รถหุ่นยนต์สำรวจ

ROBOTIC SURVEILLANCE CAR

โดย

นายทศพล สนทิม

นายธิติพันธุ์ ทองศิริ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

รถหุ่นยนต์สำรวจ

ROBOTIC SURVEILLANCE CAR

โดย

นายทศพล สนทิม 44010182

นายธิติพันธุ์ ทองศิริ 44010216

อาจารย์ที่ปรึกษา

อาจารย์พลศาสตร์ เลิศประเสริฐ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง รถหุ่นยนต์สำรวจ

ผู้จัดทำ

1. นายทศพล สนทิม 44010182
2. นายธิติพันธุ์ ทองศิริ 44010216

..... อาจารย์ที่ปรึกษา

(อาจารย์พลศาสตร์ เลิศประเสริฐ)

รถหุ่นยนต์สำรวจ

SURVEY ROBOTIC CAR

นายทศพล สนทิม 44010182

นายธิติพันธุ์ ทองศิริ 44010216

โครงงานนี้ได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....

(อาจารย์พลศาสตร์ เลิศประเสริฐ)

อาจารย์ที่ปรึกษา

I

รถหุ่นยนต์สำรวจ

นายทศพล สนทิม
นายธิติพันธุ์ ทองศิริ
อ.พลศาสตร์ เลิศประเสริฐ(อาจารย์ที่ปรึกษา)
ภาคการศึกษาที่ 2 ปีการศึกษา 2547

บทคัดย่อ

โครงงานนี้เป็นโครงงานที่เกี่ยวกับการควบคุมการเคลื่อนที่ของรถโดยรถจะขับเคลื่อนด้วยมอเตอร์เกียร์ 2 ตัว ซึ่งสามารถปรับความเร็วของมอเตอร์ทั้งสองตัวได้แบบอิสระตามการเปลี่ยนแปลงของจอยสติกแบบคันโยก โดยอาศัยไมโครคอนโทรลเลอร์ วงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล วงจรแปลงสัญญาณดิจิตอลเป็นอนาลอก วงจรขับมอเตอร์และวงจรรับส่งวิทยุ ใช้ร่วมกันในการควบคุมการเคลื่อนที่ของรถแบบไร้สาย โดยไมโครคอนโทรลเลอร์จะเป็นอุปกรณ์หลัก โดยในภาคส่งจะนำข้อมูลไปมอดดูเลทเพื่อส่งข้อมูลออกไปด้วยเครื่องส่งวิทยุ และในภาครับจะทำการแปลงข้อมูลที่รับได้ให้เป็นสัญญาณอนาลอกเพื่อนำไปควบคุมความเร็วของมอเตอร์ ในส่วนของการควบคุมการเลี้ยวซ้าย,ขวา,เดินหน้า,ถอยหลังจะใช้ไมโครคอนโทรลเลอร์ในการควบคุม

SURVEY ROBOT CAR

Mr.Totsapon Sontim

Mr.Titipan Thongsiri

Polasart Lertprasert(Adviser)

2st Semester Academic year 2004

Abstract

This project is about movement control of a robotic car. This robotic car runs by two gear motors. The speed of the two gear motor can be controlled by a joystick using microcontroller , analog to digital converter , multiplexer , driver motor and transmitter - receiver radio to wireless control movement of the robotic car. The microcontroller is the main device of this project.Transmission part it modulates digital data with carrier signal and it is transmitted to the air. In the receiver part , the received signal is digital data ,use digital to analog converter transform digital data to analog for use driving motor. For controlling the movement of the car used microcontroller.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงได้ด้วยดีนั้น ทางผู้จัดทำขอขอบคุณ อาจารย์ที่ปรึกษา อาจารย์พลศาสตร์ เลิศประเสริฐ ที่ได้ให้แนวคิด คำปรึกษาและความช่วยในขั้นตอนการทำงานต่างๆ ขอขอบคุณอาจารย์ในภาควิชาทุกท่านที่ได้ให้คำแนะนำ ขอขอบคุณรุ่นพี่และเพื่อนๆทุกคนที่ได้ให้ความช่วยเหลือในด้านต่างๆ และขอขอบคุณ บิดา มารดา รวมทั้งผู้มีพระคุณทุกท่าน

สารบัญ

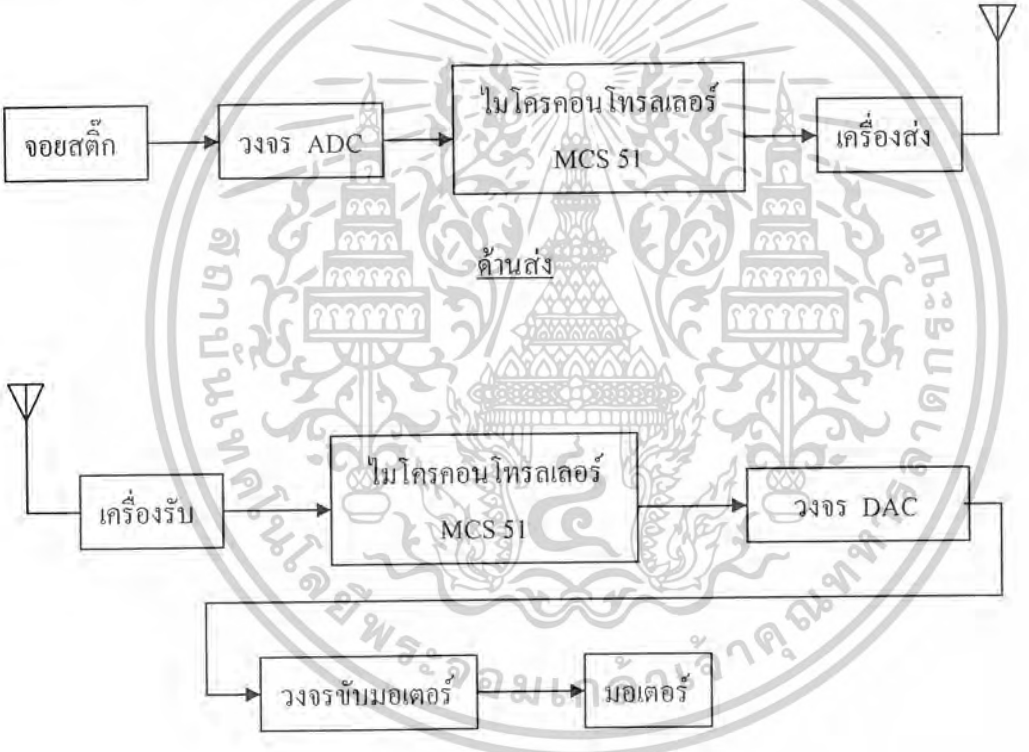
สารบัญรูป

สารบัญตาราง

ตอนที่ 1
บทนำ

1.1 ความเป็นมาและโครงสร้างของโครงงาน

ในปัจจุบันเทคโนโลยีทางด้านหุ่นยนต์ได้ถูกพัฒนาอย่างต่อเนื่อง เพื่อจะนำเอาความสามารถของหุ่นยนต์เหล่านี้ไปใช้ในงานที่เกินขีดความสามารถของมนุษย์หรือในงานที่เป็นอันตรายต่อมนุษย์ จากความสำคัญดังกล่าวจึงทำให้เกิดโครงงานนี้ขึ้น ซึ่งเป็นการนำเอาเทคโนโลยีต่างๆมาสร้างเป็นรถหุ่นยนต์ บล็อกไดอะแกรมของรถหุ่นยนต์แสดงดังรูปที่ 1.1



รูปที่ 1 บล็อกไดอะแกรมของรถหุ่นยนต์

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาการแปลงสัญญาณอนาลอกเป็นดิจิตอล
2. เพื่อศึกษาการใช้งานไมโครคอนโทรลเลอร์ในการควบคุมและต่อใช้งานร่วมกับอุปกรณ์ภายนอก
3. เพื่อศึกษาการรับ-ส่งสัญญาณควบคุมแบบไร้สาย
4. เพื่อศึกษาการควบคุมมอเตอร์โดยใช้ไมโครคอนโทรลเลอร์

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1. เข้าใจหลักการควบคุม DC มอเตอร์ ด้วยความกว้างของพัลส์ที่ใช้ขับ
2. สามารถใช้งานไมโครคอนโทรลเลอร์ร่วมกับอุปกรณ์อื่นๆได้
3. สามารถใช้งานวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลได้
4. สามารถควบคุมการทำงานของรถแบบไร้สายได้

1.4 ข้อกำหนดของโครงงาน

1. ใช้ไมโครคอนโทรลเลอร์ในการควบคุมการทำงาน
2. มีความละเอียดและแม่นยำพอควร
3. แสดงผลการทำงานให้เห็นได้ง่าย
4. ใช้อุปกรณ์รับส่งกำลังต่ำเพื่อพัฒนาเป็นเครื่องส่งกำลังสูงต่อไป

ตอนที่ 2
ทฤษฎีที่ใช้ในการออกแบบ

1. การแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาลอก (Digital to Analog)

1.1 คุณสมบัติของ DAC (DAC Characteristics)

เมื่อเอ่ยถึงวงจร DAC แล้วคุณสมบัติที่สำคัญบางประการที่ควรทราบ คือ

1) วงจร DAC นี้มีความแตกต่างของค่าสัญญาณอนาลอกเอาท์พุทมากหรือน้อยแค่ไหน

2) แรงดันอนาลอกเอาท์พุทมีการเปลี่ยนแปลงมากหรือน้อยแค่ไหน เมื่อสัญญาณดิจิตอลทางอินพุทเปลี่ยนแปลงไปหนึ่งบิท LSB (Least Significant Bit)

3) ฟังก์ชันถ่ายโอน (Transfer Function) ของวงจรมีความสัมพันธ์เป็นอย่างไร ทั้งนี้เพื่อให้สามารถคาดการณ์ค่าแรงดันอนาลอกเอาท์พุทได้เมื่อทราบแรงดันดิจิตอลอินพุท

1.1.1 รีโซลูชัน (Resolution)

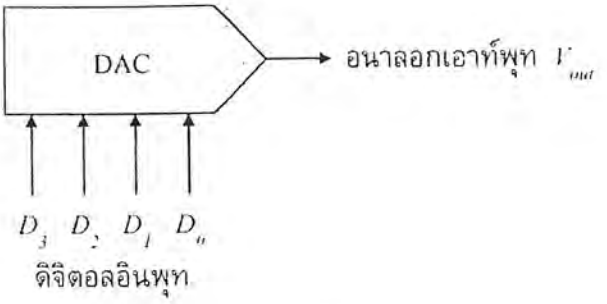
สัญลักษณ์และคุณสมบัติระหว่างสัญญาณดิจิตอลอินพุทกับสัญญาณอนาลอกเอาท์พุทของวงจร DAC ขนาด 4 บิท แสดงได้ดังรูปที่ 1 ซึ่งประกอบด้วยช่องรับสัญญาณดิจิตอลอินพุท 4 ช่อง ซึ่งแสดงถึงลักษณะของวงจร DAC ที่มีขนาด 4 บิท สภาวะของสัญญาณดิจิตอลอินพุททางไฟฟ้าในแต่ละบิทเป็นไปได้สองสภาวะ คือ โลจิก 1 (High) หรือ โลจิก 0 (Low) โดยที่ D_0 คือ บิทนัยสำคัญต่ำสุด หรือ บิท LSB (Least Significant Bit) และ D_3 คือ บิทนัยสำคัญสูงสุด หรือ บิท MSB (Most Significant Bit) รูปที่ 1 (ข) แสดงกราฟคุณสมบัติระหว่างค่าแรงดันอนาลอกเอาท์พุทเทียบกับดิจิตอลอินพุทจำนวน 16 ค่า

ในที่นี้คำว่า รีโซลูชัน (Resolution) สามารถนิยามได้สองลักษณะ ดังนี้คือ

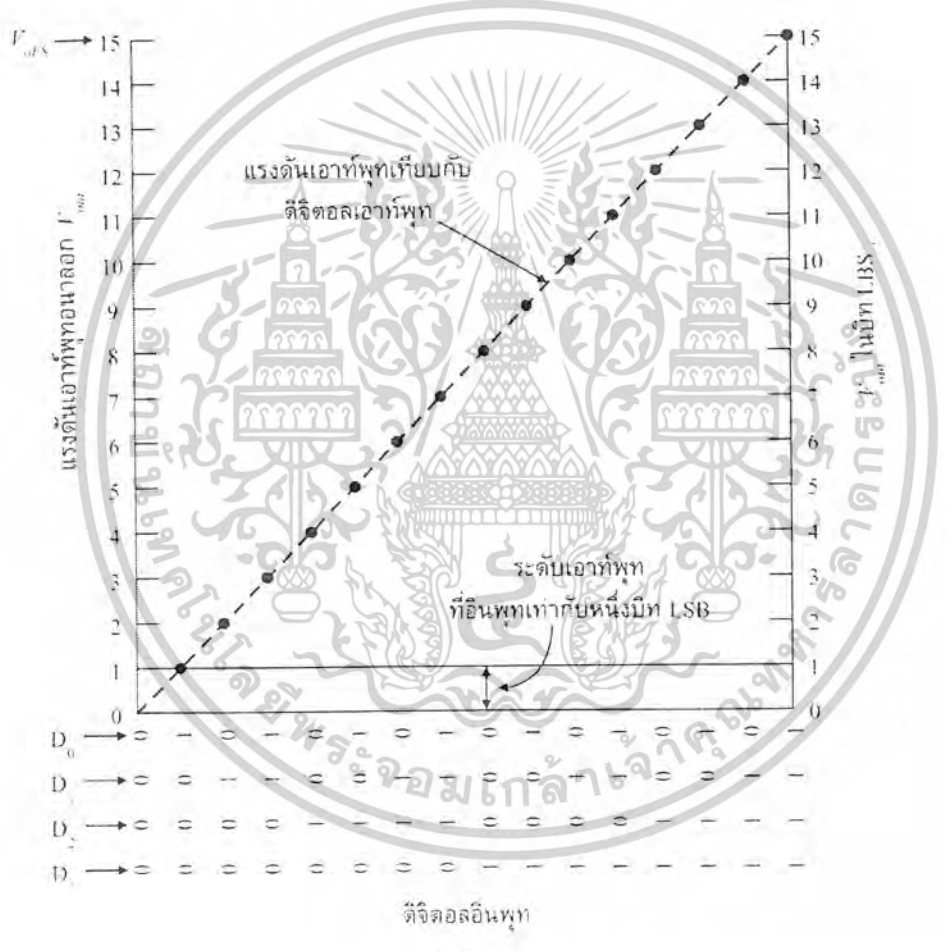
1) ค่ารีโซลูชัน หมายถึง ตัวเลขที่แสดงถึงค่าระดับความแตกต่างของสัญญาณอนาลอกเอาท์พุทของวงจร DAC นั้น ๆ สำหรับวงจร DAC ขนาด n บิทแล้ว สามารถเขียนอธิบายได้ว่า

$$\text{resolution} = 2^n$$

2) ค่ารีโซลูชัน หมายถึง อัตราส่วนของการเปลี่ยนแปลงสัญญาณอนาลอกเอาท์พุทต่อสัญญาณดิจิตอลอินพุทที่เปลี่ยนแปลงไปหนึ่งบิท LSB



(ก)



(ข)

รูปที่ 2 วงจร DAC ขนาด 4 บิท

(ก) สัญลักษณ์ของวงจร

(ข) กราฟคุณสมบัติระหว่างอินพุทกับเอาท์พุท

ในการคำนวณหาค่ารีโซลูชันของวงจร DAC ใดๆ ก็ตามจำเป็นต้องทราบคุณสมบัติประจำตัวของวงจร DAC คือ ขนาดแรงดันอนาลอกเอาท์พุทเต็มสเกล V_{OFS} (Full-Scale Output Voltage) และจำนวนของดิจิตอลอินพุท n ซึ่งคุณสมบัติทั้งสองค่านี้สามารถเปิดหาดูได้จาก Data Sheet ประจำตัววงจร DAC นั้น ดังนั้นค่ารีโซลูชันจึงสามารถคำนวณได้จากสมการดังต่อไปนี้

$$\text{resolution} = V_{OFS} / (2^n - 1)$$

โดยที่ V_{OFS} คือ แรงดันอนาลอกเอาท์พุทกรณีที่ดิจิตอลอินพุทมีสภาวะเป็นลอจิก 1 ทั้งหมดทุกบิท ตัวอย่างเช่น จากกราฟความสัมพันธ์ของวงจร DAC ขนาด 4 บิท ดังรูปที่ 1 (ข) จะได้ว่าแรงดันเอาท์พุท V_{OUT} มีค่าระดับความแตกต่างเท่ากับ $2^4 = 16$ ค่า ซึ่งมีค่าเปลี่ยนแปลงจาก 0 จนถึง 15 และได้ค่า $V_{OFS} = 15$ V เมื่อดิจิตอลอินพุทที่ป้อนให้กับวงจรมีค่าเท่ากับ 1111 ขณะที่เลขฐานสิบของ 1111 คือ 15 ดังนั้นกรณีนี้ค่ารีโซลูชันของวงจรจึงมีค่าเท่ากับ 15 V /15 = 1 V/LSB

1.1.2 ฟังก์ชันถ่ายโอน (Transfer Function)

ฟังก์ชันถ่ายโอน เป็นสมการแสดงถึงความสัมพันธ์ระหว่างสัญญาณอนาลอกเอาท์พุทกับสัญญาณดิจิตอลอินพุทของวงจร ซึ่งสามารถหาได้จากผลคูณของค่ารีโซลูชันกับค่าเปลี่ยนแปลงของสัญญาณดิจิตอลอินพุทในบิท LSB ดังนั้นจะได้

$$V_{out} = \text{resolution} \times D$$

โดยที่ V_{out} คือ แรงดันอนาลอกเอาท์พุท D คือ เลขฐานสิบของดิจิตอลอินพุท

1.2 คุณสมบัติของ ADC (ADC Characteristics)

1.2.1 สมการอินพุท-เอาท์พุท (Output-Input Equation)

สัญลักษณ์และคุณสมบัติระหว่างสัญญาณอนาลอกอินพุทกับสัญญาณดิจิตอลเอาท์พุทของวงจร ADC ขนาด 4 บิท แสดงดังรูปที่ 2 ประกอบด้วยช่องรับสัญญาณอนาลอกอินพุท 1 ช่องสัญญาณและช่องจ่ายสัญญาณดิจิตอลเอาท์พุทจำนวน 4 ช่องซึ่งแสดงถึงลักษณะของวงจร ADC สามารถกำหนดความหมายได้สองลักษณะเช่นกัน ดังนี้คือ

1) ค่ารีโซลูชันของวงจร ADC ขนาด n บิท มีค่าขึ้นกับจำนวนบิทสูงสุดของสัญญาณดิจิตอลเอาท์พุท ซึ่งสามารถเขียนอธิบายได้ว่า

$$\text{resolution} = 2^n$$

2) ค่ารีโซลูชัน หมายถึง อัตราส่วนการเปลี่ยนแปลงของสัญญาณอนาลอกอินพุท V_{in} ที่ทำให้สัญญาณดิจิตอลเอาท์พุทเปลี่ยนแปลงไปหนึ่งบิท LSB ดังนั้นหากทราบค่าแรงดันอนา

ลอกอินพุทเต็มสเกล V_{iFS} (Full-Scale Input Voltage) ที่ทำให้สัญญาณดิจิตอลเอาท์พุทมีสภาวะลอจิกเป็น 1 ทั้งหมดทุกบิทแล้ว จะสามารถคำนวณหาค่ารีโซลูชันของวงจรได้เท่ากับ

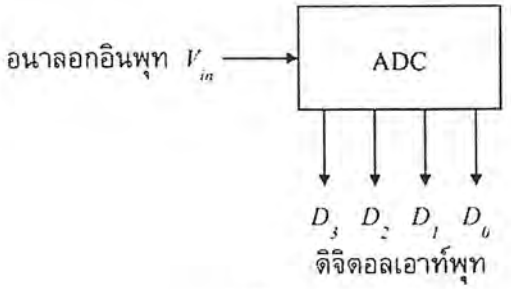
$$\text{resolution} = V_{iFS} / (2^n - 1)$$

สำหรับสมการอินพุท-เอาท์พุทของวงจร ADC สามารถหาได้จากความสัมพันธ์ ดังนี้

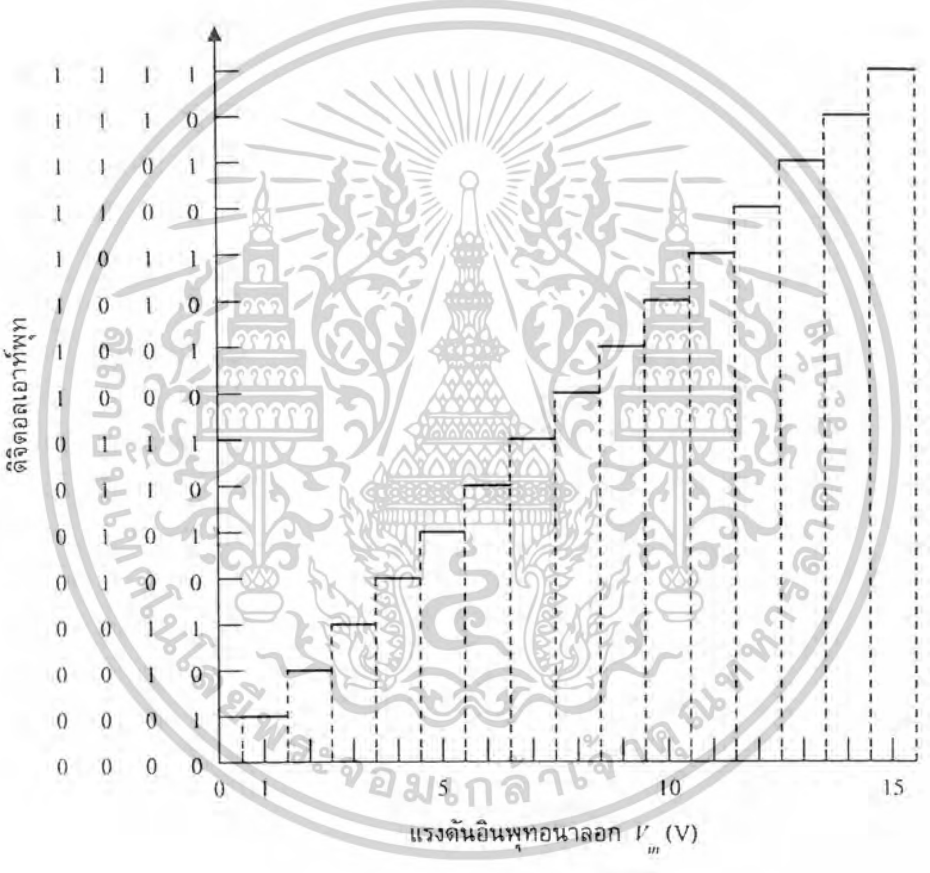
สัญญาณดิจิตอลเอาท์พุท = เลขฐานสองของ D

เมื่อ D มีค่าเท่ากับเลขฐานสิบของสัญญาณดิจิตอลเอาท์พุท หรือ D มีค่าเท่ากับจำนวณ LSB ของสัญญาณดิจิตอลเอาท์พุท ซึ่ง D สามารถหาได้จาก

$$D = Vin / \text{resolution}$$



(ก)



(ข)

รูปที่ 3 วงจร ADC ขนาด 4 บิท

(ก) สัญลักษณ์ของวงจร

(ข) กราฟคุณสมบัติระหว่างอินพุทกับเอาท์พุท

1.3 กระบวนการแปลงสัญญาณดิจิตอลให้เป็นสัญญาณอนาลอก

1.3.1 การทำงาน

หลักการทำงานพื้นฐานของกระบวนการแปลงสัญญาณดิจิตอลให้เป็นสัญญาณอนาลอกแสดงดังรูปที่ 3 วงจรสร้างแรงดันอ้างอิง V_{REF} ต่ออยู่กับวงจรแลดเดอร์ R-2R (R-2R ladder network) สัญญาณดิจิตอลอินพุทจะถูกส่งผ่านสวิตช์ควบคุมแบบดิจิตอล ในแต่ละบิทให้กับวงจรแลดเดอร์ R-2R เพื่อทำการเปลี่ยนสัญญาณดิจิตอลอินพุทให้เป็นสัญญาณอนาลอกในรูปของกระแสเอาท์พุท I_{out} และถูกเปลี่ยนให้เป็นแรงดันอนาลอกเอาท์พุท V_{out} โดยใช้วงจรเปลี่ยนกระแสให้เป็นแรงดัน ซึ่งกระแสเอาท์พุท I_{out} และแรงดันเอาท์พุท V_{out} ที่ได้จากวงจรจะมีค่าแปรผันเป็นสัดส่วนโดยตรงกับสัญญาณดิจิตอลอินพุทที่ป้อนให้กับวงจร

1.3.2 วงจรแลดเดอร์ R-2R (R-2R ladder network)

วงจรแลดเดอร์ R-2R แสดงดังรูปที่ 4 สัญญาณดิจิตอลอินพุทที่ป้อนให้กับวงจรมีตำแหน่งควบคุมสวิตช์กระแสตรงตามตำแหน่งบิทนั้นๆ เมื่อทำการพิจารณาค่าความต้านทานเทียบเท่าที่มองจากโหนด 0 (R_0) เข้าไปจะได้เท่ากับ $2R/(R+R) = R$ เช่นเดียวกันค่าความต้านทานเทียบเท่าที่มองจากโหนด 2 (R_2) และโหนด3 (R_3) เข้าไปต่างมีค่าเท่ากับ R เช่นกัน ดังนั้นจึงสามารถสรุปได้ว่า $R_0 = R_1 = R_2 = R_3 = R$ ทำให้กระแสอ้างอิง I_{REF} มีค่าประมาณ

$$I_{REF} = V_{REF} / R_3 = V_{REF} / R$$

ที่โหนด 3 กระแสอ้างอิง I_{REF} จะไหลแยกออกเป็นสองส่วนมีค่าเท่ากันโดยมีค่าเท่ากับ

$$I_3 = I_{REF} / 2$$

และเมื่อทำการพิจารณากระแสที่ไหลแยกออกจากโหนดแต่ละโหนด จะได้ความสัมพันธ์ของกระแสในแต่ละสาขา ดังนี้คือ

$$I_2 = I_3 / 2 = I_{REF} / 4$$

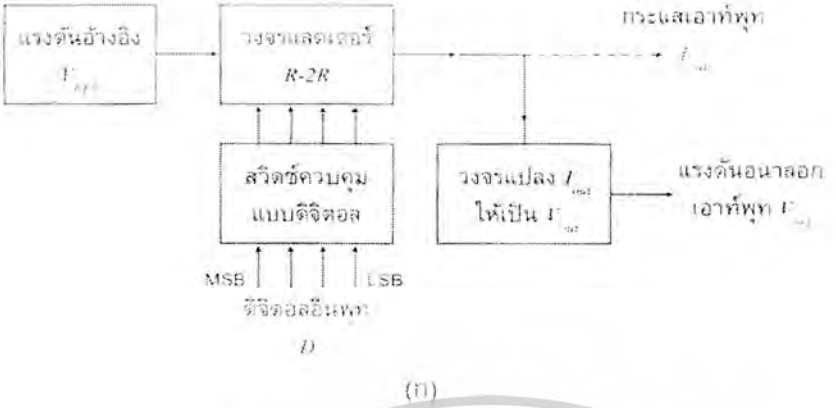
$$I_1 = I_2 / 2 = I_{REF} / 8$$

$$I_0 = I_1 / 2 = I_{REF} / 16$$

ผลรวมของกระแสเอาท์พุท I_{out} ที่โหนดเอาท์พุทหรือที่ output bus เมื่อสวิตช์กระแสทุกบิทอยู่ในตำแหน่ง 1 มีค่าเท่ากับ

