การประมวลผลภาพและหุ่นยนต์สำหรับหุ่นยนต์เตะฟุตบอล

Image Processing and Robot for RoboCup Small Size

โดย

นายสุรชัย อุทัยสมัยรัตน์

นายอธิกิต สุภาพ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

Image Processing and Robot for Robocup small size

By

MR.SURACHAI UTHAISAMAIRAT

MR.ATHIKIT SUPAB

A THESIS SUBBMITTED IN PARTIAL FULLFILMENT OF

THE REQUIREMENT FOR THE DEGREE OF

BACHELOR IN DEPARTMENT ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2004

หัวข้อปริญญานิพนธ์	การประมวลผลภาพและหุ่นยนต์แตะฟุตบอล
ชื่อนักศึกษา	นายสุรชัย อุทัยสมัยรัตน์ รหัสนักศึกษา 44010554
	นายอธิกิต สุภาพ รหัสนักศึกษา 44010577
อาจารย์ที่ปรึกษา	รศ.ดรปีติเขต สู้รักษา
	อ.บุญย์ชนะ ภู่ระหงษ์
ระดับการศึกษา	ปริญญาตรีวิศวกรรมศาสตรบัณฑิต
	สาขาวิศวกรรมสารสนเทศ
ภาควิชา	วิศวกรรมสารสนเทศ
ปีการศึกษา	2547

ปริญญานิพนธ์ฉบับนี้ได้รับความเห็นชอบจากอาจารย์ที่ปรึกษาเป็นที่เรียบร้อยแล้ว

.....

(รศ.ดรปีติเขต สู้รักษา)

.....

(อ.บุญย์ชนะ ภู่ระหงษ์)

อาจารย์ผู้ควบคุมปริญญานิพนธ์

หัวข้อโครงงาน	การประมวลผลภาพและหุ่นยนต์สำหรับหุ่นยนต์เตะฟุตบอล
จัดทำโดย	นายสุรชัย อุทัยสมัยรัตน์
	นายอธิกิต สุภาพ
อาจารย์ที่ปรึกษา	รศ.ดรปิติเขต สู้รักษา
	อ.บุญ์ชนะ ภู่ระหงษ์
ระดับการศึกษา	วิศวกรรมศาสตรบัณฑิต
ภาควิชา	วิศวกรรมสารสนเทศ
ปีการศึกษา	2547

บทคัดย่อ

ในปัจจุบันการจัดการแข่งขันหุ่นยนต์ต่างๆ นั้นเริ่มเป็นที่จับตามองของผู้คนเพิ่มมากขึ้น ซึ่งการแข่งขันหุ่นยนต์แตะฟุตบอล(RoboCup) ก็เป็นอีกการแข่งขันหนึ่งที่ได้รับความนิยมทั้งในและนอกประเทศ และโครงงานนี้ก็ได้รับแรงบันดาลใจจากการแข่งขั้นนี้เช่นกันซึ่งการทำหุ่นยนต์แตะฟุตบอลนี้ ต้องมีกระบวนการต่างๆ ทางวิศวกรรมมากมาย ที่ต้องนำมาใช้ร่วมกัน ไม่ว่าจะเป็นกระบวนการประมวลผลภาพ(Image Processing) หลักการด้านการต่อวงจรไฟฟ้า

สำหรับโครงงานนี้จะทำเกี่ยวกับการประมวลผล และการออกแบบหุ่นยนต์ โดยที่การประมวลผลภาพ จำนำไปใช้ในการตัดสินใจในโปรแกรมปัญญาประดิษฐ์อีกทีหนึ่ง และปัญญาประดิษฐ์จะทำการสั่งงานอีกครั้งหนึ่ง

Thesis Title	Image Processing and Robot for Robocup small size		
Student	Mr.SURACHAI	UTHAISAMAIRAT	ID.44010554
	Mr.ATHIKIT	SUPAB	ID.44010577
Advisor	Assoc.Prof.Dr.PITIKHATE SOORAKSA		
	Mr.BOONCHANA PHURAHONG		
Graduate level	Bachelor Degree of Information Engineering		
Department	Information Engineering		
Academic	2004		

