป็นศูนย์ ดังรูปแบบต่อไปนี้

รูปแบบคำสั่ง if - else

```
if (Condition)
{
Statement - 1;
}
Else
{
Statement - 2;
}
```

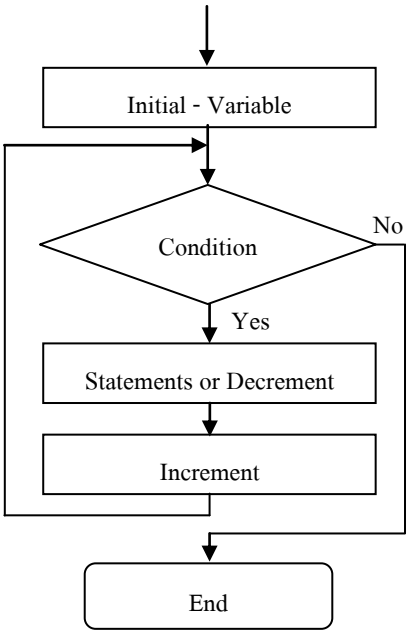


ภาพที่ 2.13 Flow Chart รูปแบบคำสั่ง if - else

คำสั่งให้ทำงานวนรอบซ้ำ for เป็นคำสั่งที่ให้โปรแกรมมีการทำงานแบบวนรอบซ้ำ ในการทำงานรอบซ้ำนั้นจะถูกกำหนดด้วยการตรวจสอบเงื่อนไขเพื่อให้โปรแกรมวนรอบซ้ำเป็นจำนวนกี่รอบ รูปแบบคำสั่งดังต่อไปนี้

รูปแบบคำสั่ง for

```
for ( Initial - Variable ; Condition ; Increment or Decrement)
{
Statements;
}
```



ภาพที่ 2.14 Flow Chart รูปแบบคำสั่ง for

โดยที่

Initial - Variable หมายถึง ค่าเริ่มต้นที่กำหนดให้กับตัวแปรที่จะนำมาใช้ในการตรวจสอบเงื่อนไขให้ทำงานรอบวนซ้ำ

Condition หมายถึง เงื่อนไขที่ใช้ตรวจสอบเพื่อให้มีการทำงานวนซ้ำ โดยถ้าเงื่อนไขเป็นจริงจะมีค่าเป็น “1” โปรแกรมจะไปทำงานที่ Statements ภายในวงเล็บปีกกา {} ของคำสั่ง for และถ้าเงื่อนไขเป็นเท็จจะมีค่า “0” โปรแกรมก็จะออกไปทำงานคำสั่งต่อไปจากคำสั่ง for

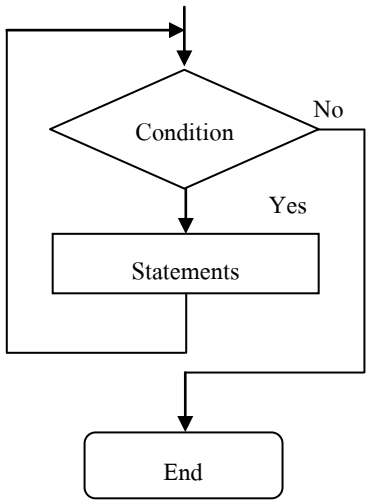
Statements หมายถึง กลุ่มคำสั่งที่อยู่ภายในวงเล็บปีกกา {} ของคำสั่ง for

Increment or Decrement หมายถึง การกำหนดค่าให้เพิ่มขึ้นหรือลดลงให้กับตัวแปรที่นำมาเป็นเงื่อนไขตรวจสอบในคำสั่ง for

คำสั่งวนรอบ while เป็นคำสั่งให้โปรแกรมทำงานวนรอบซ้ำโดยมีการตรวจสอบเงื่อนไข โดยถ้าเงื่อนไขที่ตรวจสอบเป็นจริงจะมีค่าเป็น “1” ก็จะทำงานวนรอบซ้ำ และถ้าเงื่อนไขเป็นเท็จจะมีค่าเป็น “0” ก็จะออกจะคำสั่ง while ไปทำงานคำสั่งต่อไป รูปแบบคำสั่ง ดังนี้

รูปแบบคำสั่ง while

```
while(Condition)
{
Statements;
}
```



ภาพที่ 2.15 Flow Chart รูปแบบคำสั่ง while

โดยที่

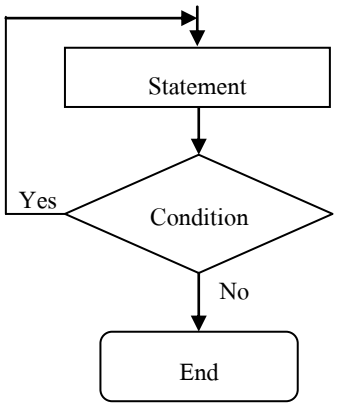
Condition หมายถึง เงื่อนไขที่ใช้ในการตรวจสอบ เพื่อให้มีการทำงานวนรอบ โดยถ้าเงื่อนไขเป็นจริงจะมีค่าเป็น “1” โปรแกรมจะไปทำงานที่ Statements ภายในวงเล็บปีกกา {} ของคำสั่ง while และถ้าเงื่อนไขเป็นเท็จจะมีค่า “0” โปรแกรมก็จะออกไปทำงานคำสั่งต่อไปจากคำสั่ง while

Statements หมายถึงกลุ่มคำสั่งที่อยู่ในวงเล็บปีกกา {} ของคำสั่ง while การใช้คำสั่ง while นั้นสามารถเป็นคำสั่งที่ทำให้เกิดการวนรอบทำงานแบบไม่รู้จบได้โดยการกำหนดให้ส่วน Condition มีค่าเป็นจริงคือ “1” หรือให้มีค่าไม่เท่ากับศูนย์โปรแกรมก็จะทำงานภายในคำสั่ง while แบบไม่รู้จบ เช่น while(1), while(5) เป็นต้น

คำสั่งวนรอบ do - while เป็นคำสั่งให้โปรแกรมทำงานวนรอบซ้ำโดยมีการตรวจสอบเงื่อนไข เมื่อกับคำสั่ง while แต่จะแตกต่างกันตรงที่จะมีการกระทำตรงคำสั่งที่ต้องการให้ทำงานก่อน ซึ่งจะอยู่ในส่วนของคำสั่ง do จากนั้นถึงมาตรวจสอบงานเงื่อนไข โดยถ้าเงื่อนไขเป็นจริงมีค่าเป็น “1” ก็จะทำงานวนรอบซ้ำในส่วนของคำสั่ง do และถ้าเงื่อนไขเป็นเท็จมีค่าเป็น “0” ก็จะออกจากคำสั่ง do - while ไปทำงานคำสั่งต่อไป รูปแบบคำสั่งดังนี้

รูปแบบคำสั่ง do - while

```
do
{
Statement;
}
while(Condition)
```



ภาพที่ 2.16 Flow Chart รูปแบบคำสั่ง do

โดยที่

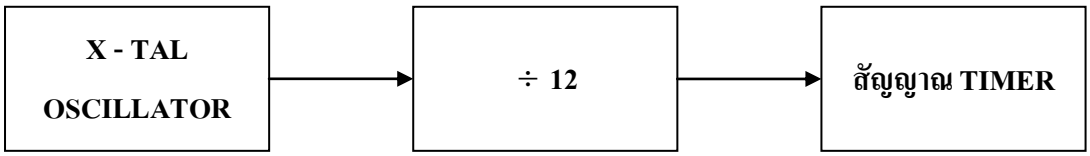
Statements หมายถึงกลุ่มคำสั่งที่อยู่ในวงเล็บปีกกา {} ของคำสั่ง do - while

Condition หมายถึง เงื่อนไขที่ใช้ในการตรวจสอบ เพื่อให้มีการทำงานวนรอบ โดยถ้าเงื่อนไขเป็นจริงจะมีค่าเป็น “1” โปรแกรมจะไปทำงานที่ Statement ภายในวงเล็บปีกกา {} ของคำสั่ง do - while และถ้าเงื่อนไขเป็นเท็จจะมีค่า “0” โปรแกรมก็จะออกไปทำงานคำสั่งต่อไปจากคำสั่ง do - while

2.2.11 ไทเมอร์ และเคาน์เตอร์ในไมโครคอนโทรลเลอร์

ในการใช้งานไทเมอร์ และเคาน์เตอร์ภายในตัวไมโครคอนโทรลเลอร์จะมีคุณสมบัติเป็นทั้งตัวไทเมอร์และเคาน์เตอร์ การใช้งานไทเมอร์ สัญญาณนาฬิกาพัลซ์ที่เกิดขึ้นภายในตัว

ไมโครคอนโทรลเลอร์ จะเกิดจากวงจรผลิตความถี่ (Oscillator) ป้อนให้ตัวไมโครคอนโทรลเลอร์ผ่านวงจรหาร 12 ดังภาพที่ 2.16



ภาพที่ 2.17 ฐานเวลาของไทเมอร์

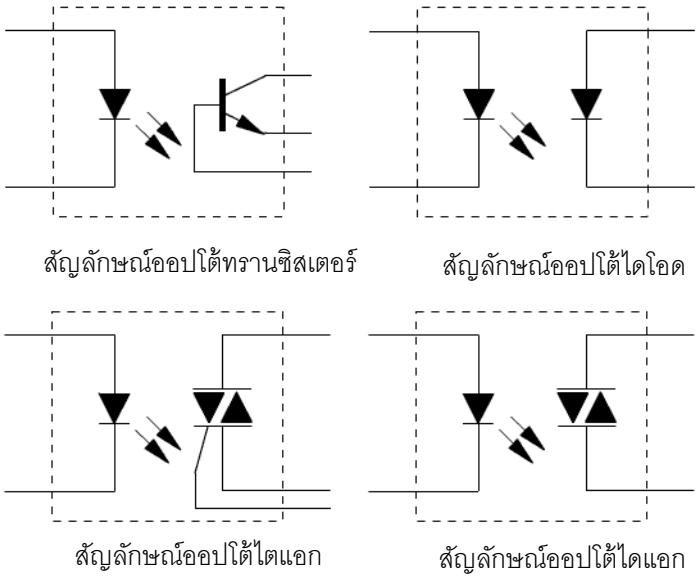
ตัวอย่าง ถ้ามีการใช้คริสตรอล (Crytal) ความถี่ 12 MHz ผลิตความถี่ป้อนให้ไมโครคอนโทรลเลอร์ สัญญาณนาฬิกาพัลซ์ที่ได้ 12 MHz / 12 = 1 MHz จากสูตร t = 1/f ดังนั้นคาบเวลาที่ได้คือ 1 μs แต่ส่วนคริสตอล (Crytal) ที่ใช้กับไมโครคอนโทรลเลอร์ MCS - 51 มักจะใช้ความถี่ 11.0592 MHz ดังนั้นคาบเวลาจะเท่ากับ 1.085 μs จากภาพที่ 2.16 สัญญาณไทเมอร์คาบเวลาที่ได้ก็คือ ค่าแมชชีนไซเคิลของไมโครคอนโทรลเลอร์ที่มีต่อการเปลี่ยนแปลงสัญญาณพัลซ์ 1 ลูกพัลซ์ ซึ่งจะมีคาบเวลาเท่ากับ 1.085 μs นั้นเอง

2.3 ออปโต้คัปเปลอร์ [3]

อุปกรณ์เชื่อมต่อทางแสง (Opto - Isolator) หรือที่เรียกว่าออปโต้คัปเปลอร์ (Opto - Coupler) เป็นอุปกรณ์อิเล็กทรอนิกส์ที่ใช้ในการเชื่อมต่อทางแสงโดยใช้หลักการเปลี่ยนสัญญาณไฟฟ้าเป็นสัญญาณแสง และเปลี่ยนกลับจากแสงเป็นไฟฟ้าตามเดิม ใช้สำหรับการเชื่อมต่อสัญญาณระหว่างสองวงจรที่ต้องการแยกทางไฟฟ้าอย่างเด็ดขาดเพื่อป้องกันการรบกวนกันทางไฟฟ้า แบ่งออกเป็นหลายชนิดแต่ละชนิดจะประกอบด้วย LED ส่งแสงซึ่งปติจะเป็นชนิดอินฟาเรดและตัวรับแสงที่เป็นโฟโต้ทรานซิสเตอร์หรือโฟโต้ไดโอด โดยจะถูกผลิตรวมอยู่ในตัวเดียวกัน

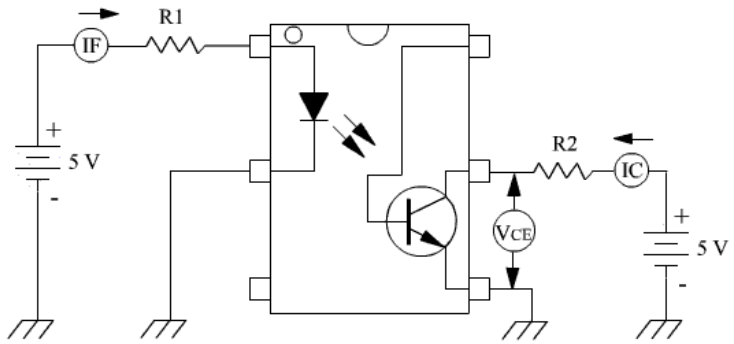
2.3.1 โครงสร้างสัญลักษณ์อุปกรณ์เชื่อมต่อทางแสงงสร้างสัญลักษณ์อุปกรณ์

โครงสร้างสัญลักษณ์อุปกรณ์เชื่อมต่อทางแสงจะเหมือนกับอุปกรณ์ประเภทโฟโต้ แต่จะเพิ่มอุปกรณ์ส่งแสงอินฟาเรดคือไดโอดเปล่งแสงอินฟาเรดเข้าไปอีกหนึ่งตัว เช่นโฟโต้ทรานซิสเตอร์จะเพิ่มไดโอดเปล่งแสงอินฟาเรดเข้าไปอีกหนึ่งตัวจะได้ ออปโต้ทรานซิสเตอร์ อุปกรณ์ออปโต้ตัวอื่นก็เช่นเดียวกัน



ภาพที่ 2.18 สัญลักษณ์อุปกรณ์เชื่อมต่อทางแสงชนิดต่างๆ

จากภาพที่ 2.17 เป็นวงจรใช้งานเบื้องต้นของออปโต้คัปเปลอร์ โดยมีไดโอดเปล่งแสงเป็นอินพุท และโฟโต้ทรานซิสเตอร์เป็นเอาท์พุตของวงจร เมื่อมีกระแสไหลผ่าน LED โดยมี R1 เป็นตัวจำกัดกระแส LED จะส่องแสงไปที่โฟโต้ทรานซิสเตอร์ ทำให้โฟโต้ ทรานซิสเตอร์นำกระแสมีแรงดันเอาท์พุตตกคร่อมที่ R2 ซึ่งจะเห็นได้ว่าเอาท์พุตของวงจรจะถูกควบคุมโดยอินพุท โดยทั้งอินพุทและเอาท์พุตแยกกันทางไฟฟ้าโดยสิ้นเชิง วงจรนี้นิยมนำไปใช้ในวงจรควบคุมแรงดันแหล่งจ่ายไฟสวิตชิ่งในเครื่องรับโทรทัศน์ วงจรควบคุมไฟวิ่งวัตต์สูง เป็นต้น



ภาพที่ 2.19 แสดงวงจรใช้งานออปโต้คัปเปลอร์เบื้องต้น

หัวใจสำคัญของอุปกรณ์เชื่อมต่อทางแสงนั้นคือ “Cerrent Transfer Ratio” (CTR) ซึ่งก็หมายถึง อัตราส่วนระหว่างกระแสอินพุท (Iin) ต่อกระแสเอาท์พุต (Iout) ดังนั้นสามารถเขียนสมการได้ดังนี้

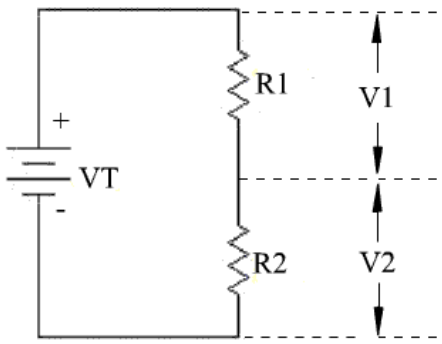
$$CTR = \frac{I_{in}}{I_{out}} \tag{2.1}$$

2.4 วงจรแบ่งแรงดัน [4]

วงจรแบ่งแรงดันไฟฟ้า เรียกว่า “โวลต์เตจ ดีไวเดอร์” (Voltage Divider) ใช้หลักการของวงจรไฟฟ้าแบบอนุกรม (Series Circuit) เนื่องจากวงจรอนุกรมมีแรงดันตกคร่อมตัวต้านทานหรือโหลดไม่เท่ากัน วงจรแบ่งแรงดันแบ่งออกเป็น 2 ชนิดคือ วงจรแบ่งแรงดันไฟฟ้าที่ไม่มีโหลด (Unloaded Voltage Divider) และวงจรแบ่งแรงดันฟ้าแบบมีโหลด (Loaded Voltage Divider)

2.4.1 วงจรการแบ่งแรงดันที่ไม่มีโหลด (Unloaded voltage divider)

วงจรการแบ่งแรงดันที่ไม่มีโหลด Unloaded voltage divider ก็คือ วงจรแบบอนุกรมทั่ว ๆ ไปนั่นเอง ซึ่งเราสามารถที่จะแบ่งแรงดันได้หลาย ๆ ค่า เพื่อนำไปจ่ายให้กับโหลดที่ต้องการแรงดันในระดับต่าง ๆ ที่แตกต่างกันออกไป ซึ่งทั้งนี้แรดันที่ถูกแบ่งทั้งหมดจะได้มาจากแหล่งกำเนิดแรงดันเพียงตัวเดียวเท่านั้น



ภาพที่ 2.20 วงจรแบ่งแรงดันไฟฟ้าแบบไม่มีโหลด (Unloaded Voltage Divider)

จากวงจรจะได้สูตร V1, V2 ดังนี้

$$V_1 = V_T\left(\frac{R_1}{R_2+R_1}\right) \tag{2.2}$$

$$V_2 = V_T \left(\frac{R_2}{R_1+R_2}\right)$$

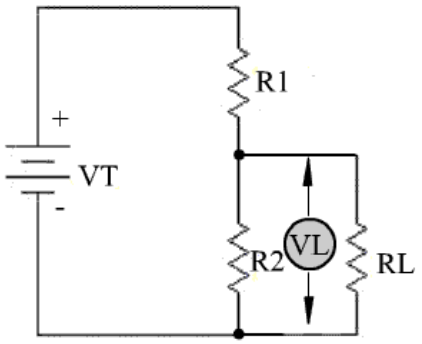
(2.3)

จากภาพที่ 2.19 เป็นวงจรไฟฟ้าแบบอนุกรม (Series Circuit) การแบ่งแรงดันไฟฟ้า (Voltage Dividers) ตัวต้านทาน R1 และ R2 จะทำหน้าที่เป็นตัวแบ่งแรงดันไฟฟ้า เพราะแรงดันไฟฟ้าตกคร่อมตัวต้านทาน R1 และ R2 ไม่เท่ากัน

2.4.2 วงจรการแบ่งแรงดันแบบมีโหลด (Loaded voltage divider)

ในวงจรอนุกรมที่คำนวณแรงดันตกคร่อมความต้านทานต่างๆ เมื่อนำเอาโหลด R_L มาต่อคร่อมตัวต้านทานตัวใดตัวหนึ่งก็จะได้วงจรแบ่งแรงดันแบบมีโหลด

อย่างไรก็ตามจะมีการคำนวณค่ากระแสของแหล่งจ่ายให้เพียงพอต่อการจ่ายกระแสให้โหลดด้วย เพราะเนื่องจากถ้าโหลดใช้กระแสมาก แรงดันที่จ่ายให้โหลดจะลดลงจากกรณีที่ไม่มีโหลด วงจรแบ่งแรงดันแบบมีโหลด (Loaded Voltage Divider) ดังภาพที่ 2.16



ภาพที่ 2.21 วงจรแบ่งแรงดันแบบมีโหลด

จากภาพที่ 2.20 มีโหลด (RL) มาต่อคร่อมอยู่กับตัวต้านทาน R2 ดังนั้นในการหาค่าแรงดันไฟฟ้าที่แบ่งมาให้โหลด (RL) เพื่อการคำนวณที่ง่ายและรวดเร็วขึ้นจึงมีสูตรคำนวณเฉพาะที่ใช้กับวงจรแบ่งแรงดันไฟฟ้าแบบมีโหลด (RL) ดังนี้