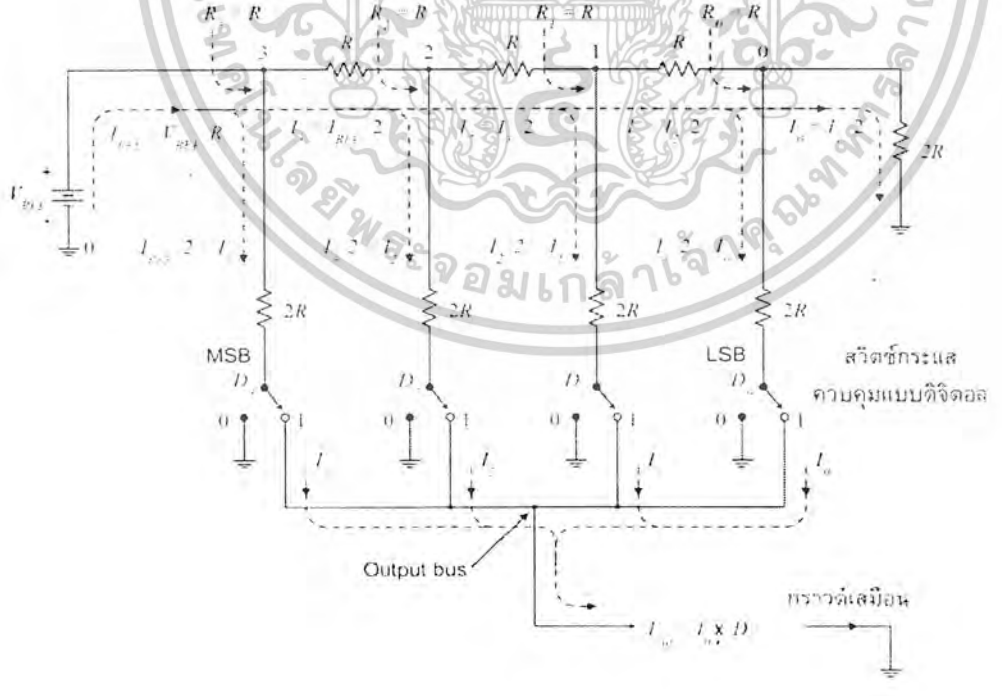
$$I_{out} = I_0 \times D$$

โดยที่ D มีค่าเท่ากับเลขฐานสิบของสัญญาณดิจิตอลอินพุท และ I_0 คือค่ากระแสต่ำสุดในวงจรแลดเดอร์ เมื่อกำหนดให้ I_0 คือค่ารีโซลูชันของวงจรแลดเดอร์



รูปที่ 4 การแปลงสัญญาณดิจิตอลให้เป็นสัญญาณอนาลอก

(ก) ผังการทำงาน (ข) สัญลักษณ์ของวงจร



รูปที่ 5 วงจรแลดเดอร์ R-2R

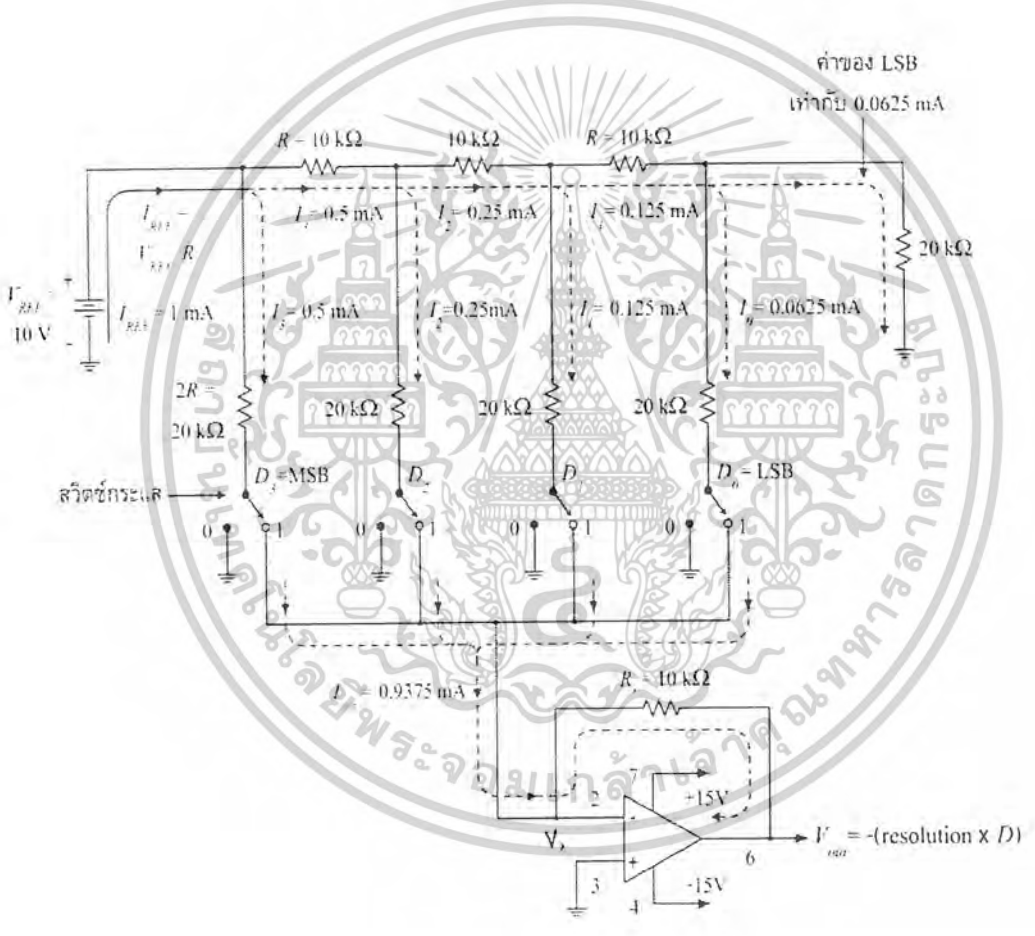
แปลงสัญญาณดิจิตอลให้เป็นกระแสอนาลอกเอาท์พุท I_{out}

1.3.3 แรงดันอนาลอกเอาท์พุทของวงจร DAC (Voltage Output DAC)

กระแสอนาลอกเอาท์พุท I_{out} ของวงจร DAC ในรูปที่ 4 สามารถเปลี่ยนให้อยู่ในรูปของแรงดันอนาลอกเอาท์พุท V_{out} ได้โดยการใช้ออปแอมป์ต่อร่วมกับตัวต้านทานป้อนกลับ R_f เพิ่มเข้าไปในวงจรเพื่อทำหน้าที่เปลี่ยนกระแสให้เป็นแรงดัน ดังรูปที่ 5

แรงดันเอาท์พุทของวงจร V_{out} มีค่าเท่ากับ

$$V_{out} = - I_{out} R_f$$



รูปที่ 6 การเปลี่ยนกระแสอนาลอกเอาท์พุท I_{out} ของวงจรแลดเดอร์ R-2R ให้เป็นแรงดันอนาลอกเอาท์พุท V_{out} โดยใช้ออปแอมป์กับตัวต้านทานป้อนกลับ R_f

แรงดันเอาท์พุทที่ได้คือ

$$V_{out} = -(1/2^n)\times(V_{REF}/R)\times R_f \times D$$

2. การแปลงสัญญาณอนาลอกให้เป็นสัญญาณดิจิตอล (Analog to Digital)

ADC สามารถแบ่งออกได้ตามช่วงเวลาที่ใช้ในการแปลงสัญญาณได้เป็น 3 ชนิดใหญ่ๆ ดังนี้คือ

1) แบบ Dual-slope integrating ADC โดยวิธีการนี้จะใช้เวลาในการแปลงสัญญาณประมาณ 300 ms

2) แบบประมาณค่าด้วยวิธีซัคเซสซีฟ (Successive approximation ADC) วิธีนี้ใช้เวลาในการแปลงสัญญาณค่อนข้างรวดเร็วในหน่วยของ μs นิยมใช้ในการประมวลผลสัญญาณเสียงดิจิตอล

3) แบบเฟลช (Flash ADC) เป็นแบบที่ใช้เวลาน้อยที่สุด แต่ก็มีราคาแพงมากที่สุดเช่นกัน มักใช้ในการประมวลผลสัญญาณวีดีโอดิจิตอล

2.1 <u>ADC แบบ Dual-Slope Integrating</u>

ผังการทำงานของ ADC แบบ Dual-slope integrating แสดงดังรูปที่ 6 ประกอบด้วยวงจรลอจิกควบคุม เป็นส่วนที่ทำหน้าที่ที่ควบคุมลอจิกการทำงานของวงจร และควบคุมสวิตช์อนาลอกเพื่อทำการแปลงแรงดันอนาลอกอินพุทให้เป็นสัญญาณดิจิตอลเอาท์พุท โดยมีวงจรสร้างสัญญาณนาฬิกาที่มีความถี่เท่ากับ 12 kHz ป้อนให้กับวงจรลอจิกควบคุม ซึ่งความถี่ของสัญญาณนาฬิกานี้สามารถแปรค่าได้โดยการกำหนดค่า R_T และ C_T จากภายนอก กระบวนการแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอลแบบนี้สามารถแบ่งเฟสในการทำงานได้ 3 เฟส คือ

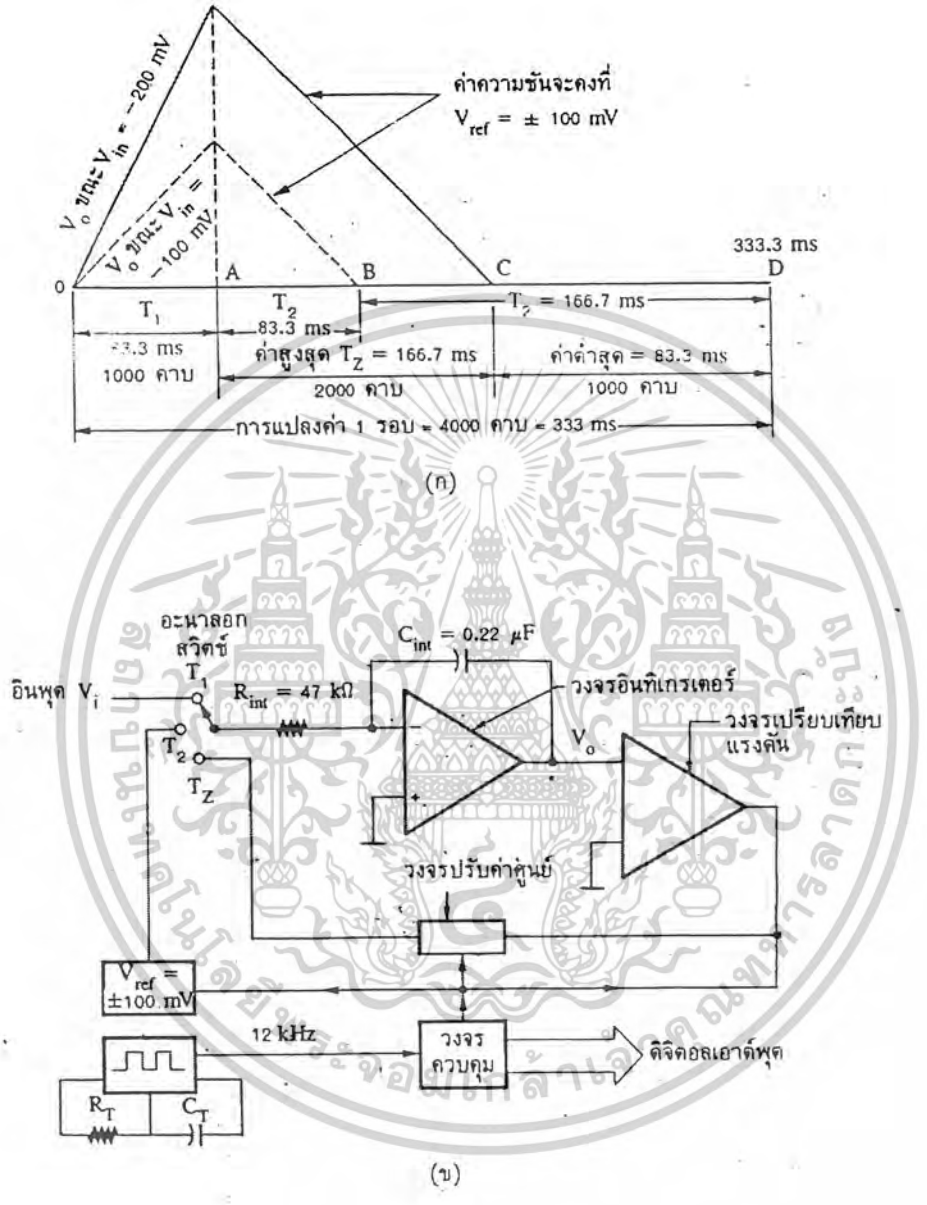
1) เฟสสัญญาณ (Signal Integrate Phase ,T_1)

2) เฟสอ้างอิง (Reference Integrate Phase ,T_2)

3) เฟสศูนย์อัตโนมัติ (Auto-Zero ,T_Z)

2.1.1 <u>เฟสสัญญาณ (Signal Integrate Phase ,T_1)</u>

การทำงานในเฟส T1 ของวงจรเริ่มต้นจากวงจรลอจิกควบคุมในรูปที่ 6(ข) จะทำการควบคุมให้สวิตช์อนาลอกต่อวงจรเพื่อป้อน Vin ให้กับวงจรอินทิเกรเตอร์ โดยแรงดันเอาท์พุทที่ได้จากวงจรอินทิเกรเตอร์นี้จะมีลักษณะเห็นสัญญาณลาด และมีทิศทางลาดขึ้นหรือลาดลงนั้นก็ขึ้นอยู่กับทิศทางของ Vin นั่นเอง ในจณะที่ความชันของสัญญาณลาดมีค่าขึ้นอยู่กับค่าของ Rint และ Cint หรือค่าคงที่เวลา (time constant)ของวงจร



รูปที่ 7 ADC แบบ Dual-Slope Integrating

(ก) ผังสัญญาณเวลาที่ T_1, T_2, T_Z

(ข) ผังแสดงการทำงาน

2.1.2 เฟสอ้างอิง (Reference Integrate Phase , T_2)

ขณะที่ทำงานในเฟส T_1 นั้นวงจรลอจิกควบคุมจะทำการประจุเข้าตัวเก็บประจุ C_{REF} (ไม่ได้แสดงไว้ในรูป) ซึ่งอยู่ภายในวงจรสร้างแรงดันอ้างอิง V_{REF} เอาไว้ด้วย เพื่อให้ V_{REF} มีค่าเป็น +100mV ดังนั้นเมื่อมีการเริ่มต้นทำงานในเฟส T_2 วงจรลอจิกควบคุมทำการควบคุมสวิตช์อนาลอกให้ต่อวงจรป้อน V_{REF} (ซึ่งมีทิศทางตรงกันข้ามกับ Vin) ให้กับวงจรอินทิเกรเตอร์ แรงดันเอาท์พุท จึงมีลักษณะเป็นสัญญาณลาดที่มีทิศทางลาดลงจรมีค่าเป็นศูนย์ และเนื่องจาก V_{REF} มีค่าคงที่ทำให้สัญญาณลาดที่ได้จากวงจรมีความชันคงที่ ดังแสดงดังรูปที่ 6 (ก)

หลังจาก Vout มีค่าลดลงจนเป็นศูนย์ วงจรเปรียบเทียบสัญญาณ จะทำการส่งสัญญาณให้กับวงจรลอจิกควบคุมเพื่อบอกให้รู้ว่าสิ้นสุดการทำงานในช่วง T_2 แล้ว และเริ่มต้นเข้าสู่การทำงานในเฟสศูนย์อัตโนมัติ Tz ต่อไป จากการทำงานในเฟส T_2 และกราฟสัญญาณในรูปที่ 6 (ก) พบว่าช่วงเวลา T2 นั้นมีค่าเป็นสัดส่วนโดยตรงกับ Vout สามารถสรุปความสัมพันธ์ได้ดังนี้

$$T_2 = T_1(V_{in}/V_{REF})$$

2.1.3 เฟสศูนย์อัตโนมัติ (Auto-Zero , Tz)

เมื่อเริ่มต้นการทำงานในเฟสที่สามหรือเฟสสุดท้ายของการแปลงสัญญาณ (Tz) วงจรลอจิควบคุมจะควบคุมให้สวิตช์อนาลอกต่อเข้ากับส่วนของวงจร Auto-Zero เพื่อทำการประจุให้กับตัวเก็บประจุ D_{AZ} (Auto-Zero Capacitor) ซึ่งอยู่ภายในวงจร C_{AZ} จะถูกประจุให้แรงดันตกคร่อมตัวมันมีค่าประมาณแรงดันคลาดเคลื่อนเฉลี่ย (average error voltage) ซึ่งเกิดจากวงจรอินทิเกรเตอร์และผลของแรงดันออฟเซต ซึ่งเกิดจากวงจรเปรียบเทียบสัญญาณ ดังนั้นหลังจากช่วงเวลา T_1 และ T_2 ผ่านพ้นไปแล้ว ค่าแรงดันคลาดเคลื่อนซึ่งประจุไว้ใน C_{AZ} จึงถูกต่อเข้ากับวงจรเพื่อหักล้างผลของแรงดันคลาดเคลื่อนที่สะสมไว้ใน C_{REF} ทำให้สัญญาณจาก ADC มีค่าเป็นศูนย์อย่างอัตโนมัติในทุกๆ ครั้งหลังจากการแปลงสัญญาณเสร็จสิ้นลง

2.2 ADC แบบประมาณค่าด้วยวิธีซัคเซสซีฟ (Successive approximation ADC)

ผังการทำงานของ ADC แบบประมาณค่าด้วยวิธีซัคเซสซีฟ แสดงดังรูปที่ 7 ซึ่งประกอบด้วยวงจร DAC วงจรเปรียบเทียบสัญญาณ และรีจิสเตอร์แบบSuccessive approximation (Successive approximation register)หรือ SAR โดยมีขั้วรับสัญญาณอินพุทที่เป็นแรงดันอนาลอกอินพุท หนึ่งขั้วสัญญาณ และมีสัญญาณดิจิตอลเอาท์พุทที่สามารถนำออกไปใช้งานได้ทั้งแบบขนานและแบบอนุกรม สัญญาณควบคุมการทำงานของระบบประกอบด้วยสัญญาณสามชุด คือ

1. สัญญาณเริ่มต้นการแปลงสัญญาณ (Start of Conversion) เป็นสัญญาณที่ป้อนให้กับ ADC เพื่อสั่งเริ่มต้นกระบวนการแปลงสัญญาณอนาลอกให้เป็นสัญญาณดิจิตอล

2. สัญญาณสิ้นสุดการแปลงสัญญาณ (End of Conversion) เป็นสัญญาณที่ป้อนให้กับ ADC เพื่อบอกให้รู้ว่ากระบวนการแปลงสัญญาณได้เสร็จสิ้นสมบูรณ์แล้ว

3. สัญญาณนาฬิกาภายนอก (Clock in) เป็นสัญญาณนาฬิกาจากภายนอกที่ป้อนให้กับ ADC เพื่อกำหนดฐานเวลาอ้างอิงให้กับกระบวนการแปลงสัญญาณ

2.2.1 หลักการทำงาน (Circuit Operation)

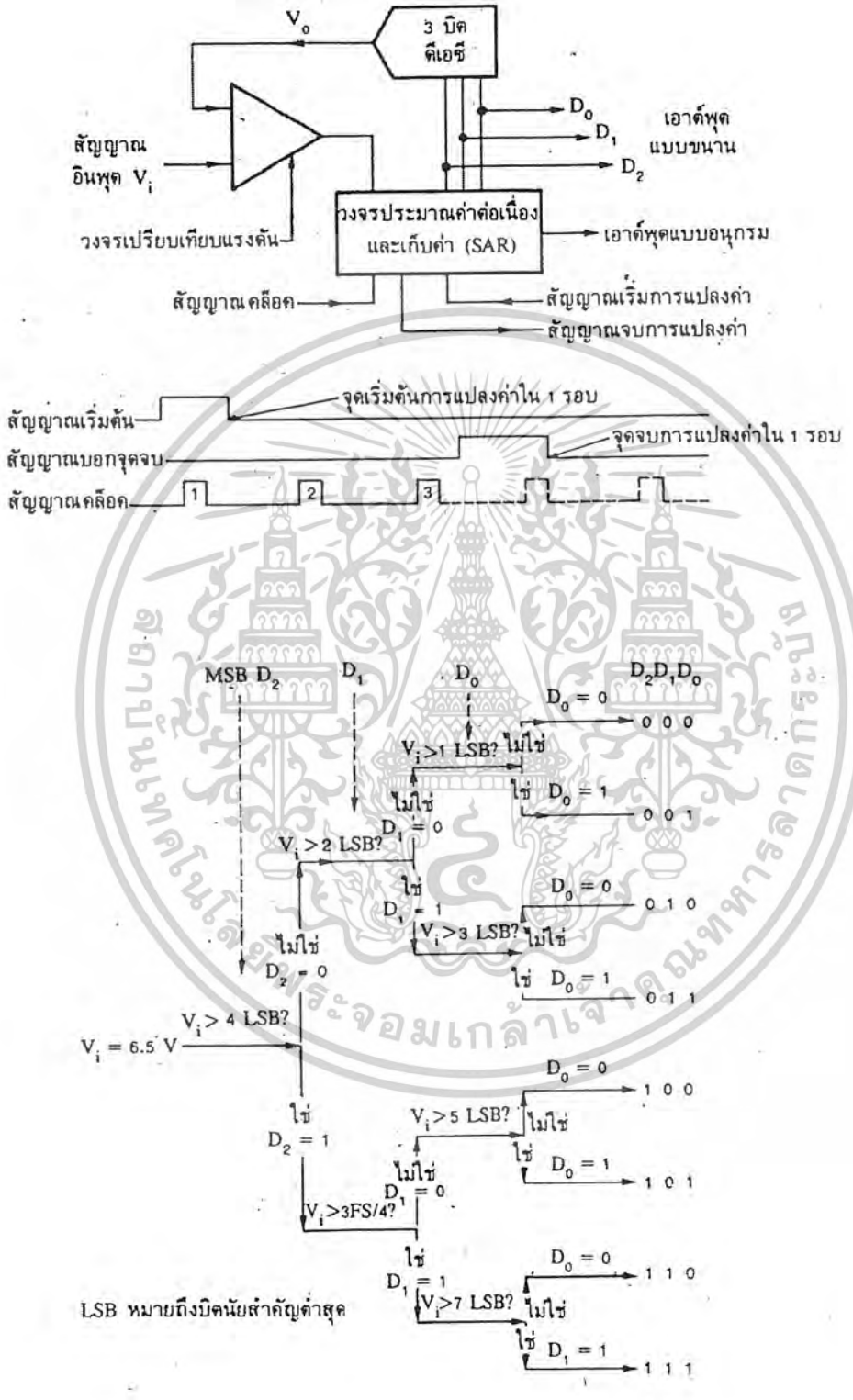
จากรูปที่ 7 กระบวนการแปลงสัญญาณอนาลอกให้เป็นสัญญาณดิจิตอลเริ่มต้นจากสัญญาณคำสั่งเริ่มต้นการแปลงสัญญาณ (Start of Conversion) จะเป็นสัญญาณกระตุ้นให้ ADC เริ่มต้นกระบวนการแปลงสัญญาณในไซเคิลแรก รีจิสเตอร์ SAR ซึ่งต่อตรงกับ DAC โดยเรียงข้อมูลตามลำดับบิทแต่ละบิทของ DAC จากนั้น DAC ทำการเปลี่ยนข้อมูลดิจิตอลในรีจิสเตอร์ SAR ให้เป็นแรงดันอนาลอกเอาท์พุท V_{out}

หลังจากนั้นวงจรเปรียบเทียบสัญญาณจะนำแรงดันอนาลอกเอาท์พุท Vout ไปเปรียบเทียบกับแรงดันอนาลอกอินพุท Vin ของวงจร สัญญาณเอาท์พุทที่ได้จากวงจรเปรียบเทียบสัญญาณจะเป็นสัญญาณบอกให้รีจิสเตอร์ SAR ทราบว่า Vin มีค่ามากกว่าหรือน้อยกว่า Vout ที่ได้จาก DAC เพื่อนำค่านั้นไปเปรียบเทียบแล้วส่งออกไปเป็นสัญญาณดิจิตอลเอาท์พุทของวงจร กรณีดิจิตอลเอาท์พุทขนาด 3 บิทแล้วการเปรียบเทียบจะเกิดขึ้น 3 ครั้ง

โดยการเปรียบเทียบเริ่มต้นจากบิท MSB ไปจนสิ้นสุดที่บิท LSB เมื่อการเปรียบเทียบที่บิท LSB สิ้นสุดลง SAR จะส่งสัญญาณ End of Conversion ออกมาเพื่อบอกให้ทราบว่าขณะนี้ขั้นตอนการแปลงสัญญาณของ ADC ได้เสร็จเรียบร้อยแล้ว และสัญญาณดิจิตอลเอาท์พุทของวงจรซึ่งเป็นค่าแปรผันตรงมาจาก Vin ก็จะปรากฏที่เอาท์พุทของ SAR

2.2.2 การประมาณค่าด้วยวิธีซัคเซสซีฟ (Successive Aproximation Analogy)

จากแผนผังแสดงหลักการประมาณค่าด้วยวิธีซัคเซสซีฟขนาด 3 บิท ดังรูปที่ 7 เมื่อสัญญาณ Start เริ่มขึ้นกระบวนการแปลงสัญญาณก็เริ่มต้นเช่นกัน ค่าน้ำหนักของบิทแต่ละบิทของ SAR จะถูกวงจรเปรียบเทียบสัญญาณนำค่าไปเปรียบบเทียบกับ Vin ดังรูปที่ 7 โดยเริ่มเปรียบเทียบจากบิท MSB (D_2) ไปจนถึงบิท LSB (D_0) หาก Vin มีค่ามากกว่าสัญญาณเอาท์พุทของ SAR จะถูกปรับให้เป็นลอจิก 0



รูปที่ 8 แผนผังแสดงหลักการประมาณค่าด้วยวิธีซักเซสซีฟขนาด 3 บิทเมื่อ Vin = 6.5 V

2.2.3 ช่วงเวลาของการแปลงสัญญาณ (Conversion Time)

จากหลักการแปลงสัญญาณโดยใช้หลักการประมาณค่าด้วยวิธีซัคเซสซีฟ ดังรูปที่7 จะเห็นว่า SAR ต้องใช้สัญญาณนาฬิกาหนึ่งลูกเพื่อใช้ในการเปรียบเทียบในแต่ละบิทดังนั้นช่วงเวลาที่ใช้สำหรับการแปลงสัญญาณอนาลอกให้เป็นสัญญาณดิจิตอลในหนึ่งรอบนั้นจึงมีค่าขึ้นอยู่กับคาบเวลาของสัญญาณนาฬิกาที่ใช้รวมทั้งจำนวนบิทของวงจรอีกด้วย ซึ่งสามารถเขียนความสัมพันธ์ได้เป็น

$$T_C = T \times (n+1)$$

โดยที่ T_C คือ ช่วงเวลาที่ใช้สำหรับการแปลงสัญญาณ T คือ คาบเวลาของสัญญาณนาฬิกาที่ใช้ในวงจร และ n คือ จำนวนบิทของวงจร ADC

2.3 ADC แบบเฟลช (Flash ADC)

2.3.1 หลักการทำงาน (Circuit Operation)

ADC ที่มีช่วงเวลาที่ใช้ในการแปลงสัญญาณรวดเร็วที่สุดในบรรดา ADC แบบต่างๆ ก็คือ ADC แบบเฟลช หรือแบบขนาน แรงดันอ้างอิง V_{REF} และส่วนของตัวต้านทานแบ่งแรงดันทำให้ค่ารีโซลูชันของวงจรเท่ากับ 1 V/LSB ขาอินพุทบวกของวงจรเปรียบเทียบแรงดันทุกตัวทำหน้าที่เป็นขารับแรงดันอนาลอกอินพุท Vin ของวงจรเพื่อนำไปเปรียบเทียบกับค่าของแรงดันที่ได้จากการแบ่งแรงดันในแต่ละโหนด จากนั้นแรงดันเอาท์พุทของวงจรเปรียบเทียบแรงดันจะป้อนเข้าวงจรเข้ารหัส (8-to-3 line encoder)เพื่อนำสัญญาณเอาท์พุทออกไปใช้งานเพียง 3 เส้น (เนื่องจากเป็นวงจร ADC ที่มีขนาด 3 บิท) โดยที่สัญญาณดิจิตอลเอาท์พุท 3 บิทที่ได้มีผลเป็นสัดส่วนโดยตรงกับค่าของแรงดันอนาลอกอินพุท Vin ที่ป้อนให้กับวงจรนั่นเอง

2.3.2 ช่วงเวลาของการแปลงสัญญาณ (Conversion Time)

ช่วงเวลาที่ใช้ในการแปลงสัญญาณของ ADC แบบเฟลชนั้นขึ้นอยู่กับสมรรถนะผลตอบสนองทางเวลาของวงจงเปรียบเทียบแรงดัน วงจรลอจิก และวงจรเกทต่างๆที่ใช้ในวงจร หรือ กล่าวได้ว่าหาก ADC แบบเฟลชที่มีความเร็วสูงมากขึ้นเท่าใด ราคาก็แพงและค่ารีโซลูชันของวงจรก็มากขึ้นตามไปด้วยเช่นกัน จากรายละเอียดของวงจรเปรียบเทียบแรงดันจำนวน 7 ตัว ดังนั้นจึงสามารถสรุปได้ว่าจำนวนวงจรเปรียบเทียบแรงดันที่ต้องใช้ในวงจร ADC แบบเฟลชขนาด n บิท มีค่าเท่ากับ

$$\text{Number of comparators} = 2^n - 1$$

2.3.3 ผลตอบสนองทางความถี่ของวงจร ADC (Frequency Response of ADC)

สำหรับวงจร ADC แล้วความถี่ปฏิบัติงานสูงสุดของแรงดันอนาลอกอินพุท Vin รูปคลื่นซายน์ที่จะแปลงให้สัญญาณดิจิตอลเอาท์พุทที่มีความแม่นยำ □ □ LSB มีค่าเท่ากับ

$$f_{max} = 1/(2 \times Tc \times 2^n)$$

โดยที่ T_c คือ ช่วงเวลาที่ใช้ในการแปลงสัญญาณหนึ่งรอบของ ADC และ n คือ จำนวนบิทของวงจร ADC

3. ไมโครคอนโทรลเลอร์ MCS 51

ไมโครคอนโทรลเลอร์ตระกูล MCS 51 ที่ใช้ในการทำโครงงานนี้จะอ้างอิงถึงไมโครคอนโทรลเลอร์ตระกูล MCS 51 ซึ่งมีหน่วยความจำแบบแฟลช ของ Atmel Corporation เบอร์ที่ขึ้นต้นด้วย AT89

3.1 จัดขาของไมโครคอนโทรลเลอร์ 8051

V_{cc}: สำหรับแหล่งจ่ายไฟฟ้า (+5v.)