Abstract

At present,many people keep an eye on robot contest and the Soccer Robot is the favourites robot contest in Thailand and foreign country. This project has an idea from this contest too. The way for making Soccer Robot must have join several engineering process i.e. Image processing,circuit

This project using Artificial Intelligence and designing robot. We use Image processing in order that decide AI problem and use AI in order that command robot activities

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงไม่อาจสำเร็จไปได้ หากไม่ได้รับความช่วยเหลือและความร่วมมือจากหลายๆฝ่ายด้วยกัน เริ่มจากอาจารย์ที่ปรึกษาปริญญานิพนธ์ซึ่งได้แก่ รศ.ดรปิติเขต สู้รักษา และอาจารย์บุณย์ชนะ ภู่ระหงษ์ ที่คอยให้ความช่วยเหลือในด้านต่างๆอย่างดีเยี่ยมมาโดยตลอดระยะเวลาที่ทำปริญญานิพนธ์นี้ จนกระทั่งโครงงานนี้ผ่านไปได้ด้วยดี

ขอขอบคุณ พี่ๆในห้อง INFO-DYNAMICS Laboratory ที่ให้คำแนะนำต่างๆและให้ความช่วยเหลือเป็นอย่างดี และขอขอบคุณคณาจารย์และเพื่อนๆในภาควิชาวิศวกรรมสารสนเทศทุกคนที่ให้คำติชม และคอยช่วยเหลือเสมอมา

สุดท้ายนี้คณะผู้จัดทำต้องขอกราบขอบพระคุณ บิดา มารดา บุคคลที่มีความสำคัญที่สุดในชีวิตที่ทำให้มีทุกวันนี้ คอยให้กำลังใจ ช่วยเหลือ และสนับสนุนในทุกๆด้าน จึงขอขอบพระคุณมา ณ ที่นี้ด้วย

คณะผู้จัดทำ

สารบัญ

สารบัญ (ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

สารบัญรูปภาพ (ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 แนวคิดและที่มา

ปัจจุบัน มนุษย์เราต้องการความสะดวกสบายเพิ่มมากขึ้น ดังนั้นจึงมีหลายๆความคิดที่ จะหาสิ่งที่จะมาอำนวยความสะดวกให้แก่คนเรา หนึ่งในนั้นก็คือหุ่นยนต์ ซึ่งเราได้มีการนำหุ่นยนต์มาใช้ประโยชน์ต่งต่างๆมากมาย ทั้งในด้านอุตสาหกรรม และในชีวิตประจำวันของคนเรา โดยหุ่นยนต์เหล่านี้จะทำงานตามโปรแกรมที่มนุษย์ได้เขียนขึ้น ดังนั้น การที่เราต้องการควบคุมให้หุ่นยนต์ทำตามคำสั่งนั้นเราจำเป็นที่จะต้องเข้าใจถึงโปรแกรมที่ใช้ในการควบคุมหุ่นยนต์นั้นๆ

หลักการทั่วๆไปของหุ่นยนต์ การที่หุ่นนต์จะทำงานได้นั้น หุ่นยนต์จะต้องได้รับข้อมูลหรือคำสั่ง (input) เมื่อหุ่นยนต์ได้รับข้อมูลมาแล้วจะทำการประมวลผลต่อไปตามที่โปรแกรมที่เราได้เขียนขึ้นเพื่อวัตถุประสงค์ต่างๆ ซึ่ง หุ่นยนต์ภายในโครงงานนี้ ก็มีหลักการเดียวกัน คือนำข้อมูลที่ได้นั่นคือ ภาพในมุมมอง 2 มิติ มาทำการประมวลผล เพื่อให้ได้ผลลัพท์ ตามที่โปรแกรมต้องการ