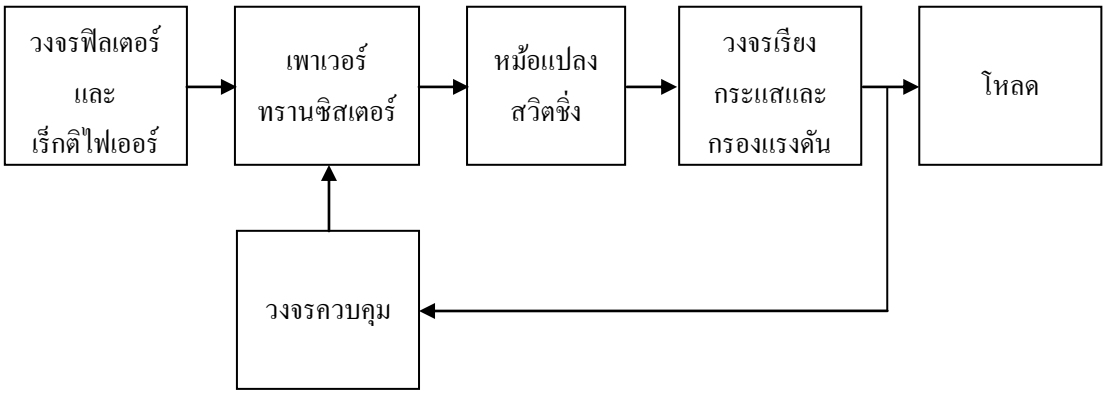
$$V_L = V_T \left\{\frac{R_2 // R_L}{(R_2 // R_L)+R_1}\right\} \qquad (2.4)$$

2.5 สวิตชิ่งเพาเวอร์ซัพพลาย [5]

สวิตชิ่งเพาเวอร์ซัพพลาย (Switching Power Supply) เป็นแหล่งจ่ายไฟตรงคงค่าแรงดันแบบหนึ่ง และสามารถเปลี่ยนแรงดันไฟจากไปสลับVสูง ให้เป็นแรงดันไฟตรงค่าต่ำ เพื่อใช้ในงานอิเลคทรอนิกส์ได้เช่นเดียวกันแหล่งจ่ายไฟเชิงเส้น (Linear Power Supply) ถึงแม้เพาเวอร์ซัพพลายทั้งสองแบบจะต้องมีการใช้หม้อแปลงในการลดทอนแรงดันสูงให้เป็นแรงดันต่ำเช่นเดียวกัน แต่สวิตชิ่งเพาเวอร์ซัพพลายจะต้องการใช้หม้อแปลงที่มีขนาดเล็ก และน้ำหนักน้อย เมื่อเทียบกับแหล่งจ่ายไฟเชิงเส้น อีกทั้งสวิตชิ่งเพาเวอร์ซัพพลายยังมีประสิทธิภาพสูงกว่าอีกด้วย

2.5.1 หลักการทำงานของสวิตชิ่งเพาเวอร์ซัพพลาย

สวิตชิ่งเพาเวอร์ซัพพลายโดยทั่วไปมีองค์ประกอบพื้นฐานที่คล้ายคลึงกัน และไม่ซับซ้อนมากนัก ดังแสดงในภาพที่ 2.21 หัวใจสำคัญของสวิตชิ่งเพาเวอร์ซัพพลายจะอยู่ที่คอนเวอร์เตอร์ เนื่องจากทำหน้าที่ทั้งลดทอนแรงดันและคงค่าแรงดันเอาท์พุทด้วย องค์ประกอบต่างๆ ทำงานตามลำดับดังนี้



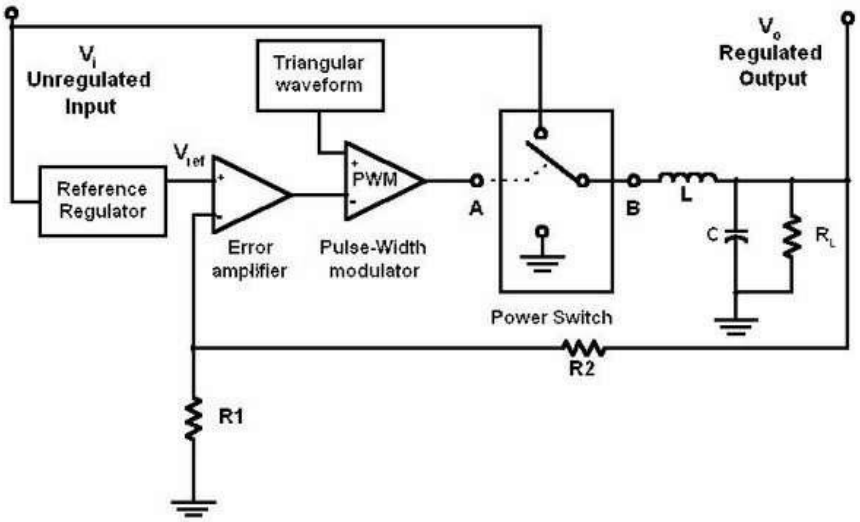
ภาพที่2.22 องค์ประกอบพื้นฐานของสวิตชิ่งเพาเวอร์ซัพพลาย

แรงดันไฟสลับค่าสูงจะผ่านเข้ามาทางวงจร RFI ฟิลเตอร์ เพื่อกรองสัญญาณรบกวนและแปลงเป็นไฟตรงค่าสูงด้วยวงจรเรกติไฟเออร์ เพาเวอร์ทรานซิสเตอร์จะทำงานเป็นเพาเวอร์คอนเวอร์เตอร์โดยการตัดต่อแรงดันเป็นช่วงๆ ที่ความถี่ประมาณ 20 - 200 kHz จากนั้นจะผ่านไปยังหม้อแปลงสวิตชิ่งเพื่อลดแรงดันลง เอาท์พุทของหม้อแปลงจะต่อกับวงจรเรียงกระแส และกรองแรงดันให้เรียบ การคงค่าแรงดันจะทำได้โดยการป้อนกลับคาแรงดันที่เอาท์พุทกลับมายังวงจรควบคุม เพื่อ

ควบคุมให้เพาเวอร์ทรานซิสเตอร์นำกระแสมากขึ้นหรือน้อยลงตามการเปลี่ยนแปลงของแรงดันที่เอาท์พุท ซึ่งจะมีผลทำให้แรงดันเอาท์พุทคงที่ได้ ภาพที่ 2.22 แสดงวงจรซึ่งแบ่งส่วนตามองค์ประกอบหลัก

2.5.1 Basic Switching Regulator Topology

วงจรสวิตชิ่งเรกูเลเตอร์แบบพื้นฐานแสดงไว้ในภาพที่ 2.22 โดยที่ Vin คือ แรงดันอินพุท (ยังไม่ผ่านการเรกูเลต), Vout คือ แรงดันเอาท์พุท (ผ่านการเรกูเลตแล้ว), R_L เป็นโหลด



ภาพที่ 2.23 วงจรสวิตชิ่งเรกูเลเตอร์แบบพื้นฐาน

ในวงจรแบ่งได้เป็นส่วนต่างๆ ดังนี้

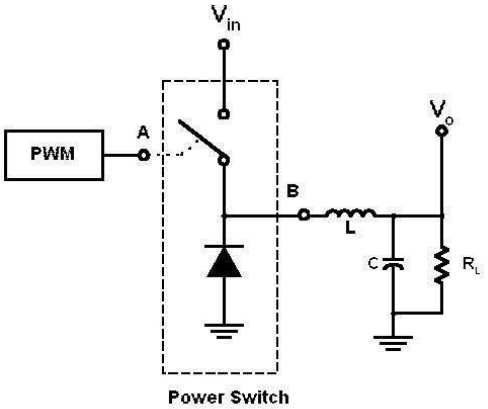
Reference regulator ทำหน้าที่สร้าง Vref ซึ่งเป็นแรงดันอ้างอิงและเป็นไฟเลี้ยงให้กับ Error - amplifier และ Pulse - width modulator ซึ่งทั้ง Reference regulator, Error - amplifier, Pulse - width modulator และส่วนที่ทำหน้าที่สร้างแรงดันคลื่นสามเหลี่ยม สามารถผลิตรวมกันไว้ในไอซีตัวเดียวได้ ดังนั้นส่วนนี้จะเป็นส่วนที่ใช้กำลังต่ำจึงไม่มีผลกระทบกับประสิทธิภาพรวมของระบบ

Error - amplifier ทำหน้าที่เปรียบเทียบ Vref กับค่าแรงดันที่ได้จากการป้อน Vo กลับมาผ่าน R1, R2 แล้วขยายความแตกต่างแล้วส่งเป็นเอาท์พุท Vm ให้กับ Pulse - width modulator (PWM)

PWM ทำหน้าที่มอดูเลต Vm กับแรงดันคลื่นสามเหลี่ยมที่มีคาบเท่ากับ T เพื่อสร้างเป็นแรงดัน VA ที่เป็นคลื่นสี่เหลี่ยมมีคาบ T มี duty cycle แปรผันตรงกับ Vm

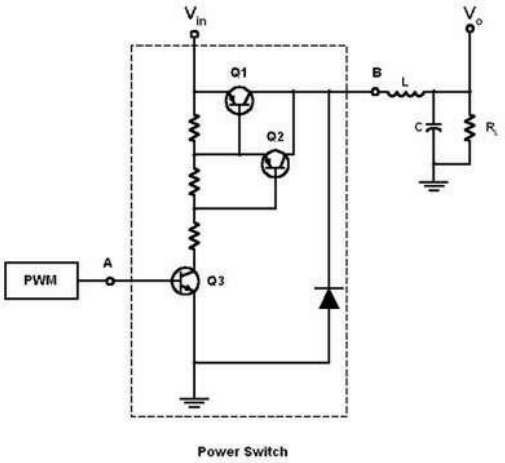
Power switch เป็นสวิตช์เดี่ยวแบบสองทาง (Single Pole - Double Throw, SPDT) ทำหน้าที่สวิตช์ระหว่าง Vin และกราวด์ การสวิตช์ควบคุมโดย VA ซึ่งเราจะได้ VB เป็นแรงดันคลื่นสี่เหลี่ยมที่มีคาบ, มี duty cycle เท่ากับ VA และมีแอมพลิจูดเป็น Vin

LC Filter ทำหน้าที่เป็น Low - pass filter โดยกำจัดองค์ประกอบที่มีความถี่สูงใน VB ไม่ให้ผ่านไปที่โหลด ดังนั้นจึงมีเฉพาะกระแสตรงเท่านั้นที่ผ่านไปได้ Vo จึงมีค่าคงที่



ภาพที่ 2.24 ใช้ไดโอดและสวิตช์เดี่ยวทางเดียวที่ควบคุมโดย PWM

วงจรในภาพที่ 2.23 ใช้ไดโอดและสวิตช์เดี่ยวทางเดียว (Single Pole - Single Throw, SPST) ที่ควบคุมโดย PWM เมื่อสวิตช์ปิดไดโอดจะไม่นำกระแส กระแสจึงไหลผ่านตัวเหนี่ยวนำและได้ VB = Vin เมื่อสวิตช์เปิดกระแสที่ผ่านตัวเหนี่ยวนำจะยังคงที่อยู่ ดังนั้นจะมีกระแสไหลจากกราวด์ผ่านไดโอดไปยังตัวเหนี่ยวนำและโหลด ถ้าไม่คิดความต่างศักย์ที่คร่อมไดโอด จะได้ VB = 0 เราเรียกไดโอดที่ทำงานในลักษณะนี้ว่า Flyback diode



ภาพที่ 2.25 ใช้ทรานซิสเตอร์ที่ควบคุมโดย PWM

สวิตช์เดี่ยวทางเดียวในภาพที่ 2.23 สามารถแทนได้ด้วยทรานซิสเตอร์ Q1 ในภาพที่ 2.24 สำหรับในกรณีที่โหลดต้องการกระแสมาก เราจะต้องต่อทรานซิสเตอร์เป็นแบบดาร์ลิงตัน (Darlington pair) โดยใช้ Q2 เป็นตัวช่วยในการดึงกระแสเบสของ Q1 ให้มากขึ้น

จากการที่เราใช้ทรานซิสเตอร์แบบ PNP แทนสวิตช์ และเราต้องการให้สวิตช์ปิดเมื่อ VA มีค่าสูงเราจึงต้องใช้ Q3 ที่เป็นชนิด NPN ในการควบคุมสวิตช์ให้เปิดปิดถูกต้อง เมื่อ VA เป็นบวก Q3 จะทำงานกระแสคอลเลคเตอร์ของ Q3 จะไหลผ่านตัวต้านทานแต่ละตัวทำให้ Q1 และ Q2 ทำงาน ซึ่งจะได้ VB เกือบเท่ากับ Vin เมื่อ VA เป็นลบหรือศูนย์ Q3 จะไม่ทำงานจึงไม่มีกระแสไปกระตุ้น Q1 และ Q2 ทำให้สวิตช์เปิด Flyback diode ก็จะทำหน้าที่เป็นทางผ่านให้กับกระแสที่ไหลจากกราวด์ผ่านไปยังตัวเหนี่ยวนำ การที่ LC Filter จะมีประสิทธิภาพในการกันองค์ประกอบที่มีความถี่สูง ค่าของ reactance ของ C จะต้องน้อยกว่าค่าของ Reactance ของ L มากๆ ที่ความถี่มูลฐาน ดังนั้น $T/(2\pi C) << 2\pi L/T$ ถ้าเราพิจารณาสมการของ Error - amplifier จะได้ว่า

$$V_{ref} = \frac{(R_2 \times V_O)}{(R_1 + R_2)} \tag{2.5}$$

หรือ $$V_o = V_{ref}(1 + \frac{R_1}{R_2}) \tag{2.6}$$

ซึ่งจะเห็นว่า Vo ไม่ขึ้นกับ Vin และกระแสโหลด แต่จากการที่ Vo เป็นค่าเฉลี่ยของ VB ซึ่ง VB มีค่าสูงสุดเป็น Vin ดังนั้น Vo ที่ได้จะน้อยกว่า Vin เสมอ

ในด้านของประสิทธิภาพวงจรที่สร้างโดยหลักการนี้ถ้าใช้ Power switch ที่มีการสูญเสียต่ำ เช่น ทรานซิสเตอร์ที่มี $V_{CE(SAT)}$ ต่ำและมีความเร็วในการสวิตช์สูง และใช้ ตัวเหนี่ยวนำที่มีค่า Q (Quality factor) สูง หรือมีค่าความต้านทานภายในต่ำจะได้ประสิทธิภาพไม่ต่ำกว่า 90 เปอร์เซนต์

2.5.2 Additional Switching Regulator Topologies

จากวงจรในภาพที่ 2.25 แรงดันเอาท์พุทที่ได้จะน้อยกว่าแรงดันอินพุทและมีค่าเป็นบวกเสมอ เราจึงคิดหาทางแก้ไขข้อจำกัดอันนี้ โดยวงจรในภาพที่ 2.26 เป็นวงจรที่สร้างแรงดันเอาท์พุทที่เป็นบวกและมีค่ามากกว่า Vin

ในช่วงที่สวิตช์ปิด (สมมติว่าปิดเป็นเวลา T1) ไดโอดจะไม่นำกระแส ในช่วงนี้ตัวเก็บประจุจะคายประจุให้กับโหลด โดยเราจะต้องเลือกให้ $C \times R_L >> T1$ เพื่อให้ค่า Vo เปลี่ยนแปลงน้อย ในช่วงนี้แรงดันคร่อมตัวเหนี่ยวนำ คือ Vin จะทำให้กระแส I_L เพิ่มขึ้นจนถึงช่วงที่สวิตช์เปิดตามสมการ

$$di = V_{in} \times \frac{dt}{L} = V_{in} \times \frac{T_1}{L} \tag{2.7}$$

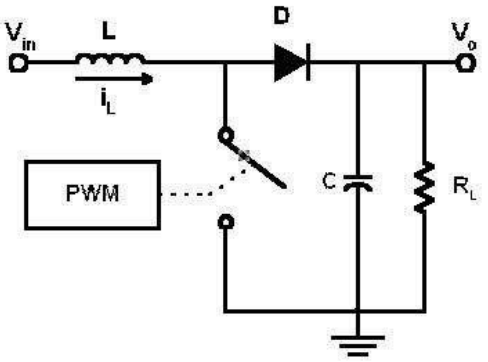
ในช่วงที่สวิตช์เปิด(สมมติว่าเปิดเป็นเวลา T2) กระแสที่ไหลในตัวเหนี่ยวนำไม่สามารถเปลี่ยนแปลงในทันที ดังนั้นกระแสที่ไหลในขณะสวิตช์เริ่มเปิดยังคงมีค่าคงเดิม มีผลทำให้ไดโอดนำกระแส กระแสจากตัวเหนี่ยวนำจึงไหลไปยังตัวเก็บประจุและโหลด ในช่วงเวลานี้กระแสของตัวเหนี่ยวนำจะลดลงเท่ากับ Vin × T1/L ทำให้แรงดันคร่อมตัวเหนี่ยวนำมีค่าเป็นลบ (Ldi/dt < 0) ดังนั้น

$$V_o = V_{in} - \frac{Ldi}{dt} \tag{2.8}$$

$$= V_{in} + L \times \frac{(V_{in} \times T_1 / L)}{T_2}$$

$$= V_{in} + (1 + \frac{T_1}{T_2}) > V_{in}$$

เมื่อวงจรนี้เข้าสู่ภาวะสมดุล ค่าของ Vo จะมีค่ามากกว่า Vin คือมีค่าเท่ากับ Vin×(1+T1/T2) เมื่อสวิตช์ปิดค่าของ Vo จะลดลงเล็กน้อยขึ้นอยู่กับค่าของ C×R_L ดังนั้นเราสามารถควบคุมค่าของ Vo โดยควบคุม duty cycle ของ PWM



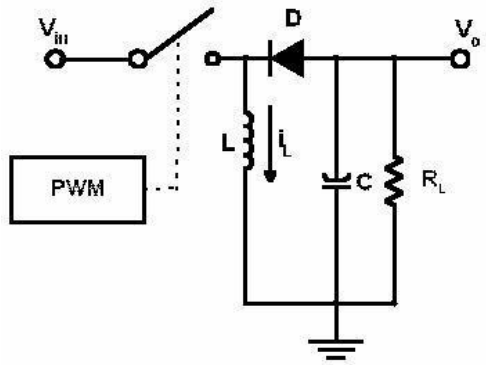
ภาพที่ 2.26 เป็นวงจรที่สร้างแรงดันเอาท์พุทที่เป็นบวก

วงจรในภาพที่ 2.25 เป็นวงจรที่ให้ค่า Vo เป็นลบ ซึ่งหลักการของวงจรคล้ายกับหลักการของวงจรในภาพที่ 2.24 คือ ในช่วงที่สวิตช์ปิดเป็นเวลา T1 ไดโอดจะไม่นำกระแส ตัวเก็บประจุจะคายประจุให้กับโหลดและแรงดันคร่อมตัวเหนี่ยวนำ คือ Vin จะทำให้กระแสที่ไหลผ่านตัวเหนี่ยวนำมีค่าเพิ่มขึ้นเท่ากับ Vin×T1/L ในช่วงที่สวิตช์เปิดเป็นเวลา T2 ดังนั้นไดโอดจะถูกบังคับให้นำกระแส กระแสของตัวเหนี่ยวนำ จะไหลไปยังตัวเก็บประจุและโหลดทางขั้วที่อยู่ด้านล่าง โดยค่าของกระแสนี้จะลดลงเท่ากับ Vin×T1/L ในช่วงเวลา T2 ดังนั้น

$$V_O = \frac{Ldi}{dt} \tag{2.9}$$

$$= L \times \frac{(-V_{in} \times T_1 / L)}{T_2}$$

$$= -V_{in} \times \frac{T_1}{T_2}$$



ภาพที่ 2.27 เป็นวงจรที่ให้ค่า Vo เป็นลบ

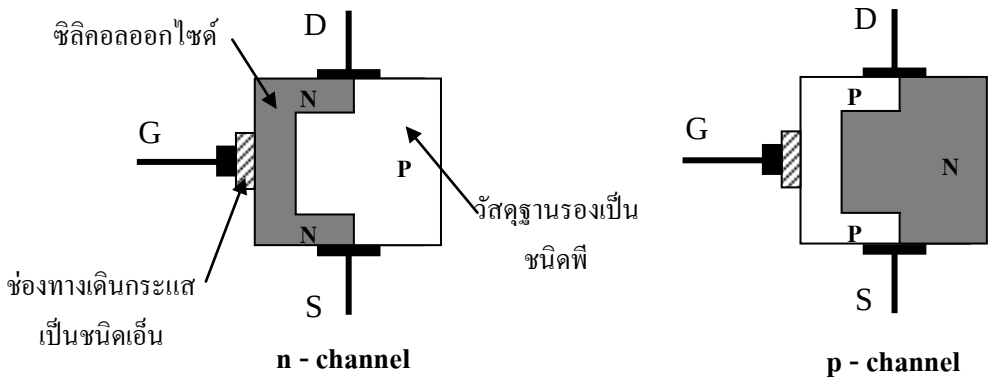
2.6 เฟตชนิดออกไซด์ของโลหะ [6] [7]

MOSFET แตกต่างจาก JFET ที่โครงสร้างภายใน JFET นั้นระหว่างเกตกับช่วงทางเดินกระแสมีโครงสร้างเป็นรอยต่อ p - n แต่ MOSFET นั้นระหว่างเกตกับช่องทางเดินกระแสมีโครงสร้างเป็นชั้นของซิลิคอลออกไซด์ (SiO2) MOSFET มี 2 ชนิด คือ MOSFET