V_{ss}: สำหรับต่อกราวด์

P0 : เป็นขาพอร์ต 0 ของ 8051 ที่มีขนาด 8 บิตชนิดสองทิศทาง ซึ่งแต่ละบิตสามารถกำหนดให้เป็นได้ทั้งอินพุตและ เอาต์พุตสำหรับใช้งานทั่วไปหากต้องการให้เป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล 1 ไปยังบิตนั้น โดยแต่ละบิตเมื่อเป็นเอาต์พุตจะสามารถต่อพ่วงกับอุปกรณ์ TTL แบบ LS ได้ 8 ตัว และยังเป็นขาให้สัญญาณ Multiplex ระหว่างสัญญาณข้อมูลกับสัญญาณ Address 8 บิตแรก ในกรณีที่ใช้หน่วยความจำภายนอก

P1: เป็นขาพอร์ต 1 ของ 8051 ขนาด 8 บิต ชนิดสองทิศทางแบบ Quasi bi-directional ซึ่งแต่ละบิตสามารถกำหนดให้เป็นได้ทั้งอินพุต และเอาต์พุตสำหรับใช้งานทั่วไปหากต้องการให้เป็นอินพุต สามารถทำได้โดยการเขียนข้อมูล 1 ไปยังบิตนั้น และสามารถต่อพ่วงกับอุปกรณ์ LS TTL ได้ 4 ตัว

P2: เป็นขาพอร์ต 2 ของ 8051 ขนาด 8 บิต ชนิดสองทิศทางแบบ Quasi bi-directional เช่นเดียวกับพอร์ต 1 นอกจากนี้พอร์ต2นี้ยังทำหน้าที่ให้สัญญาณ Address 8 บิตบน ในกรณีที่ใช้หน่วยความจำภายนอก ในกรณีอ้าง Address หน่วยความจำขนาด 16 บิต ดังนั้นขณะที่ใช้หน่วยความจำภายนอก จะต้องไม่มีการเขียนข้อมูลใด ๆ ไปที่พอร์ต 2 จะทำให้เกิดความผิดพลาดการทำงานได้

P3: เป็นขาพอร์ต 3 ของ 8051 ขนาด 8 บิต ชนิดสองทิศทางแบบ Quasi bi-directional เช่นเดียวกับขาพอร์ต 1 และพอร์ต 2 แต่พอร์ต 3 นี้จะมีหน้าที่พิเศษดังตารางที่ 1

ตารางที่ 1 หน้าที่พิเศษของขาต่าง ๆ ของ PORT 3

ขาพอร์ต	หน้าที่พิเศษ
P3.0	R x D (สำหรับรับข้อมูลแบบอนุกรม)
P3.1	T x D (สำหรับส่งข้อมูลแบบอนุกรม)
P3.2	INT0 (ขาอินเตอร์รัพท์ภายนอก 0)
P3.3	INT1 (ขาอินเตอร์รัพท์ภายนอก 1)
P3.4	T0 (ขาอินพุตของ Timer 0)
P3.5	T1 (ขาอินพุตของ Timer 1)
P3.6	R(สำหรับสัญญาณเขียนหน่วยความจำข้อมูลภายนอก)
P3.7	RD (สำหรับสัญญาณอ่านหน่วยความจำข้อมูลภายนอก)

ดังนั้น เมื่อมีการใช้สัญญาณดังกล่าว จึงไม่ควรเขียนข้อมูลไปที่พอร์ต 3 จะทำให้การทำงานของ 8051 ผิดพลาดได้

RST: เป็นขาสำหรับรีเซตการทำงานของ 8051 โดยการให้ลอจิกหนึ่งเป็นเวลาอย่างน้อย 2 ช่วง Machine Cycle

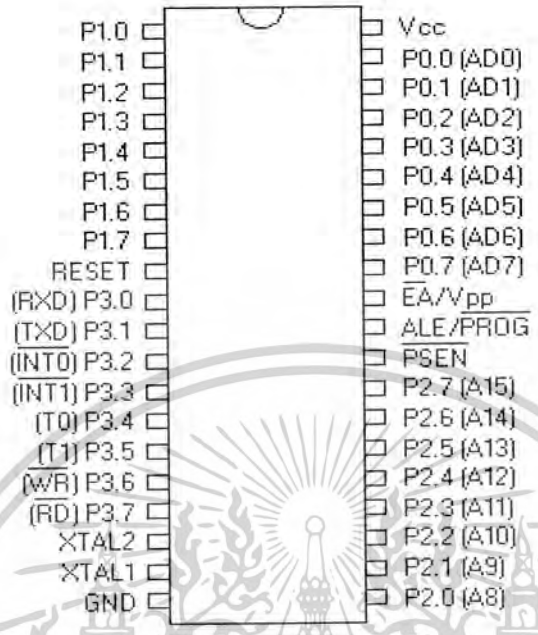
ALE: เป็นขาที่ใช้ในการควบคุมการแลตช์ของขา พอร์ต 0 เมื่อมีการใช้งานหน่วยความจำภายนอก

PSEN: เป็นขาสัญญาณเพื่อร้องขอติดต่อหน่วยความจำโปรแกรมภายนอกเมื่อไมโครคอนโทรลเลอร์ต้องการอ่านหน่วยความจำโปรแกรมภายนอก

EA:เป็นขาใช้สำหรับเลือกการติดต่อหน่วยความจำโปรแกรมภายนอกหรือภายในไมโครคอนโทรลเลอร์ โดยที่ให้ลอจิก 0 จะอ่านหน่วยความจำโปรแกรมภายนอก และลอจิก 1 จะอ่านหน่วยความจำโปรแกรมภายใน

XTAL1: ขาเข้าของวงจรกำเนิดความถี่อ้างอิงภายในของ 8051

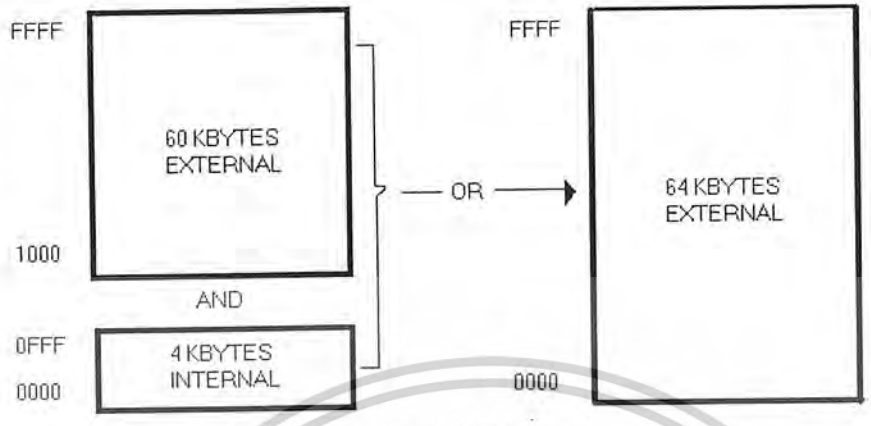
XTAL2: ขาออกของวงจรกำเนิดความถี่อ้างอิงภายในของ 8051



รูปที่ 9 การจัดขาของ 8051

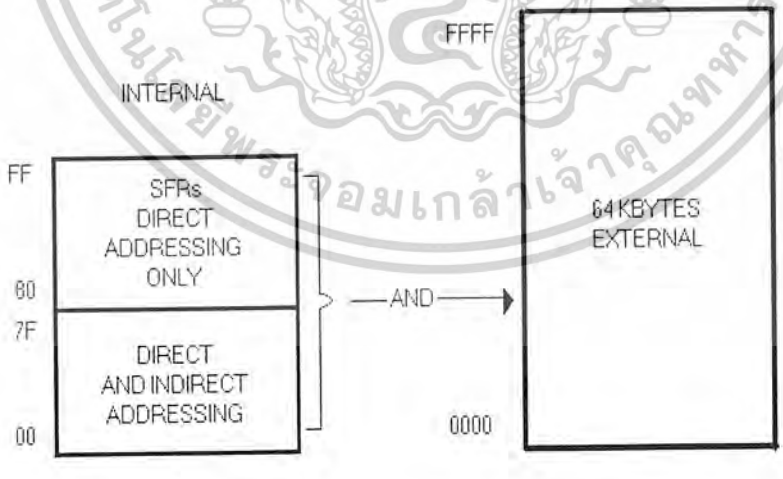
3.2 <u>โครงสร้างหน่วยความจำของ 8051</u>

ดังที่กล่าวมาแล้ว 8051 จะแบ่งหน่วยความจำออกเป็นสองส่วน ได้แก่ หน่วยความจำสำหรับโปรแกรมและหน่วยความจำสำหรับเก็บข้อมูล โดยมีขนาดของแต่ละส่วนเท่ากับ 64 กิโลไบต์ ในส่วนของหน่วยความจำโปรแกรมจะเป็นส่วนหน่วยความจำสำหรับอ่านอย่างเดียว โดยที่ 8051 จะใช้สัญญาณ PSEN ในการอ่านเท่านั้น แต่หน่วยความจำข้อมูลของ 8051 จะสามารถอ่านและเขียนได้โดยใช้สัญญาณ RD และ WR ตามลำดับ แต่อย่างไรก็ตาม ผู้ใช้สามารถรวมหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลเข้าด้วยกันได้ โดยนำสัญญาณ RD และ PSEN มาต่อเข้าวงจรแอนเกท สำหรับสร้างสัญญาณในการอ่านหน่วยความจำ นอกจากนี้หน่วยความจำโปรแกรมยังแบ่งออกเป็นภายนอกและภายในของ 8051 ดังแสดงใน รูปที่ 9 รูปที่ 10 โดยรูปที่ 9 แสดงหน่วยความจำโปรแกรมในกรณีที่เลือกใช้หน่วยความจำภายนอกและภายใน ในด้านซ้ายมือเป็นส่วนหนึ่งของหน่วยความจำโปรแกรมภายในที่มีขนาด 4 กิโลไบต์ของ 8051 ส่วนที่เหลือจะเป็นหน่วยความจำภายนอก ส่วนด้านขวามือแสดงหน่วยความจำโปรแกรมเมื่อเลือกให้ติดต่อหน่วยความจำภายนอกทั้งหมด



รูปที่ 10 แสดงหน่วยความจำโปรแกรมของ 8051

สำหรับหน่วยความจำข้อมูลของ 8051 สามารถแบ่งออกเป็นภายนอกและภายใน โดยหน่วยความจำภายนอกแสดงไว้ด้านขวามือของรูปที่ 10 ซึ่งมีขนาด 64 กิโลไบต์ ส่วนหน่วยความจำข้อมูลภายในแสดงไว้ด้านซ้ายของรูปที่ 10 โดยหน่วยความจำภายในของ 8051 แบ่งออกเป็นสองส่วน ได้แก่ ส่วนของหน่วยความจำข้อมูลที่สามารถอ้างอิงแบบ Direct และ Indirect ซึ่งมีขนาด 128 ไบต์ กับหน่วยความจำที่อ้างอิงได้เฉพาะแบบ Direct หรือในส่วนนี้จะเรียกอีกแบบหนึ่งว่า SFR (Special Function Register) โดยจะแบ่งกล่าวได้ดังนี้



รูปที่ 11 แสดงหน่วยความจำข้อมูลของ 8051

ในส่วนของหน่วยความจำข้อมูลภายในที่อ้างอิงแบบ direct และ Indirect นั้นจะสามารถแบ่งออกได้ 3 ส่วน ดังแสดงในรูปที่ 11 โดยมีรายละเอียดดังนี้

ส่วนที่ 1 เรียกว่า Register Banks 0-3 ซึ่งอยู่ที่ตำแหน่งความจำข้อมูลภายใน ตั้งแต่ 00H ถึง 1FH จำนวน 32 ไบต์ โดยจะแบ่งออกเป็นชุด ชุดละ 8 ไบต์จำนวน 4 ชุด ซึ่งแต่ละชุดจะมีชื่อเรียกเป็น R0 ถึง R7 จะเป็น Register ที่ใช้งาน โดยเมื่อ 8051 ถูกรีเซต Register Bank 0 จะถูกเลือกใช้

ส่วนที่ 2 เรียกว่า Bit Addressable Area ซึ่งมีขนาด 16 ไบต์ที่ตำแหน่งหน่วยความจำข้อมูล 20H ถึง 2FH ในส่วนนี้สามารถที่จะอ้างอิงข้อมูลได้เป็นระดับบิตถึง 128 บิต โดยการอ้างอิงตำแหน่งโดยตรงในลักษณะบิต ตั้งแต่ตำแหน่ง 00H ถึง 7FH

รูปที่ 12 แสดงหน่วยความจำข้อมูลภายใน

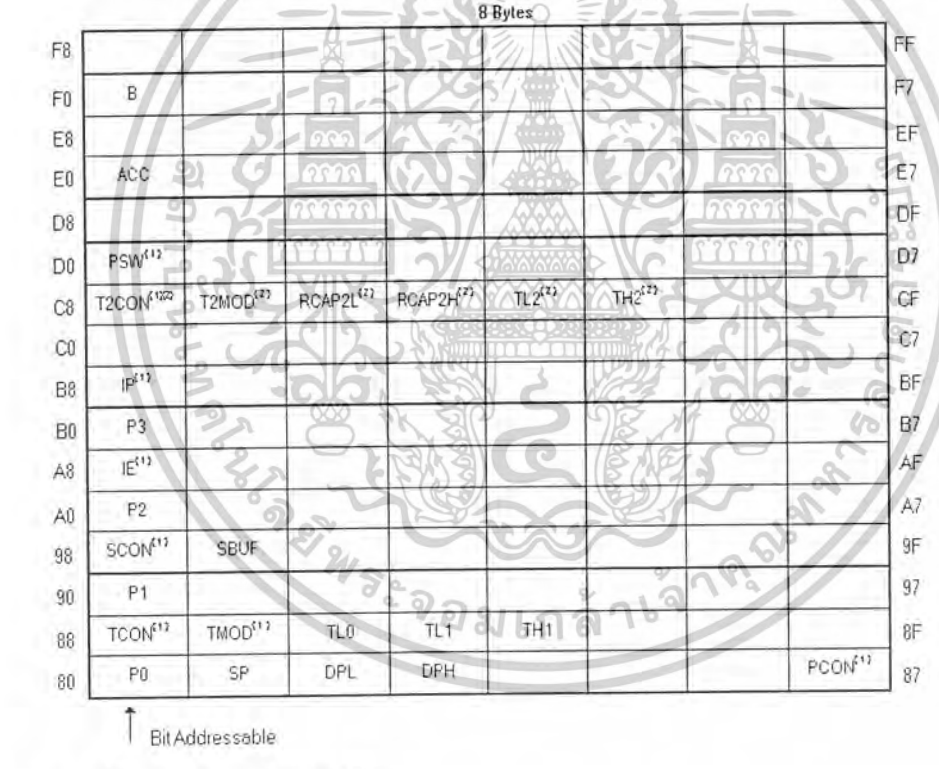
ส่วนที่ 3 เรียกว่า Scratch Pad Area จะอยู่ที่ตำแหน่งตั้งแต่ 30H ถึง 7FH ซึ่งเป็นบริเวณหน่วยความจำข้อมูลภายในเอนกประสงค์ที่ผู้ใช้สามารถใช้ได้โดยตรง นอกจากนี้ยังสามารถใช้หน่วยความจำข้อมูลบริเวณนี้สำหรับการเก็บข้อมูลแบบ Stack ได้ด้วยในส่วนของหน่วยความจำข้อมูลภายในที่ใช้อ้างอิงแบบ Direct เพียงอย่างเดียวหรือที่เรียกว่า SFR ซึ่งเป็นส่วนสำหรับเก็บหรือกำหนดการทำงานภายในของ 8051 ดังแสดงในรูปที่ 12

ในส่วนของบริเวณนี้จะมีขนาด 128 ไบต์แต่ในการใช้งานนั้นใช้ได้เฉพาะตำแหน่งซึ่งแสดงไว้ในรูปที่ 12 เท่านั้น หากผู้ใช้อ้างตำแหน่งนอกเหนือจากนี้นั้นจะได้ข้อมูลที่คาดเดาไม่ได้ โดยแต่ละตำแหน่งจะมีหน้าที่ดังนี้

ACC: เป็น Accumulator ซึ่งเป็นรีจีสเตอร์สำหรับการประมวลผลทางคณิตศาสตร์และลอจิก โดยผู้ใช้สามารถอ้างอิงได้ในรูปแบบของไบต์หรือระดับบิตได้

B: เป็นรีจีสเตอร์พิเศษสำหรับใช้กับคำสั่งในการคูณหรือหาร นอกจากนี้ยังใช้เป็นรีจีสเตอร์สำหรับเก็บพักข้อมูลได้

PSW: เป็นรีจีสเตอร์ Program Status Word หรือแฟลกจะแสดงสถานะการทำงานของ 8051 สำหรับการตรวจสอบซึ่งจะอธิบายรายละเอียดในภายหลัง



รูปที่ 13 แสดงรายละเอียดของ Special Function Register

SP: เป็นรีจีสเตอร์สำหรับชี้หน่วยความจำข้อมูลภายในสำหรับการเก็บแบบ Stack

DPTR : เป็นรีจีสเตอร์ขนาด 16 บิต โดยแบ่งเป็น 8 บิตบนและ 8 บิตล่าง ให้สำหรับชี้ตำแหน่งของหน่วยความจำข้อมูลภายนอกหรือสำหรับการอ่านตารางข้อมูลของหน่วยความจำโปรแกรม

P0: เป็นรีจีสเตอร์สำหรับพอร์ต 0 ของ 8051

P1: เป็นรีจีสเตอร์สำหรับพอร์ต 1 ของ 8051

P2: เป็นรีจีสเตอร์สำหรับพอร์ต 2 ของ 8051

P3: เป็นรีจีสเตอร์สำหรับพอร์ต 3 ของ 8051

IP: เป็นรีจีสเตอร์สำหรับกำหนดลำดับความสำคัญของการอินเตอร์รัพท์ของ 8051

IE: เป็นรีจีสเตอร์สำหรับกำหนดการรับหรือไม่รับการอินเตอร์รัพท์ของ 8051

TMOD: เป็นรีจีสเตอร์สำหรับควบคุมหน้าที่ของ Timer/Counter ของ 8051

TCON: เป็นรีจีสเตอร์สำหรับควบคุมการทำงานของ Timer/Counter ของ 8051

T2CON: เป็นรีจีสเตอร์สำหรับควบคุมการทำงานของ Timer/Counter 2 ของ 8052

TH0: เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 0 8บิตบน

TL0: เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 0 8บิตล่าง

TH1: เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 1 8บิตบน

TL1: เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 1 8บิตล่าง

TH2: เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 2 8บิตบนของ 8052

TL2: เป็นรีจีสเตอร์สำหรับเก็บข้อมูลของ Timer/Counter 2 8บิตล่างของ 8052

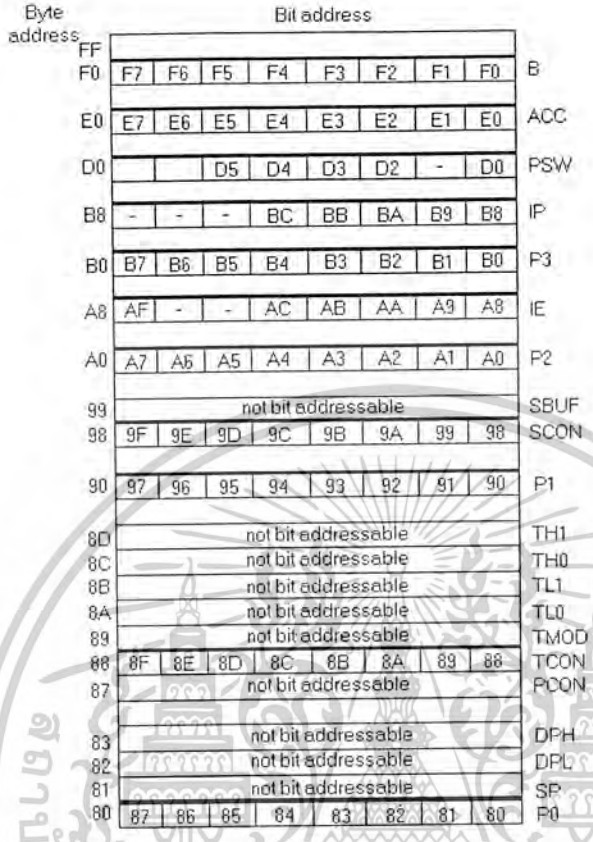
RCAP2H: เป็น Capture Register ของ Timer/Counter 2 8บิตบนของ 8052

SCON: เป็นรีจีสเตอร์สำหรับควบคุมการรับส่งข้อมูลแบบอนุกรมของ 8051

SBUF: เป็นรีจีสเตอร์สำหรับเก็บพักข้อมูลที่ได้จากการรับส่งข้อมูลแบบอนุกรมของ MCS - 8051

PCON: เป็นรีจีสเตอร์สำหรับควบคุมการทำงานของ MCS-8051 ด้านเกี่ยวกับการใช้กำลังไฟฟ้า

ในส่วนของรีจีสเตอร์ SFR นี้สามารถที่จะอ้างอิงในระดับบิตได้โดยตำแหน่งการอ้างอิงระดับบิตซึ่งได้แสดงไว้ในตาราง รูปที่ 13



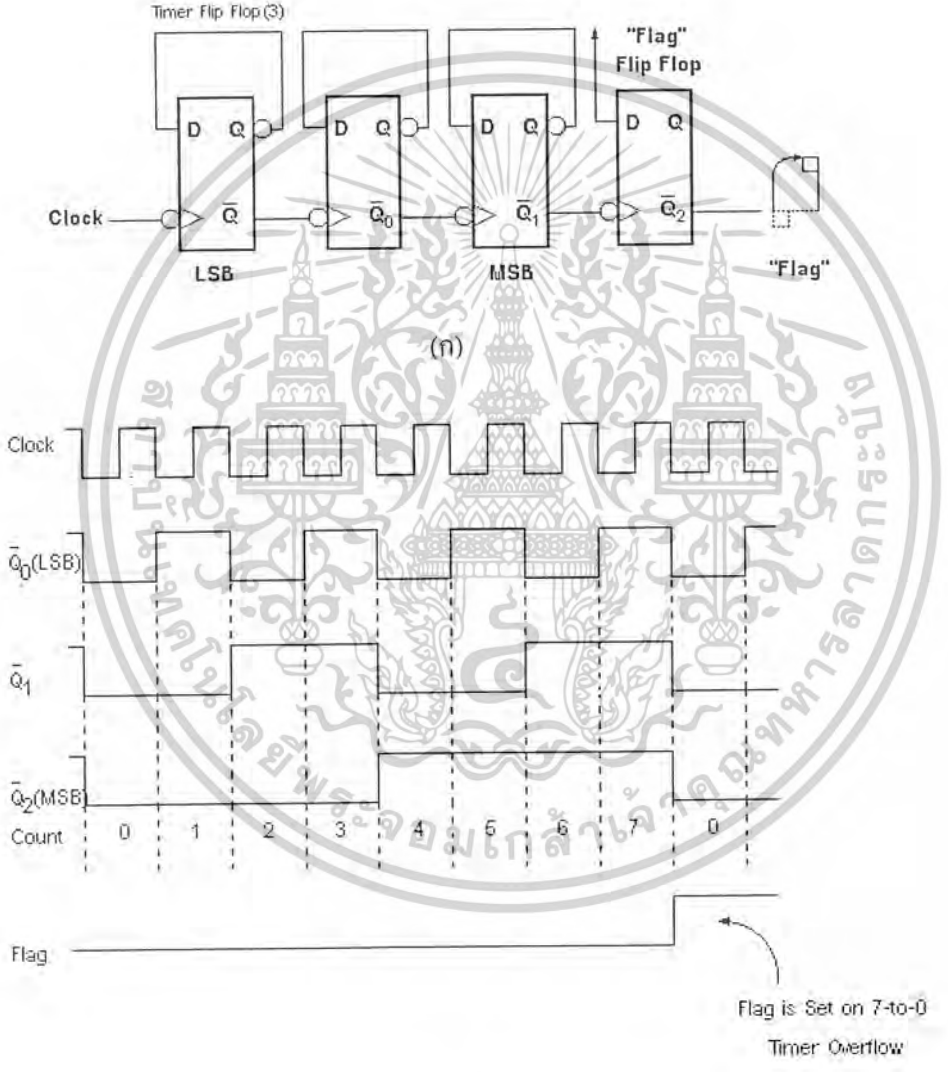
รูปที่ 14 แสดงตำแหน่งการอ้างอิงระดับบิตของรีจิสเตอร์ SFR

3.3 TIMER

ตัว Timer อาจพิจารณาได้ง่าย ๆ ว่าเป็นตัวฟลิปฟลอปมาต่อเรียงกันโดยมี Clock เป็นอินพุต สำหรับเอาต์พุทที่ออกมาจากฟลิปฟลอปแต่ละตัวจะถูกหารด้วย 2 พิจารณาการต่อฟลิปฟลอปตามรูปที่ 14 ถ้าใส่ Clock เข้าไปในฟลิปฟลอปตัวแรก ความถี่ของ Clock ที่ออกจากเอาต์พุทตัวแรกจะถูกหารด้วย 2 และเอาต์พุทนี้จะต่อกับฟลิปฟลอปตัวที่สอง และสัญญาณที่ออกมาจะถูกหารด้วย 2 อีก ดังนั้น ถ้ามีฟลิปฟลอปต่ออยู่ n Stages จะหารสัญญาณนาฬิกาได้ 2^n ถ้าให้เอาต์พุท Stage สุดท้ายของ Timer เป็น Overflow Flip-Flop หรือ Flag และจะให้เอาต์พุทออกมาเมื่อการนับเป็น Overflow เช่น ถ้าเป็นตัวนับแบบ 16 บิต (มีฟลิปฟลอปต่ออยู่ 16 ตัว) วงจรจะนับตั้งแต่ 0000H ถึง FFFFH เมื่อฟลิปฟลอปเปลี่ยนจาก FFFFH เป็น 0000H จะให้บิต Overflow ออกมา

พิจารณารูป 14 (ก) เป็น 3-bit Timer โดยฟลิปฟลอปแต่ละตัวจะนำขา Q มาต่อกับ D ซึ่งอาจเรียกว่าเป็นการใช้ฟลิปฟลอปแบบ Divide-by-two Mode โดยความถี่ของสัญญาณที่ได้จากฟลิปฟลอปแต่ละตัวจะมีค่าหารสองจากสัญญาณนาฬิกาที่เข้ามา เมื่อนับไปถึงค่า 111 (หรือ $Q_2 = 1$, $Q_1 = 1$, $Q_0 = 1$) และเปลี่ยนกลับมาเป็น 000 จะให้บิต Flag ออกมา ดังแสดงในรูปที่ 14(ข)

ใน MCS – 51 จะมีตัวจับเวลาอยู่ภายในชิพ ถ้าเป็นเบอร์ 8051 หรือ 8031 จะมี 2 ตัว คือ Timer 0 และ Timer 1 แต่ถ้าเป็นเบอร์ 8052 จะมีเพิ่มอีกหนึ่งตัวคือ Timer 2 รีจีสเตอร์ต่าง ๆ ที่เกี่ยวข้องกับการใช้ Timer แสดงได้ดังตารางที่ 2 ซึ่งจะเห็นว่ามีรีจีสเตอร์บางตัวสามารถเข้าถึงข้อมูลระดับบิตได้ด้วย นอกจากนี้ตัว Timer สามารถใช้เป็นตัวนับ (Counter) ได้อีกด้วย โดยการโปรแกรมในรีจีสเตอร์ TMOD



(ข)

รูปที่ 15 รีจีสเตอร์ที่ใช้เป็น Timer

ตารางที่ 2 รีจีสเตอร์ที่ใช้เป็น Timer

รีจีสเตอร์	หน้าที่	ตำแหน่ง	สามารถอ้างอิงตำแหน่งบิต
TCON	Control	88H	Yes
TMOD	Mode	89H	No
TL_0	Timer 0 Low-byte	8AH	No
TL_1	Timer 1 Low-byte	8BH	No
TH_0	Timer 0 High-byte	8CH	No
TH_1	Timer 1 High-byte	8DH	No
T_2CON	Timer 2 Control	C8H	Yes
$RCAP_2L$	Timer 2 Low-byte Capture	CAH	No
$RCAP_2H$	Timer 2 High-byte Capture	CBH	No
TL_2	Timer 2 Low-byte	CCH	No
TH_2	Timer 2 High-byte	CDH	No

มีในเบอร์ 8032 / 8052

3.3.1 Timer Mode Register (TMOD)

ตัวรีจีสเตอร์ TMOD เป็นรีจีสเตอร์ควบคุม Timer จะแบ่งออกเป็น 2 กลุ่ม กลุ่มละ 4 บิต โดย 4 บิตบนจะเป็นการควบคุม Timer 1 ส่วน 4 บิตล่างจะเป็นการควบคุม Timer 0 ความหมายของแต่ละบิตดูในตารางที่ 2 ซึ่งตัวรีจีสเตอร์นี้เป็นตัวเลือกการทำงานว่าจะให้ตัว Time/Counter ทำงานในโหมดใด และเป็น Timer หรือ Counter รีจีสเตอร์ TCON ไม่สามารถจะโปรแกรมเข้าไปในระดับบิตได้ (Not Bit-Addressable) ซึ่งการใช้งานมักจะโปรแกรมเข้าไปครั้งเดียวในตำแหน่งเริ่มต้นของโปรแกรม

ตารางที่ 3 รีจีสเตอร์ TMOD (Timer Mode)

บิต	ชื่อ	Timer	ความหมาย
7	GATE	1	Gate bit ถ้าบิตนี้เซตวงจรจะทำงาน เมื่อ INT1 เป็น High
A	C/T	1	เป็นบิตเลือก Counter / Timer 1 = ใช้เป็น Counter 0 = ใช้เป็น Timer
5	M_1	1	Mode bit 1 (ดูตาราง 5-3)
4	M_0	1	Mode bit 0 (ดูตาราง 5-3)
3	GATE	0	บิต Gate ของ Timer 0
2	C/T	0	บิตเลือก Counter / Timer ของ Timer 0
1	M_1	0	Timer 0 M_1 bit
0	M_0	0	Timer 0 M_0 bit

ตารางที่ 4 การใช้ Timer โหมดต่าง ๆ

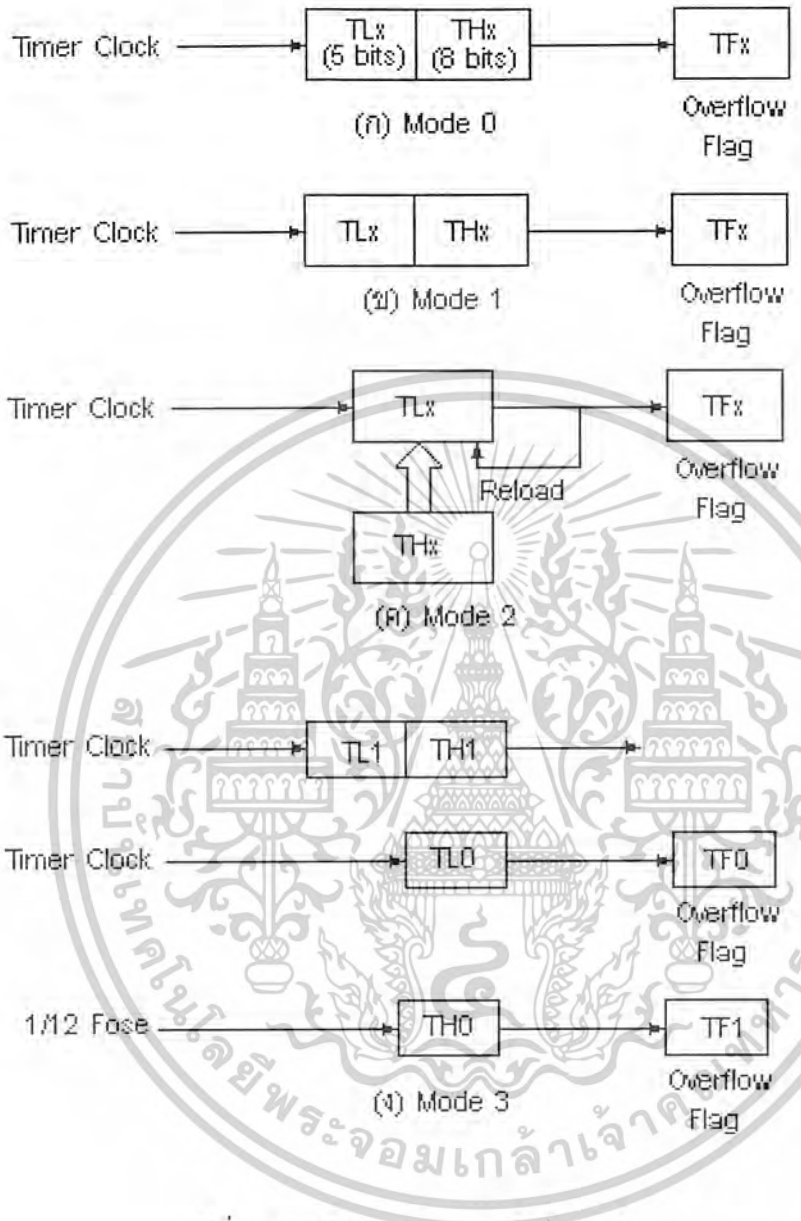
M_1	M_0	Mode	ความหมาย
0	0	0	ใช้เป็น Timer แบบ 13-bit (8048 Mode)
0	1	1	ใช้เป็น Timer แบบ 16-bit
1	0	2	ใช้เป็น Timer แบบ 8-bit Auto-reload Mode
1	1	3	Split Timer Mode แยก Timer 0 ออกเป็น Timer 8 บิตสองตัวคือ TL_0 และ TH_0 โดยไม่ใช้ Timer 1

3.3.2 Timer Control Register (TCON)

รีจีสเตอร์ TCON เป็นรีจีสเตอร์ที่บอกสถานะและควบคุมบิต Timer 0 และ Timer 1 ซึ่งดูได้จากตารางที่ 5 รีจีสเตอร์นี้สามารถเข้าถึงข้อมูลระดับบิตได้

ตารางที่ 5 แสดงความหมายแต่ละบิตของรีจีสเตอร์ TCON (Timer Control)

บิต	ชื่อ	ตำแหน่งบิต	ความหมาย
TCON.7	TF_1	8FH	บิตแฟลกแสดงการโอเวอร์โฟลว์ของ Timer 1 จะ Set โดย Hardware และ Clear โดย Software
TCON.6	TR_1	8EH	บิตควบคุมการปิด-เปิด Timer 1 Set และ Clear โดย Software
TCON.5	TF_0	8DH	แฟลกแสดงการโอเวอร์โฟลว์ของ Timer 0
TCON.4	TR_0	8CH	บิตควบคุมการปิด-เปิด Timer 0
TCON.3	IE_1	8BH	บิตแฟลกแสดงการอินเทอร์รัพท์จาก INT_1 จะ Set โดย Hardware และสามารถ Clear ได้ด้วย Software
TCON.2	IT_1	8AH	บิตเลือกชนิดของสัญญาณอินเทอร์รัพท์จากอินเทอร์รัพท์ภายนอก INT_1 สามารถ Set และ Clear ได้ด้วย Software
TCON.1	IE_0	89H	บิตแฟลกแสดงการอินเทอร์รัพท์จาก INT_0
TCON.0	IT_0	88H	บิตเลือกชนิดของสัญญาณอินเทอร์รัพท์จากอินเทอร์รัพท์ภายนอก INT_0



รูปที่ 16 การทำงานของ Timer ในโหมดต่าง

3.3.3 Timer Mode And Overflow Flag

เมื่อใช้ Timer 0 และ Timer 1 จะต้องใช้รีจีสเตอร์คู่ TLx และ THx โดยค่า x จะเป็นตัวบอกว่าเป็น Timer 0 หรือ Timer 1 การใช้ Timer สามารถใช้งานได้หลายโหมด ดังแสดงในรูปที่ 15 ซึ่งเราสามารถเซตค่าโหมดการทำงานได้ โดยการโปรแกรมในรีจีสเตอร์ TMOD

1. 13-Bit Timer Mode (Mode 0)

การทำงานในโหมด 0 นี้จะเป็นการใช้ Timer แบบ 13 บิต ดังแสดงในรูป 15(ก) ซึ่งจะใช้ 5 บิตล่างของ TLx โดยไม่สนใจ 3 บิตที่เหลือ และ 8 บิต ของ THx การทำงานในโหมดนี้ เมื่อบิตของ TLx นับไปจนเป็น "1" ทุกบิตจะส่ง Clock 1 ลูกให้ หนึ่งลูกให้ THx นับต่อและเมื่อนับเป็น "1" ทุกบิต และเปลี่ยนกลับเป็น "0" จะเกิด Overflow Flag เกิดขึ้น

2. 16-Bit Timer Mode (Mode 1)

การทำงานในโหมดนี้จะเหมือนกับการทำงานในโหมด 0 แต่เป็น Timer แบบ 16 บิต ซึ่งการนับจะเริ่มตั้งแต่ 0000H,0001H,0002H ไปเรื่อย ๆ และจะเกิด Overlflow ขึ้น เมื่อมีการเปลี่ยนจาก FFFFH เป็น 0000H ดังรูปที่ 15(ข) ซึ่งเป็นการเซต Overflow Flag และค่านี้จะเกิดขึ้นในบิต TFx ของรีจีสเตอร์ TCON ซึ่งสามารถอ่านและเขียนด้วยโปรแกรม

การใช้ตัว Timer นี้ค่าของบิตสูงสุด (MSB) คือค่าบิต 7 ของ THx ส่วนบิตต่ำสุด (LSB) คือ บิต 0 ของ TLx บิต LSB จะเป็น Toggles เมื่อมีสัญญาณอินพุตเข้ามา ถูกหารด้วย 2 ดังนั้นจะพบว่าบิต MSB จะ Toggles ด้วยค่าความถี่ของสัญญาณอินพุตหารด้วย 65,536 (2^{16}) และค่า Timer รีจีสเตอร์นี้ (TLx/THx) สามารถอ่านและเขียนได้ด้วยการโปรแกรม ดังนั้นสามารถนำไปประยุกต์ใช้งานได้ตามต้องการ

3. 8-Bit Auto – Reload Mode (Mode 2)

การทำงานในโหมด 2 เรียกอีกอย่างหนึ่งว่า 8-bit Auto – reload Mode โดยใช้ Timer ไบต์ต่ำ (TLx) เป็น Timer แบบ 8 บิต เมื่อไบต์ต่ำเกิด Overflows หรือเกิดการเปลี่ยนแปลงจาก FFH เป็น 00H จะมีการโหลดค่าที่เก็บไว้ในไบต์สูง (THx) ไปเก็บไว้ในไบต์ต่ำ (TLx) ซึ่งจะเป็นค่าเริ่มต้นของการนับครั้งต่อไป นิยมใช้สร้างเป็นฐานเวลาที่สามารถโปรแกรมได้ การทำงานในโหมดนี้แสดงดังรูปที่ 15(ค)

4. Split Timer Mode (Mode 3)

การทำงานในโหมด 3 นี้ ตัว Timer 1 จะไม่ทำงาน ตัว Timer 0 จะแยกเป็น 2 ตัว ตัวละ 8 บิต คือ TL_0 และ TH_0 เมื่อ Timer เกิด Overflows จะมีการเซตบิต TF_0 และ TF_1 ดังแสดงในรูปที่ 15(ง)

การทำงานในโหมด 3 นี้ Timer 1 จะไม่ถูกใช้งานแต่เราสามารถสวิตช์ให้ Timer 1 ไปทำงานในโหมดอื่นได้ แต่การทำงานของ Timer 1 จะไม่มีการอินเทอร์รัพท์เกิดขึ้น เพราะบิต TF_1 ถูกใช้ในการนับของ TH_0 ในการทำงานของโหมด 3 ไปแล้ว เราอาจมองว่าถ้าให้ Timer ทำงานใน3

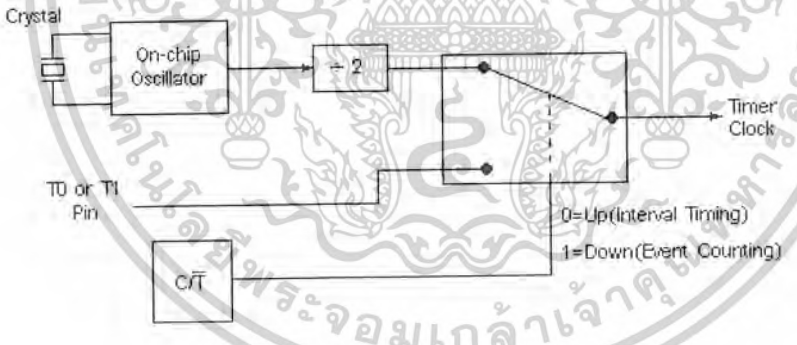
โหมด 3 ทำให้เรามี Timer เพิ่มขึ้น คือ TH_0 และ TL_0 ใน Timer 0 โหมด 3 และโปรแกรมให้ Timer 1 ไปทำงานในโหมดอื่น ๆ

5. Clocking Source

ในรูปที่ 15 ไม่ได้แสดงว่า Timer Clock นำมาจากที่ใดซึ่งการใช้ Timer นี้สามารถใช้ได้ 2 หน้าที่ คือเป็นตัวจับเวลา (Timer) และเป็นตัวนับ (Counter) ซึ่งสามารถโปรแกรมได้โดยการเซต หรือรีเซตบิต C / T ในรีจีสเตอร์ TMOD

6. การใช้เป็นตัวจับเวลา (Timer)

ถ้าบิต C / T ใน TMOD เป็นลอจิก "0" จะเป็นการเลือกให้ Timer นำ Clock มาจากวงจร Oscillator ในชิพ ซึ่งสัญญาณนาฬิกาจะเข้ามาทุก ๆ Machine Cycle หรืออาจกล่าวได้ว่าค่าใน THx และ TLx จะมีค่าเพิ่มขึ้นด้วยอัตราการนับแต่ละครั้งใช้เวลาเท่ากับ 1/12 ของความถี่ของสัญญาณนาฬิกาที่ใช้บนชิพ ดังแสดงในรูปที่ 16 ถ้า MCS – 51 ใช้สัญญาณนาฬิกา 12 MHz การนับจะมีความถี่เท่ากับ 1 MHz



รูปที่ 17 ความถี่ของสัญญาณนาฬิกาที่เข้าขา Timer

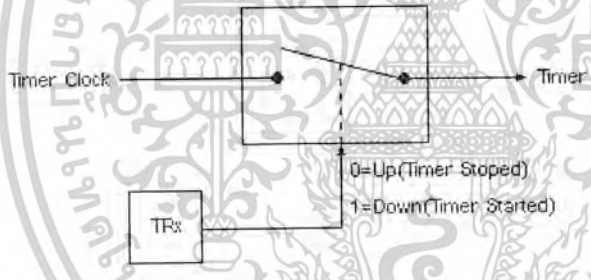
7. การใช้เป็นตัวนับ (Counter)

ถ้าบิต C / T เป็น "1" ตัว Timer จะนำ Clock มาจากภายนอกโดยใช้ขา $P_{3.4}$ หรือ T_0 เป็นขา Input Clock ให้กับ Timer 0 และใช้ขา $P_{3.5}$ หรือ T_1 เป็น Input Clock ให้กับ Timer 1 ดังรูปที่ 16 หรืออาจมองว่า ถ้าจะให้นับอะไรสัญญาณที่จะนับให้ต่อกับขา T_0 และ T_1 ในการใช้เป็น Counter สัญญาณที่เข้ามามีการเปลี่ยนแปลงจาก "1" เป็น "0" จะทำให้วงจรนับ TLx มีค่าเพิ่มขึ้น 1 ภายใน MCS – 51 นี้จะตรวจสอบขาอินพุต T_0 และ T_1 ในช่วงเวลาเฟส 2 ของ State 5 (S_5P_2) ถ้าพบว่ามีค่า

เป็น "1" ต่อมาในอีกหนึ่ง Machine Cycle ที่เฟส 2 ของ State 5 (S_5P_2) ลอจิกอินพุตเปลี่ยนเป็น "0" จะทำให้ค่าใน Timer เพิ่มขึ้น 1 ดังนั้น จะเห็นได้ว่าการนับ 1 ครั้งจะต้องใช้เวลา 2 Machine Cycles ดังนั้นความถี่สูงสุดที่จะให้ Timer ทำงานเป็น Counter นับได้ จะมีค่ามากที่สุด 500 kHz ถ้า MCS – 51 ทำงานที่ความถี่สัญญาณนาฬิกา 12 MHz

3.3.4 การเริ่ม , หยุด และการควบคุม Timer

ในรูปที่ 15 จะแสดงลักษณะของ Timer Registers ซึ่งจะเห็นว่าประกอบด้วย TLx และ THx และเมื่อเกิด Overflow จะเกิดเอาต์พุทที่บิต TFx สำหรับสัญญาณนาฬิกาที่จะเข้าไปใน Time จะมาจาก 2 ส่วนดังแสดงในรูปที่ 16 ต่อไปจะกล่าวถึงว่าเราจะควบคุมให้เริ่ม , หยุดตัว Timer ได้อย่างไร วิธีเริ่มและหยุดตัว Timers สามารถควบคุมได้ที่บิต TRx ในรีจีสเตอร์ TCON โดยปกติแล้ว TRx จะเคลียหลังจากที่ระบบถูกรีเซต ซึ่งจะเป็นการให้ Timer ไม่นับและ TRx นี้จะเซตได้จากชุดคำสั่งหรือการโปรแกรม พิจารณารูปที่ 17



รูปที่ 18 การใช้บิตควบคุม TR

ตัวบิต TRx จะเป็นส่วนที่สามารถเข้าถึงข้อมูลในระดับบิตได้ (Bit Addressable)ในรีจีสเตอร์ TCON ถ้าจะให้ TIMER 0 เริ่มทำงานจะเขียนคำสั่งได้ดังนี้

SETB TR0

ถ้าจะหยุดทำงานเขียนคำสั่งได้ดังนี้

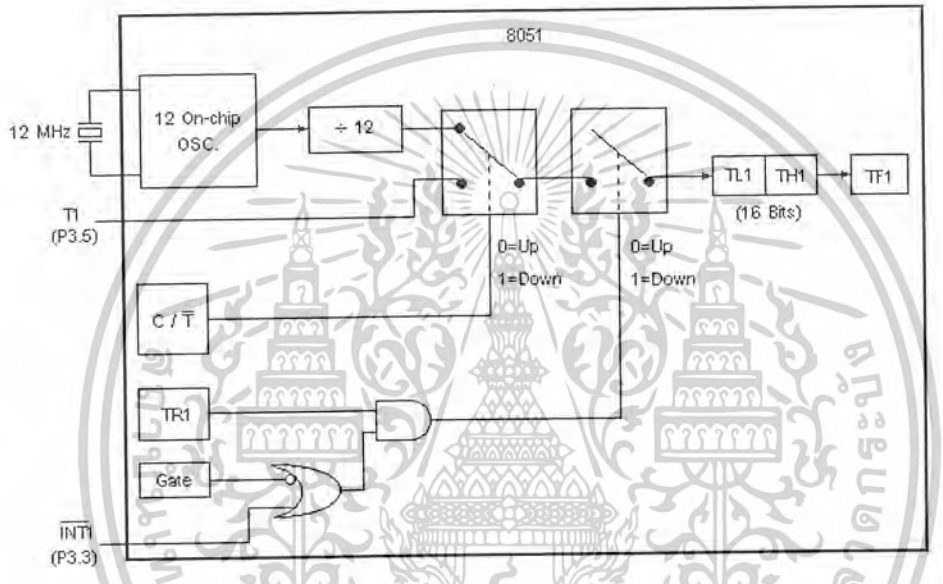
CLR TR0

ในการเขียนโปรแกรมภาษาแอสแซมบลี สามารถใช้สัญลักษณ์ TR0 ในคำสั่ง SETB TR0 เลยได้ เพราะตัวแอสแซมบลีจะตีความ TR0 เป็น Bit Address ตำแหน่ง 8 CH

วิธีควบคุม Timer สามารถควบคุมได้ที่บิต GATE ใน TMOD และขาอินเตอร์รัพท์จากภายนอก INTx ถ้า INT0 เป็นลอจิก "0" และโปรแกรมให้ Timer 0 ทำงานในโหมด 2 เมื่อ TL0/TH0 = 0000H, GATE = 1 และ TR0 = 1 เมื่อ INT0 ขึ้นเป็นลอจิก "1" ตัว Timer จะ "Gate On" และจะให้

สัญญาณนาฬิกาความถี่ 1 MHz เมื่อ INT0 ลงเป็น "0" ตัว Timer " Gate Off " สัญญาณที่ได้จะมีความกว้างของสัญญาณนาฬิกา 1 μS ส่งเข้าไปใน TL_0/TH_0

รูปที่ 18 จะเป็นระบบที่สมบูรณ์ของ Timer 1 เมื่อทำงานในโหมด 1 ซึ่งเป็น 16-bit Timer โดยใช้รีจีสเตอร์ TL_1 / TH_1 และ Overflow Flag TF_1 ในรูปจะเห็นถึงการควบคุมแหล่งกำเนิด Clock การเริ่มทำงาน และการหยุดทำงาน



รูปที่ 19 ระบบทั้งหมดของ Timer 1

3.3.5 Intializing And Accessing Timer Register

การใช้งาน Timer เริ่มแรกจะต้องโปรแกรมเพื่อเลือกโหมดการทำงานของ Timer ก่อนเมื่อเริ่มใช้งานก็โปรแกรมให้ เริ่มทำงาน, หยุดทำงาน, อ่าน และ เคลียร์ค่า Flag Bits อ่านค่า Timer Registers ตามลำดับ เพื่อนำไปประยุกต์การใช้งานต่อไป

TMOD คือ รีจีสเตอร์ที่ต้องโปรแกรม โดยเซตโหมดการทำงานก่อน ตัวอย่างเช่น ถ้าให้ Timer 1 เป็น 16-bits Timer (โหมด 1) นับสัญญาณนาฬิกาบนชิพ สามารถเขียนคำสั่งได้ดังนี้

```
MOV TMOD, #00010000B
```

ผลที่ได้จากคำสั่งข้างบนคือ เซตบิต $M_1 = 0$ และ $M_0 = 1$ ซึ่งเป็นการเลือกโหมด 1 และให้ C / T= 0 และ GATE = 0 ซึ่งเป็นการใช้สัญญาณนาฬิกาจากภายในหรือใช้เป็น Timer และตัว Timer นี้จะยังไม่ทำงาน ถ้าบิตควบคุม TR_1 ยังไม่ได้เซต

ถ้าให้ Timer นี้นับขึ้นโดยใช้รีจีสเตอร์ TL1 / TH1 และจะเซตบิต Overflow Flag เมื่อรีจีสเตอร์เปลี่ยนจาก FFFFH เป็น 0000H โดยให้นับเวลาไป 100 μS หรือให้ TL1 / TH1 นับสัญญาณนาฬิกาได้ 100 ลูก ดังนั้นค่าเริ่มต้นของ TL1 / TH1 จะไม่เริ่มที่ 0000H จะต้องเริ่มที่ FFFFH ลบด้วย 100 ลูก หรือ FF9CH เพื่อให้นับไปถึง FFFFH และเปลี่ยนเป็น 0000H ได้สัญญาณนาฬิกา 100 ลูกพอดี สามารถเขียนคำสั่งได้ดังนี้

MOV TL1 , # 9CH

MOV TH1 , # 0FFH

ถ้าให้ Timer เริ่มทำงานก็ให้บิตควบคุมดังนี้

SETB TR1

จากนั้นบิต Overflow Flag จะส่งออกมาหลังเวลาผ่านไป 100 μS ซึ่งเราสามารถเขียนโปรแกรมเป็นโปรแกรมวนลูป 100 μS ได้ โดยตรวจสอบบิต TF1 ว่าถูกเซตหรือไม่ ถ้าไม่เซตก็ให้วนลูปต่อไปดังนี้