1. ดีพีทชั่น (Depletion, D)
2. เอนแฮนซ์เมนต์ (Enhancement, E)

2.6.1 MOSFET ชนิดดีพีทชั่น (Depletion MOSFET, D - MOSFET)

จากภาพที่ 2.27 คือโครงสร้างพื้นฐานของ D - MOSFET เป็นชนิด (n - channel) ช่องทางเดิน กระแส ระหว่างเดรน และซอร์สจะเป็น สารกึ่งตัวนำชนิด n และวัสดุฐานรอง (Substance) เป็นสารกึ่งตัวนำชนิด p ดังภาพที่ 2.27 สำหรับ D - MOSFET ชนิด n - channel จะมีช่องทางเดินกระแสระหว่างเดรนและซอร์สเป็นสารชนิด p และมีฐานรองเป็นสารชนิด n ดังภาพที่ 2.27 และมีเกตติดอยู่ระหว่างช่องทางเดินกระแส โดยมีซิลิคอนออกไซด์ (SiO2) เป็นฉนวนกั้นระหว่างเกตและช่องทางเดินกระแส



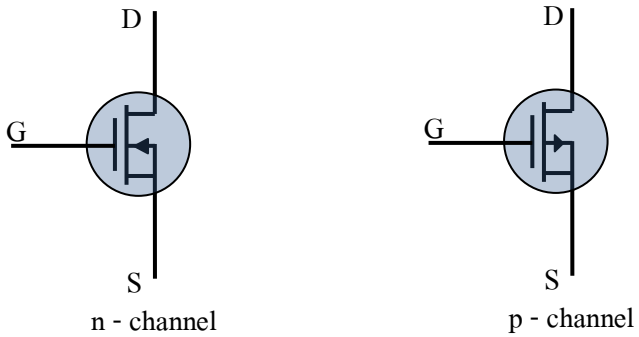
ภาพที่ 2.28 โครงสร้างพื้นฐาน D - MOSFET

เนื่องจาก D - MOSFET ทำงานได้ใน 2 ลักษณะคือ

1. ดีพีทชันโหมด (Depletion Mode)
2. เอนแฮนซ์โหมด (Enhancemnet Mode)

ดีพีทชันโหมด (Depletion Mode) ทำงานด้วยหลักการของคุณสมบัติตัวเก็บประจุ เนื่องจากเกตของ D - MOSFET เป็นโลหะและมีฉนวน (SiO2) กั้นกลางระหว่างเพลทของเกตกับแซนเนลของ D - MOSFET เพื่อให้แรงดันเกตเป็นลบ ($-V_{GG}$) จะเกิดประจุลบที่เกตและประจุบวกที่แซนเนลภายในตัว D - MOSFET ประจุบวกจะทำให้ทางเดินของกระแสในแซนเนลระหว่างเดรนกับซอร์สแคบลงทำให้กระแสเดรน (I_D) ไหลได้น้อย แต่ถ้าให้ V_{GG} = 0V กระแสเดรนจะไหลได้สูงสุด ดังนั้นสามารถควบคุมกระแสเดรน (I_D) ได้ด้วยค่าแรงดันเกตระหว่าง $V_{GG} = V_{GG}$ (OFF) จนถึง 0V

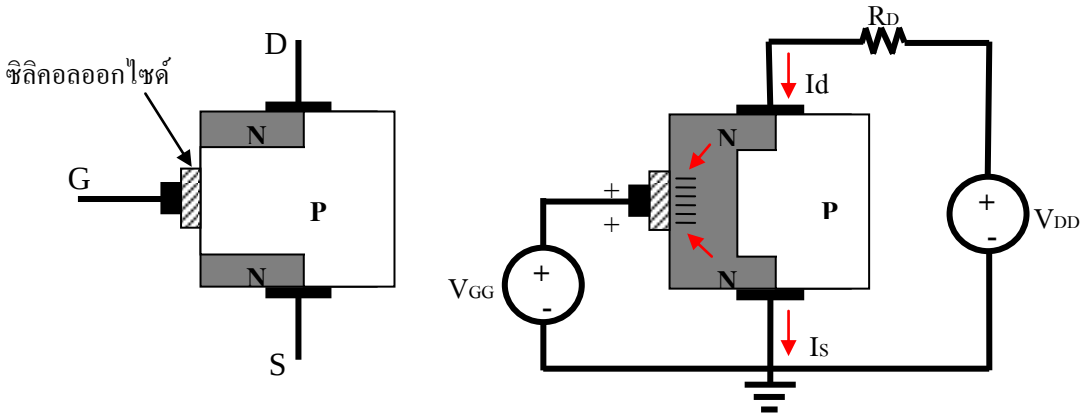
เอนแฮนซ์โหมด (Enhancement Mode) เอนแฮนซ์โหมดคือการไบอัสของ D - MOSFET ด้วยแรงดันบวกดังภาพที่ 2.27 ขาเกตของ D - MOSFET จะได้รับประจุบวกจากแหล่งจ่าย V_{GG} ทำให้ในแซนเนลของ D - MOSFET เป็นประจุลบ ทำให้ช่องทางเดินกระแสระหว่างเดรนกับซอร์สไม่มี ประจุชนิดตรงข้ามกับแซนเนล คอยบีบแซนเนลให้แคบลง ทำให้กระแสเดรน (I_D) ไหลได้จำนวนมาก ถ้าให้ V_{GG} = 0 V จะทำให้กระแสเดรน (I_D) ไหลได้น้อยลงเพราะประจุลบในแซนเนลมีค่าลดลงเป็น 0



ภาพที่ 2.29 สัญลักษณ์ของ D - MOSFET

2.6.2 เอนแฮนซ์เมนมอสเฟต (Enhancement MOSFE, E - MOSFET)

เอนแฮนซ์เมนมอสเฟต ทำงานได้ลักษณะของเอนแฮนซ์เมนโหมดเพียงอย่างเดียว ไม่สามารถทำงานในดีพีทชันโหมดได้ โครงสร้าง E - MOSFET แตกต่างจาก D - MOSFET ที่ช่องทางเดินกระแสของ E - MOSFET จะถูกสร้างขึ้นโดยการไบอัสที่เกต ในสภาวะที่เกตไม่มีการไบอัสจะไม่มีช่องทางเดินกระแสเชื่อมต่อระหว่างเดรนกับซอร์ส ดังภาพที่ 2.29 เป็น E - MOSFET (n - channel) ส่วนเดรนและซอร์สเป็นสารกึ่งตัวนำชนิดเอ็น แต่ไม่มีแชนเนลต่อถึงกัน มีสารชนิดพีเป็นวัสดุฐานรองและระหว่างเกตกับวัสดุฐานรองเป็น (SiO2) เป็นฉนวนกั้นกลาง



ภาพที่ 2.30 โครงสร้างและการทำงานของ E - MOSFET

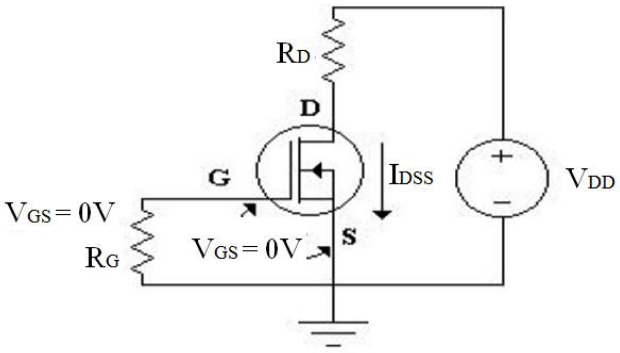
การทำงานของ E - MOSFET ที่เพลตของเกตจะเกิดประจุบวก และวัสดุฐานรองของ E - MOSFET จะเกิดประจุลบ ทำให้เกิดประจุลบเหนี่ยวนำขึ้นเป็นช่องทางเดินกระแสเชื่อมต่อระหว่างเดรนกับซอร์ส ทำให้กระแสเดรน (I_D) สามารถไหลข้ามช่องทางเดินกระแสซอร์สได้ จะไหลมากหรือน้อยขึ้นอยู่กับขนาดของแรงดันไบอัสที่เกตของ E - MOSFET เพราะขนาดของแชนเนลขึ้นอยู่กับขนาดของ V_{GG}

2.6.3 การไบอัส MOSFET

การไบอัส D - MOSFET สามารถทำงานได้ด้วยแรงดันเกต - ซอร์สเป็นบวกหรือลบ วิธีที่จะไบอัส D – MOSFET ทำได้โดยไบอัสด้วยแรงดันศูนย์ คือ $V_{GS} = 0V$ ดังภาพที่ 2.30

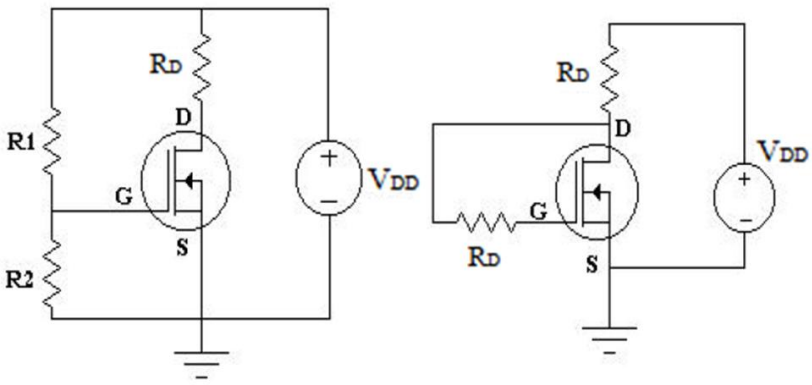
กรณี $$V_{GS} = VI_D = I_{DSS} \quad (2.10)$$

ดังนั้น $$V_{DS} = V_{DD} - I_{DSS}R_D \quad (2.11)$$



ภาพที่ 2.31 วงจรไบอัส D - MOSFET แบบไบอัสด้วยแรงดันศูนย์

การไบอัส E - MOSFET การควบคุมแรงดัน V_{GS} ให้มีค่ามากกว่าแรงดันเทรซโฮล $V_{GS(TH)}$ จากภาพที่ 2.31 สามารถไบอัส E - MOSFET ได้ 2 วิธีคือไบอัสแบบแบ่งแรงดัน โดยใช้ R1 และ R2 ดังภาพที่ 2.31 และไบอัสด้วยการป้อนกลับที่เดรน ด้วยการต่อ R_G เข้ากับเดรนของ E - MOSFET ดังภาพที่ 2.31



ภาพที่ 2.32 วงจรไบอัส E - MOSFET

วงจรไบอัสแบบแบ่งแรงดันในภาพที่ 2.31 สามารถคำนวณหาค่าของ V_{GS} ได้ดังสมการ

$$V_{GS} = \frac{R_2}{(R_1 + R_2)} V_{DD} \tag{2.11}$$

$$V_{DS} = V_{DD} - I_D R_D \tag{2.12}$$

เมื่อ

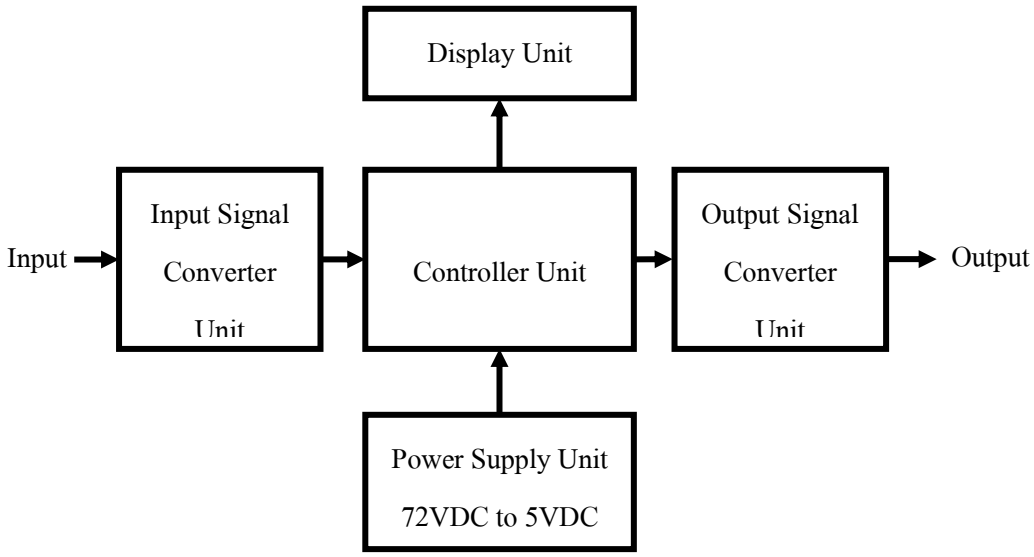
$$I_D = k(V_{GS} - V_{GS(TH)})^2 \tag{2.13}$$

และ k หาได้จาก

$$k = \frac{I_{D(ON)}}{(V_{GS(ON)} - V_{GS(TH)})^2} \tag{2.14}$$

บทที่ 3
การออกแบบ

การออกแบบของเครื่องควบคุมระบบป้องกันพนักงานรถจักรหมดสติของรถจักรดีเซลไฟฟ้าของการรถไฟแห่งประเทศไทย แบ่งออกเป็น 5 ส่วนได้แก่ ชุดแปลงสัญญาณด้านอินพุท ชุดควบคุม ชุดแหล่งจ่าย ชุดแสดงผล และแปลงสัญญาณด้านเอาท์พุท ดังภาพที่ 3.1



ภาพที่ 3.1 บล็อกไดอะแกรมแสดงส่วนต่างๆของระบบ

3.1 การออกแบบชุดแปลงสัญญาณด้านอินพุท (Input Signal Converter Unit)

สัญญาณอินพุทเป็นไฟฟ้ากระแสตรงแรงดัน 72V และ 0V ซึ่งสัญญาณอินพุทนี้มาจากการรีเซทตามจุดต่างๆดังนี้

- Speed Meter จะมีสัญญาณแรงดันไฟฟ้ากระแสตรง 72V ส่งไปยังระบบ Vigilance เมื่อความเร็วของรถไฟต่ำกว่า 8 km/hrs. และเป็น 0V เมื่อความเร็วเป็น 8 km/hrs.ขึ้นไป
- Engine Speed Up จะมีสัญญาณแรงดันไฟฟ้ากระแสตรง 72V ส่งไปยังระบบ Vigilance เมื่อมีการรีเซท
- Engine Speed Down จะมีสัญญาณแรงดันไฟฟ้ากระแสตรง 72V ส่งไปยังระบบ Vigilance เมื่อมีการรีเซท

- Foot Switch and Push Button Switch จะมีสัญญาณแรงดันไฟฟ้ากระแสตรง 72V ส่งไปยังระบบ Vigilance เมื่อมีการรีเซท

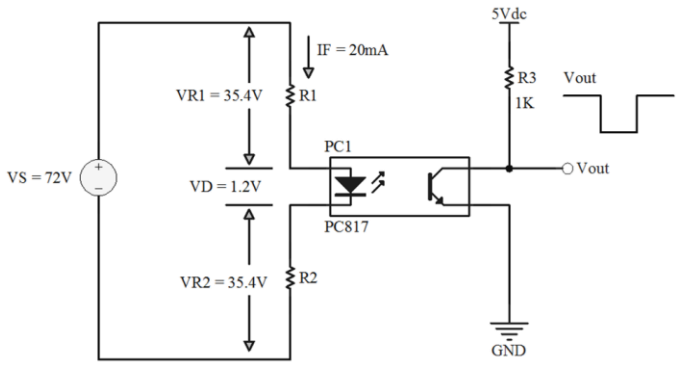
- Sending Switch จะมีสัญญาณแรงดันไฟฟ้ากระแสตรง 0V เมื่อมีการกดปุ่ม

เนื่องจากว่าสัญญาณอินพุทมีขนาดแรงดันเป็น 72V จึงต้องทำการเปลี่ยนให้เป็นสัญญาณดิจิตอลในระดับแรงดัน 5V เพื่อส่งไปยังชุดควบคุม ซึ่งเป็นไมโครคอนโทรลเลอร์ โดยการเปลี่ยนสัญญาณอินพุทนี้ ใช้วงจร Opto - Isolator หลักการทำงานคือ เมื่อ LED ของออปโต้ทำงานจะทำให้ทรานซิสเตอร์ของออปโต้ทำงานด้วย ซึ่งในส่วนทรานซิสเตอร์นั้นได้ต่อไฟ 5V รอไว้อยู่เมื่อออปโต้ทำงานจะมีสัญญาณดิจิตอลลอจิก “0” และในสภาวะที่ออปโต้ไม่ทำงานจะเป็นเป็นลอจิก “1” ส่งไปยังพอร์ตอินพุทของชุดควบคุม

การคำนวณวงจรแบ่งแรงดัน เพื่อให้คำนวณหาค่าต่างๆได้ง่ายจึงได้แบ่งแรงดันทั้งสองด้านให้เท่ากัน และจาก Datasheet ของ PC817 ในสภาวะของ I_F = 20mA จะได้ Forward Voltage เท่า 1.2V ดังนั้นการแบ่งแรงดันให้เท่ากันทั้งสองด้านจะได้

$$V_{R1} = V_{R2} = \frac{(72-1.2)}{2} \tag{3.1}$$

$$V_{R1} = 35.4V$$



ภาพที่ 3.2 วงจรชุดแปลงสัญญาณอินพุท (Input Signal Converter Circuit)

ดังนั้น R1 และ R2 จะได้

$$R_1 = R_2 = \frac{V_{R1}}{I_F} \tag{3.2}$$

$$= \frac{34.4V}{20mA} = 1.7k\Omega$$

หากำลังไฟฟ้าที่ R1 และ R2 จะได้ $P_{R1} = P_{R2} = I^2 R1$ (3.3)

$$= 20mA^2 \times 1.77k\Omega = 0.71\text{W}$$

ดังนั้นเลือกใช้ R1 และ R2 = 1. 8$k\Omega$ 1W

3.2 การออกแบบชุดควบคุม (Controller Unit)

ชุดควบคุมเป็นตัวความคุมการทำงานของเอาท์พุทได้แก่ แตรไฟฟ้า กระดิ่งไฟฟ้า และห้ามล้อฉุกเฉิน ซึ่งการทำงานของเอาท์พุทเหล่านี้จะขึ้นอยู่กับอินพุทที่ป้อนให้ชุดควบคุม ชุดควบคุมได้ออกแบบโดยเลือกใช้ไมโครคอนโทรลเลอร์ MCS - 51 เบอร์ AT89C51 ซึ่งเป็นเบอร์ที่นิยมใช้กันมากในตระกูล MCS - 51

การกำหนดพอร์ตไมโครคอนโทรลเลอร์ของชุดควบคุมได้กำหนดพอร์ตที่ 0 พอร์ตที่ 1 และพอร์ตที่ 3 เป็นพอร์ตเอาท์พุทในส่วนของการแสดงผลการนับเวลา ส่วนพอร์ตที่ 1 กำหนดเป็นทั้งอินพุทและเอาท์พุทร่วมกันดังนี้

P1.0 รับสัญญาณอินพุทจากชุดแปลงสัญญาณอินพุท Speed Meter

P1.1 รับสัญญาณจากชุดแปลงสัญญาณอินพุท Engine Speed Up

P1.2 รับสัญญาณจากชุดแปลงสัญญาณอินพุท Engine Speed Down

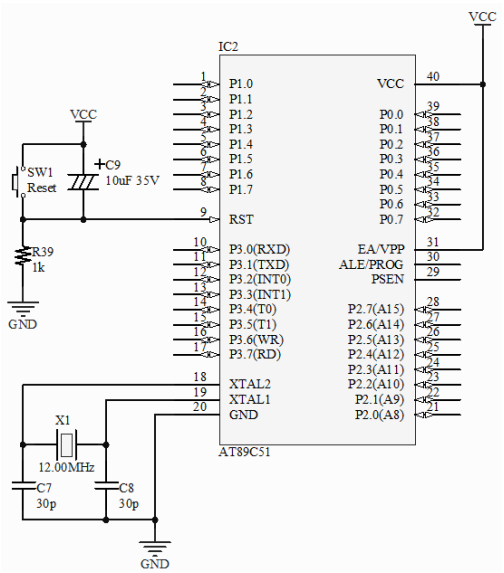
P1.3 รับสัญญาณจากชุดแปลงสัญญาณอินพุท Foot Switch and Push Button Switch

P1.4 รับสัญญาณอินพุทจาก Sending Switch

P1.5 ส่งสัญญาณเอาท์พุทไปยังชุดแปลงสัญญาณเอาท์พุทเพื่อสั่งให้สัญญาณเสียงเตือนชุดที่ 2 ทำงาน

P1.6 ส่งสัญญาณเอาท์พุทไปยังชุดแปลงสัญญาณเอาท์พุทเพื่อสั่งให้สัญญาณเสียงเตือนชุดที่ 1 ทำงาน

P1.7 ส่งสัญญาณเอาท์พุทไปยังชุดแปลงสัญญาณเอาท์พุทเพื่อสั่งห้ามล้อฉุกเฉิน



ภาพที่ 3.3 วงจรชุดควบคุม (Controller Circuit)