CLR TR1

CLR TF1

<u>การใช้แบบ Readind a Timer "On the Fly"</u>

การใช้งานแบบประยุกต์บางงานจะต้องอ่านค่าจาก Timer Register เนื่องจากตัว Timer Register มีขนาด 2 ไบต์ ถ้าหากไบต์ต่ำเกิด Overflow จะทดเข้าไบต์สูง ถ้าหากเขียนโปรแกรมให้อ่านค่าจากไบต์ต่ำก่อน แล้วจึงอ่านไบต์สูงข้อมูลที่ได้อาจเกิดข้อผิดพลาดได้เนื่องจากไบต์ต่ำมีการเปลี่ยนแปลงเร็วกว่าไบต์สูง การอ่านข้อมูลควรอ่านจากไบต์สูงก่อน แล้วจึงกลับมาอ่านไบต์ต่ำ จากนั้นอ่านข้อมูลไบต์สูงอีกครั้ง ถ้าค่าไบต์สูงที่อ่านได้ไม่มีการเปลี่ยนแปลงให้ใช้ค่านั้นได้เลย แต่ถ้ามีการเปลี่ยนแปลงให้อ่านอีกครั้ง ถ้าต้องการอ่านข้อมูลจาก TL1 / TH1 เข้าในรีจีสเตอร์ R6 / R7 อาจเขียนโปรแกรมได้ดังนี้

```
AGAIN :  MOV A, TH1
         MOV R6, TL1
         CJNE A, TH1, AGAIN
         MOV R7, A
```

3.3.6 <u>Short Intervals And Long Intervals</u>

ถ้า MCS – 51 ทำงานที่ความถี่สัญญาณนาฬิกา 12 MHz ถ้าให้ Timers ใช้วงจร Oscillator บนชิพ สัญญาณนาฬิกาจะถูกหารด้วย 12 และ Timer จะทำงานด้วยความถี่ 1 MHz ถ้าต้องการใช้

โปรแกรมสร้างสัญญาณนาฬิกาออกมาอาจทำได้โดยง่าย ซึ่งพิจารณาจากการทำงานชุดคำสั่งต่าง ๆ ของ MCS – 51 ใน 1 Machine Cycle จะใช้เวลา 1μS ในตารางที่ 6 จะแสดงความกว้างของสัญญาณที่สร้างขึ้นจาก MCS – 51 ที่ทำงานด้วย Crystal ความถี่ 12 MHz

ตารางที่ 6 ค่าสูงสุดของการใช้ Timer โหมดต่าง ๆ

Maximum Interval in Microseconps	**Technique**
≈10	Software Tuing
256	8 – bit Timer with Auto-reload
65536	8 – bit Timer
No Limit	16 – bit Timer Plus Software Loops

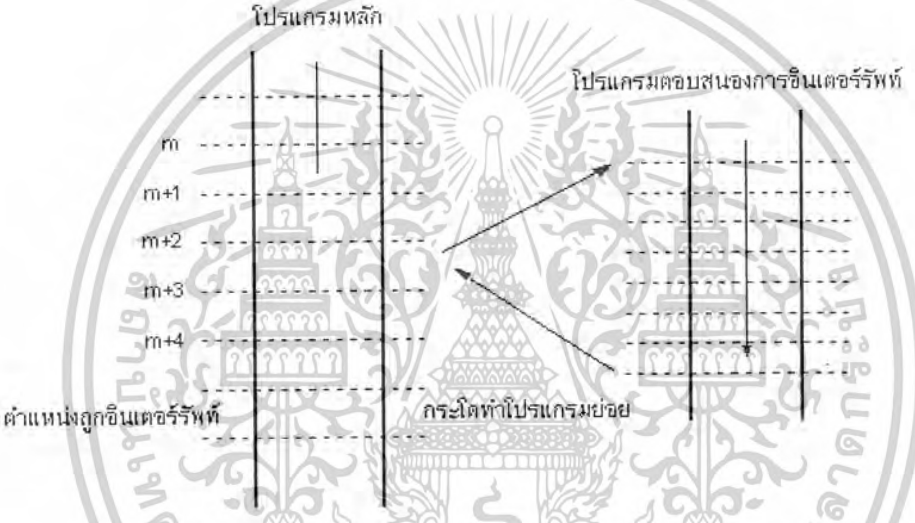
3.4 การอินเตอร์รัพท์

การทำงานของระบบคอมพิวเตอร์โดยทั่วไปมักมีอุปกรณ์ภายนอกต่อร่วมอยู่ถ้าคอมพิวเตอร์ต้องการทำงานกับอุปกรณ์ภายนอกจะต้องคอยตรวจสอบอุปกรณ์เหล่านั้นเสมอ ตัวอย่างเช่น ถ้าหากให้คอมพิวเตอร์พอร์ทหนึ่งต่ออยู่กับหลอด LED 7 ส่วน อีกพอร์ทหนึ่งต่อกับสวิตช์ ถ้าระบบของเราทำงานเป็นนาฬิกาเดินไปให้คอยตรวจสอบสวิตช์ด้วยว่ามีการกดหรือยัง การทำงานแบบนี้เรียกว่า Polling Method คือตัวไมโครโปรเซสเซอร์จะต้องคอยตรวจสอบอุปกรณ์อินพุตตลอดเวลาว่ามีข้อมูลเข้ามาหรือยัง การทำงานแบบนี้ ถ้ามีอุปกรณ์ภายนอกหลายตัวระบบต้องตรวจสอบอุปกรณ์ภายนอกหลายตัว ทำให้เสียเวลาในการทำงานหลักไป การทำงานอีกแบบหนึ่งจะให้ CPU ทำงานหลัก ถ้ามีการกดสวิตช์เมื่อไรให้นาฬิกาหยุดเดินทันที การทำงานในลักษณะนี้ CPU ไม่ต้องเสียเวลาในการตรวจอุปกรณ์ภายนอก ถ้าอุปกรณ์ภายนอกต้องการติดต่อกับ CPU อุปกรณ์ภายนอกจะส่งสัญญาณมาบอก CPUเอง ระบบนี้เรียกว่า การอินเทอร์รัพท์ (Interrupt)

3.4.1 ขบวนการเกิดอินเตอร์รัพท์

ถ้าหากคอมพิวเตอร์กำลังทำงานโปรแกรมหลักอยู่เมื่อมีการอินเทอร์รัพท์เข้ามาคอมพิวเตอร์จะละทิ้งโปรแกรมหลัก แต่ไปทำงานโปรแกรมตอบสนองการอินเตอร์รัพท์ (Interrupt Service Routine) เมื่อทำโปรแกรมตอบสนองอินเตอร์รัพท์เสร็จ คอมพิวเตอร์จะกลับมาทำโปรแกรมเดิม พิจารณารูปที่ 19

ถ้า CPU กำลังทำงานโปรแกรมหลักอยู่ เช่นกำลังทำคำสั่งในตำแหน่งของหน่วยความจำที่ m, m+1, m+2 ไปเรื่อย ๆ โดย PC จะชี้ที่ตำแหน่งที่จะอ่านค่าคำสั่งถัดมา เมื่อโปรแกรมทำงานมาถึงตำแหน่งที่ m+3 แล้วเกิดการอินเตอร์รัพท์ขึ้น (ขณะนั้น PC อยู่ที่ m+4) โปรแกรมจะต้องทำงานโปรแกรมตอบสนองการอินเตอร์รัพท์ โดยย้าย PC ไปที่ตำแหน่งที่เก็บโปรแกรมตอบสนองการอินเตอร์รัพท์ จากนั้นจะเก็บค่า PC เดิมลงในหน่วยความจำสแตค เมื่อคอมพิวเตอร์ทำงานโปรแกรมตอบสนองการอินเตอร์รัพท์เสร็จสิ้นลง จะคืนค่าใน สแตค (m+4) ให้กับ PC ทำโปรแกรมหลักต่อไป



รูปที่ 20 ขั้นตอนการทำงานของโปรแกรมเมื่อถูกอินเตอร์รัพท์

3.4.2 สัญญาณอินเตอร์รัพท์

แหล่งกำเนิดสัญญาณอินเตอร์รัพท์ที่ใช้กับ MCS – 51 มีสองชนิดคือ อินเตอร์รัพท์ภายในและภายนอก โดยอินเทอร์รัพท์ภายในจะเกิดขึ้นจากภายในตัว MCS – 51 เอง ได้แก่สัญญาณจากไทเมอร์แฟลก 0 (TF0) ไทเมอร์แฟลก 1 (TF1) และพอร์ทอนุกรม สำหรับอินเตอร์รัพท์ภายนอกเกิดจากสัญญาณที่กระตุ้นเข้ามาทางขา INT0 และ INT1 เมื่อมีสัญญาณอินเตอร์รัพท์จากแหล่งต่างๆ เข้ามา เราสามารถโปรแกรมได้ว่าจะให้ MCS – 51 ยอมให้มีการอินเตอร์รัพท์ได้หรือไม่ โดยการโปรแกรมไปที่ รีจีสเตอร์ IE (Interrupt Enable) และถ้ามีสัญญาณอินเตอร์รัพท์มากจากแหล่งต่าง ๆ หลายแหล่งพร้อมกันเราสามารถจัดลำดับได้ว่า จะให้อินเตอร์รัพท์ใดเกิดก่อน โดยการโปรแกรมไปที่ อินเตอร์รัพท์ไพออริตี้ IP (Interrupt Priority) รีจีสเตอร์ทั้งสองตัวมีรายละเอียดดังนี้

1. Interrupt Enables

เป็นรีจีสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิตได้ ใช้สำหรับกำหนดค่าว่าถ้าเกิดการอินเตอร์-รัพท์จากแหล่งต่าง ๆ จะทำอินเตอร์รัพท์เหล่านั้นหรือไม่ โดยรายละเอียดของบิตต่าง ๆ มีดัง-ตารางที่ 7

บิต 7 → บิต 0

EA	X	ET2	ES	ET1	EX1	ET0	EX0

ตารางที่ 7 บิตต่าง ๆ ของรีจีสเตอร์ IE

บิต	ชื่อบิต	ตำแหน่งบิต	รายละเอียด
IE.7	EA	AFH	ถ้าเซตยอมให้มีการอินเตอร์รัพท์
IE.6	-	AEH	ไม่ใช้งาน
IE.5	ET2	ADH	Enable อินเตอร์รัพท์จาก Timer 2 (ใช้กับ 8052)
IE.4	ES	ACH	Enable อินเตอร์รัพท์จากพอร์ทอนุกรม
IE.3	ET1	ABH	Enable อินเตอร์รัพท์จาก Timer 1
IE.2	EX1	AAH	Enable อินเตอร์รัพท์จาก INT1
IE.1	ET0	A9H	Enable อินเตอร์รัพท์จาก Timer 0
IE.0	EX0	A8H	Enable อินเตอร์รัพท์จาก INT0

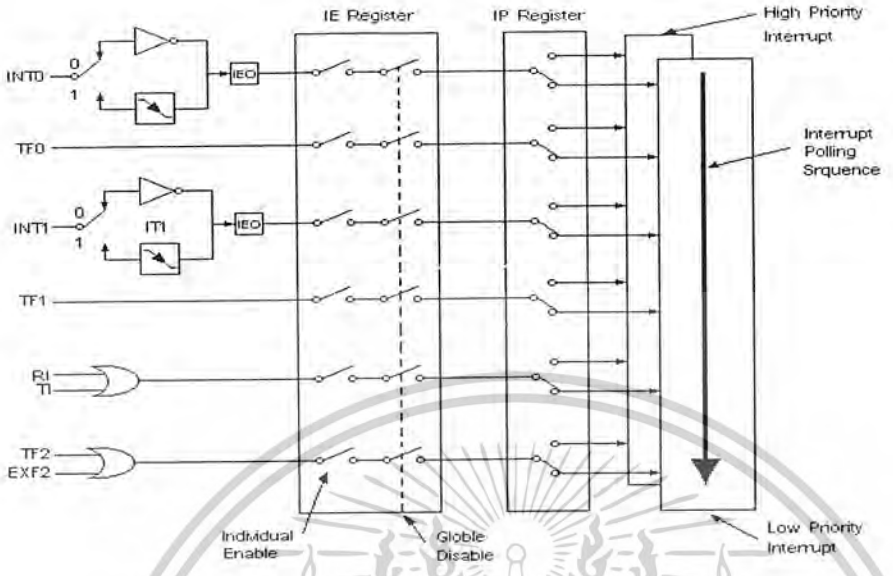
2. Interrupt Priority

เป็นรีจีสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิตได้ใช้ในการจัดลำดับความสำคัญของการอิน-เทอร์รัพท์ซึ่งสามารถจัดได้สองลำดับ ถ้าเป็น "1" หมายความว่ามีความสำคัญสูงสุด ถ้าเป็น "0" หมายความว่ามีลำดับความสำคัญต่ำสุด ความหมายของบิตต่าง ๆ แสดงได้ดังตารางที่ 8 ถ้าหากกำหนดให้มีความสำคัญเป็น "1" เหมือนกันหมด MCS – 51 จะจัดลำดับความสำคัญใหม่ดังนี้

ลำดับ	อินเตอร์รัพท์
1 (สูงสุด)	IE0
2	TF0
3	IE1
4	TF1
5 (ต่ำสุด)	Serial Port

ตารางที่ 8 บิตและหน้าที่ต่าง ๆ ของรีจีสเตอร์ IP

บิต	ชื่อบิต	ตำแหน่งบิต	รายละเอียด
IP.7	-	-	ไม่ใช้งาน
IP.6	-	-	ไม่ใช้งาน
IP.5	PT2	0BDH	ใช้กับ Timer 2 (8052)
IP.4	PS	0BCH	ใช้กับพอร์ทอนุกรม
IP.3	PT1	0BBH	ใช้กับ Timer 1
IP.2	PX1	0BAH	ใช้กับอินเทอร์รัพท์จาก INT1
IP.1	PT0	0B9H	ใช้กับ Timer 0
IP.0	PX0	0B8H	ใช้กับอินเทอร์รัพท์จาก INT0



รูปที่ 21 รีจิสเตอร์ต่าง ๆ ที่เกี่ยวข้องกับการอินเตอร์รัพท์

จากรูปที่ 20 แสดงการอินเตอร์รัพท์จากแหล่งต่าง ๆ ที่มีผลกับ MCS – 51 ถ้าเป็นเบอร์ 8051 8031 จะถูกอินเตอร์รัพท์ได้ 5 แหล่ง ถ้าเป็นเบอร์ 8052,8032 จะถูกอินเตอร์รัพท์ได้ 6 แหล่ง โดยเพิ่มอินเตอร์รัพท์จาก Timer 2 ในรูปที่ 2.17 จะแสดงให้เห็นว่า ถ้า MCS – 51 จะถูกอินเตอร์รัพท์ได้จะต้องเซตค่า Global Enable ในรีจีสเตอร์ IE นอกจากนี้ยังกำหนดได้ว่าจะให้อินเตอร์รัพท์ใดเกิดได้ โดยการเซตค่า Interrupt Enable ของอินเตอร์รัพท์จากแหล่งต่าง ๆ ในรีจีสเตอร์ IE จากรูปยังแสดงให้เห็นอีกว่าเมื่อมีการอินเทอร์รัพท์เข้ามาจะมีผลต่อแฟลกใด เช่นถ้า INT0 เป็น “1” บิต IE0 จะเป็น “1” หมายความว่าถูกอินเตอร์รัพท์ โดยแฟลกต่าง ๆ ที่มีผลจากการถูกอินเตอร์รัพท์แสดงได้ดังตารางที่ 9

ตารางที่ 9 แฟลกที่จะทำงานเมื่อถูกอินเตอร์รัพท์

อินเตอร์รัพท์	แฟลก	ประกอบอยู่ในรีจีสเตอร์
External 0	IE0	TCON.1
External 1	IE1	TCON.3
Timer 1	TF1	TCON.7
Timer 0	TF0	TCON.5
Serial port	T1	SCON.1
Serial port	RI	SCON.0
Timer 2	TF2	T2CON.7 (8052)
Timer 2	EXF2	T2CON.6 (8052)

จากตารางจะเห็นว่า ถ้ามีการอินเตอร์รัพท์จากภายนอกเข้ามา ตัวที่จะอินเตอร์รัพท์ MCS – 51 คือ บิตแฟลก IE0 ซึ่งอยู่ในรีจีสเตอร์ TCON ถ้ามีการสื่อสารแบบอนุกรม เมื่อข้อมูลถูกส่งไปหมดแล้วจะอินเตอร์รัพท์ MCS – 51 ทางบิตแฟลก TI ถ้ารับข้อมูลหมดแล้วจะอินเตอร์รัพท์ MCS – 51 ทางบิตแฟลก RI ซึ่งอยู่ในรีจีสเตอร์ SCON และถ้าใช้ Timer 0 ในการนับเมื่อเกิด Overflow สามารถอินเตอร์รัพท์ MCS – 51 ได้ทางบิต TF0

3.4.3 การทำงานของระบบหลังถูกอินเตอร์รัพท์

เมื่อ MCS – 51 ถูกอินเทอร์รัพท์จะต้องกระโดดไปทำโปรแกรมตอบสนองการอินเทอร์รัพท์โดยตำแหน่งที่จะกระโดดไปเรียกว่า อินเตอร์รัพท์เวกเตอร์ (Interrupt Vectors) เมื่อทำโปรแกรมตอบสนองการอินเตอร์รัพท์เรียบร้อยแล้ว MCS – 51 จะกระโดดมาทำงานยังตำแหน่งเดิม โดยก่อนที่จะกระโดดไปทำโปรแกรมตอบสนองการอินเทอร์รัพท์จะต้องเก็บค่าตำแหน่งเดิมไว้ โดยเก็บค่า PC ลงหน่วยความจำสแตคซึ่งอยู่ที่หน่วยความจำที่ถูกชี้โดยรีจีสเตอร์ SP เมื่อทำโปรแกรมตอบสนองการอินเทอร์รัพท์เสร็จแล้วจะคืนค่าในหน่วยความจำสแตคให้ PC ตามเดิม ค่าอินเทอร์รัพท์เวกเตอร์ของ MCS – 51 แสดงได้ดังตารางที่ 10

ตารางที่ 10 อินเตอร์รัพท์เวกเตอร์ของอินเตอร์รัพท์ต่าง ๆ

อินเทอร์รัพท์	อินเทอร์รัพท์เวกเตอร์
System Reset	0000H
External 0	0003H
Timer 0	000BH
External 1	0013H
Timer 1	001BH
Serial Port	0023H
Timer 2	002BH

จากตารางจะเห็นว่าถ้าระบบถูกอินเตอร์รัพท์จากภายนอกทาง INT0 ตัว MCS – 51 จะกระโดดไปทำงานที่ตำแหน่ง 0003H ถ้าระบบถูกอินเตอร์รัพท์จาก Timer 0 จะกระโดดไปทำงานตำแหน่ง 000BH

3.4.4 การออกแบบโปรแกรมอินเตอร์รัพท์

ในการเขียนโปรแกรมหลัก (Main Program) จะต้องกำหนดค่าว่าจะให้ MCS – 51 ถูกอินเตอร์รัพท์ด้วยอะไร และจะให้ MCS – 51 ถูกอินเตอร์รัพท์ได้หรอไม่ โดยการโปรแกรมค่าต่าง ๆ ใน IE รีจีสเตอร์ ถ้ามีการอินเทอร์รัพท์จากสองแหล่งขึ้นไปควรมีการจัดลำดับความสำคัญในรีจีสเตอร์ IP ดังนั้นในโปรแกรมหลักจะต้องมีการโปรแกรมต่อไปนี้

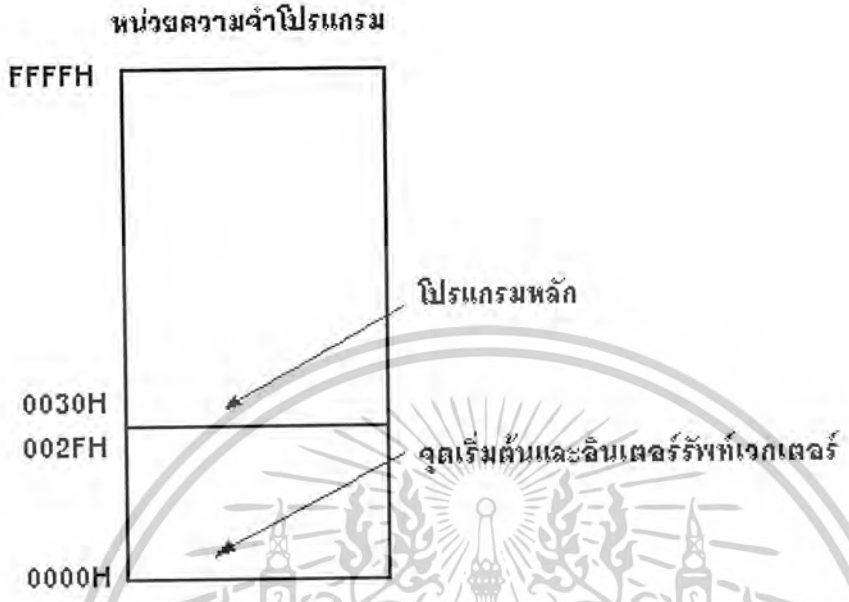
1. โปรแกรมค่าในรีจีสเตอร์ IE

โปรแกรมค่าในรีจีสเตอร์ IP

สำหรับโปรแกรมตอบสนองการอินเทอร์รัพท์ถือว่าเป็นโปรแกรมย่อยโปรแกรมหนึ่ง แต่จะต้องจบโปรแกรมย่อยด้วยค่า RETI (Return From Interrupt)

จากตารางอินเตอร์รัพท์เวกเตอร์ จะเห็นว่าถ้ากด Reset หรือให้ระบบเริ่มทำงาน โปรแกรมจะเริ่มทำงานที่ตำแหน่ง 0000H และจะเห็นว่า ตำแหน่งที่เก็บโปรแกรมหลักมีโอกาส อย่างมากที่จะทับกับหน่วยความจำโปรแกรมที่เก็บค่าอินเทอร์รัพท์เวกเตอร์ที่ตำแหน่ง 0003H ถ้าโปรแกรมยาวมากอาจจะไปทับตำแหน่ง 000BH ได้ซึ่งเป็นตำแหน่งของอินเทอร์รัพท์เวกเตอร์ของ Timer 0 ดังนั้นในการเขียนโปรแกรมหลัก ภายใน 3 ตำแหน่งแรก คือ 0000H,0001H,0002H จะต้องกระโดดไปที่อื่นก่อนเพื่อให้ข้ามอินเทอร์รัพท์เวกเตอร์ไป ซึ่งอาจเขียนโปรแกรมได้ดังนี้

```
ORG    0000H          ; เริ่มต้นโปรแกรม
LJMP   MAIN           ; กระโดดไปโปรแกรมหลัก
.....                 ; เพื่อหนีอินเทอร์รัพท์เวกเตอร์
.....
ORG    0030H          ; ตำแหน่งเริ่มต้นของโปรแกรม
MAIN : .....          ; เริ่มต้นโปรแกรมหลัก
.....
```



รูปที่ 22 การจัดตำแหน่งโปรแกรมในหน่วยความจำ

จากตัวอย่างโปรแกรมจะเห็นว่า เมื่อเริ่มต้นโปรแกรมหรือระบบถูกรีเซต ระบบจะทำงานตำแหน่งแรก คือคำสั่งกระโดดไปโปรแกรมหลัก ซึ่งอยู่ต่อจากโปรแกรมตอบสนองการอินเตอร์-รัพท์ที่อยู่ที่ตำแหน่ง 0030H

<u>โปรแกรมตอบสนองการอินเตอร์รัพท์แบบสั้น</u>

จากตารางอินเตอร์รัพท์เวกเตอร์ จะเห็นว่าที่เก็บโปรแกรมอินเตอร์รัพท์แต่ละแหล่งจะห่างกัน 8 ไบต์ ดังนั้นถ้ามีการอินเตอร์รัพท์จากแหล่งต่าง ๆ หลาย ๆ แหล่งและโปรแกรมตอบสนองการอินเตอร์รัพท์บางโปรแกรมมีขนาดยาวเกิน 8 ไบต์ จะทำให้โปรแกรมไปทับกับตำแหน่งของโปรแกรมตอบสนองการอินเทอร์รัพท์ของอินเทอร์รัพท์ถัดไป แต่ถ้าโปรแกรมตอบสนองการอินเทอร์รัพท์ไม่ยาวมากเกินไปเราสามารถเขียนไปในตำแหน่งนั้นได้เลยดังโปรแกรมต่อไปนี้

```
ORG     0000H
LJMP    MAIN        ; กระโดดไปโปรแกรมหลัก
ORG     000BH       ; ตำแหน่งเริ่มต้นของอินเตอร์รัพท์  Timer 0
TOISR :.....
```

```
        ......
        RETI                ; กลับโปรแกรมหลัก
MAIN : ......               ; โปรแกรมหลัก
        ......
```

จากตัวอย่างโปรแกรมจะใช้อินเตอร์รัพท์จาก Timer 0 เมื่อระบบเริ่มทำงานจะทำตำแหน่ง 0000H โดยกระโดดไปโปรแกรมหลักซึ่งอยู่ที่ตำแหน่งต่อจากโปรแกรมตอบสนองการอินเทอร์รัพท์เมื่อมีการอินเทอร์รัพท์ Timer 0 ระบบจะทำโปรแกรมตำแหน่งที่ 000BH ซึ่งเป็นอินเตอร์รัพท์-เวกเตอร์ของ Timer 0 โดยโปรแกรมตอบสนองการอินเตอร์รัพท์จะจบด้วยคำสั่ง RETI เพื่อกลับสู่โปรแกรมหลักต่อไป