สำหรับวงจรชุดความคุมมีส่วนที่ต้องคำนวณคือวงจรผลิตความถี่ เนื่องจากการทำงานของระบบ Vigilance มีส่วนของการนับเวลาอยู่ด้วย ดังนั้นเพื่อความแม่นยำในการนับเวลาจึงเลือกใช้คริสตอลความถี่ 12MHz ผลิตความถี่ป้อนให้ไมโครคอนโทรลเลอร์ สัญญาณนาฬิกาพัลซ์ที่ได้ 12MHz/12 = 1MHz จากสูตร t = 1/f ดังนั้นคาบเวลาที่ได้คือ 1 ไมโครวินาที

ในโปรแกรมควบคุมจะมีส่วนที่ต้องคำนวณก็คือโปรแกรมย่อยสำหรับนับเวลา การนับเวลาได้ใช้ไทเมอร์ในไมโครคอนโทรลเลอร์ การคำนวณเวลาเป็นดังนี้

ต้องการนับเวลาขึ้นทีละ 1 วินาที โดยใช้ไทเมอร์ 0 โหมด 1 สร้างเวลา 10,000 ไมโครวินาที และให้เกิดการวนซ้ำกัน 100 รอบ ซึ่งจะได้เวลา 1,000,000 ไมโครวินาที หรือ 1 วินาที ค่าสูงสุดของการนับในไทเมอร์ 0 โหมด 1 คือ FFFF H นั่นคือสัญญาณพัลซ์ 65,536 ลูก หรือ 65,536 ไมโครวินาที ค่าที่ต้องการคือ 65,536 - (10,000/1) = 55,536 ไมโครวินาที หรือ D8F0 H เมื่อได้แล้ว นำข้อมูลไบต์ด้านสูง D8 กำหนดให้ TH0 และข้อมูลไบต์ด้านต่ำ F0 กำหนดให้ TL0 สามารถเขียนโปรแกรมภาษาซีกำหนดค่าเริ่มต้นนับของไทเมอร์ 0 ได้ดังนี้

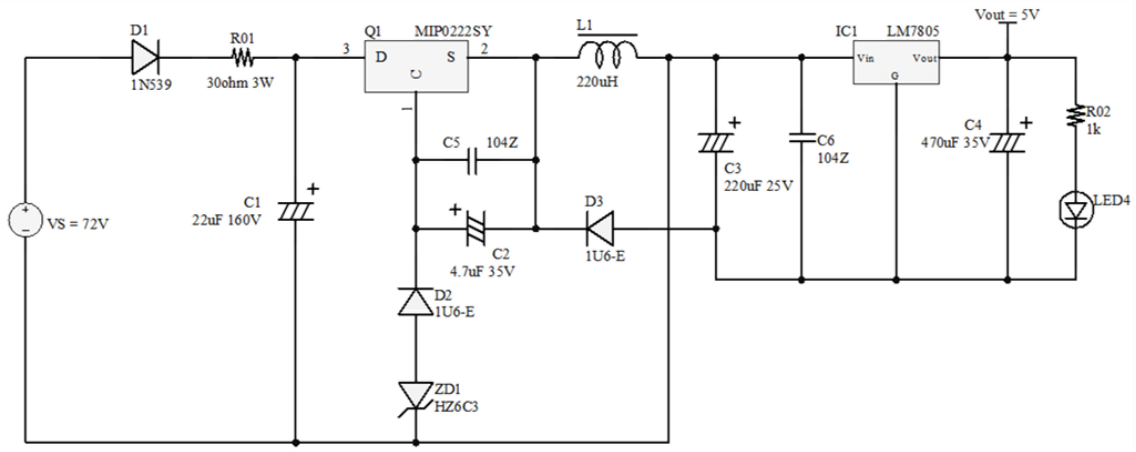
TH0 = 0xD8; และ TL0 = 0xF0;

และโปรแกรมย่อยในส่วนของการนับเวลาสามารถเขียนได้ดังนี้(โปรแกรมส่วนอื่นดูในภาคผนวก)

```
void count_1second()
{
unsigned char x;
for(x=0;x<100;x++)          //วน 100 รอบ x 10,000 ไมโครวินาที จะได้ 1 วินาที
{
TMOD = 0x01;                //เลือกใช้ไทเมอร์ 0โหมด 1 (TMOD = 0000 0001)
TH0 = 0xD8;                 //ค่าที่ต้องการ 65,536 - (10,000/1) = D8F0 H
TL0 = 0xF0;                 //ป้อนค่าข้อมูลไบต์ด้านสูง - ต่ำ (TH0, TL0)
TR0 = 1;                    //สั่งให้ไทเมอร์ 0 เริ่มทำงาน
while(TF0==0);              //รอให้ไทเมอร์เกิดโอเวอร์โฟลว์ TF = 1
TR0=0;                      //เมื่อเกิดโอเวอร์โฟลว์แล้ว สั่งให้ไทเมอร์หยุดทำงาน TR = 0
TF0 = 0;                    //เคลียร์บิตแฟลกโอเวอร์โฟลว์
}
}
```

3.3 การคำนวณชุดแหล่งจ่าย (Power Supply Unit)

ชุดแหล่งจ่ายออกแบบขึ้นเพื่อจ่ายไฟเลี้ยงวงจรของเครื่องควบคุมระบบป้องกันพนักงานรถจักรหมดสติของรถจักรดีเซลไฟฟ้าของการรถไฟแห่งประเทศไทยได้แก่ชุดแปลงสัญญาณอินพุท ชุดควบคุม ชุดแสดงผล และแปลงสัญญาณเอาท์พุท โดยการออกแบบวงจรชุดแหล่งจ่ายใช้หลักการของวงจร สวิตช์ชิ่งเพาเวอร์ซัพพลาย



ภาพที่ 3.4 วงจรชุดแหล่งจ่าย (Power Supply Circuit)

ค่าคงที่จากตัวอุปกรณ์สวิตช์ชิ่ง (MIP0222SY)

แรงดันอินพุท (Input Voltage)	72Vdc
แรงดันเอาท์พุท (Output Voltage)	12Vdc
กระแสเอาท์พุท(Output Current, $I_{o\,max}$)	300 mAdc
ความถี่สวิตช์ชิ่ง(Oscillating Frequency)	100kHz ± 10kHz

คำนวณค่าตัวเหนี่ยวนำเพื่อหาค่า L

$$L = \frac{(V_{in} - V_{out})V_{out}}{(2I_{outmax}V_{in}f_{osc})} \tag{3.4}$$

$$L = \frac{(72V - 12V)12V}{(2 \times 300mA \times 72V \times 110kHz)}$$

$$L = 184.1\mu H$$

เลือกใช้ตัวเหนี่ยวนำ L ค่า 220μH เพราะฉะนั้นแรงดันเอาท์พุท Vout จะได้

$$V_{out} = \frac{2I_{outmax} \times V_{in} f_{osc} L}{V_{in} - 1} \tag{3.5}$$

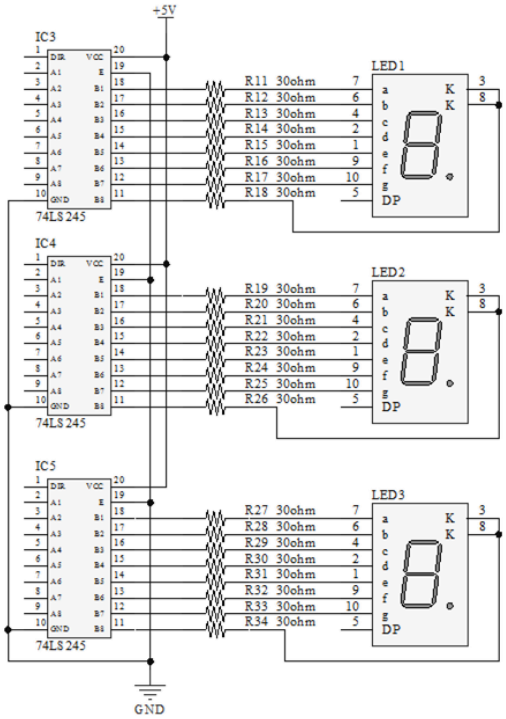
$$V_{out} = \frac{2 \times 300mA \times 72V \times 110kHz \times 220\mu H}{72V - 1}$$

$$V_{out} = 11.29V$$

ต้องการไฟเลี้ยงวงจร 5Vdc ดังนั้นในส่วนของแรงดันเอาท์พุทจึงใช้ IC Regulate 78L05 เพื่อลดแรงดันเอาท์พุทจาก 11.29V ให้เหลือ 5 V

3.4 ชุดแสดงผล (Display Unit)

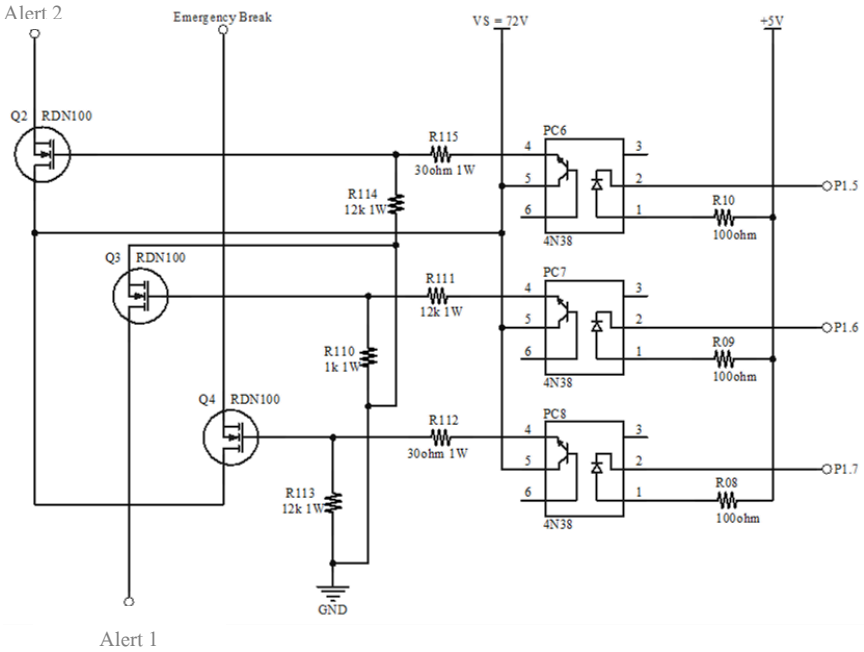
ในการแสดงผลเป็นส่วนที่แสดงการนับเวลา เลือกใช้ 7 Segment แสดงการนับเวลา เนื่องจากมีการนับเวลาสูงสุดที่ 102 วินาที จึงต้องใช้ 7 Segment แสดงถึง 3 หลัก เนื่องจากแต่ละขาพอร์ตของไมโครคอนโทรลเลอร์สามารถจ่ายกระแสได้สูงสุด 10mA เพื่อไม่ให้มีปัญหาในการจ่ายกระแสให้กับชุดแสดงผลที่เชื่อมต่ออยู่กับพอร์ตจึงได้ใช้ IC เบอร์ 74LS245 ช่วยขับกระแสต่อกับขาพอร์ตที่ต่อไปยังชุดแสดงผลดังแสดงในภาพที่ 3.5



ภาพที่ 3.5 วงจรชุดแสดงผล (Display Circuit)

3.5 ชุดแปลงสัญญาณเอาท์พุท (Output Signal Converter)

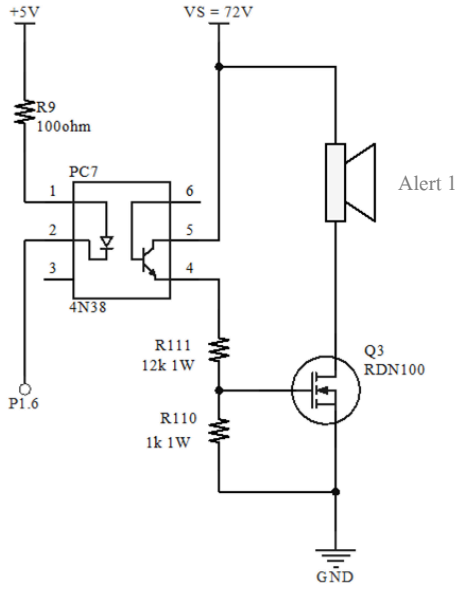
ชุดแปลงสัญญาณเอาท์พุท หลักการคือเปลี่ยนสัญญาณเอาท์พุทที่มีขนาดแรงดันประมาณ 5V เป็น 72V โดยอาศัยวงจร Opto - Isolator เป็นสวิทช์ให้สามารถส่งผ่านแรงดันไฟฟ้ากระแสตรง 72V ไปยังเอาท์พุทได้



ภาพที่ 3.6 วงจรชุดแปลงสัญญาณเอาท์พุท (Output Signal Converter Circuit)

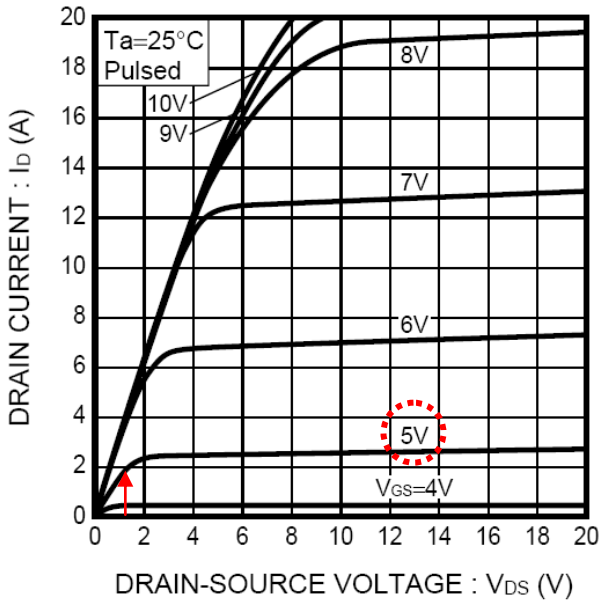
จากวงจรชุดแปลงสัญญาณเอาท์พุท เนื่องจากว่าแรงดัน V_S = 72V จึงต้องใช้ออปโต้เบอร์ 4N38 ซึ่งมีคุณสมบัติเหมาะสมกับระดับแรงดัน 72V คือในช่วงที่ทรานซิสเตอร์ไม่นำกระแส V_{CE} สามารถทนได้ถึง 80V แต่ทนกระแสเพียง 80mA ซึ่งไม่สามารถทนกระแสโหลดที่มีกระแสประมาณ 350mA ได้ ดังนั้นจึงใช้ออปโต้เป็นเพียงสวิตช์ส่งผ่านแรงดันไปไบอัสมอสฟอสเท่า และเนื่องจากกระโหลดมีประมาณ 350mA จึงได้เลือกมอสเฟตเบอร์ RDN100 ซึ่งมีคุณสมบัติทนต่อแรงดันและกระแสในระดับที่โหลดทำงานได้ เป็นสวิตช์อีกชั้นหนึ่ง

วงจรชุดแปลงสัญญาณเอาท์พุทของโหลดสัญญาณเสียงเตือนชุด 1 (Alert1)



ภาพที่ 3.7 วงจรชุดแปลงสัญญาณเอาท์พุทของสัญญาณเสียงเตือนชุด 1 (Alert1)

ต้องการให้มอสเฟตทำงานคล้ายสวิตช์ ดังนั้นจึงต้องออกแบบ V_{DS} ให้เข้าใกล้ค่า 0V มากที่สุด ในที่นี้กำหนดให้ V_{DS} = 1V และแรงดันไบอัส V_{GS} = 5V



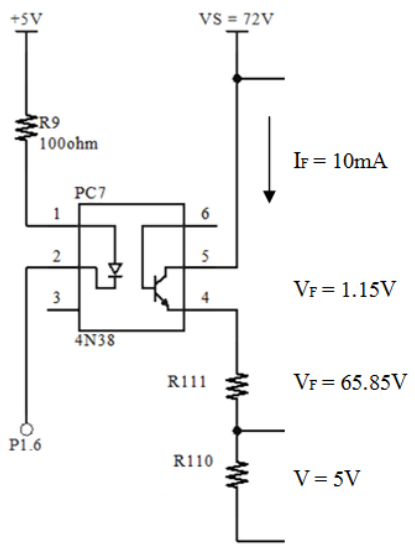
ภาพที่ 3.8 Typical Output Characteristics ของ RDN100

จากกราฟในภาพที่ 3.8 เราเลือก V_{GS} ที่ 5V ดังนั้นจากวงจรในภาพที่ 3.7 แรงดันตกคร่อมที่ R110 จะเท่ากับ $V_{R110} = V_{GS} = 5V$ เพราะฉะนั้นจากกำของเคอร์ชอฟฟ์จะได้แรงดัน V_{R111} ดังนี้

$$V_S = V_{CE(on)} + V_{R111} + V_{R110} \tag{3.6}$$

$$V_{R111} = V_S - V_{CE(on)} - V_{R110}$$

$$V_{R111} = 72V - 1.15V - 5V = 65.85V$$



ภาพที่ 3.9 แรงดันที่จุดต่างๆ ของวงจร

ดังนั้นหาค่า R111 ได้จาก

$$R111 = \frac{V_{R111}}{I_F} \tag{3.6}$$

$$= \frac{65.85V}{10mA} = 6.59k\Omega$$

หาค่ากำลังไฟฟ้าที่ R111 ได้ $P_{R111} = I^2 R111$ (3.7)

$$P_{R111} = 10mA^2 \times 6.59k\Omega = 0.66\text{W}$$

และหาค่า R110 ได้จาก

$$V_{R110} = \frac{R110 \times V_S}{(R111 + R110)} \tag{3.8}$$

$$5V = \frac{R110 \times 72V}{(6.59k\Omega + R110)}$$

$$32.95 + 5R110 = 72R110$$

$$32.95 = 72R110 - 5R110$$

$$32.95 = 67R110$$

$$R110 = 32.95 / 67$$

$$R110 = 0.49k\Omega \text{ หรือ } 490\ \Omega$$

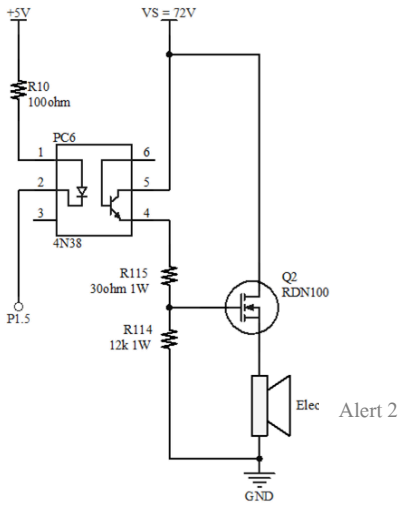
หาค่ากำลังไฟฟ้าที่ R110 ได้ $P_{R110} = 10mA^2 \text{ x } 490\,\Omega = 0.05\text{W}$

เนื่องจากว่าต้องการให้ MOSFET ทำงานเหมือนเป็นสวิตช์จึงกำหนดให้ V_{DS} = 1V ดังนั้น แรงดันที่โหลดจะมีค่าเท่ากับ

$$V_{LOAD} = V_S - V_{DS} \tag{3.9}$$

$$V_{LOAD} = 72V - 1V$$

วงจรชุดแปลงสัญญาณเอาท์พุทของสัญญาณเสียงเตือนชุด 2 (Alert2) และ Emergency Break การออกแบบแรงดันไบอัสเป็นเหมือนกัน



ภาพที่ 3.10 วงจรชุดแปลงสัญญาณเอาท์พุทของสัญญาณเสียงเตือนชุด 2 (Alert2)

ต้องการให้มอสเฟตทำงานคล้ายสวิตช์ ดังนั้นจึงต้องออกแบบ V_{DS} ให้เข้าใกล้ค่า 0 V มากที่สุด ในที่นี้กำหนดให้ V_{DS} = 1V และแรงดันไบอัส V_{GS} = 5V หาแรงดัน V_{LOAD} โดยที่ความต้านทานโหลดมีเท่ากับ 232 Ω จากกฎของเคอร์ชอฟฟ์ได้ดังนี้ โดยค่า $V_{GS(ON)}$ = 10V, $V_{GS(TH)}$ = 4V และ $I_{D(ON)}$ = 10A

$$V_G = \frac{R_2}{(R_1 + R_2)} V_S \tag{3.10}$$

$$V_{DS} = V_{DD} - I_D R_D \tag{3.11}$$

เมื่อ

$$I_D = k(V_{GS} - V_{GS(TH)})^2 \tag{3.12}$$

และ k หาได้จาก $$k = \frac{I_{D(ON)}}{(V_{GS(ON)} - V_{GS(TH)})^2} \quad (3.13)$$

$$k = \frac{10A}{(10V - 4V)^2} = 0.277A/V^2$$

ดังนั้น $$I_D = 0.277(5-4)^2 = 277mA$$

จาก $$V_{GS} = V_G - I_D R_{Load} \quad (3.14)$$

$$5V = V_G - (277mA \times 239\Omega)$$

$$V_G = (277mA \times 232\Omega) + 5V$$

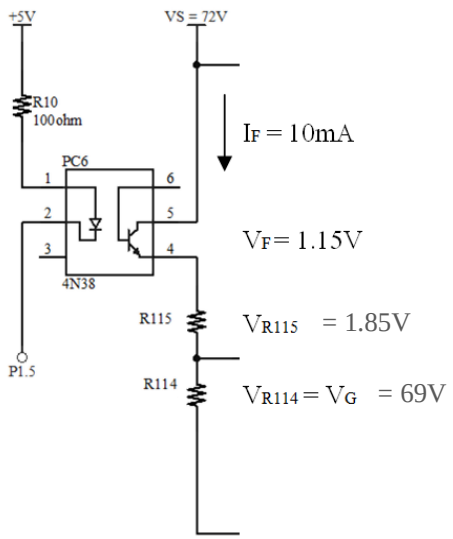
$$V_G = 69V$$

จากวงจรจะได้แรงดันตกคร่อมที่จุดๆต่างดังภาพที่ 3.11

$$V_G = V_{R114} = 69V$$

ดังนั้น $$V_{R115} = V_S - (V_F + V_{R114}) \quad (3.15)$$

$$V_{R115} = 72V - (1.15V + 69V) = 1.85V$$



ภาพที่ 3.11 แรงดันที่จุดต่างๆ ของวงจร

หาค่า R115 ได้จาก $$R115 = \frac{V_{R115}}{I_F} \quad (3.16)$$

$$R115 = \frac{1.85V}{10mA} = 185\Omega$$

หาค่ากำลังไฟฟ้าที่ R115 ได้ $$P_{R115} = I_F^2 R115 \quad (3.17)$$

$$P_{R115} = 10^2 185 = 0.0185W$$

หาค่า R114 ได้จาก

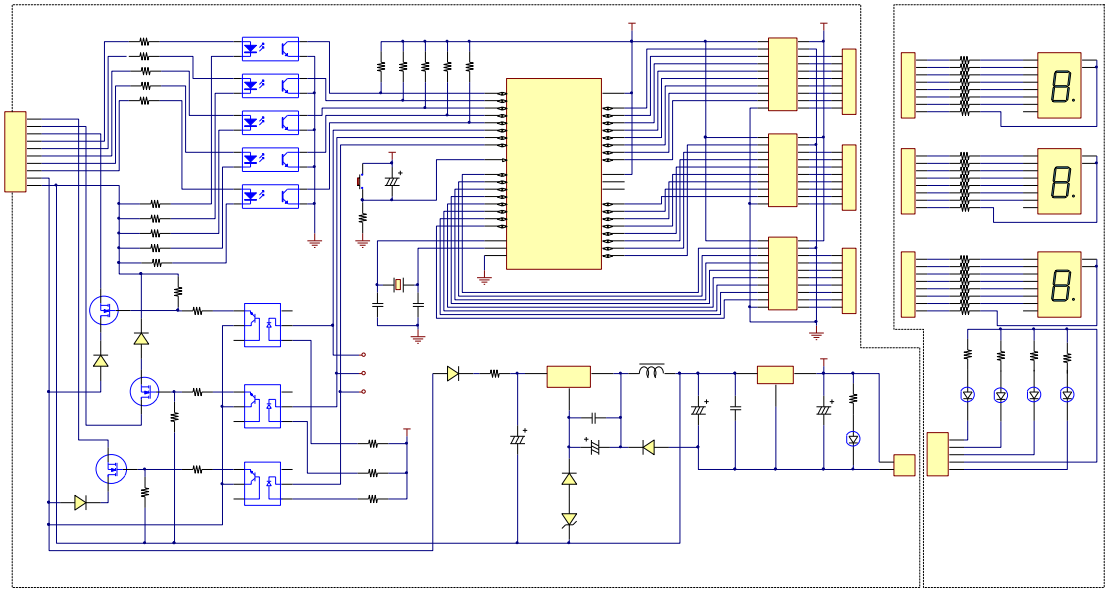
$$R114 = \frac{V_{R114}}{I_F} \tag{3.18}$$

$$R114 = \frac{69V}{10mA} = 6.9k\Omega$$

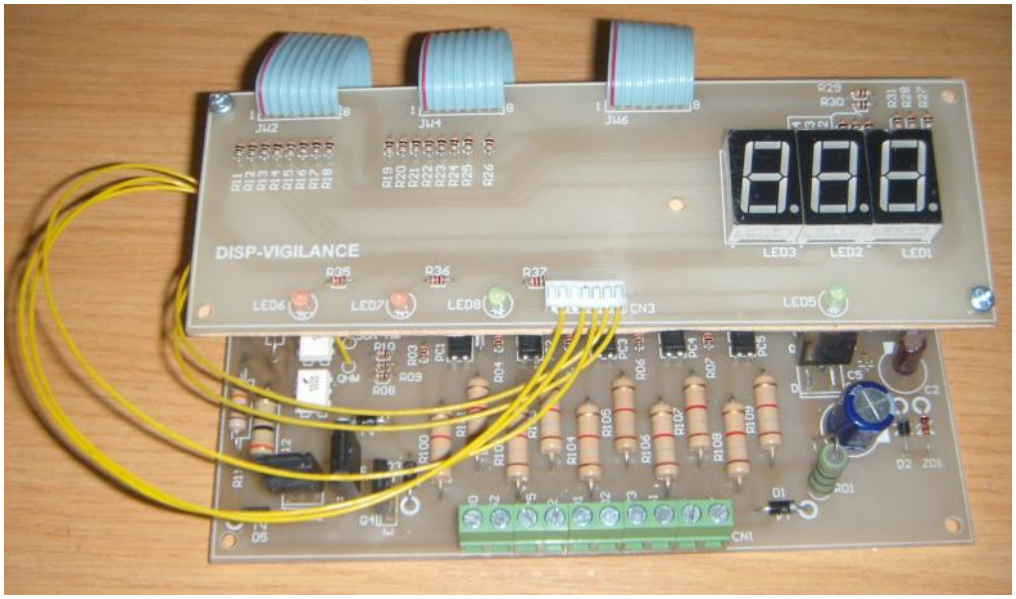
หาค่ากำลังไฟฟ้าที่ R115 ได้

$$P_{R114} = I_F^2 R114 \tag{3.19}$$

$$P_{R114} = 10^2 6.9 = 0.69W$$



ภาพที่ 3.12 Vigilance Safety Circuit



ภาพที่ 3.13 PCB Assembly of Vigilance Safety Device

บทที่ 4

การทดลองและผลการทดลอง

การทดลองเครื่องควบคุมระบบป้องกันพนักงานรถจักรหมดสติของรถจักรดีเซลไฟฟ้าของการรถไฟแห่งประเทศไทย ได้แบ่งออกเป็น 5 ส่วนคือ ชุดแปลงสัญญาณอินพุท ชุดเพาเวอร์ซัพพลาย ชุดควบคุม ชุดแปลงสัญญาณเอาท์พุท และทดลองติดตั้งที่หัวรถจักร

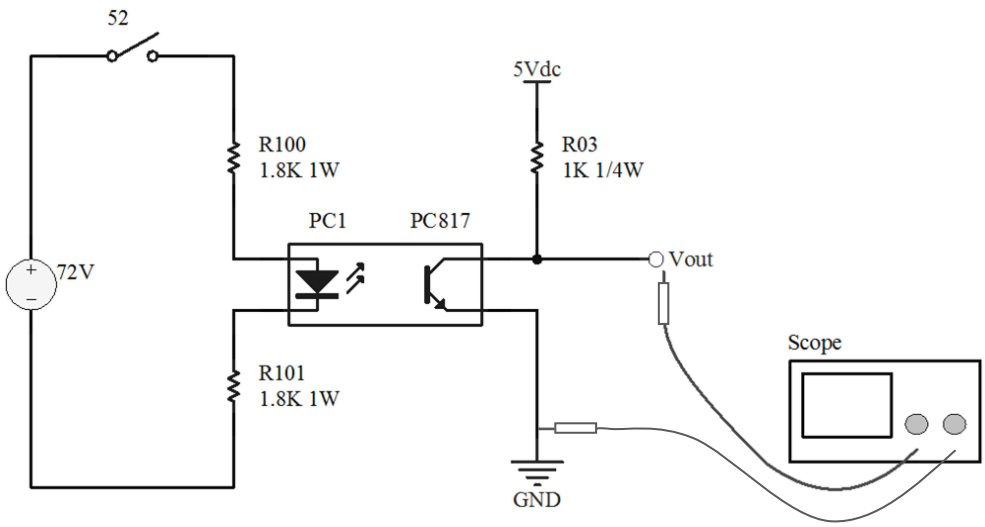
4.1 ชุดแปลงสัญญาณอินพุท

4.1.1 วัตถุประสงค์

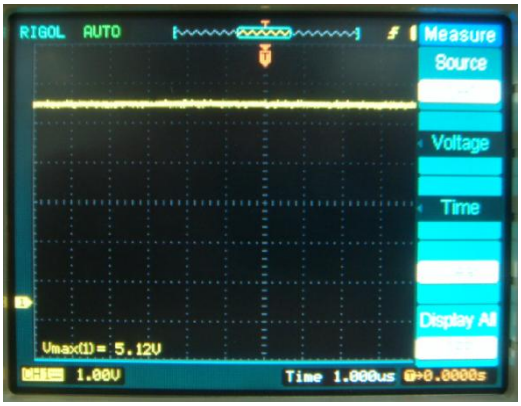
เปลี่ยนสัญญาณอินพุทจาก 72Vdc จากการรีเซทตามสวิตช์รีเซทต่างๆให้เป็นสัญญาณดิจิตอลในระดับ 5Vdc และ 0Vdc เพื่อที่จะป้อนให้พอร์ตอินพุทของไมโครคอนโทรลเลอร์ซึ่งได้แก่พอร์ต P1.0, P1.1, P1.2, P1.3, และ P1.4

4.1.2 อุปกรณ์ที่ในการทดสอบ

Digital Oscilloscopes, แหล่งจ่ายไฟ AC ชุดบริดจ์เรกติไฟเออร์, ชุดจำลองสวิตช์รีเซท (สวิตช์รีเซท 5 ตัว)



ภาพที่ 4.1 จุดที่วัดสัญญาณ Vout ของวงจรชุดแปลงสัญญาณอินพุท



ก.

ข.

ภาพที่ 4.2 ก. สัญญาณสภาวะที่ยังไม่กดสวิตช์
ข. สัญญาณสภาวะที่กดสวิตช์

จากภาพที่ 4.2 ก. เป็นสัญญาณแรงดันที่จุดวัด Vout ขณะที่ยังไม่ทำงานซึ่งมีค่าเท่ากับ 5.12V และภาพที่ 4.2 ข. เป็นสัญญาณขณะที่ทำงานจะมีแรงดันเท่ากับ 200mV หรือ 0V

4.1.3 ผลการทดลอง

ผลการทดลองสัญญาณจากการจำลองการกดสวิตช์รีเซททั้ง 5 ตัว ผลที่ได้คือ ในสภาวะที่ยังไม่ได้กดสวิตช์จะมีแรงดันที่จุด Vout = 5.12 V นั้นก็คือสัญญาณดิจิตอลลอจิก “1” ดังภาพที่ 4.2 และเมื่อกดสวิตช์พบว่าแรงดัน Vout = 200mV ซึ่งนั้นก็คือสัญญาณดิจิตอลลอจิก “0” ซึ่งผลการทดลองชุดแปลงสัญญาณอินพุท เป็นไปตามวัตถุประสงค์ที่กำหนดไว้

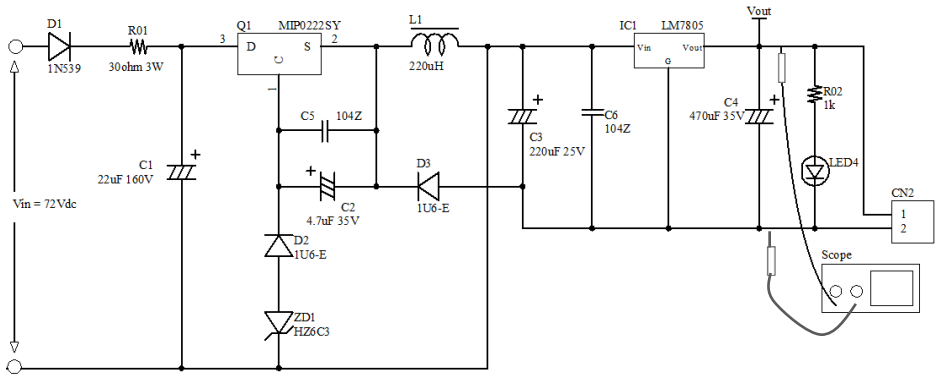
4.2 ชุดเพาเวอร์ซัพพลาย

4.2.1 วัตถุประสงค์

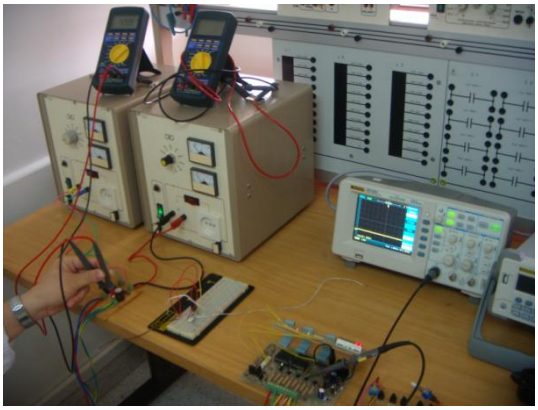
แปลงแรงดันไฟฟ้ากรแสตรง 72V เป็น 5V เพื่อจ่ายไฟเลี้ยงวงจรชุดแปลงสัญญาณอินพุท ชุดควบคุม ชุดแปลงสัญญาณเอาท์พุท และชุดแสดงผล

4.2.2 อุปกรณ์ที่ในการทดสอบ

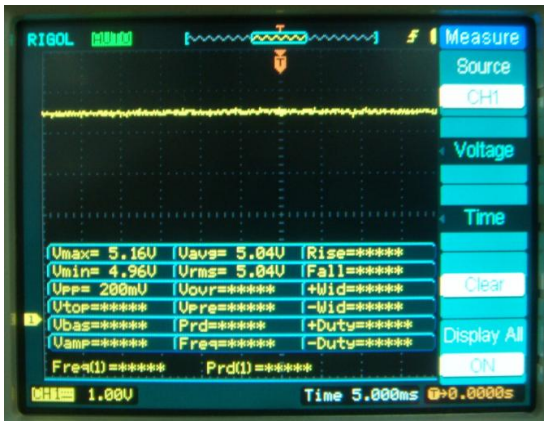
Digital Oscilloscopes, แหล่งจ่ายไฟ AC ชุดบริดจ์เรกติไฟเออร์, Digital Multimeter



ภาพที่ 4.3 จุดที่วัดสัญญาณ Vout ของวงจรชุดเพาเวอร์ซัพพลาย



ภาพที่ 4.4 เป็นการวัดแรงดันอินพุทที่ใช้ในการทดลอง



ภาพที่ 4.5 สัญญาณแรงดันเอาท์พุท (Vout) ที่วัดได้

จากภาพที่ 4.4 เป็นการวัดแรงดันอินพุท ซึ่งวัดได้เท่ากับ 72.63V และภาพที่ 4.5 เป็นแรงดันเอาท์พุทที่วัดจากจุด Vout ของวงจรเพาเวอร์ซัพพลายที่ออกแบบ ค่าที่วัดได้คือ 5.04V

4.2.3 ผลการทดลอง

ผลการทดลองชุดเพาเวอร์ซัพพลายเป็นไปตามวัตถุประสงค์ โดยแปลงแรงดันไฟฟ้ากระแสตรง 72V ให้เป็น 5V วัดแรงดันเอาท์พุท (Vout) ได้ 5.04V ดังภาพที่ 4.5

4.3 ชุดควบคุม

4.3.1 วัตถุประสงค์

ชุดควบคุมจะต้องสามารถรับอินพุทได้ 5 อินพุท และควบคุมเอาท์พุท 3 เอาท์พุท และแสดงสถานะ การนับเวลาที่ชุดแสดงผล และการทำงานของระบบความคุมจะต้องเป็นไปตามข้อกำหนดดังนี้

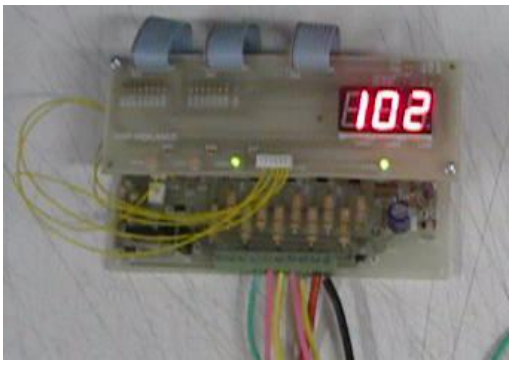
1. ความเร็ว 8 km/hrs ระบบ Vigilance เริ่มทำงานนับเวลาจาก 0 - 8 วินาที โดยที่ 7 Segment ของชุดแสดงผลเป็นตัวแสดงผลการนับเวลา และไฟแสดงสถานะห้ามล้อฉุกเฉินที่ชุดแสดงผลจะติด (LED8)

2. ภายใน 8 วินาที ถ้าไม่มีการรีเซทที่ Foot Switch หรือ Push Button Switch สัญญาณเสียงเตือนชุด1 จะดังเตือนไฟแสดงผล (LED6) ของชุดแสดงผลจะติดด้วย และระบบจะนับต่อไปอีก 4 วินาที และถ้าไม่มีการรีเซทภายใน 4 วินาที ห้ามล้อฉุกเฉินจะทำงานไฟแสดงผล (LED8) ของชุดแสดงผล จะดับ และถ้ามีการกดรีเซทภายใน 8 วินาทีหรือ 4 วินาที ที่ Foot Switch หรือ Push Button Switch ระบบจะเริ่มนับใหม่โดยเริ่มจาก 0 - 8 วินาที อีกครั้ง และถ้าเหยียบ Foot Switch ค้างระบบก็จะนับเวลา 0 - 102 วินาที โดย 7 Segment ของชุดแสดงผลเป็นตัวแสดงผลการนับเวลา

3. ภายใน 102 วินาที (เหยียบ Foot Switch ค้าง) ถ้าไม่มีการรีเซทที่ Engine Speed Up หรือ Engine Speed Down หรือ Sanding Switch สัญญาณเสียงเตือนชุด 2 (Alert2) จะดังเตือนสังเกตได้จากไฟแสดงผล (LED 7) ของชุดแสดงผลจะติด และระบบจะนับต่อไปอีก 8 วินาที ไฟแสดงผล (LED6) ของชุดแสดงผลจะติด ถ้ายังไม่มีการรีเซท สัญญาณเสียงเตือนชุด 1 (Alert1) จะดังเตือน และระบบจะนับต่อไปอีก 4 วินาที ถ้ายังไม่มีการรีเซทอีกห้ามล้อฉุกเฉินจะทำงานไฟแสดงผล (LED8) ของชุดแสดงผล จะดับ และถ้ามีการรีเซทที่ Engine Speed Up หรือ Engine Speed Down หรือ Sanding Switch ภายใน 102 วินาที หรือภายใน 8 วินาที หรือภายใน 4 วินาที ระบบจะเริ่มนับใหม่อีกครั้งสังเกตจาก 7 Segment ของชุดแสดงผล

4.3.2 อุปกรณ์ที่ในการทดสอบ

แหล่งจ่ายไฟ AC ชุดบริดจ์เรกติไฟเออร์, Digital Multimeter, ชุดจำลองสวิตช์รีเซท



ภาพที่ 4.6 การทดลองชุดควบคุม

ตารางที่ 4.1 ผลการทดลองในฟังก์ชั่น 8 วินาที

ลำดับ	เงื่อนไขการทดลอง	ผลการทดลอง	หมายเหตุ
1	ความเร็ว 8 km/hrs	7 Segment แสดงผลการนับตั้งแต่ 0 - 8 และ LED8 ติด	จำลองโดยการจ่ายไฟ 72Vdc
2	ไม่มีการรีเซทภายใน 8 วินาที	LED6 ติด และ 7 Segment แสดงผลการนับเป็น 0 - 4	
3	ไม่มีการรีเซทภายใน 4 วินาที	LED8 ดับ และ 7 Segment แสดงผล “- - -”	
4	มีการรีเซทที่ Foot Switch หรือ Push Button Switch ภายใน 8 วินาที	7 Segment แสดงผลการนับใหม่อีกครั้งโดยเริ่มจาก 0 - 8 และ LED8 ยังคงติดอยู่	
5	มีการรีเซทที่ Foot Switch หรือ Push Button Switch ภายใน 4 วินาที	LED6 ดับ และ 7 Segment แสดงผลการนับใหม่อีกครั้งโดยเริ่มจาก 0 – 8	

ตารางที่ 4.2 ผลการทดลองในฟังก์ชั่น 102 วินาที (เหยียบ Foot Switch ค้าง)