<u>โปรแกรมตอบสนองการอินเตอร์รัพท์ขนาดใหญ่</u>

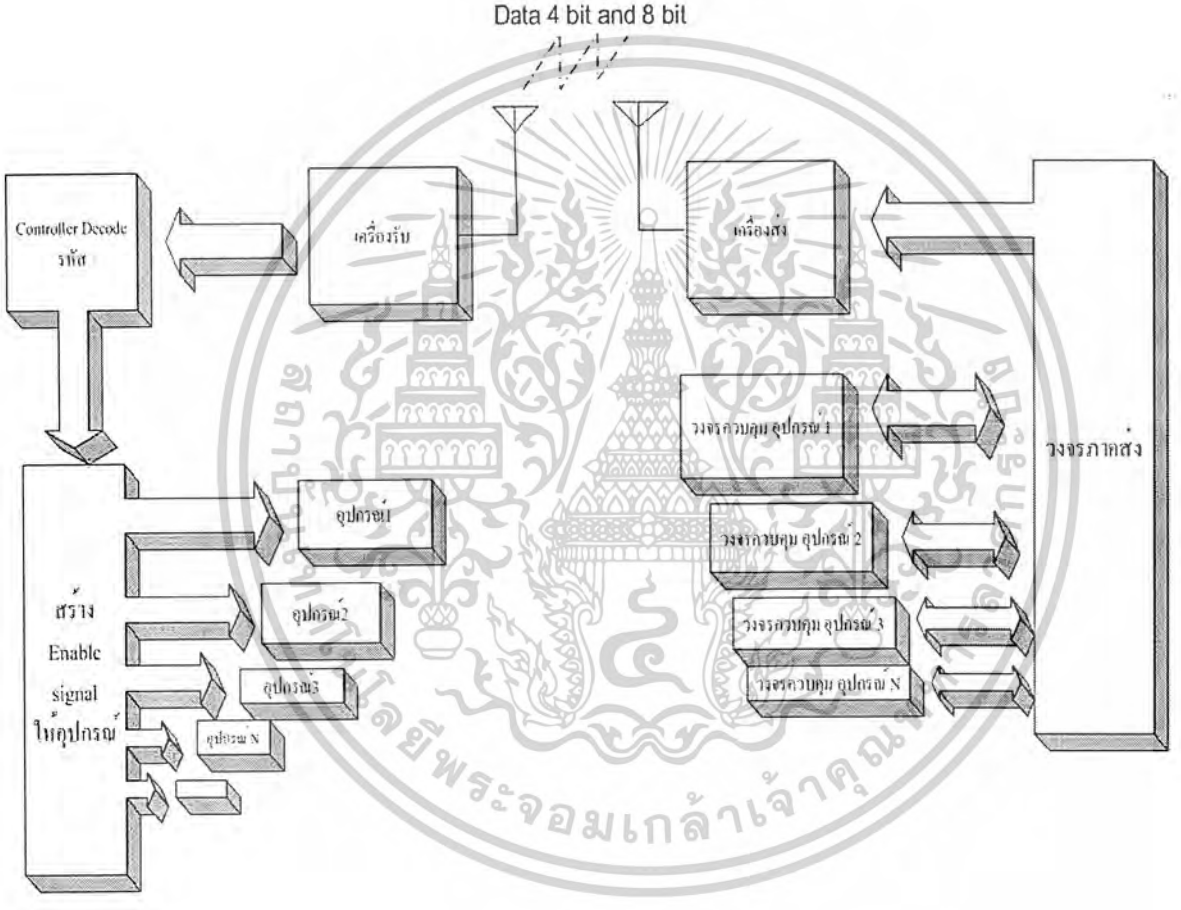
ในกรณีที่มีการอินเตอร์รัพท์จากหลายแหล่ง และโปรแกรมตอบสนองการอินเตอร์รัพท์แต่ละโปรแกรมยาวเกิน 8 ไบต์ เราไม่สามารถเขียนโปรแกรมตอบสนองการอินเทอร์รัพท์ไว้ที่ตำแหน่งของอินเทอร์รัพท์เวกเตอร์ได้ ซึ่งจะแก้ปัญหานี้ได้โดยกำหนดให้ตำแหน่งของอินเทอร์รัพท์เวกเตอร์ให้ทำโปรแกรมกระโดด โดยกระโดดไปที่ตำแหน่งเก็บโปรแกรมตอบสนองการอินเทอร์รัพท์ที่เขียนไว้ที่ตำแหน่งอื่นดังตัวอย่าง ต่อไปนี้

```
ORG    0000H        ; เริ่มโปรแกรมของระบบ
LJMP   MAIN         ; กระโดดไปโปรแกรมหลัก
ORG    000BH        ; ตำแหน่งของอินเตอร์รัพท์ Timer 0
LJMP   LED1         ; กระโดดไปโปรแกรมตอบสนองการอินเตอร์รัพท์ชื่อ LED1
ORG    0030H        ; ตำแหน่งหลังอินเตอร์รัพท์เวกเตอร์
MAIN : ......       ; โปรแกรมหลัก
......
LED1 : ......       ; โปรแกรมตอบสนองการอินเตอร์รัพท์ Timer 1
......
RETI                ; กลับสู่โปรแกรมหลัก
```

จากโปรแกรมจะเห็นว่า เมื่อระบบทำงาน จะต้องทำที่ตำแหน่ง 0000H โดยกระโดดไปทำโปรแกรมหลักที่ตำแหน่งต่อจาก 0030H เพราะตำแหน่งดังกล่าวข้ามอินเตอร์รัพท์เวกเตอร์จากแหล่งต่าง ๆ ไปแล้ว เมื่อมีการอินเตอร์รัพท์จาก Timer 0 โปรแกรมจะต้องทำงานที่ตำแหน่ง 000BH แต่โปรแกรมตอบสนองการอินเตอร์รัพท์ยาวมาก ที่ตำแหน่ง 000BH จึงให้ทำโปรแกรมกระโดดโดยกระโดดไปที่โปรแกรมตอบสนองการอินเตอร์รัพท์ชื่อ LED1 ซึ่งอยู่ท้ายโปรแกรม เมื่อจบโปรแกรมจะจบด้วยคำสั่ง RETI เพื่อกลับไปโปรแกรมหลักต่อไป

ตอนที่ 3
หลักการทำงานของวงจร

การทำงานโดยรวม



รูปที่ 23 Block Diagram System ของระบบการทำงานโดยรวม

<u>1. วงจรภาคส่ง</u>

1.1 หน้าที่

- เป็นตัว Encoder และตัว Master Controller Port อุปกรณ์ทั้งหมด

1.2 หลักการทำงาน

MCS – 51 จะสร้างสัญญาณ ENABLE โดยการส่ง CODE ออกจากport 3.0 และ 3.1 เพื่อให้ตัว DECODER1 74HC138 สร้างเป็นสัญญาณ ENABLE ให้กับ MCS – 51 เมื่อส่งสัญญาณ ENABLE ไปแล้ว

2.1 Port ที่ไม่มีอุปกรณ์จะมีสัญญาณ No Device ออกมา ทำให้ MCS 51ข้ามการทำงานที่ Port นี้ไป

2.2 Port ที่มีข้อมูลจะเริ่มการส่งข้อมูลออกมาโดยข้อมูลจะประกอบด้วยค่า address ของอุปกรณ์ (4 BIT) และข้อมูลของอุปกรณ์นั้น (8 BIT) ออกมาพร้อมกับสัญญาณ Send เพื่อบอกให้ MCS 51 ทำการอ่านข้อมูลจาก Port นั้น เพื่อไปเข้ารหัสส่งไปที่ตัวรถ และจะส่งสัญญาณ ACK เพื่อบอกอุปกรณ์ว่าได้ข้อมูลแล้วให้ทำการส่งข้อมูลชุดต่อไปได้

2.3 ขั้นตอนที่ 2.2 จะเกิดขึ้นเรื่อยๆ พร้อมกับที่ MCS 51 ทำตัวเป็น Encoder ส่งไปให้ตัวรับที่รถ จนกว่าอุปกรณ์ที่ Port จะให้สัญญาณ Finish เพื่อบอก MCS 51 ให้ทำงานที่ Port ถัดไปได้

1.3 วงจร Driver Motor ของ Joystick

1. เมื่อ MCS 51ของวงจร Driver)ซึ่งมีลักษณะเป็น Slave (ได้รับสัญญาณ Enable จากตัว Master แล้ว MCS 51 ของวงงจร Driver จะส่ง Code เพื่อเลือก Channel และ เซต สัญญาณS tart ไปที่A DC

2. หลังจากที่ ADC ได้ทำการเปรียบเทียบค่าโวลท์ที่ได้จาก Joystick แล้วให้ผลออกมาเป็นเลขฐานสอง8 BIT ADC จะส่งสัญญาณE OC ไปบอกM CS 51 เพื่อให้M CS 51อ่านค่าข้อมูล8 BIT ที่ได้

3. 51MCS จะทำการตรวจสอบและเปรียบเทียบค่าที่ได้จากA DC ดังนี้ เลขที่มีค่าน้อยกว่าA C ode address ที่ได้จะเป็นC ode ถอยหลัง แต่ถ้ามากกว่า A แต่ น้อยกว่า B ไม่ส่งสัญญาณใดๆ ถ้ามีค่ามากกว่า B ให้เอาค่า B ลบออกด้วยค่ากลางคือ A+B/2 แล้วส่งCode address ไปเป็นสัญญาณเดินหน้า

-ทำเช่นเดียวกันกับสัญญาณเลี้ยวซ้ายและขวา

4. ส่งสัญญาณ Send เพื่อบอก Masterให้อ่านข้อมูลได้ และสุดท้ายจะส่งสัญญาณ Finish ออกไปเมื่อจบรอบการทำงาน

2. วงจร ภาครับ

MCS 51 จะรับสัญญาณจากภาคส่งมาและทำการ Decode ให้เป็นสัญญาณ 2 อย่างคือ

1. สัญญาณ Enable โดย MCS 51 จะเอาCo de address ที่แปลงมาได้ส่งไปที่IC Decoder 74 154HC เพื่อสร้างเป็นสัญญาณE nable ให้กับอุปกรณ์ ที่ภาครับ เช่น Motor

2. Coded 8 BIT ที่เป็นข้อมูลของอุปกรณ์นั้นๆ เช่น joy stick

3. วงจรควบคุมการเคลื่อนที่

ส่วนควบคุมการ เดินหน้า ถอยหลัง เลี้ยวซ้ายขวา จะใช้ไมโครคอนโทรลเลอร์เป็นตัวประมวลผลและส่งข้อมูลออกมาทาง port 3.0 – 3.3 โดยที่ port 3.0 และ3.1 จะเป็นส่วนของการควบคุมการหมุนของมอเตอร์ตัวหนึ่ง โดยส่งข้อมูลเข้ายัง input ของ ไอซี drive motor คู่หนึ่ง ส่วน port 3.2 และ 3.3 จะเป็นส่วนของการควบคุมการหมุนของมอเตอร์อีกตัวหนึ่ง โดยส่งข้อมูลเข้ายัง input ของ ไอซี drive motor อีกคู่หนึ่ง ซึ่งเอาท์พุทที่ได้ของแต่ละคู่ของ input จะนำไปขับ motor 1 ตัว

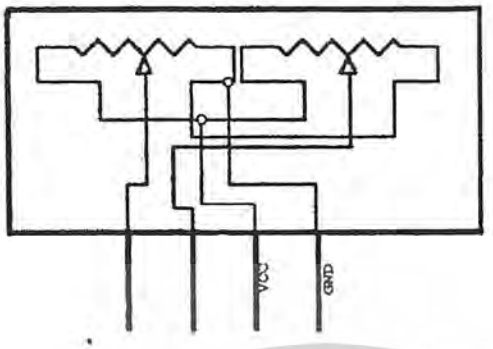
ในส่วนของการปรับความเร็วของรถ เราจะใช้การปรับ duty cycle ของสัญญาณที่นำไปขับไอซี drive motor โดยการนำข้อมูลที่เป็นดิจิตอลที่รับมาได้มาทำการแปลงเป็นสัญญาณอนาลอกโดยใช้วงจร R-2R ladder แล้วนำไป compare กับสัญญาณ sawtooth โดยใช้ตัว ไอซีออปแอมป์ comparator สัญญาณที่ได้ก็จะมี duty cycle เปลี่ยนแปลงตามแรงดันอนาลอกที่ได้มาจาก R-2R ladder แล้วส่งเอาท์พุทที่ได้ไปขับ ไอซี drive motor ซึ่งจะทำให้การหมุนของมอเตอร์เปลี่ยนแปลงตาม duty cycle ของสัญญาณเอาท์พุทที่ได้มาจากตัว comparator

4. ส่วนมอเตอร์

มอเตอร์ที่ใช้จะเป็นมอเตอร์เกียร์ขนาด 24 โวลต์เหตุที่ใช้มอเตอร์เกียร์เพราะต้องการแรงบิตที่สูงเพื่อว่าขณะที่รถวิ่งในสภาพถนนที่ไม่ดีแรงบิตที่สูงจะทำให้รถสามารถวิ่งต่อได้

5. ส่วนจอยสติ๊ก

ส่วนประกอบภายในจอยสติ๊กนั้นประกอบด้วย ตัวต้านทานปรับค่าได้ 2 ตัวดังรูปที่ 3.2 ตัว ต้านทานตัวแรกจะใช้ในการควบคุมการเดินหน้าถอยหลังของรถ ส่วนอีกตัวจะใช้ในการควบคุมการเลี้ยวซ้าย-ขวาของรถ โดยอาศัยหลักการแบ่งแรงดันของตัวต้านทานปรับค่าทำให้เอ้าท์พุทที่ได้จากจอยสติ๊กนั้นคือค่าแรงดันที่เปลี่ยนแปลงตามการโยกของคันโยก



รูปที่ 24 โครงสร้างภายในจอยสติ๊ก

6. ส่วนวงจร ADC

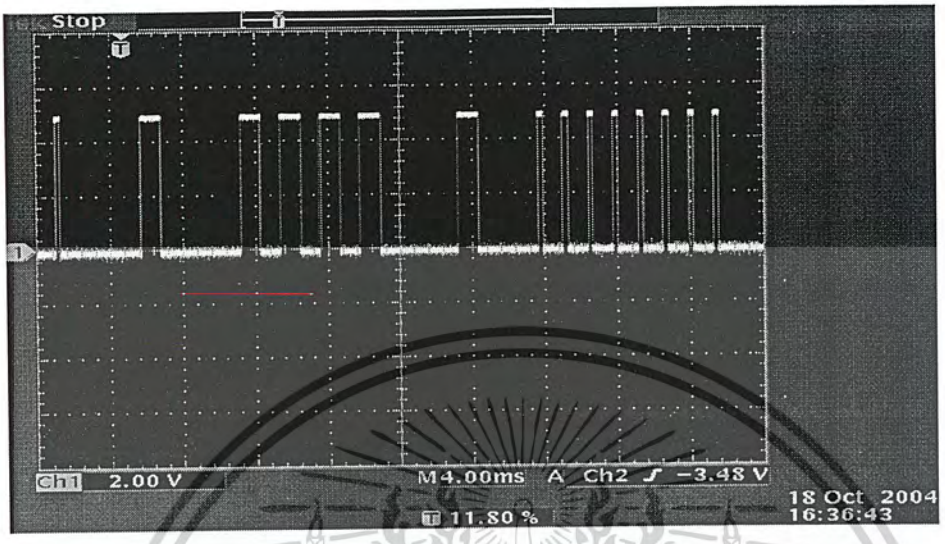
เนื่องจากแรงดันที่ได้จากจอยสติ๊กเป็นสัญญาณอนาลอกที่ MCS 51 ไม่สามารถนำไปประมวลผลได้จึงต้องอาศัยวงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล โดยใช้ไอซีเบอร์ ADC0809 ทำการแปลงสัญญาณเพื่อส่งต่อให้กับ MCS 51 ADC0809 เป็นADC ขนาดแปดบิต ดังนั้นจึงทำให้แรงดันจากจอยสติ๊กเมื่อแปลงเป็นสัญญาณดิจิตอลแล้วมีความละเอียดถึง 256 ระดับ

7. หลักการทำงานวงจรภาคส่ง

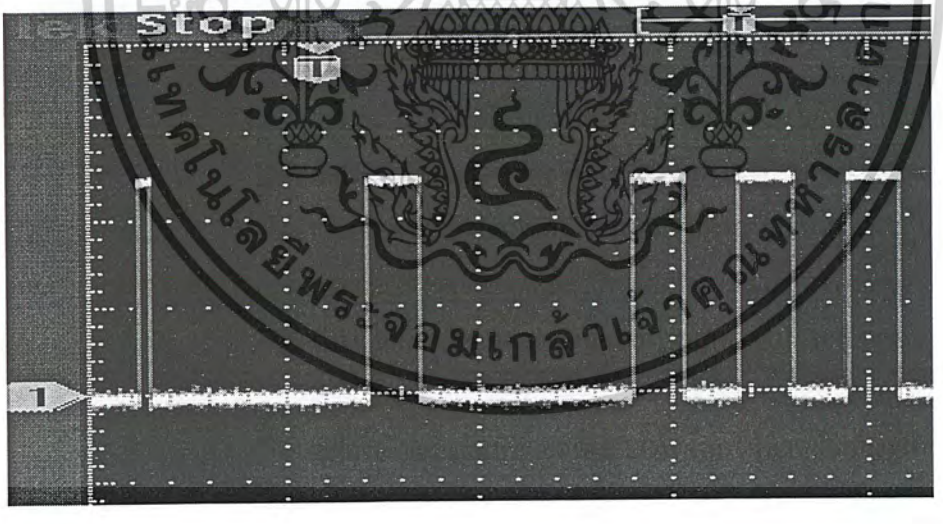
รับค่าจาก ADC 8 bit จากไมโครคอนโทรเลอร์ที่ควบคุมADC มาทำการแปลงเป็น code อนุกรมเพื่อส่งเข้าไป mod กับสัญญาณ carrie 443 MHz ของเครื่องส่งวิทยุ

รหัสที่ได้ทำการแปลงจะประกอบไปด้วย

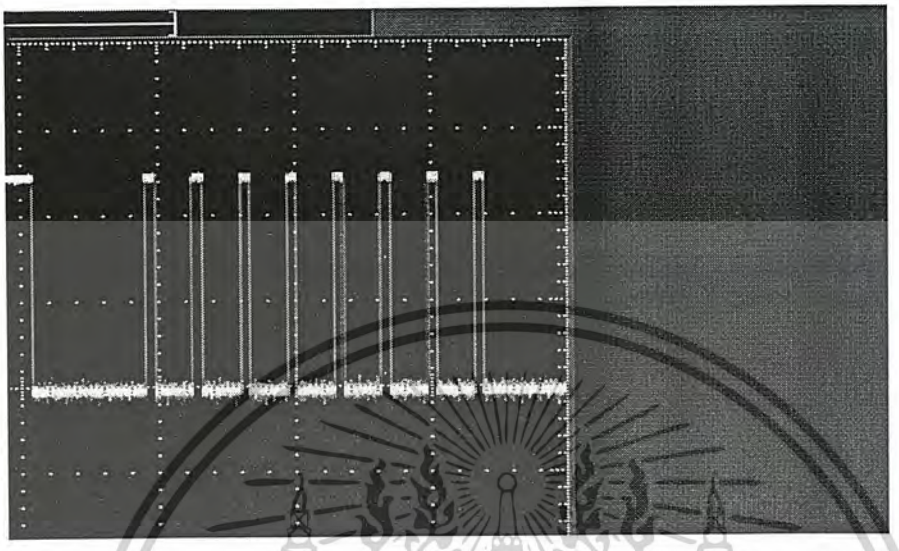
- บิตเริ่มต้น (start bit) มีขนาด 1 บิตจะอยู่ด้านหน้าของบิตรหัสอุปกรณ์และบิตรหัสข้อมูล มีหน้าที่ทำให้ฝั่งรับรู้ได้ว่ามีการเริ่มส่งของชุดข้อมูลแล้ว บิตเริ่มต้นจะตามด้วยช่องว่างมีช่วงเวลา 4 บิตเพื่อทำให้รู้ได้ว่านี่เป็นรหัสเริ่มต้น
- บิตรหัสอุปกรณ์ มีขนาด 4 บิต ทำให้จำนวนอุปกรณ์ที่ฝั่งส่งควบคุมไปยังฝั่งรับมีจำนวนสูงสุด 16 อุปกรณ์ มีหน้าที่ระบุหมายเลขอุปกรณ์ที่จะใช้ชุดบิตข้อมูลที่ตามมานั้น
- บิตรหัสข้อมูล มีขนาด 8 บิต ซึ่งอาจเป็นข้อมูลของ adc หรือข้อมูลที่กำหนดมาจากอุปกรณ์นั้นๆ



รูปที่ 25 ชุดรหัส 13 บิต



รูปที่2 6 บิตเริ่มต้นและรหัส 4 บิต



รูปที่ 27 รหัส 8 บิต

โดยที่ระหว่างการแลกเปลี่ยนข้อมูลระหว่างไมโครคอนโทรเลอร์ที่เป็นชุดควบคุมอุปกรณ์และชุดสร้างรหัสได้มีข้อตกลงในการเช็คบิตเพื่อการส่งข้อมูลที่ถูกต้อง 5 รหัสด้วยกัน ดังนี้

1. Send Signal

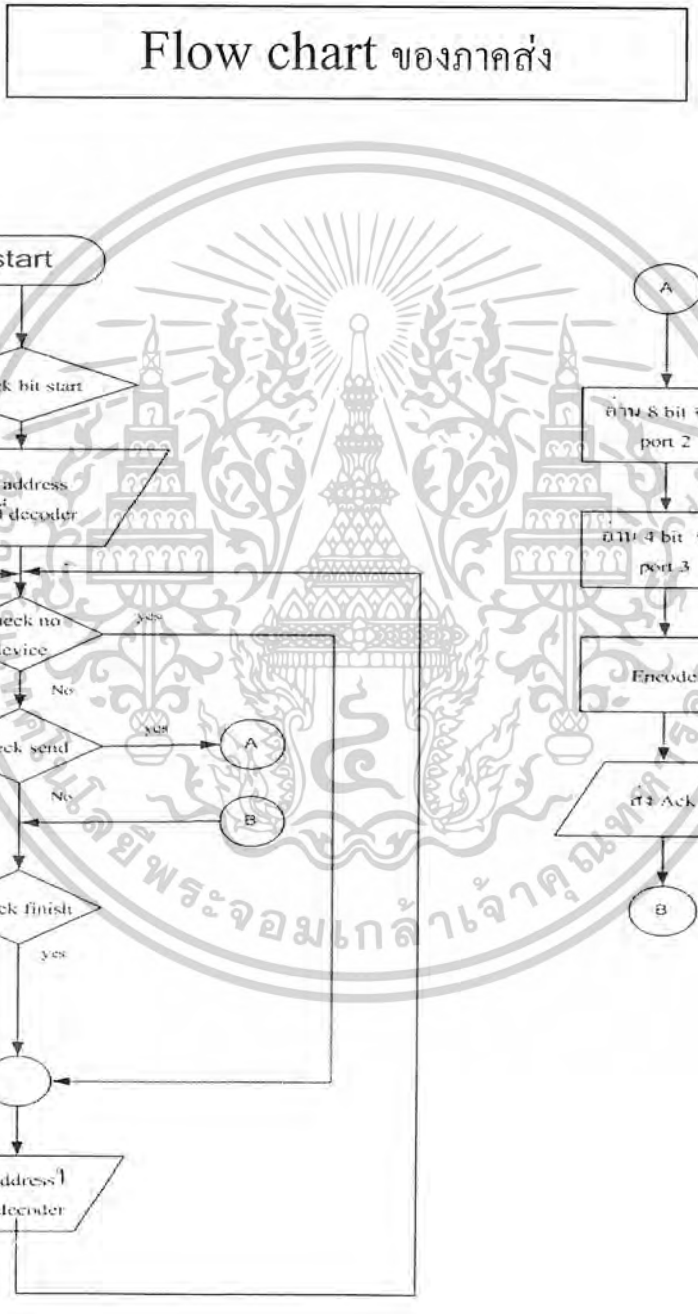
เป็นการเช็คขาที่ 5 หรือพอร์ต 1.4 ของทั้งตัวชุดสร้างรหัสและชุดควบคุมอุปกรณ์คือเมื่อชุดสร้างรหัสพร้อมที่จะรับข้อมูลจากตัวชุดควบคุมอุปกรณ์แล้วจะส่งสัญญาณ Enable ผ่านทางตัว Decoder 2 to 4 เมื่อชุดฝั่งควบคุมอุปกรณ์ได้รับสัญญาณ Enable แล้วจะเริ่มทำการควบคุม adc ให้ส่งค่าที่ sampling ได้กลับมาเพื่อทำการคิดคำนวณเพื่อจัดแยกชุดข้อมูลไปตามรหัสอุปกรณ์ ในที่นี้ได้แก่ Motor แต่ละตัวที่จะหมุนทิศทางไปตามรหัสอุปกรณ์ เช่นเมื่อ ข้อมูลที่ได้จาก adc channel 1 มีค่ามากกว่า 127 แสดงว่าเป็นรหัสเดินหน้า เมื่อได้รหัสข้อมูล และรหัสอุปกรณ์แล้วชุดควบคุมอุปกรณ์ จะทำให้บิต 1.4 เป็น LOW เพื่อให้ตัวสร้างสัญญาณรู้ได้ว่ามีการส่งข้อมูลมาแล้ว

2. Ack Signal

เป็นการเคลียบิตที่ชุดสร้างอุปกรณ์ที่พอร์ต 1.7 เพื่อให้ฝั่งควบคุมอุปกรณ์รู้ว่าข้อมูลที่ส่งไปให้กับชุดสร้างรหัสได้รับแล้ว และจะทำการส่งข้อมูลชุดใหม่ต่อไปได้No Device signalเป็นการเช็คบิตของชุดสร้างรหัสเพื่อให้รู้ว่าอุปกรณ์ที่ได้ทำการส่ง Enable ไปนั้นมีอุปกรณ์อยู่ หรือไม่ถ้าไม่จะได้ทำการข้ามไป แต่ถ้ามีจะเกิดการรอสองสัญญาณ คือ send signal และ finish signal โดยที่ถ้าพอร์ต 1.3 มีบิต เป็น LOW แสดงว่าอุปกรณ์อยู่

3. Finish Signal

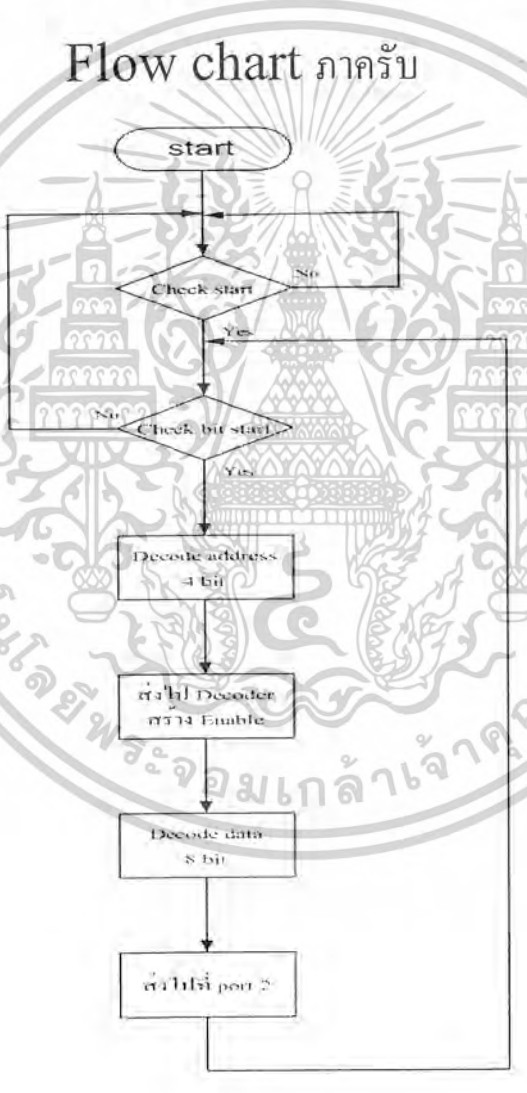
เป็นการเช็คบิตที่พอร์ต 1.6 ของชุดสร้างรหัสคือเมื่อชุดควบคุมอุปกรณ์ได้ให้ข้อมูลแก่ชุดสร้างสัญญาณไปหมดแล้วจะทำการเคลียร์บิตที่พอร์ต1.6เพื่อบอกให้รู้ว่าได้ส่งข้อมูลครบหมดแล้วให้ทำการผ่านไปยังอุปกรณ์อื่นๆได้



รูปที่ 28 Flow chart ของภาคส่ง

8. หลักการทำงานของภาครับ

ไมโครคอนโทลเลอร์จะทำการวนลูปหาบิตเริ่มต้นและเมื่อเจอแล้วจะ delay เพื่อเช็คหาว่าเป็นบิตเริ่มต้นจริงหรือไม่เมื่อหาบิตเริ่มต้นแล้วจะทำการวนลูปเพื่อทำการเช็คแต่ละบิตที่เข้ามาที่พอร์ต1.0 โดยหลักการคือ ค่าบิตที่เป็น 1 จะมีขนาดช่วง High ที่กว้างกว่า Low ทำการเช็ค 4 บิตที่เป็นชุดรหัสของอุปกรณ์ และหลังจากนั้นจะเช็ค อีก 8 ครั้งเพื่อให้ได้รหัสข้อมูล จากนั้นจะนำไปออกเป็นเอาท์พุทที่ พอร์ต2 เป็นรหัสข้อมูล 8 บิต และที่พอร์ต 3 เป็นรหัสอุปกรณ์ 4 บิต รหัส 4 บิตนี้จะนำไปเข้า Decoder 4 to 16 เพื่อเป็นสัญญาณ Enable ให้แก่อุปกรณ์ต่อไป

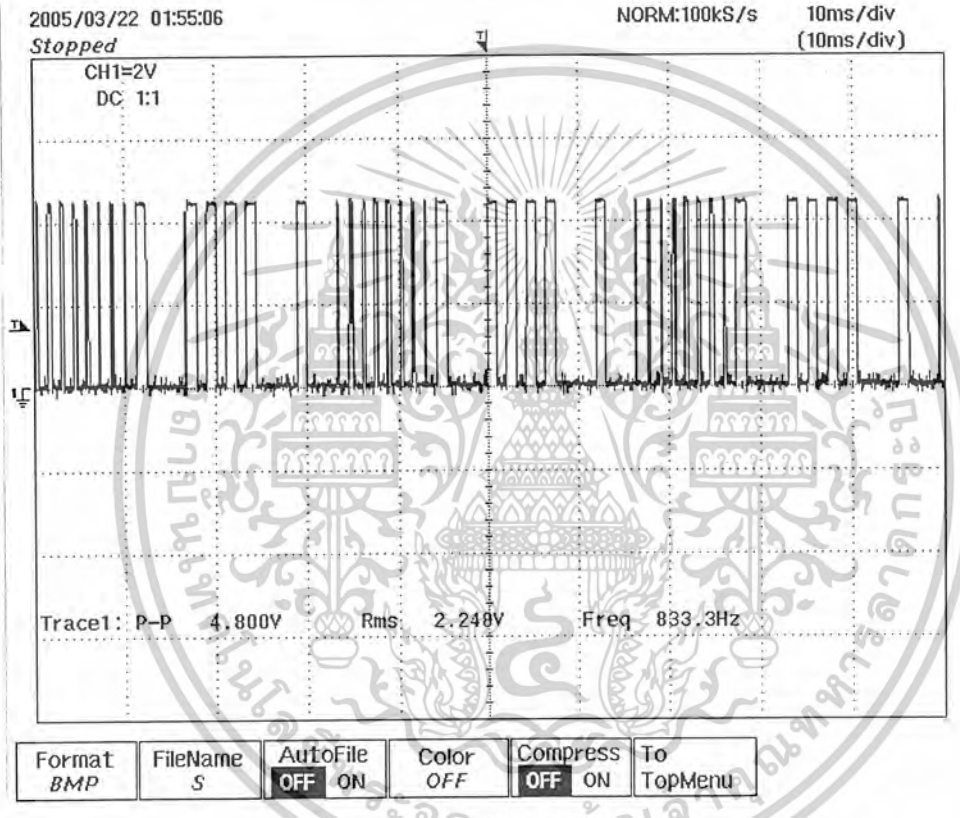


รูปที่ 29 Flow chart ของภาครับ

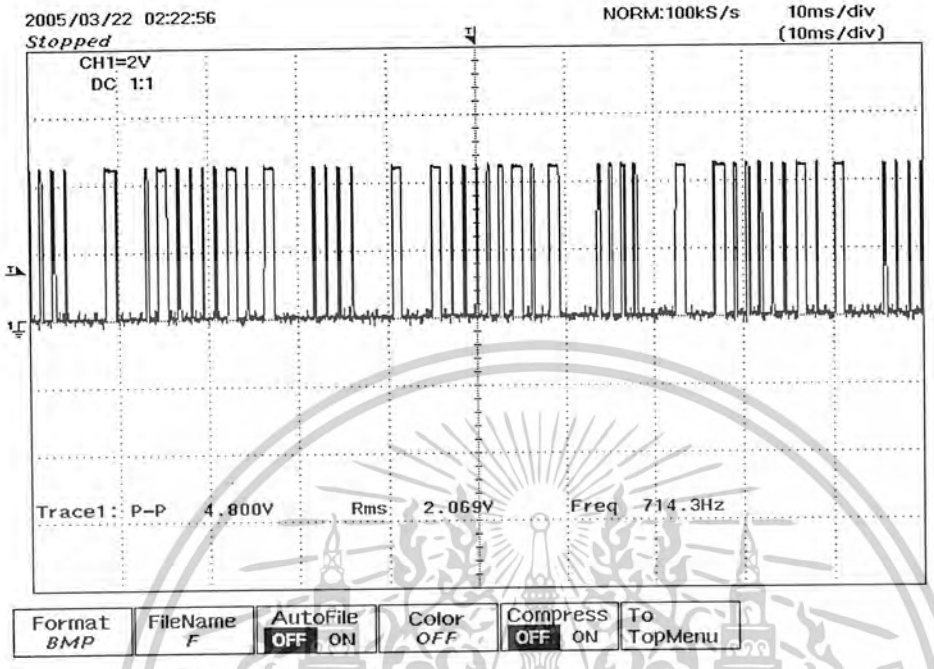
ตอนที่ 4

ผลการทดลอง

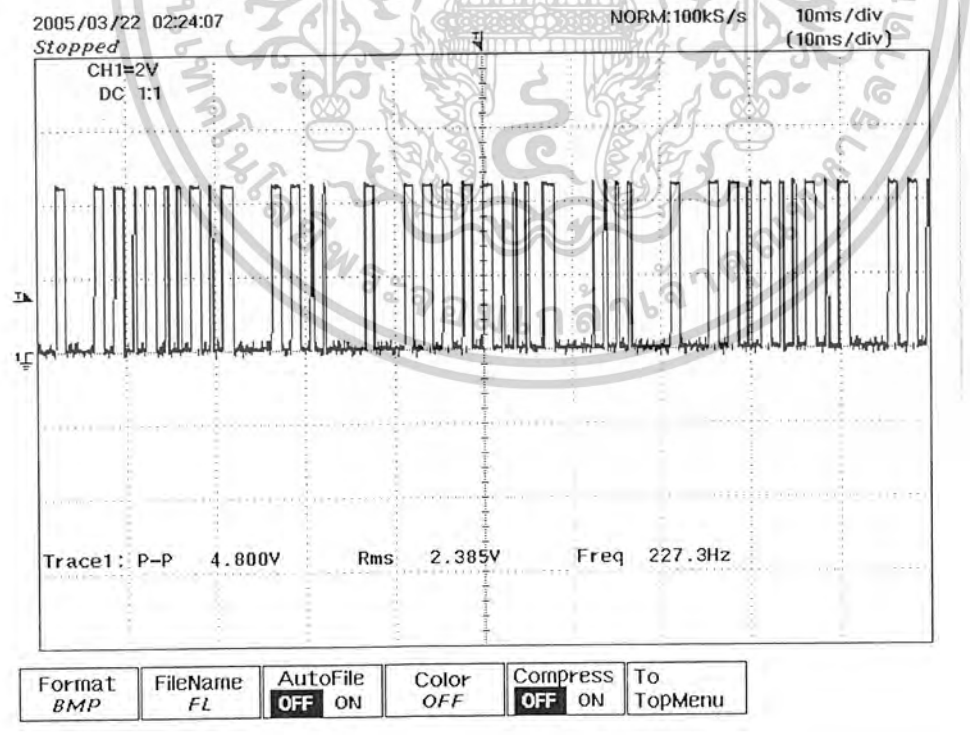
สัญญาณที่ไปควบคุมทิศทางการเคลื่อนที่ของรถ



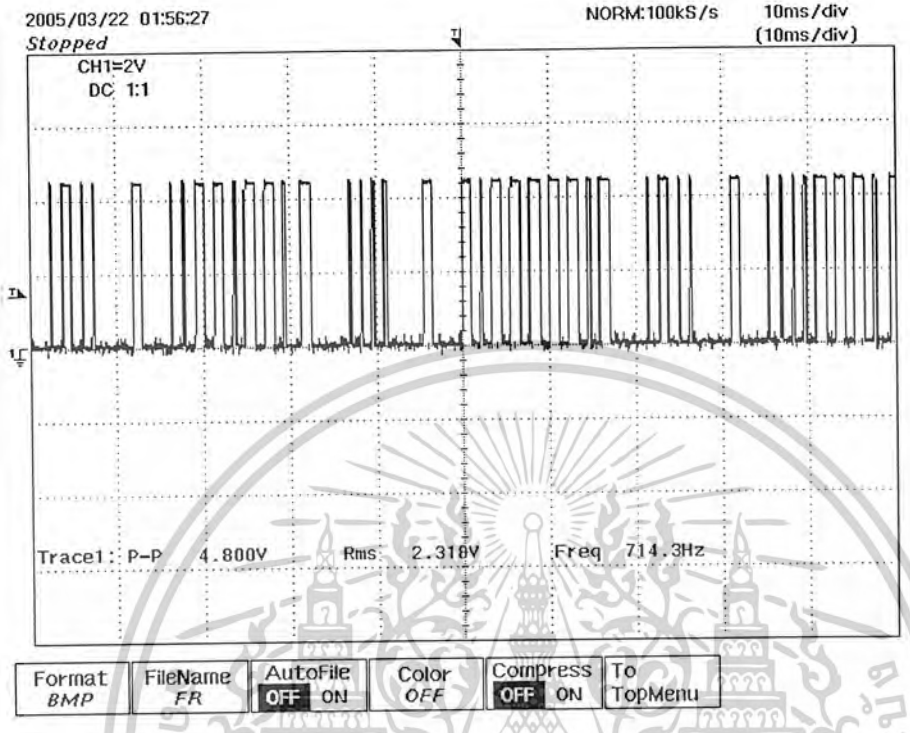
รูปที่ 30 ไม่มีการเคลื่อนที่ของรถ



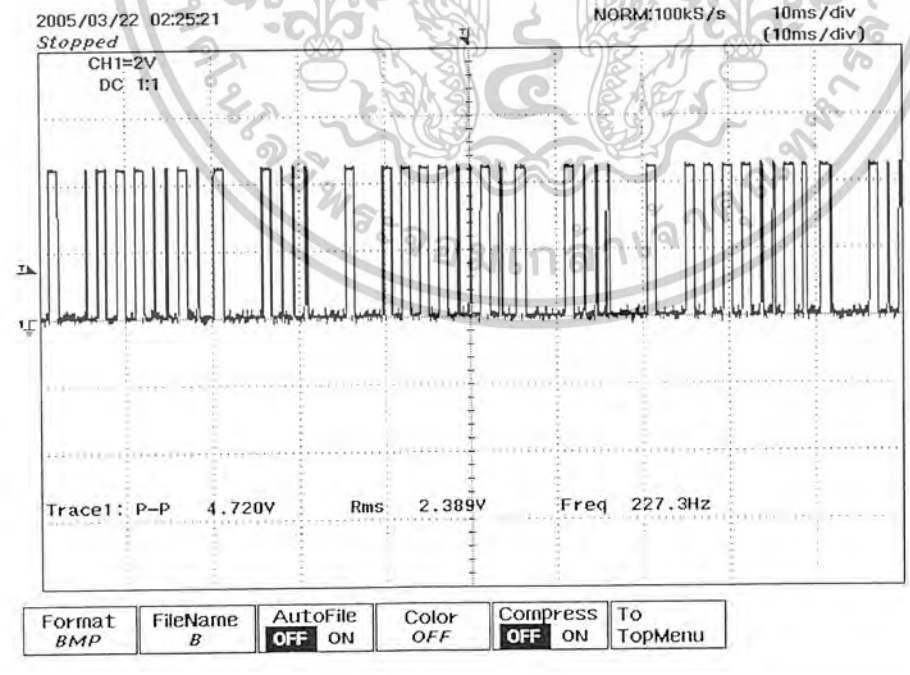
รูปที่ 31 เคลื่อนที่ไปข้างหน้า



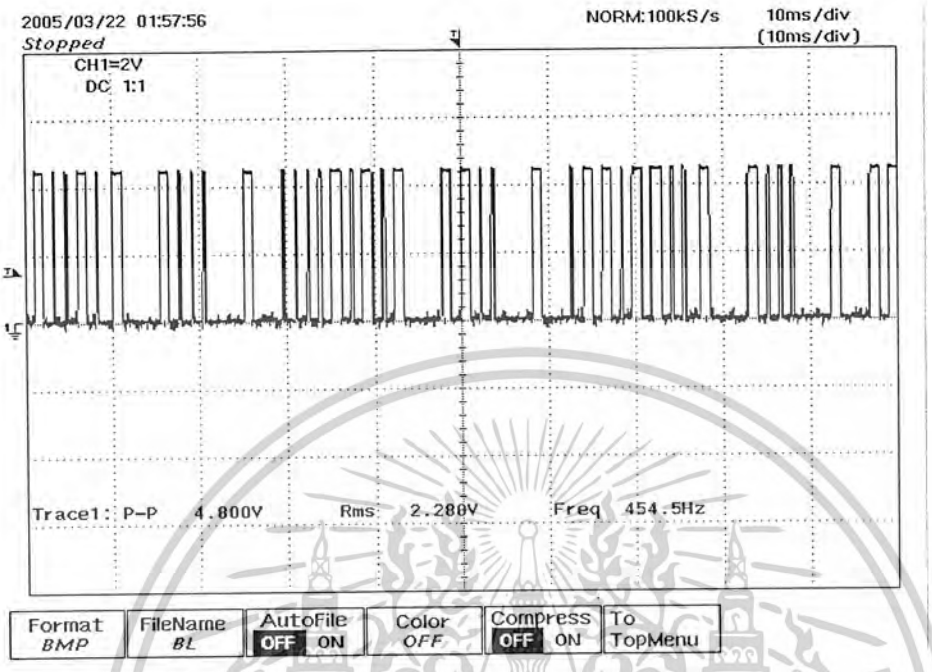
รูปที่ 32 เคลื่อนที่ไปข้างหน้าและเลี้ยวซ้าย



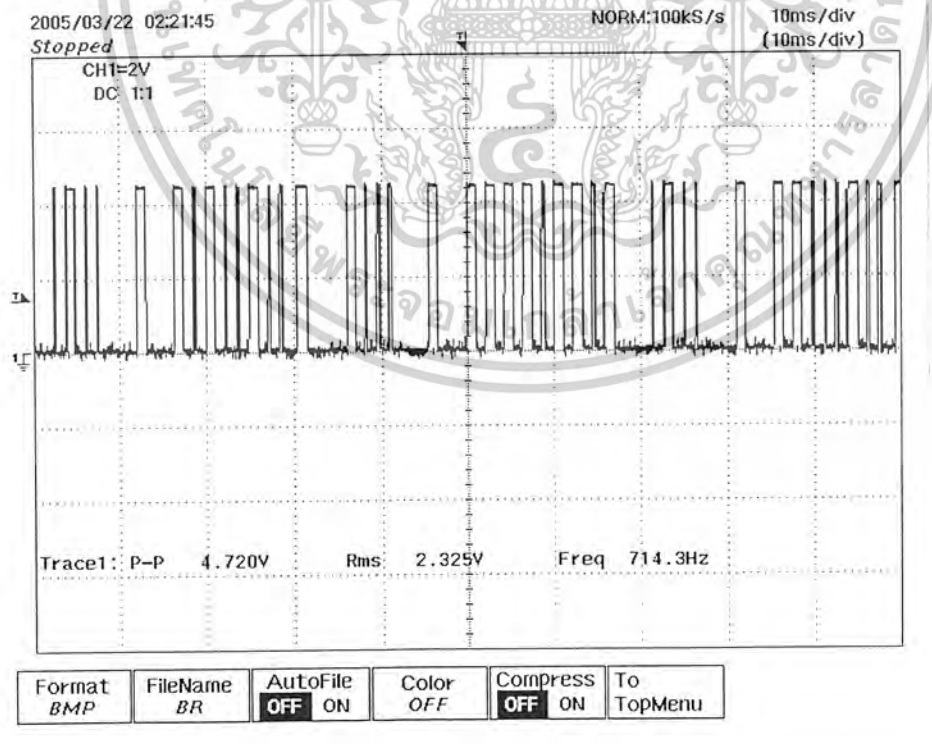
รูปที่ 33 เคลื่อนที่ไปข้างหน้าและเลี้ยวขวา



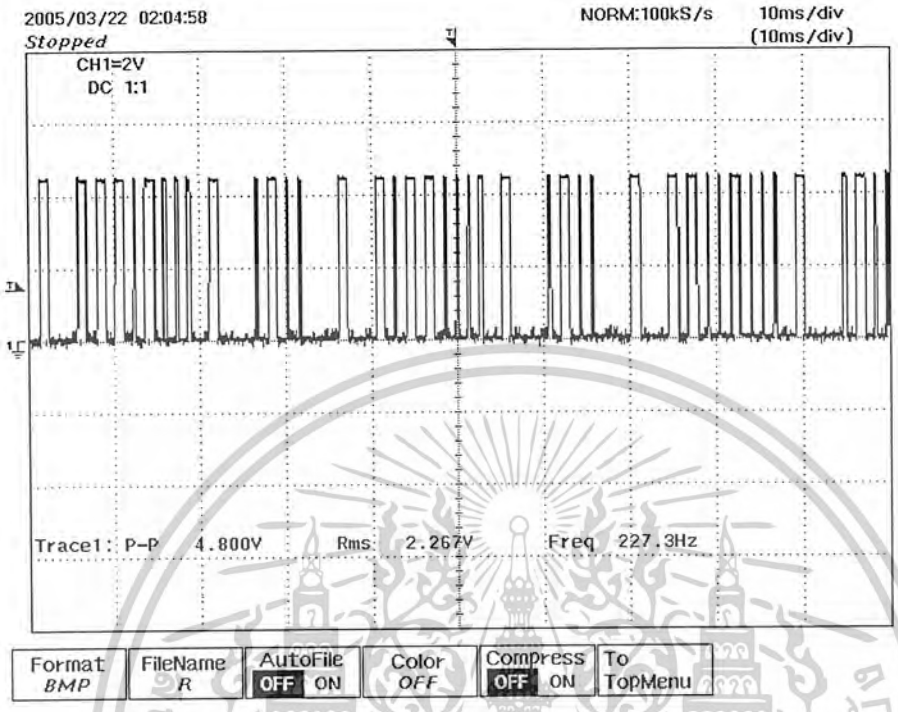
รูปที่ 34 เคลื่อนที่ถอยหลัง



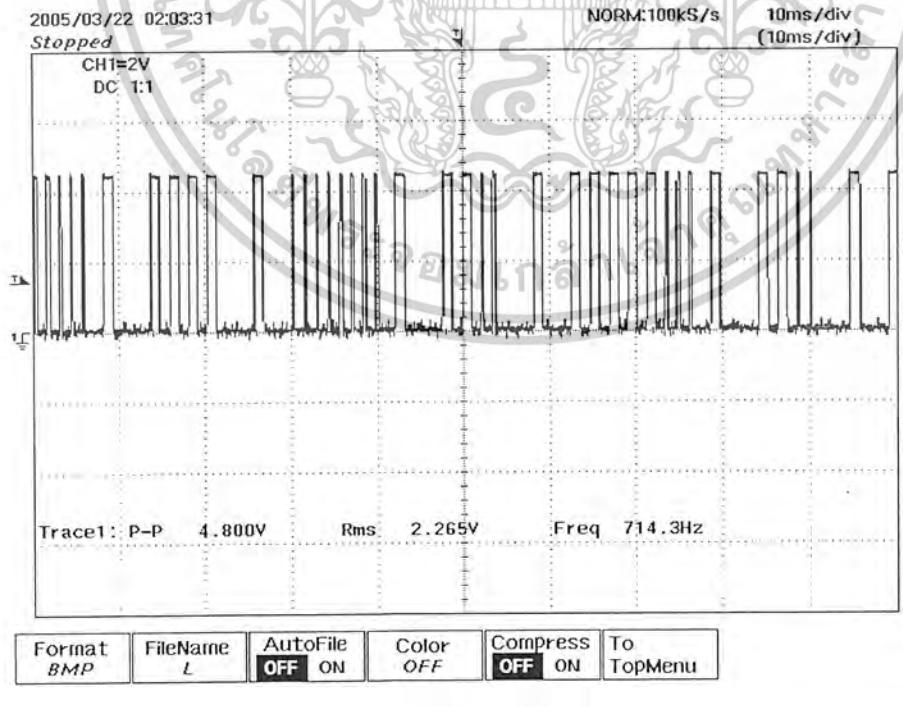
รูปที่ 35 เคลื่อนที่ถอยหลังและเลี้ยวซ้าย



รูปที่ 36 เคลื่อนที่ถอยหลังและเลี้ยวขวา

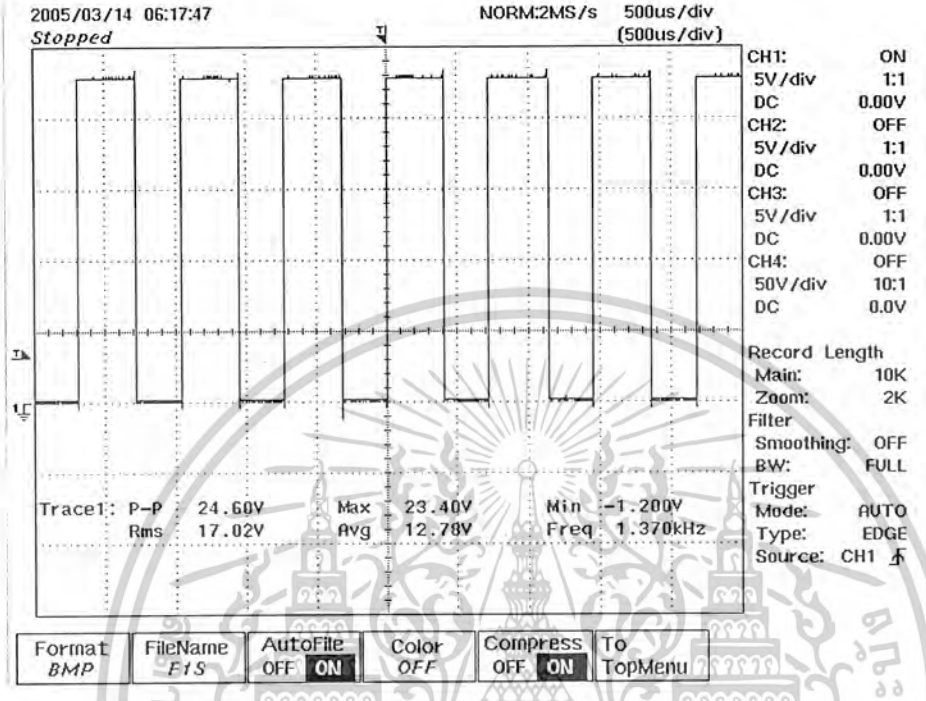


รูปที่ 37 เคลื่อนที่เลี้ยวขวา

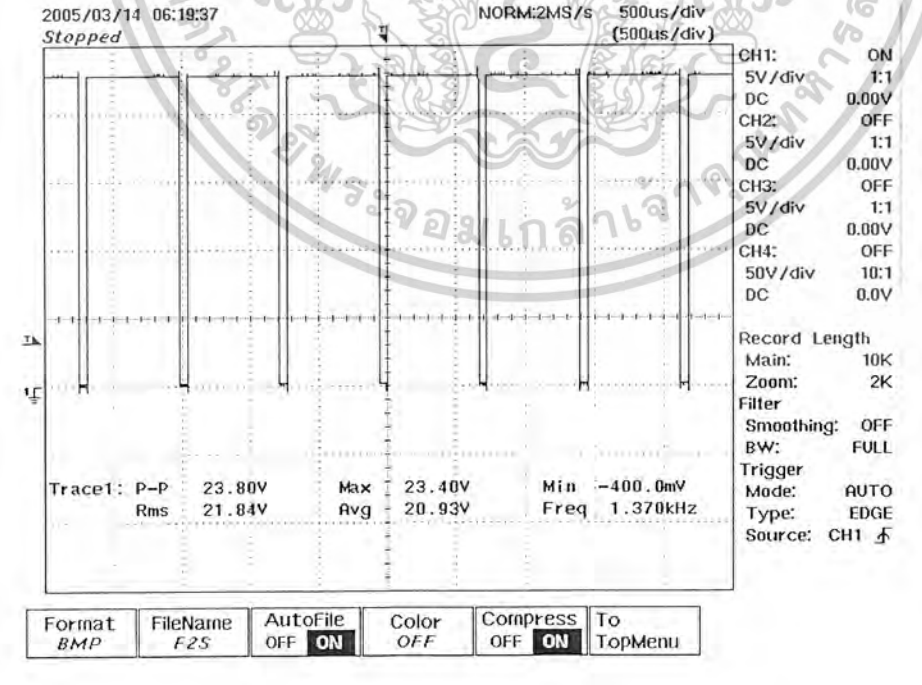


รูปที่ 38 เคลื่อนที่เลี้ยวซ้าย

สัญญาณที่นับไป Enable ไอซีขับมอเตอร์ซึ่งทำให้ความเร็วเปลี่ยนแปลง



รูปที่ 39 เคลื่อนที่ไปข้างหน้าด้วยความเร็วค่าหนึ่ง



รูปที่ 40 เคลื่อนที่ไปข้างหน้าด้วยความเร็วที่มีค่ามากกว่าค่าแรก

สรุปผลการทดลอง

1.ในการปรับความเร็วของการเคลื่อนที่ของรถเราจะใช้การปรับ Duty Cycle ของสัญญาณที่จะนำไปขับมอเตอร์โดยใช้หลักการของ Pulse Width Modulate ซึ่งทำให้ได้ค่าความเร็วตามที่เราบังคับ joystick

2.ในการควบคุมทิศทางการเลี้ยวของรถเราจะใช้หลักการของการปล่อยสัญญาณพัลส์ที่นำไปขับมอเตอร์ด้วยค่าของ Duty Cycle ที่ไม่เท่ากันของแต่ละมอเตอร์ เช่น เลี้ยวซ้ายจะให้ค่า Duty Cycle ของสัญญาณพัลส์ของล้อซ้ายมีค่าน้อยกว่าล้อขวา

บรรณานุกรม

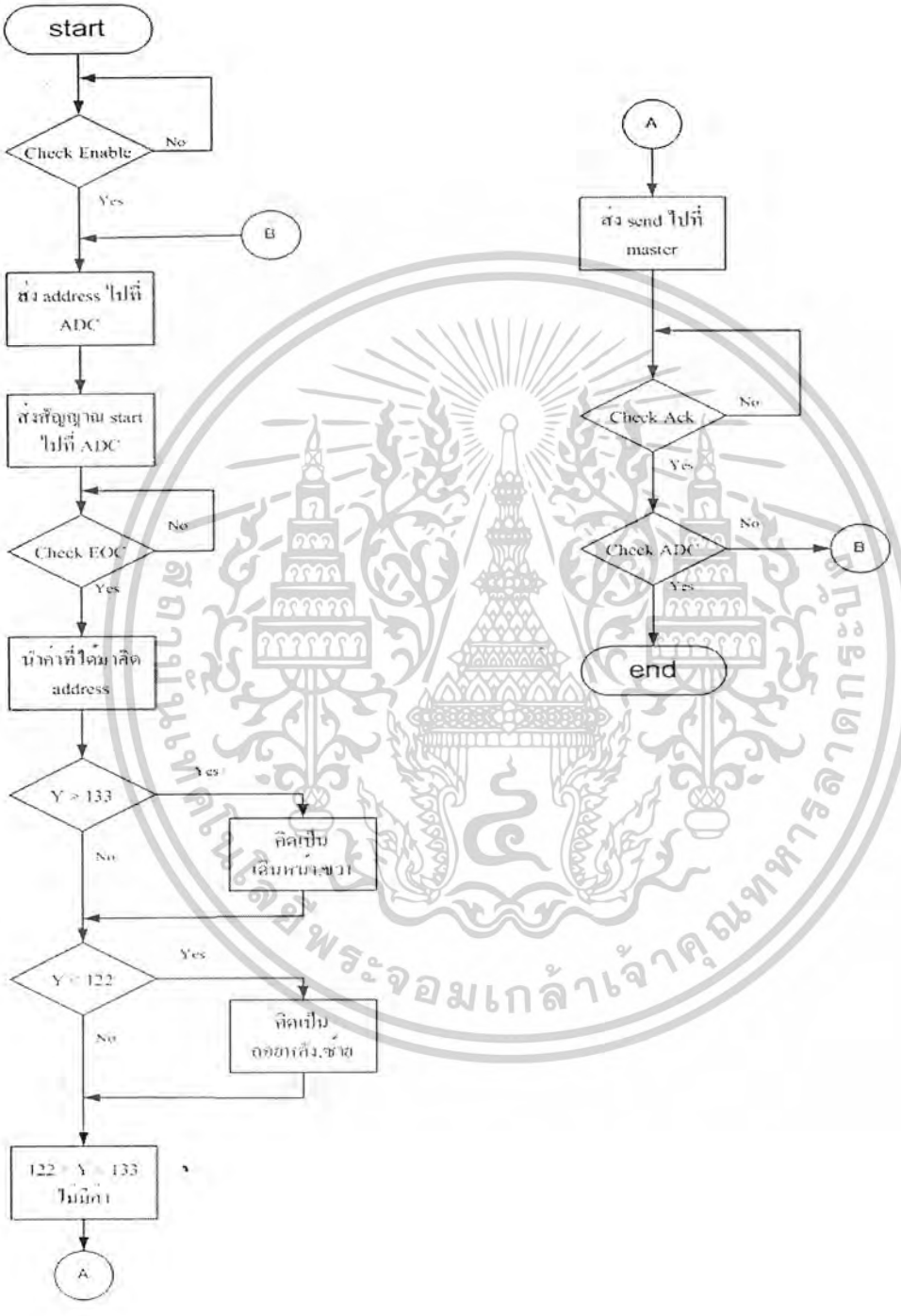
1. ผศ. วรพงศ์ ตั้งศรีรัตน์ , ออปแอมป์และการประมวลผลสัญญาณอนาลอก (OP-AMPs and Analog Signal Processing)
2. ชัยวัฒน์ ลิ้มพรจิตรวิไล,วรพจน์ กรแก้ววัฒนกุล, "เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS 51", Innovative Experiment, 476 หน้า, 2539.
3. Electronic Laboratory III ภาควิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์, KMITL
4. คู่มือเทียบเบอร์ไอซี TTL, บริษัท SE-Education จำกัด.