ลำดับ	เงื่อนไขการทดลอง	ผลการทดลอง	หมายเหตุ
1	เหยียบ Foot Switch ค้าง	7 Segment แสดงผลการนับตั้งแต่ 0 - 102 และ LED8 ติด	
2	ไม่มีการรีเซทภายใน 102 วินาที	LED7 ติด และ 7 Segment แสดงผลการนับเป็น 0 - 8	
3	ไม่มีการรีเซทภายใน 8 วินาที	LED6 ติด และ 7 Segment แสดงผลการนับเป็น 0 - 4	
4	ไม่มีการรีเซทภายใน 4 วินาที	LED8 ดับ และ 7 Segment แสดงผล “- - -”	
5	มีการรีเซทที่ Engine Speed Up หรือ Engine Speed Down หรือ ปล่อยเท้าภายใน 102 วินาที	7 Segment แสดงผลการนับใหม่อีกครั้งจาก 0 - 8	
6	มีการรีเซทที่ Engine Speed Up หรือ Engine Speed Down หรือ ปล่อยเท้าภายใน 8 วินาที	LED7 ดับ และ 7 Segment แสดงผลการนับใหม่อีกครั้งจาก 0 - 8	
7	มีการรีเซทที่ Engine Speed Up หรือ Engine Speed Down หรือ ปล่อยเท้าภายใน 4 วินาที	LED7, LED8 ดับ และ 7 Segment แสดงผลการนับใหม่อีกครั้งจาก 0 - 8	

4.3.3 ผลการทดลอง

เมื่อความเร็วถึง 8 km/hrs จำลองโดยการเปิดวงจรสวิตช์ของชุดจำลองสวิตช์รีเซท ระบบ Vigilance เริ่มทำงานไฟ LED8 ติด โดยเริ่มนับเวลาจาก 0 - 8 วินาที หลังจากนั้นไฟ LED6 ติด แสดงถึงสัญญาณเสียงเตือนชุด 1 (Alert1) ดัง และนับต่ออีก 4 วินาที หลังจากนั้นไฟ LED8 จะดับ แสดงถึงห้ามล้อฉุกเฉินทำงาน และเมื่อกด Foot Switch จำลองโดยการกดสวิตช์151 ที่ชุดจำลองสวิตช์รีเซท ภายใน 8 หรือ 4 วินาที ระบบจะหยุดนับเวลา

ในกรณีที่เหยียบ Foot Switch หรือกด Engine Speed Up หรือ Engine Speed Down ค้างไว้ ระบบจะนับ 0 - 102 วินาที หลังจากนั้นไฟ LED7 ติด แสดงถึงสัญญาณเสียงเตือนชุด 2 (Alert2) ดัง และนับต่อไปอีก 8 วินาที หลังจากนั้น LED6 ติด แสดงถึงสัญญาณเสียงเตือนชุด 1 (Alert1) ดัง และนับต่อไปอีก 4 วินาที หลังจากนั้นไฟ LED8 ดับ แสดงถึงห้ามล้อฉุกเฉินทำงาน ระบบหยุดนับเวลา และเมื่อมีการกด Foot Switch หรือ Engine Speed Up หรือ Engine Speed Down หรือ Sanding Switch ระบบก็จะเริ่มนับใหม่อีกครั้ง ซึ่งผลการทดลองของชุดควบคุมเป็นไปตามวัตถุประสงค์

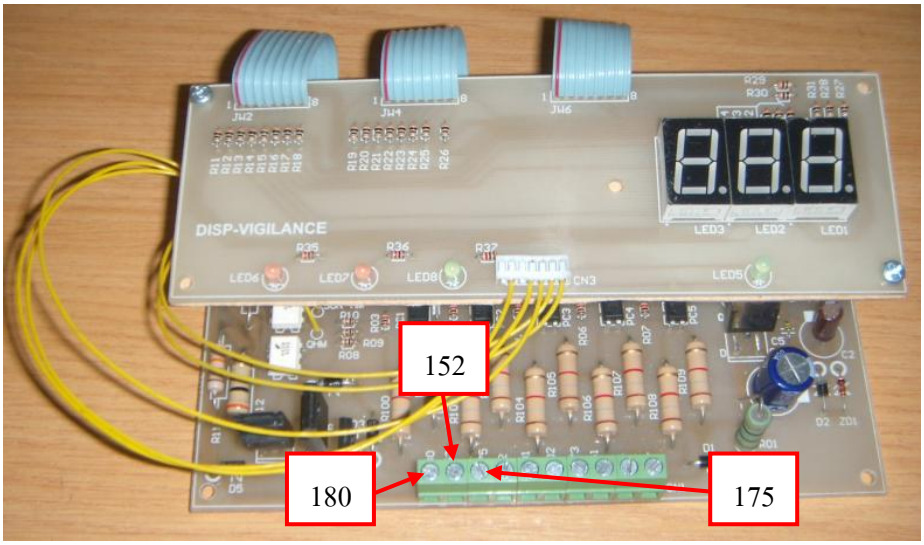
4.4 ชุดแปลงสัญญาณเอาท์พุท

4.4.1 วัตถุประสงค์

เปลี่ยนสัญญาณเอาท์พุทจาก 5Vdc จากเอาท์พุทของไมโครคอนโทรลเลอร์ได้แก่พอร์ต P1.5, P1.6 และ P1.7 ให้เป็นแรงดัน 72Vdc เพื่อไปเลี้ยงโหลดได้แก่ สัญญาณเสียงเตือนชุด 1 (Alert1) และ ชุด 2 และ Emergency Break

4.4.2 อุปกรณ์ที่ในการทดสอบ

แหล่งจ่ายไฟ AC ชุดบริดจ์เรกติไฟเออร์ ชุดจำลองสวิตชรีเซท และดิจิตอลมัลติมิเตอร์



ภาพที่ 4.7 ตำแหน่งการวัดเอาท์พุต

จากภาพที่ 4.7 เป็นการแสดงตำแหน่งการวัดเอาท์พุททั้ง 3 จุด โดยที่ตำแหน่ง 152 คือ สัญญาณเสียงเตือนชุด 1 (Alert1) ตำแหน่ง 175 คือ สัญญาณเสียงเตือนชุด 2 (Alert2) และตำแหน่ง 180 คือ ห้ามล้อฉุกเฉิน

ตารางที่ 4.3 ผลการวัดแรงดันเอาท์พุท

ลำดับ	ฟังก์ชั่น	เงื่อนไข	แรงดันเอาท์พุท (V)		
			152	175	180
1	8 วินาที	ไม่มีการรีเซทภายใน 8 วินาที	72.40	1.58	69.90
		ไม่มีการรีเซทภายใน 4 วินาที	72.30	1.58	1.53
		มีการรีเซทภายใน 8 วินาที	0.09	1.58	70.00
		ไม่มีการรีเซทภายใน 4 วินาที	0.07	1.58	69.80
2	102 วินาที (เหยียบ Foot Switch ค้าง)	ไม่มีการรีเซทภายใน 102 วินาที	0.09	69.30	71.20
		ไม่มีการรีเซทภายใน 8 วินาที	72.50	69.30	69.90
		ไม่มีการรีเซทภายใน 4 วินาที	73.90	69.30	1.51
		มีการรีเซทภายใน 102 วินาที	0.10	1.58	69.90
		มีการรีเซทภายใน 8 วินาที	0.08	1.58	69.80
		มีการรีเซทภายใน 4 วินาที	0.09	1.58	69.80

4.4.3 ผลการทดลอง

ผลการทดลองชุดแปลงเอาท์พุทเป็นไปตามวัตถุประสงค์ จากตารางที่ 4.3 ในส่วนของฟังก์ชั่น 8 วินาที และฟังก์ชั่น102 วินาที (เหยียบ Foot Switch ค้าง) แรงดันเอาท์พุทที่ 152, 175 และ 180 แสดงให้เห็นว่าเราสามารถเปลี่ยนแรงเอาท์พุทจากพอร์ตชุดควบคุมเป็นแรงดันเอาท์พุทไปจ่ายให้โหลดได้ ซึ่งผลการทดลองเป็นไปตามวัตถุประสงค์

4.5 การทดลองติดตั้งที่รถจักร

4.5.1 วัตถุประสงค์

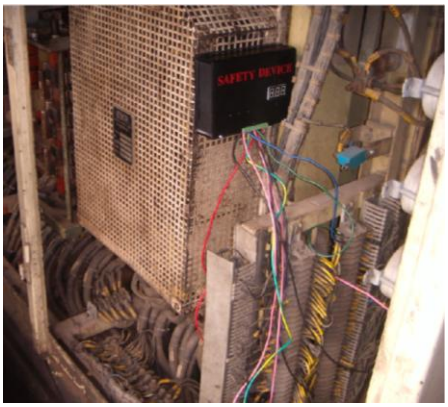
เพื่อนำเครื่องควบคุมระบบป้องกันพนักงานรถจักรหมดสติที่สร้างขึ้นไปทดลองติดตั้งที่หน้างานจริง และเพื่อตรวจสอบว่าสามารถทำงานได้ถูกต้องตามฟังก์ชั่นการทำงานของรถจักรดีเซลไฟฟ้าหรือไม่

4.5.2 อุปกรณ์ที่ในการทดสอบ

เครื่องควบคุมระบบป้องกันพนักงานรถจักรหมดสติของรถจักรดีเซลไฟฟ้าที่สร้างขึ้น และดิจิตอลมัลติมิเตอร์

ภาพที่ 4.8 รถจักรดีเซลไฟฟ้าที่ใช้ทดสอบ

จากภาพที่ 4.8 เป็นรถจักรดีเซลไฟฟ้าหมายเลข 4218 ที่ใช้ทดสอบกับเครื่องควบคุมระบบป้องกันพนักงานรถจักรหมดสติของรถจักรดีเซลไฟฟ้าที่สร้างขึ้น



ภาพที่ 4.9 การติดตั้งที่ห้องเครื่องของรถจักรดีเซลไฟฟ้า

จากภาพที่ 4.8 เป็นการนำเครื่องควบคุมระบบป้องกันพนักงานรถจักรหมดสติของรถจักรดีเซลไฟฟ้าที่สร้างขึ้นไปติดตั้งที่ห้องเครื่องของรถจักรดีเซลไฟฟ้า

ภาพที่ 4.10 ทดสอบระบบ Vigilance ของรถจักรดีเซลไฟฟ้า

จากภาพที่ 4.10 เป็นการทดลองเครื่องควบคุมระบบป้องกันพนักงานรถจักรหมดสติของรถจักรดีเซลไฟฟ้าที่สร้างขึ้นโดยการเลือกโหมดทดสอบ Vigilance ของรถจักรดีเซลไฟฟ้าที่ใช้อยู่จริง

4.5.2 อุปกรณ์ที่ในการทดสอบ

ผลการทดลองพบว่าเครื่องควบคุมระบบป้องกันพนักงานรถจักรหมดสติของรถจักรดีเซลไฟฟ้าที่สร้างขึ้น สามารถทำงานได้ถูกต้องตามฟังก์ชั่นการใช้งานของรถจักรดีเซล

ภาพที่ 4.11 การแสดงผลขณะทำงาน

จากภาพที่ 4.11 เป็นการแสดงผลการนับเวลาขณะที่ทดลองของเครื่องควบคุมระบบป้องกันพนักงานรถจักรหมดสติของรถจักรดีเซลไฟฟ้าที่สร้างขึ้น

บทที่ 5
สรุปและข้อเสนอแนะ

จากการทดลองเครื่องควบคุมระบบป้องกันพนักงานรถจักรหมดสติของรถจักรดีเซลไฟฟ้าของการรถไฟแห่งประเทศไทย พบปัญหาอยู่สองเรื่องคือ ในส่วนของชุดแปลงสัญญาณเอาท์พุท ซึ่งเดิมได้ออกแบบไว้ออปโต้เบอร์ 4N38 เป็นตัวขับโหลด สัญญาณเสียงเตือนชุด 1 และสัญญาณเสียงเตือนชุด 2 (Alert2) แต่เนื่องจากกระแสขณะที่โหลดทำงานจะมีกระแสประมาณ 350mA ซึ่งออปโต้เบอร์ 4N38 สามารถทนกระแสมากสุดได้เพียง 80mA และพบปัญหาในส่วนของชุดควบคุม คือพบว่าชุดควบคุมไม่ทำงาน ซึ่งสาเหตุเกิดจากไม่ได้กำหนด EA ให้เป็น “1” ก็คือไม่ได้จ่ายไฟให้ขา 39 ของไมโครคอนโทรลเลอร์ ซึ่งการกำหนดให้ EA เป็น “1” คือใช้หน่วยความจำภายใน และถ้ากำหนดให้ EA เป็น “0” คือใช้หน่วยความจำภายใน ซึ่งในที่นี้ต้องใช้หน่วยความจำภายใน จากปัญหาที่พบในการทดลอง มีข้อเสนอแนะในการแก้ไขปัญหาคือ ในส่วนของอุปกรณ์ที่ใช้ขับโหลดของชุดแปลงสัญญาณเอาท์พุทควรมีคุณสมบัติที่สามารถทนกระแสได้มากกว่ากระแสที่โหลดทำงานใน และในส่วนของปัญหาชุดควบคุมไม่ทำงาน จะต้องกำหนด EA ให้เป็น “1” เนื่องจากว่าไมโครคอนโทรลเลอร์ในที่นี้ใช้หน่วยความจำภายใน แนวทางกาแก้ไข ในส่วนของอุปกรณ์ที่ใช้ขับโหลดของชุดแปลงสัญญาณเอาท์พุทเปลี่ยนอุปกรณ์ขับโหลดเป็นมอสเฟตเบอร์ RDN100 ซึ่งมีคุณสมบัติที่สามารถทนกระแสได้สูง และในส่วนของปัญหาชุดควบคุมไม่ทำงานแก้ไขโดย ทำการกำหนดให้ EA เป็น “1” โดยต่อไฟ 5Vdc ให้ขา 39 ของไมโครคอนโทรลเลอร์

เอกสารอ้างอิง

[1] อดิศักดิ์ ชิณะวงศ์ “บทความ MCS - 51” แผนกวิชาช่างอิเล็กทรอนิกส์ วิทยาลัยเทคนิคน่าน http://www.adisak51.com/

[2] ขจร อนุดิตย์ “การเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ ด้วยภาษาซี” จัดพิมพ์โดย สำนักพิมพ์ Core Function 2550

[3] ฉันทนา โพธิครูประเสริฐ “เนื้อหาสาระ หน่วย ออปโต้คัปเปลอร์” วิทยาลัยเทคนิคสมุทรปราการ http://www.sptc.ac.th/prapruet/dc/

[4] ฉันทนา โพธิครูประเสริฐ “เนื้อหาสาระ หน่วย วงจรแบ่งแรงดัน” วิทยาลัยเทคนิคสมุทรปราการ http://www.sptc.ac.th/prapruet/dc/

[5] พงศ์ธวัช ชีพพิมลชัย และ อโณ โชติมณี “สวิตชิ่งเพาเวอร์ซัพพลายเบื้องต้น” ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ http://www.cpe.ku.ac.th/

[6] Damrong Kalawdee “Basic Electronics” Department of Computer Engineering Facuty of Engineering Prince of Songkla University, http://dk.coe.psu.ac.th/

[7] อ.ธนันต์ ศรีสกุล “บทความ อุปกรณ์สารกึ่งตัวนำ MOSFET” ภาควิชาครุศาสตร์ไฟฟ้า คณะครุศาสตร์อุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ http://www.numsai.com/

ภาคผนวก

ภาคผนวก ก

PCB Assembly Data

DRAWN	DESIGN	CHECKED	UNIT	SCALE	Part list of PCB assembly unit of VIGILANCE SAFETY DEVICE
			mm	NTS	

PRODUCT NAME	MODEL NAME	PARTS NAME
VIGILANCE SAFETY DEVICE	VIGILANCE FOR ALSTHOM	Parts List of PCB assembly unit of VIGILANCE SAFETY DEVICE

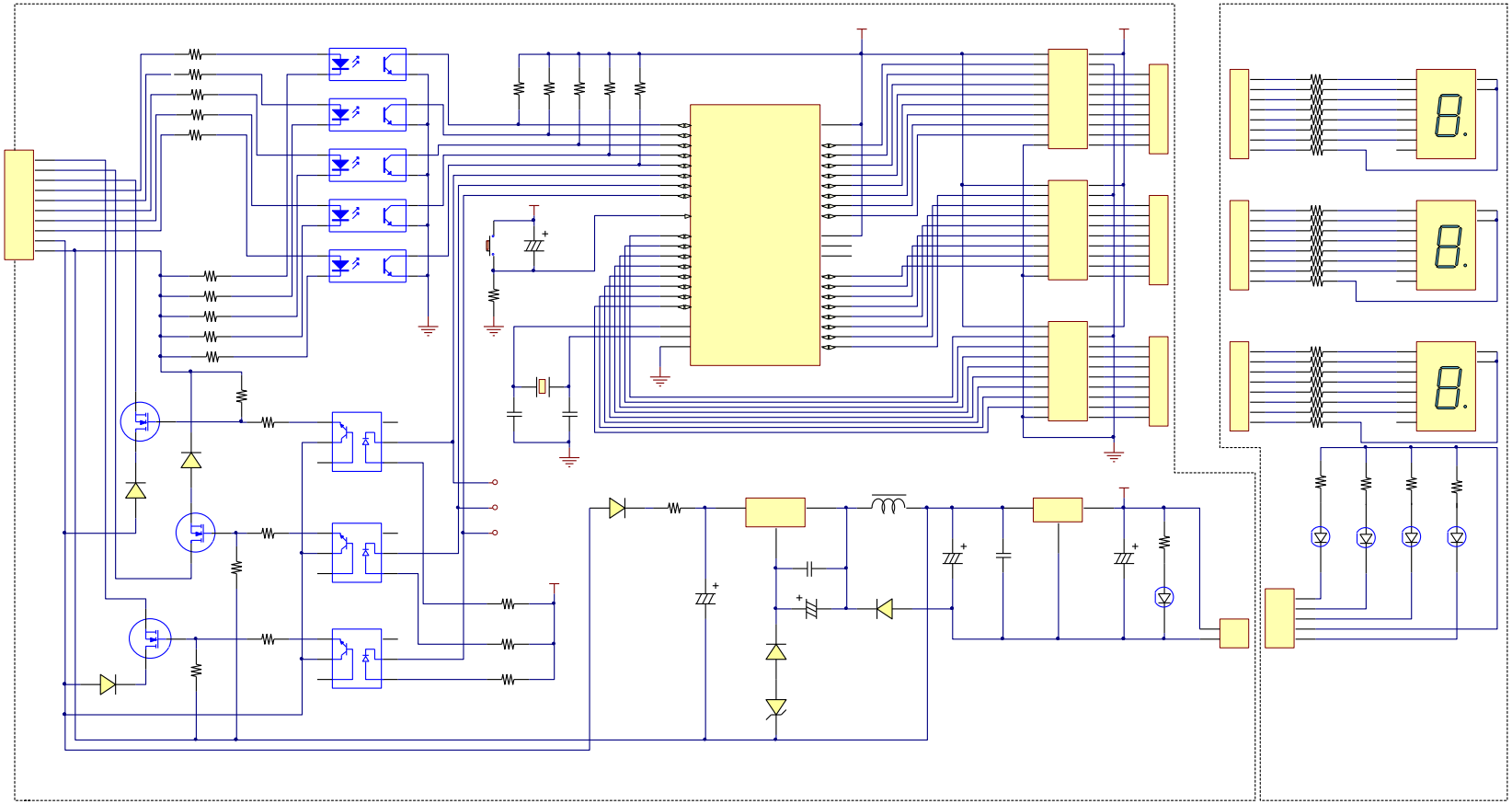
PARTS CODE	PARTS NAME
	DISP-VIGILANCE
	CON-VIGILANCE