ภาคผนวก

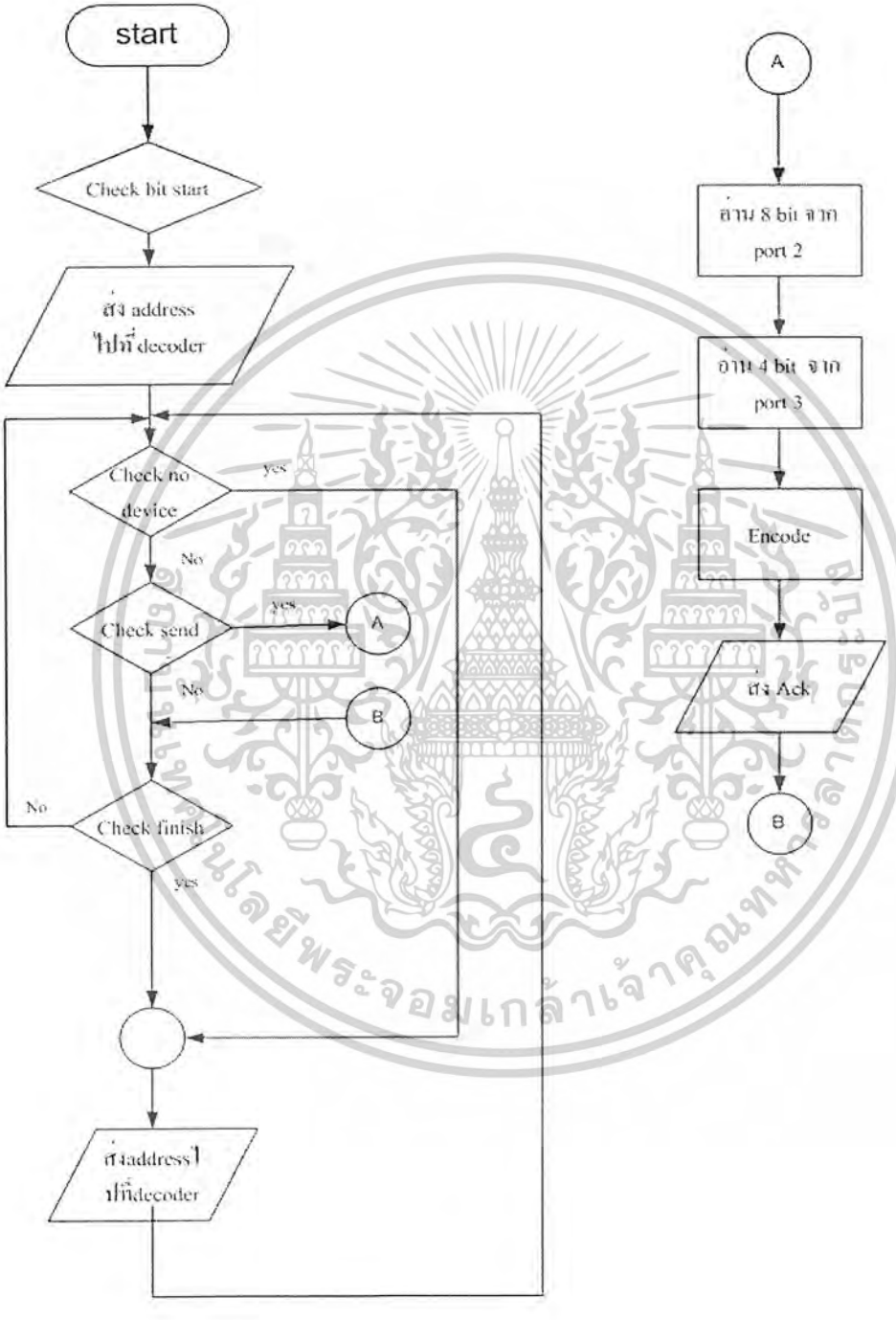
ภาคผนวก ก.

แสดง Flowchart ของโปรแกรมที่ใช้ในโครงงาน

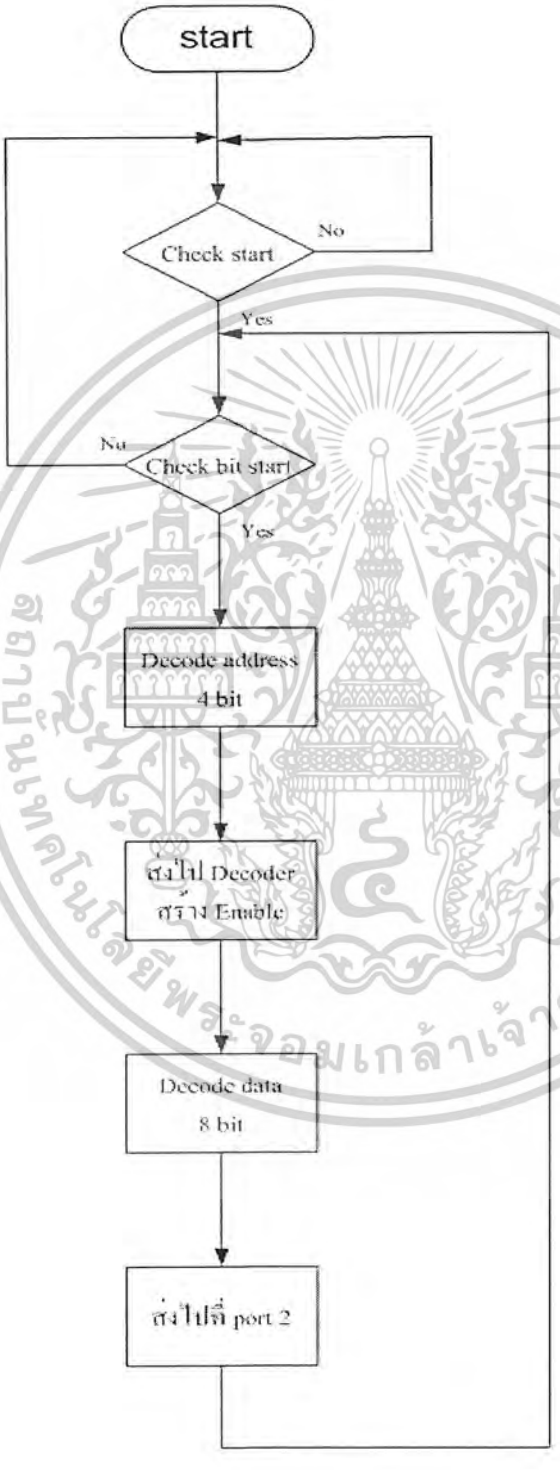
Flowchart ส่วนควบคุม Joystick



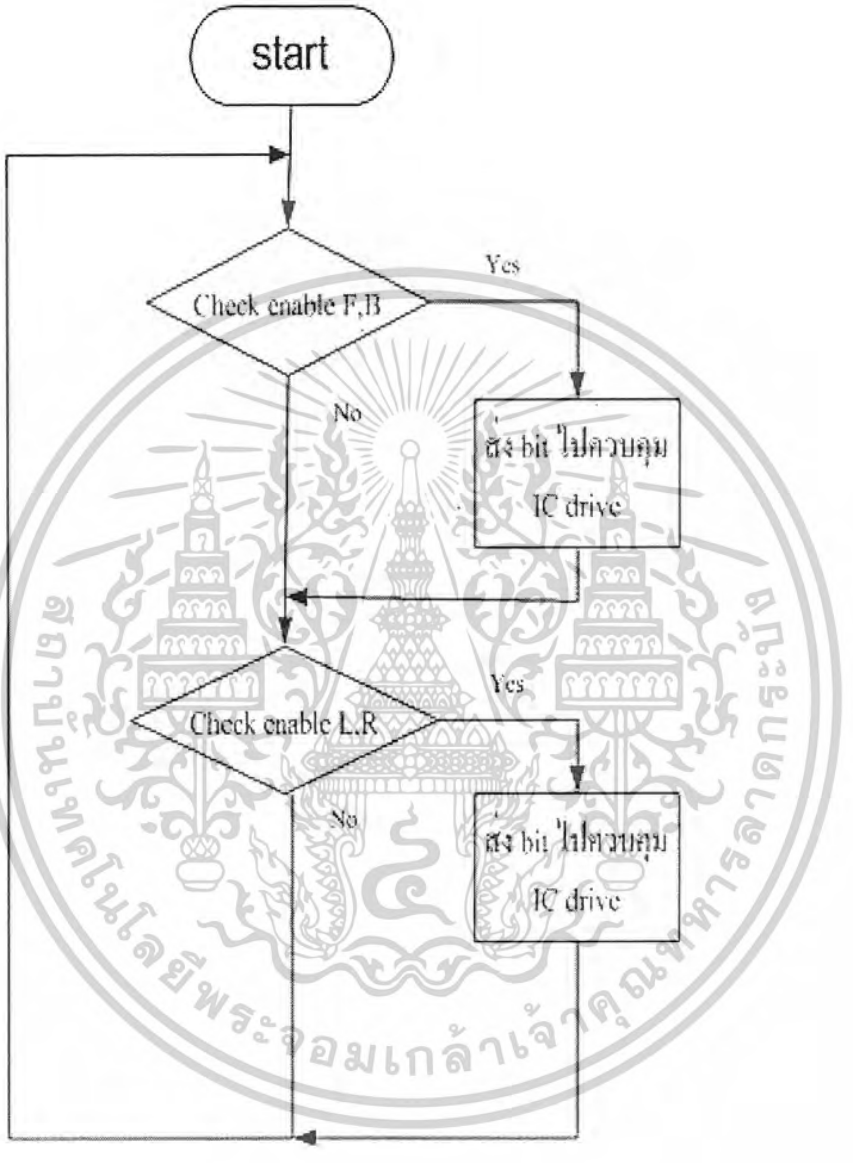
Flowchart ส่วนของวงจรภาคส่ง



Flowchart ส่วนของวงจรภาครับ

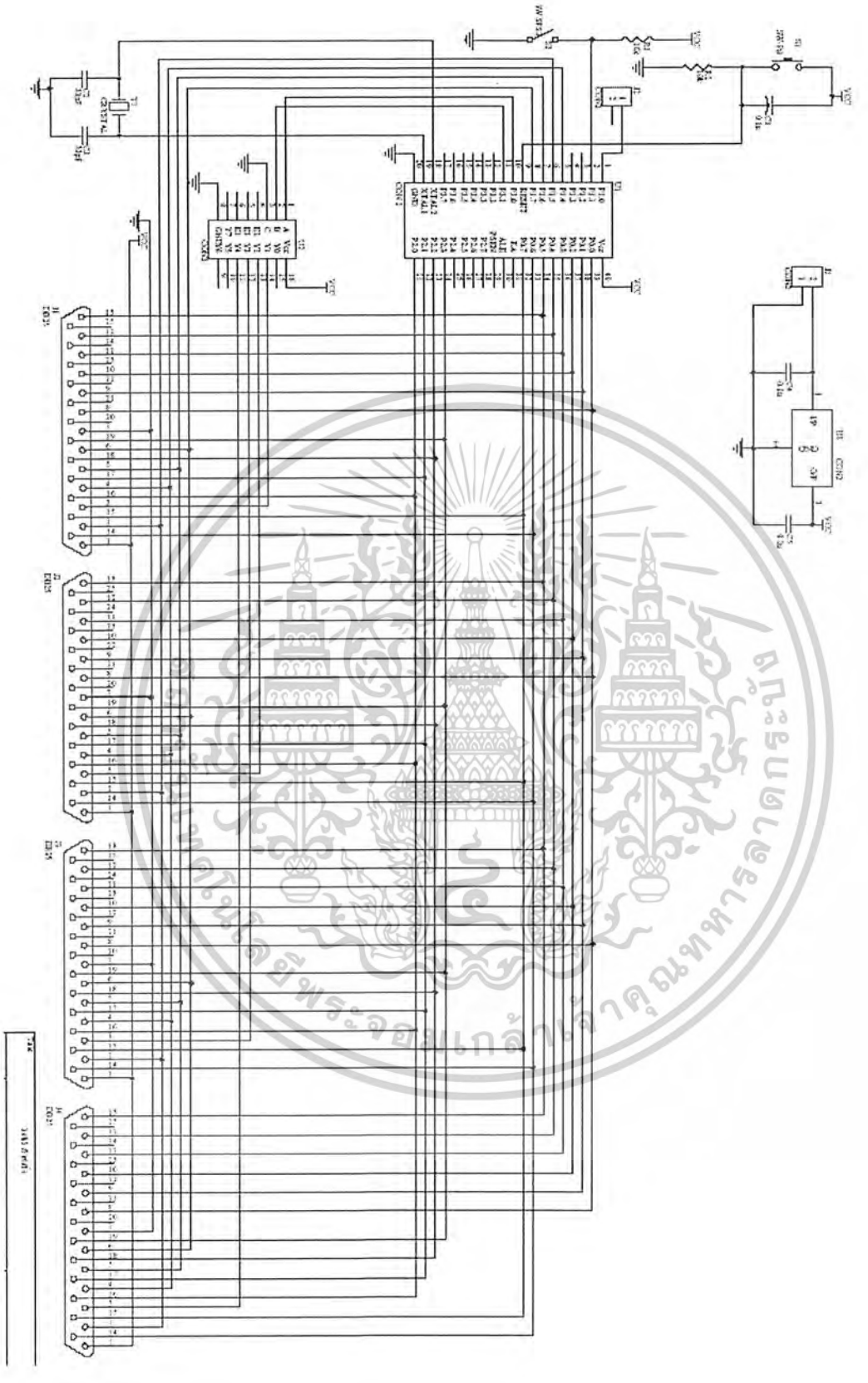


Flowchart ส่วนควบคุมทิศทาง

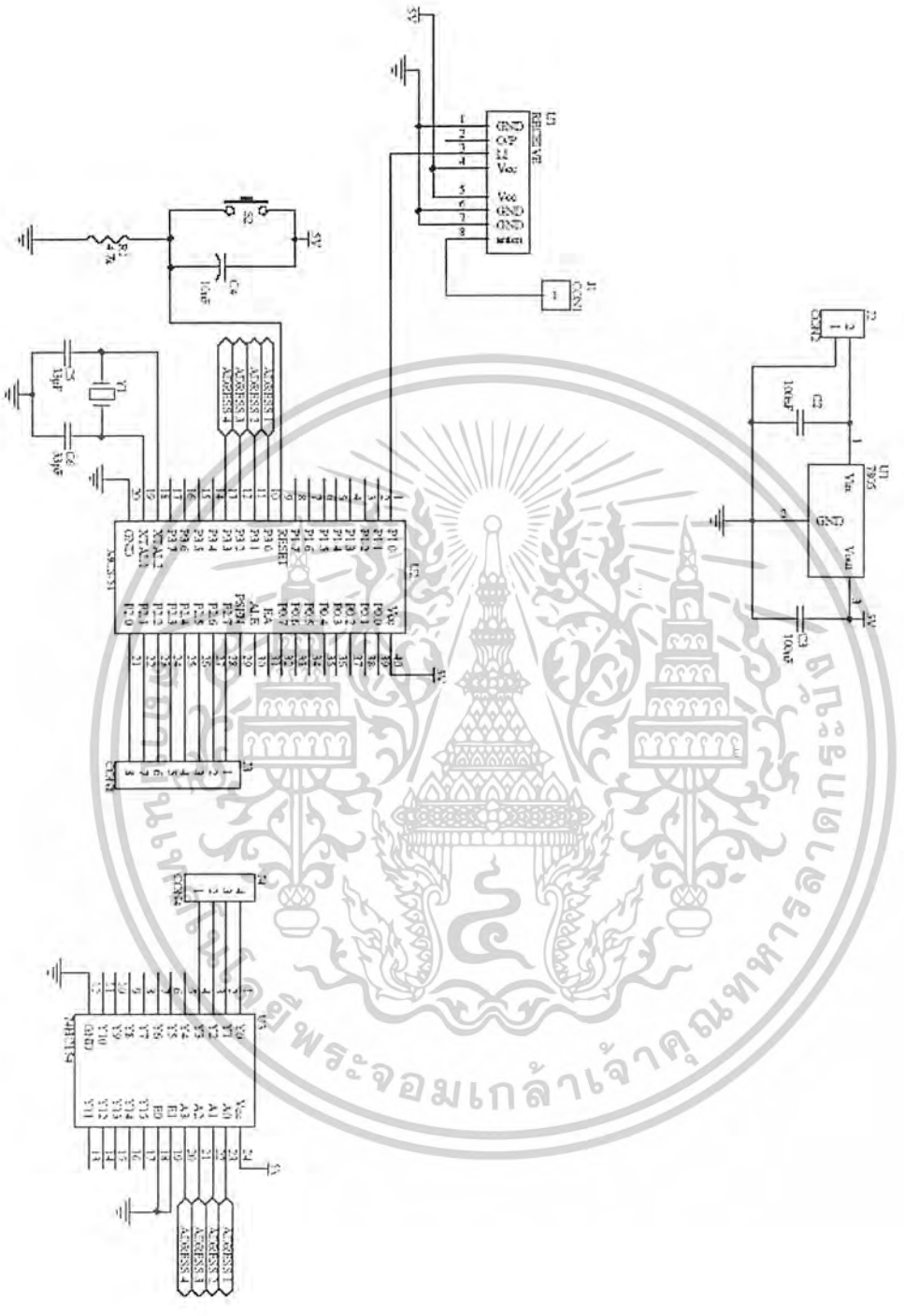


ภาคผนวก ข.

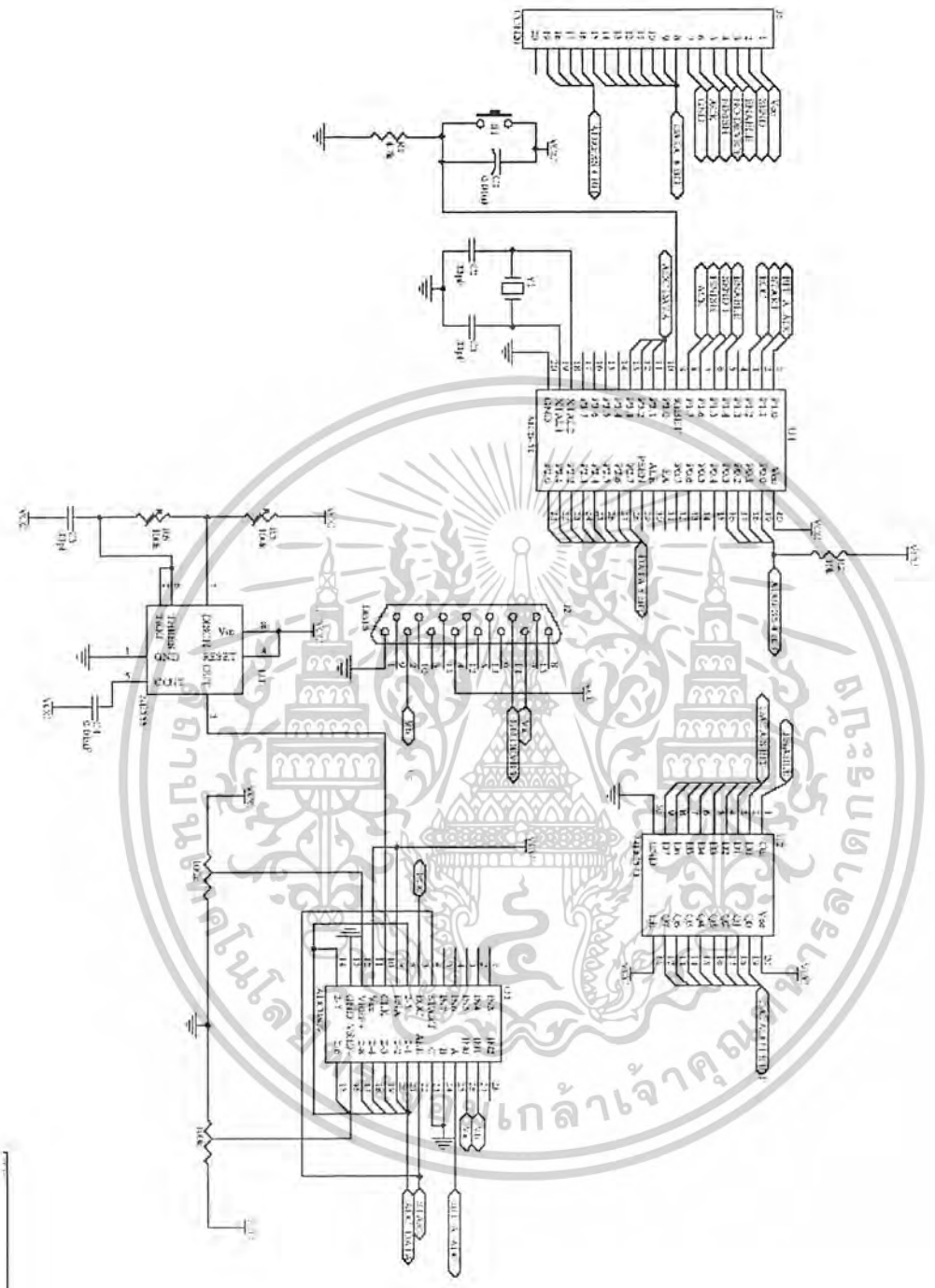
แสดงรายละเอียดของวงจรที่เกี่ยวข้องกับโครงงาน



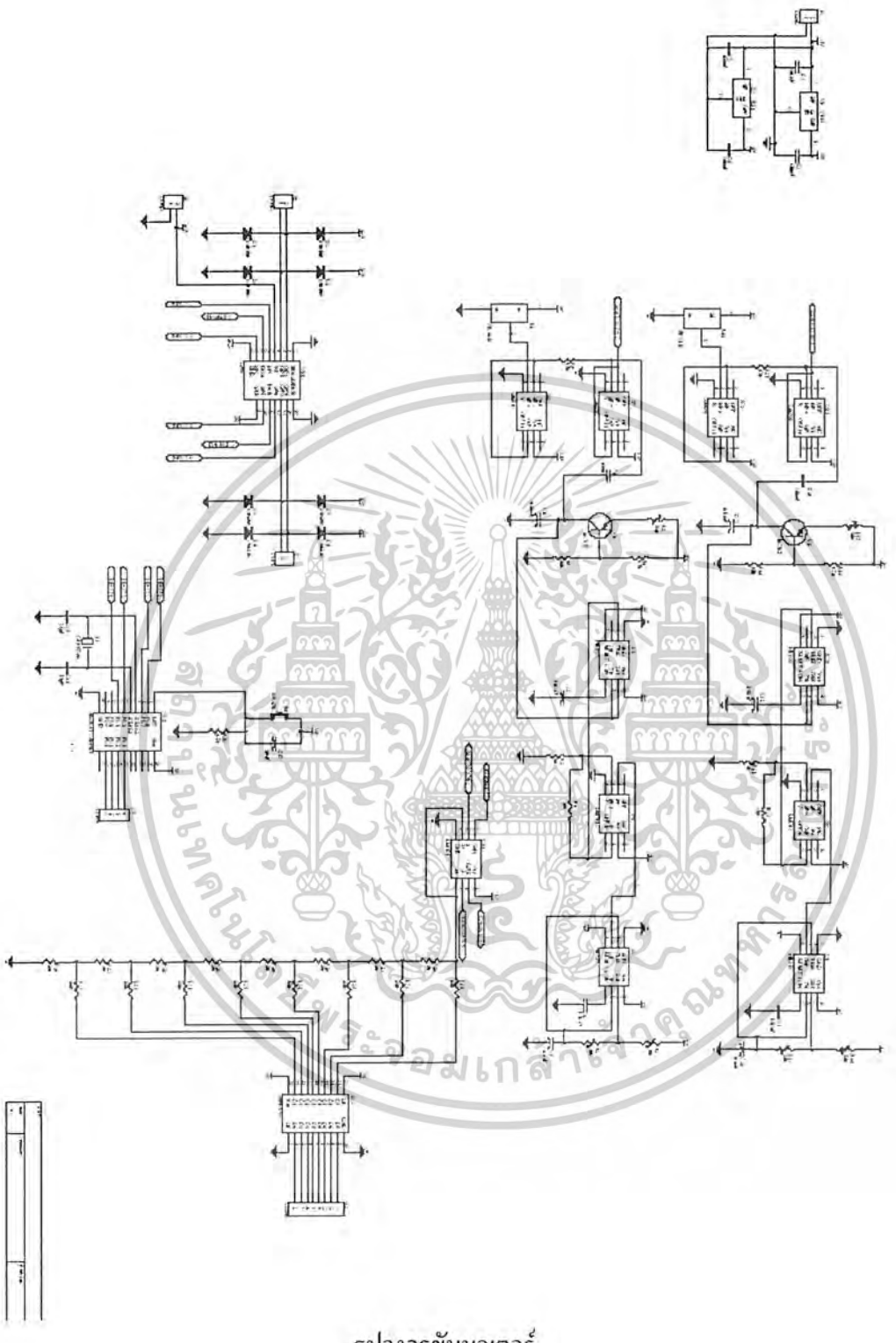
รูปวงจรภาคส่ง



รูปวงจรภาครับ



รูปวงจรของส่วน Joy Stick



รูปวงจรขับมอเตอร์

ภาคผนวก ค.

แสดงรายการไอซีที่เกี่ยวข้องกับโครงงาน

รายการไอซีที่ใช้มีดังนี้

1. NE555 จำนวน 3 ตัว
2. LM358 จำนวน 5 ตัว
3. LM741 จำนวน 2 ตัว
4. L298 จำนวน 4 ตัว
5. 74HC573 จำนวน 3 ตัว
6. 74HC154 จำนวน 1 ตัว
7. MCS8051 จำนวน 3 ตัว
8. ADC0809 จำนวน 1 ตัว