Qty 2A	Qty 1A	No.	LEVEL	PARTS CODE	PARTS NAME	POSITION	VENDOR	NOTES
				PARTS				
	1	1		MOS3C300J	RESISTOR, MOS3C300J (RoHS)	R01	ROYAL OHM	30Ω 3W
	2	2		068352012300	R 1W 5% 30E ROYAL	R112, R115	NPE	30Ω 1W
	1	3		068352014100	R 1W 5% 1K ROYAL	R110	NPE	1kΩ 1W
	3	4		068352015120	R 1W 5% 12K ROYAL	R111, R113, R114	NPE	12kΩ 1W
	3	5		BCS4J0102A10	RESISTOR,R-BCS4J0102A10	R02-R07, R39	ROYAL OHM	1kΩ 1/4W
	3	6		BCS4J0300A10	RESISTOR,R-BCS4J0300A10	R08-R10	ROYAL OHM	30Ω 1/4W
	10	7		068352014180	R 1W 5% 1.8K ROYAL	R100-R109	NPE	1.8kΩ 1W
	4	8		1N5399	DIODE, 1N5399 (TAPE & AMMO / 26 mm)	D1, D4-D6	EIC	
	2	9		1U6-E	DIODE, 1U6-E	D2, D3	GAGNER	
	1	10		ZD-HZ6C3-TD-E	ZENER DIODE, ZD-HZ6C3-TD-E	ZD1	KYOEI	
	1	11		56601542	EGS226M2AE11RRSAP	C1	SAMXON	22μF 100V
	1	12		ECA1VMH470B	CAPACITOR C-ECA1VMH470B	C2	PANASONIC	47μF 35V
	1	13		ECA1EMH221B	CAPACITOR, ECA1EMH221B	C3, C9	PANASONIC	220μF 25V
	1	14		56600076	EGS477M1EG1BRRSAP	C4	SAMXON	470μF 25V
	2	15		UP050F104Z-A	CAPACITOR,C-UP050F104Z-A	C5, C6	SIIX	0.1μF
	2	16		059291508030	CERAMIC,30PF 50V OFFER	C7, C8	NPE	30pF
	1	17		37000016	X-TAL 12.00MHZ	X1	HKC	12.00MHz
	1	18		LHLP12NB221K	COIL,L-LHLP12NB221K	L1	TOSHIBA ASIA	220μH
	1	19		9050254009	LD-EMPG3863XT-H-F228K	LED5, LED8	STANLEY	9mm green
	1	20		MIP0222SY	TRANSISTOR,MIP0222SY	Q1	PANASONIC	
	3	21		PCNL22360115010	TRANSISTOR,FET RDN100N20	Q2-Q4	ROHM	
	1	22		SKRGAED010	SWITCH TACT,SKRGAED010	SW1	HOSODA	
	3	23		PCFPJ0726008000	FLAT JUMPER 8X195	JW1, JW3, JW5	TTEI	80mm. 8Pin
	1	24		00500787	TERMINAL BLOCK 10PINS,300V 10A ,PITCH 7.62 MM	CN1	DINKLE	
	1	25		08300063	POSITIVE VOLTAGE REGULATOR +5 VOLTS. 1.5 AMP.	IC1	SGS TOMSAN	
	1	26		06500834	BIT MICROCONTROLLER,ATMEL ,AT89C51RC-24PI	IC2	ATMEL	
	3	27		08602395	OCTAL BUS TRANSCEIVERS,SN74LS245N	IC3-IC5	TEXAS	
	5	28		LTV-817	IC, PHOTO COUPLER LTV-817-A (RoHS)	PC1-PC5	LITE-ON	
	3	29		12300113	HIGH VOLTAGE PHOTOTRANSISTOR,4N38	PC6-PC8	FAIRCHILD	
	1	30			PC BOARD, CON-VIGILANCE	CON-VIGILANCE	INOTECH	
	1	31			PC BOARD, DISP-VIGILANCE	DISP-VIGILANCE	INOTECH	
38		32		BCS4J0300A10	RESISTOR,R-BCS4J0300A10	R11-R38	ROYAL OHM	30Ω 1/4W
3		33		42800115	7-SEGMENT 1 DIGIT,4.00 INCH,RED COLOR	LED1-LED3	TOYO	Red
2		34		9050254009	LD-EMPG3863XT-H-F228K	LED5, LED8	STANLEY	9mm green
2		35		9050255009	LD-MVR3863XT-H-F228K	LED6, LED7	STANLEY	9mm Red
3		36		PCFPJ0726008000	FLAT JUMPER 8X195	JW2, JW4, JW6	TTEI	80mm. 8Pin

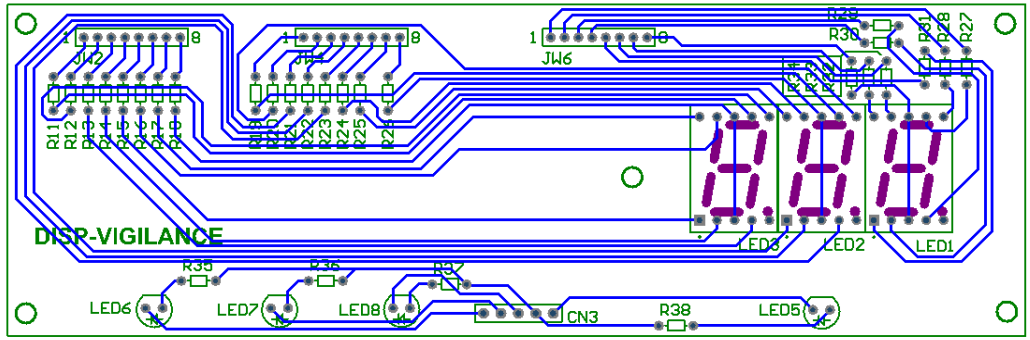
DATE	CONTENS	REVISED	APROVED

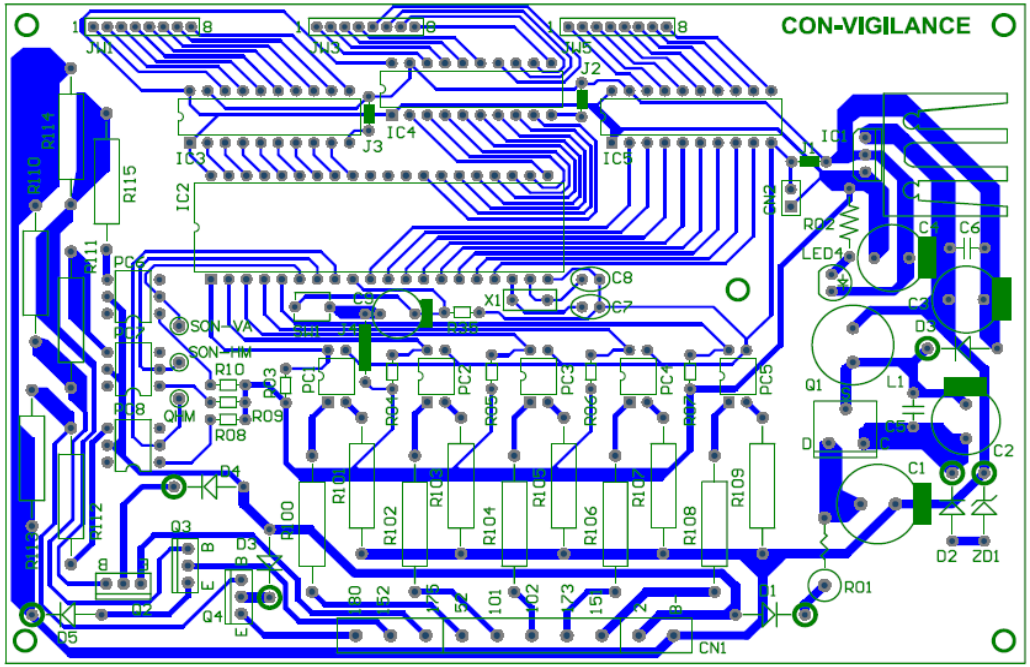
Use for PCB Prototype

Part list of PCB assembly unit of VIGILANCE SAFETY DEVICE

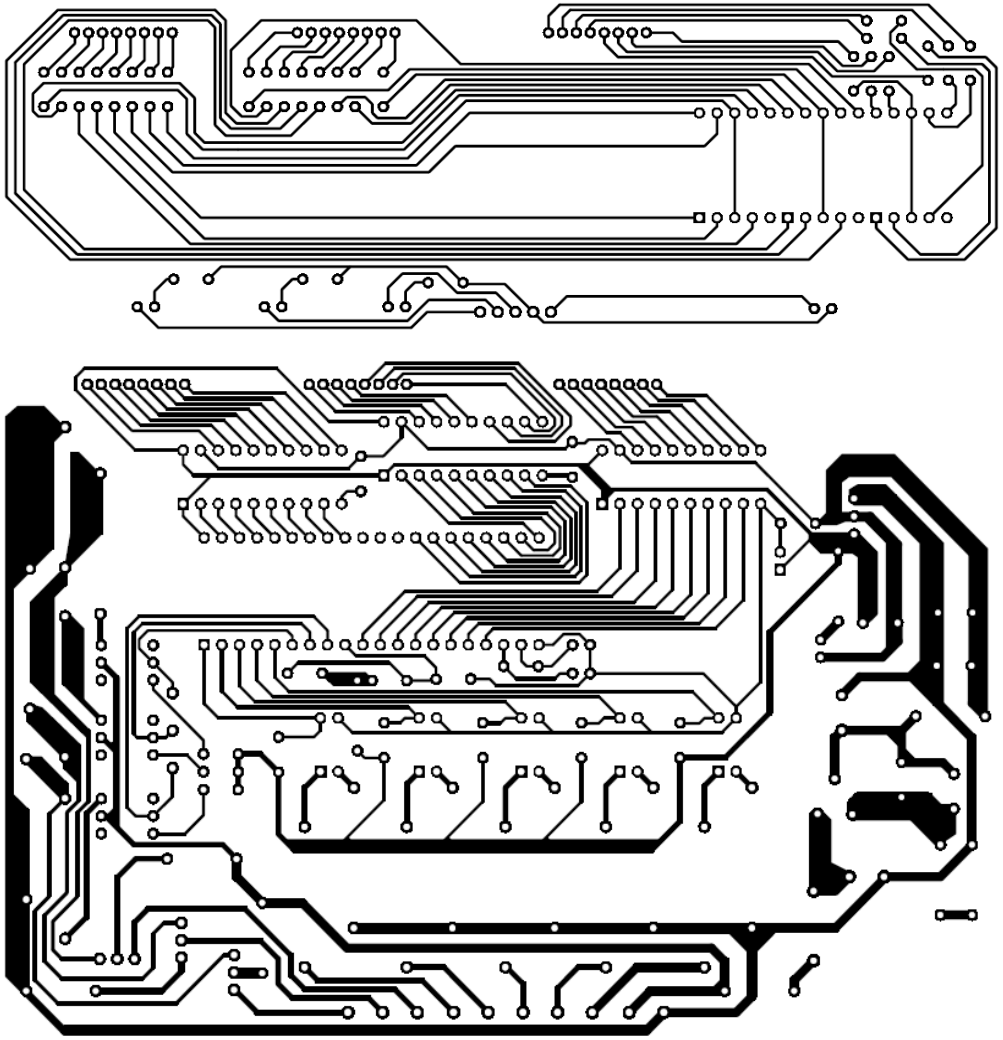


Title: VIGILANCE SAFETY DEVICE FOR LOCOMOTIVE		
Name: Circuit Diagram	**Sheet of:** 1/4	**Revision:** 0
Date: 3/26/2011	**Draw by:** Apichai Chaiprates	

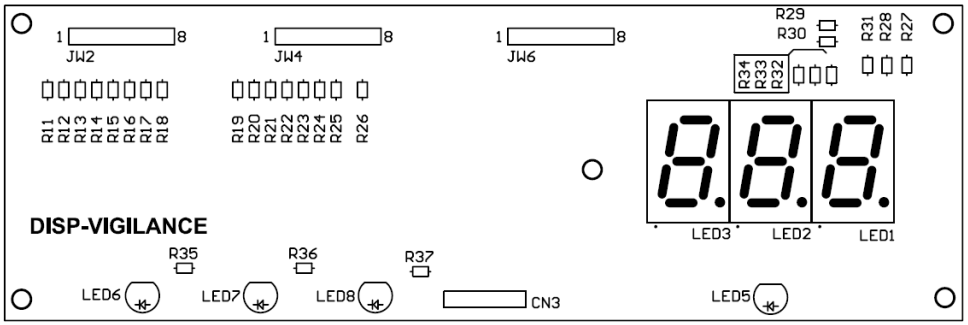


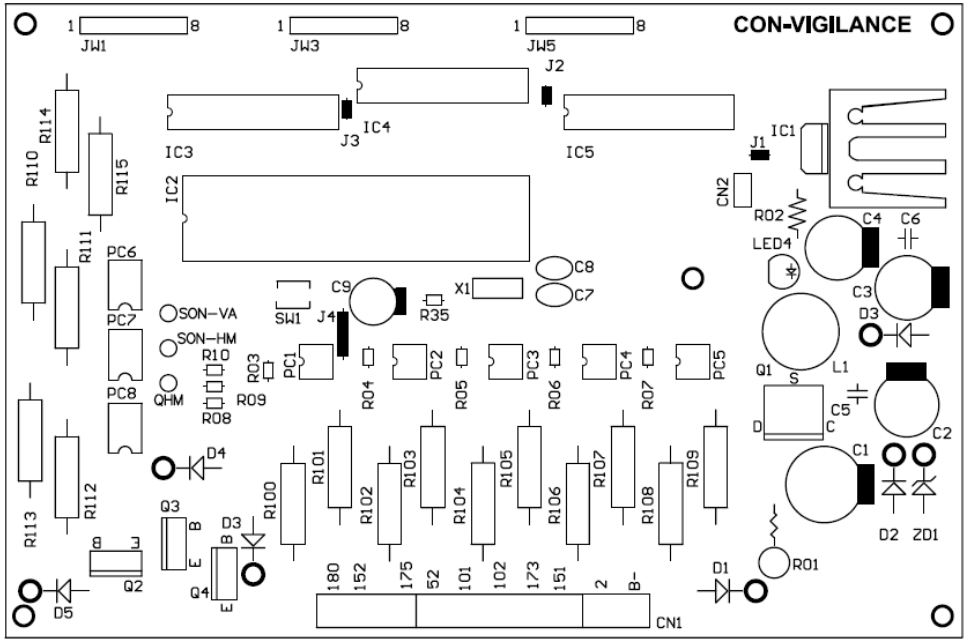


Title: VIGILANCE SAFETY DEVICE FOR LOCOMOTIVE		
Name: Schematic	**Sheet of:** 2/4	**Revision:** 0
Date: 3/26/2011	**Draw by:** Apichai Chaiprates	



Title: VIGILANCE SAFETY DEVICE FOR LOCOMOTIVE		
Name: Button Layer	**Sheet of:** 3/4	**Revision:** 0
Date: 3/26/2011	**Draw by:** Apichai Chaiprates	





Title: VIGILANCE SAFETY DEVICE FOR LOCOMOTIVE		
Name: Top Overlay	**Sheet of:** 4/4	**Revision:** 0
Date: 3/26/2011	**Draw by:** Apichai Chaiprates	

ภาคผนวก ข

โปรแกรมควบคุม

```
#include<reg51.h>
void system_stop();                    //Sub program of stop the vigilance systems
void mode_count8();                    //Sub program of start count 8 second
void mode_count102();                  //Sub program of start count 102 second
void mode_InputMonitoring();           //Sub program of monitoring these input
void mode_ESU();                       // Sub program of the reset by engine speed up
void mode_ESD();                       // Sub program of the reset by engine speed down
void mode_delay(unsigned char run);    //Sub program of delay time for bounce of switch
                                       contact
//Specification location of Input and Output signal
sbit SpeedMeter = P1^0;                //Set SpeedMeter = Port P1^0
sbit EngineSpeed_Up = P1^1;            //Set EngineSpeed_Up = Port P1^1
sbit EngineSpeed_Down = P1^2;          //Set EngineSpeed_Up = Port P1^2
sbit SandingSwitch = P1^3;             //Set SandingSwitch = Port P1^2
sbit PushAndFootSwitch = P1^4;         //Set FootSwitch = Port P1^3
sbit ElectricBell = P1^5;              //Set ElectricBell = P1^5
sbit ElectricHorn = P1^6;              //Set ElectricHorn = P1^6
sbit EmergencyBreak = P1^7;            //Set Standby = P1^7
//Sub program of Counter 1 second
unsigned char num[11] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,0x40};
void count_1second()
{
unsigned char x;
for(x=0;x<100;x++)
{
TMOD = 0x01;                   //Select TIMER0 mode0(16 bit) TMOD = 0000 0001
TH0 = 0xD8;                    //Require 65536 - (10000/1) = D8F0 H
TL0 = 0xF0;                    //Data of high - Low(TH0, TL0)
TR0 = 1;                       //Start TIMER0
```

```
while(TF0==0);                  //Wait over flow of TIMER0(TF= 1)
TR0=0;                          //After over flow TIMER0 stop
TF0 = 0;                        //Clear bit flag over flow
}
}
//Sub program of Sub program of Start the vigilance systems
void system_stop()
{
while(SpeedMeter==0)            //If speed meter is less than 8km/hrs. P0, P2 and P3 = 0x00
{
P0=0x00;                        //Clear Port P0=0x00
P2=0x00;                        //Clear Port P2=0x00
P3=0x00;                        //Clear Port P3=0x00
}
}
//Sub program of Start count 8 second
void mode_count8()
{
unsigned char a = 0, b = 0;     //Set initial value of a = 0 and b = 1
P0=0x00;                        //Clear Port P0=0x00
P2=0x00;                        //Clear Port P2=0x00
P3=0x00;                        //Clear Port P3=0x00
ElectricBell = 1;               //ElectricBell = 1(not yet operate)
ElectricHorn = 1;               //ElectricHorn = 1(not yet operate)
EmergencyBreak = 0;             //EmergencyBreak = 0(not yet operate)
while(PushAndFootSwitch==1)     //Monitoring operation condition of Push and Foot switch
{
if(SpeedMeter==0)               //Monitoring Speed Meter condition(start at 8km/hrs.)
{
ElectricBell = 1;               //ElectricBell = 1(not yet operate)
```

```
ElectricHorn = 1;                //ElectricHorn = 1(not yet operate)
a = 0;                           //Setup the initial value of parameter "a" = 0
system_stop();                   //The vigilance systems was stop
}
P3=num[a];                       //Seven segment bit 3 start count 0
count_1second();                 //Sub program count_1second operate
a++;                             //Increase parameter "a" were 1
if(a>8)                          //Monitoring, if a more than 8 the electric bell will be warn
{
ElectricBell = 1;                //Electric bell no operate
ElectricHorn = 0;                //Electric horn warn
a=0;
while(PushAndFootSwitch==1)      //Monitoring operation condition of Push and Foot switch
{
if(SpeedMeter==0)                //Monitoring, if speed meter is less than 8km/hrs. The electric
                                 bell and electric horn was stop
{
ElectricBell = 1;                //Electric bell no operate
ElectricHorn = 1;                //Electric horn no operate
system_stop();                   //The vigilance systems was stop
mode_count8();                   //If speed meter is more than 8km/hrs. and there is no input, the
                                 system will be count 8 second
}
ElectricBell = 1;                //Electric bell no operate
ElectricHorn = 0;                //Electric horn warn
P3=num[b];
count_1second();                 //Sub program count_1second oeprate
b++;
if(b>4)                          //If more than 4 second the seven segment display " -   -   - "
                                 blink
```

```
{
EmergencyBreak = 1;             //If count over 4 second the automatic emergency break was
                                operate
while(SpeedMeter==1)
{
P0=num[10];                     //Port P0 display " - "
P2=num[10];                     //Port P2 display " - "
P3=num[10];                     //Port P3 display " - "
count_1second();
P0=0x00;                        //Port P0 display " "
P2=0x00;                        //Port P2 display " "
P3=0x00;                        //Port P3 display " "
count_1second();
}
P0=0x00;                        //Clear Port P0=0x00
P2=0x00;                        //Clear Port P2=0x00
P3=0x00;                        //Clear Port P3=0x00
system_stop();
a = 0, b = 0;
}
}
}
}
}
//Sub program of Start count 102 second
void mode_count102()
{
unsigned char i,j,k,l;          //Specify parameter i, j, k and l are unsigned char
i = 0;                          //Setup initial value of i = 0
j = 1;                          //Setup initial value of j = 1
```

```
k = 1;                          //Setup initial value of k = 1
l = 0;                          //Setup initial value of l = 0
P0=0x00;                        //Clear port P0
P2=0x00;                        //Clear port P2
P3=0x00;                        //Clear port P3
ElectricBell = 1;               //Setup the electric bell is stop condition
ElectricHorn = 1;               //Setup the electric horn is stop condition
EmergencyBreak = 0;             //EmergencyBreak = 0(not yet operate)
while(PushAndFootSwitch==0)     //Monitoring operation condition of Push and Foot switch
{
if(SpeedMeter==0)               //Monitoring Speed Meter condition(start at 8km/hrs.)
{
ElectricBell = 1;               //ElectricBell = 1(not yet operate)
ElectricHorn = 1;               //ElectricHorn = 1(not yet operate)
i = 0;                          //Setup initial value of i = 0
j = 1;                          //Setup initial value of j = 1
k = 1;                          //Setup initial value of k = 1
mode_count8();
}
P3 = num[i];                    //Seven segment bit 3 start count 0
count_1second();                //Call Sub program of Counter 1 second
i++;
if(EngineSpeed_Up==0)           //Monitoring, if reset the engine up the system will be again
                                count
{
ElectricBell = 1;               .//ElectricBell = 1(not yet operate)
ElectricHorn = 1;               //ElectricHorn = 1(not yet operate)
i = 0;                          //Reset initial value of i = 0
j = 1;                          //Reset initial value of j = 1
k = 1;                          //Reset initial value of k = 1
```

```
l = 0;                          //Reset initial value of l = 0
mode_ESU();                     //Call  sub program of reset by engine speed up
}
if(EngineSpeed_Down==0)         //Monitoring, if reset the engine down the system will be again
                                count
{
ElectricBell = 1;               //ElectricBell = 1(not yet operate)
ElectricHorn = 1;               //ElectricHorn = 1(not yet operate)
i = 0;                          //Reset initial value of i = 0
j = 1;                          //Reset initial value of j = 1
k = 1;                          //Reset initial value of k = 1
l = 0;                          //Reset initial value of l = 0
mode_ESD();                     //Call  sub program of reset by engine speed down
}
if(SandingSwitch==1)            //Monitoring, if reset the Sanding Switch the system will be
                                again count
{
ElectricBell = 1;               //ElectricBell = 1(not yet operate)
ElectricHorn = 1;               //ElectricHorn = 1(not yet operate)
P0=0x00;                        //Clear Port P0=0x00
P2=0x00;                        //Clear Port P2=0x00
P3=0x00;                        //Clear Port P3=0x00
i = 0;                          //Reset initial value of i = 0
j = 1;                          //Reset initial value of j = 1
k = 1;                          //Reset initial value of k = 1
l = 0;                          //Reset initial value of l = 0
system_stop();                  //The vigilance systems was stop
}
if(i>8 && l>0)                  //After count to 102 second, seven segment bit 3 will be
                                continue count 8 second
```

```
{
i=0;
l++;
ElectricHorn = 0;
}
if(i>4 && l>1)                  //After count 8 second, seven segment bit 3 will be continue
                                count 4 second
{
EmergencyBreak = 1;             //If count over 4 second the automatic emergency break was
                                operate
while(SpeedMeter==1)            //Seven segment bit1, 2 and 3  display "  -  -  -  " and blink
{
P0=num[10];                     //Port P0 display " - "
P2=num[10];                     //Port P2 display " - "
P3=num[10];                     //Port P3 display " - "
count_1second();
P0=0x00;                        //Port P0 display " "
P2=0x00;                        //Port P2 display " "
P3=0x00;                        //Port P3 display " "
count_1second();
}
P0=0x00;                        //Clear Port P0=0x00
P2=0x00;                        //Clear Port P2=0x00
P3=0x00;                        //Clear Port P3=0x00
system_stop();
i=0,j=1,k=1,l=0;
}
if(i>9)                         //Monitoring, if count is more than 9 parameter i become 0
{
```

```
i = 0;

P2 = num[j];                    //Seven segment bit 2 start count 1
j++;                            //Increse j = 1
if(EngineSpeed_Up==0)           //Monitoring, if reset the engine up the system will be again
                                count
{
ElectricBell = 1;               //ElectricBell = 1(not yet operate)
ElectricHorn = 1;               //ElectricHorn = 1(not yet operate)
i = 0;                          //Reset initial value of i = 0
j = 1;                          //Reset initial value of j = 1
k = 1;                          //Reset initial value of k = 1
l = 0;                          //Reset initial value of l = 0
mode_ESU();                     //Call  sub program of reset by engine speed up
}
if(EngineSpeed_Down==0)         //Monitoring, if reset the engine down the system will be again
                                count
{
ElectricBell = 1;               //ElectricBell = 1(not yet operate)
ElectricHorn = 1;               //ElectricHorn = 1(not yet operate)
i = 0;                          //Reset initial value of i = 0
j = 1;                          //Reset initial value of j = 1
k = 1;                          //Reset initial value of k = 1
l = 0;                          //Reset initial value of l = 0
mode_ESD();                     //Call  sub program of reset by engine speed down
}
if(SandingSwitch==1)            //Monitoring, if reset the Sanding Switch the system will be
                                again count
{
ElectricBell = 1;               //ElectricBell = 1(not yet operate)
```

```
ElectricHorn = 1;          //ElectricHorn = 1(not yet operate)
P0=0x00;                   //Clear Port P0=0x00
P2=0x00;                   //Clear Port P2=0x00
P3=0x00;                   //Clear Port P3=0x00
i = 0;                     //Reset initial value of i = 0
j = 1;                     //Reset initial value of j = 1
k = 1;                     //Reset initial value of k = 1
l = 0;                     //Reset initial value of l = 0
system_stop();             //The vigilance systems was stop
}
if(j>10)                   //Monitoring, if bit 2 count is more than 9 parameter j = 1
{
j = 1;
P2 = num[i];               //Seven segment bit 2 display number conform value of i
P0 = num[k];               //Seven segment bit 1 display number conform value of i
k++;                       //Increse k = 1
if(EngineSpeed_Up==0)      //Monitoring, if reset the engine up the system will be again
                           count
{
ElectricBell = 1;          //ElectricBell = 1(not yet operate)
\ElectricHorn = 1;         //ElectricHorn = 1(not yet operate)
i = 0;                     //Reset initial value of i = 0
j = 1;                     //Reset initial value of j = 1
k = 1;                     //Reset initial value of k = 1
l = 0;                     //Reset initial value of l = 0
mode_ESU();                //Call  sub program of reset by engine speed up
}
if(EngineSpeed_Down==0)    //Monitoring, if reset the engine down the system will be again
                           count
{
```

```
ElectricBell = 1;                //ElectricBell = 1(not yet operate)
ElectricHorn = 1;                //ElectricHorn = 1(not yet operate)
i = 0;                           //Reset initial value of i = 0
j = 1;                           //Reset initial value of j = 1
k = 1;                           //Reset initial value of k = 1
l = 0;                           //Reset initial value of l = 0
mode_ESD();                      //Call  sub program of reset by engine speed down
}
if(SandingSwitch==1)             //Monitoring, if reset the Sanding Switch the system will be
                                 again count
{
ElectricBell = 1;                //ElectricBell = 1(not yet operate)
ElectricHorn = 1;                //ElectricHorn = 1(not yet operate)
P0=0x00;                         //Clear Port P0=0x00
P2=0x00;                         //Clear Port P2=0x00
P3=0x00;                         //Clear Port P3=0x00
i = 0;                           //Reset initial value of i = 0
j = 1;                           //Reset initial value of j = 1
k = 1;                           //Reset initial value of k = 1
l = 0;                           //Reset initial value of l = 0
system_stop();                   //The vigilance systems was stop
}
}
}
//Monitoring, if k is more than 1 and i is more than 2 call sub program count_1second
if(k>1)
{
if(i>2)                          //Monitoring, if count to 102 second
{
i = 0;                           //Setup initial value of i = 0
```

```
j = 1;                          //Setup initial value of j = 1
k = 1;                          //Setup initial value of k = 1
P0 = 0x00;                      //Seven segment bit 1 not display
P2 = 0x00;                      //Seven segment bit 2 not display
P3 = num[l];                    //Seven segment bit 3 display number conform the value of l
l++;
ElectricBell = 0;
if(EngineSpeed_Up==0)           //Monitoring, if reset the engine up/down the system will be
                                again count
{
ElectricBell = 1;               //ElectricBell = 1(not yet operate)
ElectricHorn = 1;               //ElectricHorn = 1(not yet operate)
i = 0;                          //Reset initial value of i = 0
j = 1;                          //Reset initial value of j = 1
k = 1;                          //Reset initial value of k = 1
l = 0;                          //Reset initial value of k = 0
mode_ESU();                     //Call  sub program of reset by engine speed up
}
if(EngineSpeed_Down==0)         //Monitoring, if reset the engine down the system will be again
                                count
{
ElectricBell = 1;               //ElectricBell = 1(not yet operate)
ElectricHorn = 1;               //ElectricHorn = 1(not yet operate)
i = 0;                          //Reset initial value of i = 0
j = 1;                          //Reset initial value of j = 1
k = 1;                          //Reset initial value of k = 1
l = 0;                          //Reset initial value of l = 0
mode_ESD();                     //Call  sub program of reset by engine speed down
}
```

```
if(SandingSwitch==1)            //Monitoring, if reset the Sanding Switch the system will be
                                again count
{
ElectricBell = 1;               //ElectricBell = 1(not yet operate)
ElectricHorn = 1;               //ElectricHorn = 1(not yet operate)
P0=0x00;                        //Clear Port P0=0x00
P2=0x00;                        //Clear Port P2=0x00
P3=0x00;                        //Clear Port P3=0x00
i = 0;                          //Reset initial value of i = 0
j = 1;                          //Reset initial value of j = 1
k = 1;                          //Reset initial value of k = 1
l = 0;                          //Reset initial value of l = 0
system_stop();                  //The vigilance systems was stop
}
}
}
}
mode_count8();                  //Call sub program mode_count8
}
//Sub program of reset by engine speed up
void mode_ESU()
{
unsigned char ii,jj,kk,ll;      //Specify parameter i, j, k and l are unsigned char
ii = 0;                         //Setup initial value of i = 0
jj = 1;                         //Setup initial value of j = 1
kk = 1;                         //Setup initial value of k = 1
ll = 0;                         //Setup initial value of l = 0
P0=0x00;                        //Clear port P0
P2=0x00;                        //Clear port P2
P3=0x00;                        //Clear port P3
```

```
ElectricBell = 1;            //Setup the electric bell is stop condition
ElectricHorn = 1;            //Setup the electric horn is stop condition
EmergencyBreak = 0;          //EmergencyBreak = 0(not yet operate)
while(EngineSpeed_Up==0)     //Monitoring operation condition of Push and Foot switch
{
if(SpeedMeter==0)            //Monitoring Speed Meter condition(start at 8km/hrs.)
{
ElectricBell = 1;            //ElectricBell = 1(not yet operate)
ElectricHorn = 1;            //ElectricHorn = 1(not yet operate)
ii = 0;                      //Setup initial value of i = 0
jj = 1;                      //Setup initial value of j = 1
kk = 1;                      //Setup initial value of k = 1
mode_count8();
}
P3 = num[ii];                //Seven segment bit 3 start count 0
count_1second();             //Call Sub program of Counter 1 second
ii++;
if(SandingSwitch==1)         //Monitoring, if reset the Sanding Switch the system will be
                             again count
{
ElectricBell = 1;            //ElectricBell = 1(not yet operate)
ElectricHorn = 1;            //ElectricHorn = 1(not yet operate)
P0=0x00;                     //Clear Port P0=0x00
P2=0x00;                     //Clear Port P2=0x00
P3=0x00;                     //Clear Port P3=0x00
ii = 0;                      //Reset initial value of i = 0
jj = 1;                      //Reset initial value of j = 1
kk = 1;                      //Reset initial value of k = 1
ll = 0;                      //Reset initial value of l = 0
system_stop();               //The vigilance systems was stop
```

```
}
if(ii>8 && ll>0)            //After count to 102 second, seven segment bit 3 will be
                            continue count 8 second
{
ii=0;
ll++;
ElectricHorn = 0;
}
if(ii>4 && ll>1)            //After count 8 second, seven segment bit 3 will be continue
                            count 4 second
{
EmergencyBreak = 1;         //If count over 4 second the automatic emergency break was
                            operate
while(SpeedMeter==1)        //Seven segment bit1, 2 and 3  display " -  -  - " and blink
{
ii=0,jj=1,kk=1,ll=0;
P0=num[10];                 //Port P0 display " - "
P2=num[10];                 //Port P2 display " - "
P3=num[10];                 //Port P3 display " - "
count_1second();
P0=0x00;                    //Port P0 display " "
P2=0x00;                    //Port P2 display " "
P3=0x00;                    //Port P3 display " "
count_1second();
}
P0=0x00;                    //Clear Port P0=0x00
P2=0x00;                    //Clear Port P2=0x00
P3=0x00;                    //Clear Port P3=0x00
system_stop();
ii = 0, jj = 0, kk = 0, ll = 0;
```

```
}
if(ii>9)                    //Monitoring, if count is more than 9 parameter i become 0
{
ii = 0;
P2 = num[jj];               //Seven segment bit 2 start count 1
jj++;                       //Increse j = 1
if(SandingSwitch==1)        //Monitoring, if reset the Sanding Switch the system will be
                            again count
{
ElectricBell = 1;           //ElectricBell = 1(not yet operate)
ElectricHorn = 1;           //ElectricHorn = 1(not yet operate)
P0=0x00;                    //Clear Port P0=0x00
P2=0x00;                    //Clear Port P2=0x00
P3=0x00;                    //Clear Port P3=0x00
ii = 0;                     //Reset initial value of i = 0
jj = 1;                     //Reset initial value of j = 1
kk = 1;                     //Reset initial value of k = 1
ll = 0;                     //Reset initial value of l = 0
system_stop();              //The vigilance systems was stop
}
if(jj>10)                   //Monitoring, if bit 2 count is more than 9 parameter j = 1
{
jj = 1;
P2 = num[ii];               //Seven segment bit 2 display number conform value of i
P0 = num[kk];               //Seven segment bit 1 display number conform value of i
kk++;                       //Increse k = 1
if(SandingSwitch==1)        //Monitoring, if reset the Sanding Switch the system will be
                            again count
{
ElectricBell = 1;           //ElectricBell = 1(not yet operate)
```

```
ElectricHorn = 1;                //ElectricHorn = 1(not yet operate)
P0=0x00;                         //Clear Port P0=0x00
P2=0x00;                         //Clear Port P2=0x00
P3=0x00;                         //Clear Port P3=0x00
ii = 0;                          //Reset initial value of i = 0
jj = 1;                          //Reset initial value of j = 1
kk = 1;                          //Reset initial value of k = 1
ll = 0;                          //Reset initial value of l = 0
system_stop();                   //The vigilance systems was stop
}
}
}
//Monitoring, if k is more than 1 and i is more than 2 call sub program count_1second
if(kk>1)
{
if(ii>2)                         //Monitoring, if count to 102 second
{
ii = 0;                          //Setup initial value of i = 0
jj = 1;                          //Setup initial value of j = 1
kk = 1;                          //Setup initial value of k = 1
P0 = 0x00;                       //Seven segment bit 1 not display
P2 = 0x00;                       //Seven segment bit 2 not display
P3 = num[ll];                    //Seven segment bit 3 display number conform the value of l
ll++;
ElectricBell = 0;
if(SandingSwitch==1)             //Monitoring, if reset the Sanding Switch the system will be
                                 again count
{
ElectricBell = 1;                //ElectricBell = 1(not yet operate)
ElectricHorn = 1;                //ElectricHorn = 1(not yet operate)
```

```
P0=0x00;                    //Clear Port P0=0x00
P2=0x00;                    //Clear Port P2=0x00
P3=0x00;                    //Clear Port P3=0x00
ii = 0;                     //Reset initial value of i = 0
jj = 1;                     //Reset initial value of j = 1
kk = 1;                     //Reset initial value of k = 1
ll = 0;                     //Reset initial value of l = 0
system_stop();              //The vigilance systems was stop
}
}
}
}
mode_count8();              //Call sub program mode_count8
}
//Sub program of reset by engine speed up

void mode_ESP()
{
unsigned char ii,jj,kk,ll;  //Specify parameter i, j, k and l are unsigned char
ii = 0;                     //Setup initial value of i = 0
jj = 1;                     //Setup initial value of j = 1
kk = 1;                     //Setup initial value of k = 1
ll = 0;                     //Setup initial value of l = 0
P0=0x00;                    //Clear port P0
P2=0x00;                    //Clear port P2
P3=0x00;                    //Clear port P3
ElectricBell = 1;           //Setup the electric bell is stop condition
ElectricHorn = 1;           //Setup the electric horn is stop condition
EmergencyBreak = 0;         //EmergencyBreak = 0(not yet operate)
while(EngineSpeed_Up==0)    //Monitoring operation condition of Push and Foot switch
```

```
{
if(SpeedMeter==0)          //Monitoring Speed Meter condition(start at 8km/hrs.)
{
ElectricBell = 1;          //ElectricBell = 1(not yet operate)
ElectricHorn = 1;          //ElectricHorn = 1(not yet operate)
ii = 0;                    //Setup initial value of i = 0
jj = 1;                    //Setup initial value of j = 1
kk = 1;                    //Setup initial value of k = 1
mode_count8();
}
P3 = num[ii];              //Seven segment bit 3 start count 0
count_1second();           //Call Sub program of Counter 1 second
ii++;
if(SandingSwitch==1)       //Monitoring, if reset the Sanding Switch the system will be
                           again count
{
ElectricBell = 1;          //ElectricBell = 1(not yet operate)
ElectricHorn = 1;          //ElectricHorn = 1(not yet operate)
P0=0x00;                   //Clear Port P0=0x00
P2=0x00;                   //Clear Port P2=0x00
P3=0x00;                   //Clear Port P3=0x00
ii = 0;                    //Reset initial value of i = 0
jj = 1;                    //Reset initial value of j = 1
kk = 1;                    //Reset initial value of k = 1
ll = 0;                    //Reset initial value of l = 0
system_stop();             //The vigilance systems was stop
}
if(ii>8 && ll>0)           //After count to 102 second, seven segment bit 3 will be
                           continue count 8 second
{
```

```
ii=0;
ll++;
ElectricHorn = 0;
}
if(ii>4 && ll>1)            //After count 8 second, seven segment bit 3 will be continue
                            count 4 second
{
EmergencyBreak = 1;         //If count over 4 second the automatic emergency break was
                            operate
while(SpeedMeter==1)        //Seven segment bit1, 2 and 3  display "  -  -  -  " and blink
{
ii=0,jj=1,kk=1,ll=0;
P0=num[10];                 //Port P0 display " - "
P2=num[10];                 //Port P2 display " - "
P3=num[10];                 //Port P3 display " - "
count_1second();
P0=0x00;                    //Port P0 display " "
P2=0x00;                    //Port P2 display " "
P3=0x00;                    //Port P3 display " "
count_1second();
}
P0=0x00;                    //Clear Port P0=0x00
P2=0x00;                    //Clear Port P2=0x00
P3=0x00;                    //Clear Port P3=0x00
system_stop();
ii = 0, jj = 0, kk = 0, ll = 0;
}
if(ii>9)                    //Monitoring, if count is more than 9 parameter i become 0
{
ii = 0;
```

```
P2 = num[jj];                    //Seven segment bit 2 start count 1
jj++;                            //Increse j = 1
if(SandingSwitch==1)             //Monitoring, if reset the Sanding Switch the system will be
                                 again count
{
ElectricBell = 1;                //ElectricBell = 1(not yet operate)
ElectricHorn = 1;                //ElectricHorn = 1(not yet operate)
P0=0x00;                         //Clear Port P0=0x00
P2=0x00;                         //Clear Port P2=0x00
P3=0x00;                         //Clear Port P3=0x00
ii = 0;                          //Reset initial value of i = 0
jj = 1;                          //Reset initial value of j = 1
kk = 1;                          //Reset initial value of k = 1
ll = 0;                          //Reset initial value of l = 0
system_stop();                   //The vigilance systems was stop
}
if(jj>10)                        //Monitoring, if bit 2 count is more than 9 parameter j = 1
{
jj = 1;
P2 = num[ii];                    //Seven segment bit 2 display number conform value of i
P0 = num[kk];                    //Seven segment bit 1 display number conform value of i
kk++;                            //Increse k = 1
if(SandingSwitch==1)             //Monitoring, if reset the Sanding Switch the system will be
                                 again count
{
ElectricBell = 1;                //ElectricBell = 1(not yet operate)
ElectricHorn = 1;                //ElectricHorn = 1(not yet operate)
P0=0x00;                         //Clear Port P0=0x00
P2=0x00;                         //Clear Port P2=0x00
P3=0x00;                         //Clear Port P3=0x00
```

```
ii = 0;                    //Reset initial value of i = 0
jj = 1;                    //Reset initial value of j = 1
kk = 1;                    //Reset initial value of k = 1
ll = 0;                    //Reset initial value of l = 0
system_stop();             //The vigilance systems was stop
}
}
}
//Monitoring, if k is more than 1 and i is more than 2 call sub program count_1second
if(kk>1)
{
if(ii>2)                   //Monitoring, if count to 102 second
{
ii = 0;                    //Setup initial value of i = 0
jj = 1;                    //Setup initial value of j = 1
kk = 1;                    //Setup initial value of k = 1
P0 = 0x00;                 //Seven segment bit 1 not display
P2 = 0x00;                 //Seven segment bit 2 not display
P3 = num[ll];              //Seven segment bit 3 display number conform the value of l
ll++;
ElectricBell = 0;
if(SandingSwitch==1)       //Monitoring, if reset the Sanding Switch the system will be
                           again count
{
ElectricBell = 1;          //ElectricBell = 1(not yet operate)
ElectricHorn = 1;          //ElectricHorn = 1(not yet operate)
P0=0x00;                   //Clear Port P0=0x00
P2=0x00;                   //Clear Port P2=0x00
P3=0x00;                   //Clear Port P3=0x00
ii = 0;                    //Reset initial value of i = 0
```

```
jj = 1;                         //Reset initial value of j = 1
kk = 1;                         //Reset initial value of k = 1
ll = 0;                         //Reset initial value of l = 0
system_stop();                  //The vigilance systems was stop
}
}
}
}
mode_count8();                  //Call sub program mode_count8
}
//Sub program of monitoring these inputs
void mode_InputMonitoring()
{
if(PushAndFootSwitch==0)        //Monitoring input condition
{
mode_delay(10);                 //Delay time for bounce of switch contact
if(PushAndFootSwitch==1)        //If, do not press Push or Foot switch system will be count 8
                                second
{
mode_count8();                  //The systems count 8 second
}
mode_count102();                //The systems count 102 second
}
else if(PushAndFootSwitch==1)
{
mode_count8();
}
}
//Main program
void main()
```

```
{
ElectricBell = 1;
ElectricHorn = 1;
EmergencyBreak = 0;
while(SpeedMeter==1)
{
mode_InputMonitoring();
}
}
//Sub program of Delay time for bounce of switch contact
void mode_delay(unsigned char run)
{
unsigned int x;
for(x=0;x<run;x++);
}
```

ภาคผนวก ค

DataSheet