โดยในโครงงานชิ้นนี้ แบ่งส่วนใหญ่ๆออกเป็น 2 ส่วนนั่นคือ การประมวลผลภาพ(image processing) และส่วนของหุ่นยนต์ โดยที่ **การประมวลผลภาพ** คือ เป็นการประยุกต์ใช้งาน การประมวลผลสัญญาณ บนสัญญาณ 2 มิติ เช่น ภาพนิ่ง (ภาพถ่าย) หรือภาพเคลื่อนไหว (วิดีโอ) และยังรวมถึง สัญญาณ 2 มิติ อื่นๆ ที่ไม่ใช่ภาพด้วย โดย กระบวนการนี้จะทำการแปลงข้อมูลที่เป็นรูปภาพให้มาอยู่ในรูปของสัญญาณไฟฟ้า เพื่อที่จะนำสัญญาณฟ้านั้นไปใช้ประโยชน์ ตามวัตถุประสงค์ของเราต่อไป

1.2 วัตถุประสงค์ของโครงงาน

เพื่อทำการศึกษาหลักการประมวลผลภาพ และ นำความรู้ที่ได้ไปใช้ประโยชน์กับหุ่นยนต์ โดยในโครงงานนี้จะใช้ภาษา C++ และภาษา C ของ ไมโครคอนโทรลเลอร์ (MCS-51)

1.3 ขอบเขตของโครงงาน

โครงงานชิ้นนี้ ได้ทำการศึกษาหลักการทำงานของหุ่นยนต์ หลักการประมวลผลภาพ เพื่อนำมาใช้ในการทำงานของตัวหุ่นให้สามารถเคลื่อนไหวได้ตามวัตถุประสงค์ที่เรากำหนด นั่นคือ หุ่นยนต์สามารถเคลื่อนที่เข้าหาลูกบอล และสามารถยิงโดยใช้การประมวลผลจากภาพ ได้อย่างมีประสิทธิภาพ ในบทต่อไปจะกล่าวถึงทฤษฎีที่เกี่ยวข้องกับโครงงานนั้นคือ กระบวนการประมวลผลภาพ

บทที่ 2
ทฤษฎีและหลักการที่ใช้ในโครงงาน

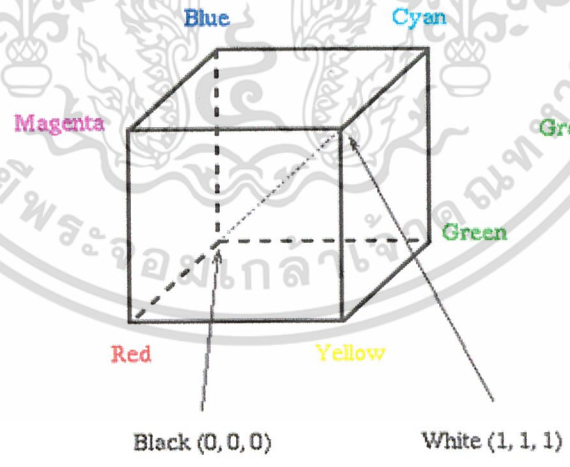
2.1Image Processing

Image Processing แบ่งทฤษฎีออกได้เป็น 2 ส่วนคือ

- RGB color model
- HSI color model

2.1.1 RGB Color model

ใน RGB model นั้น จะประกอบด้วยแม่สีต่างๆที่แยกออกเป็นหลักๆคือ สีแดง สีเขียว และสีน้ำเงิน Model นี้เป็นพื้นฐานของ Cartesian coordinate system สีที่แยกย่อยออกมาดังที่แสดงในรูปที่ 1 ในรูปจะเห็นมุมของลูกบาศก์ 3 มุม ที่ประกอบด้วยสีน้ำเงินเขียว(cyan),สีม่วงแดง (magenta)และสีเหลือง(yellow) ซึ่งอยู่แต่ละมุมของลูกบาศก์ สี ดำอยู่ที่จุดกำเนิด และสีขาวจะอยู่ตรงข้ามสีดำ และ b scale จะเป็นเส้นที่เชื่อมระหว่างจุดสีดำและสีขาว

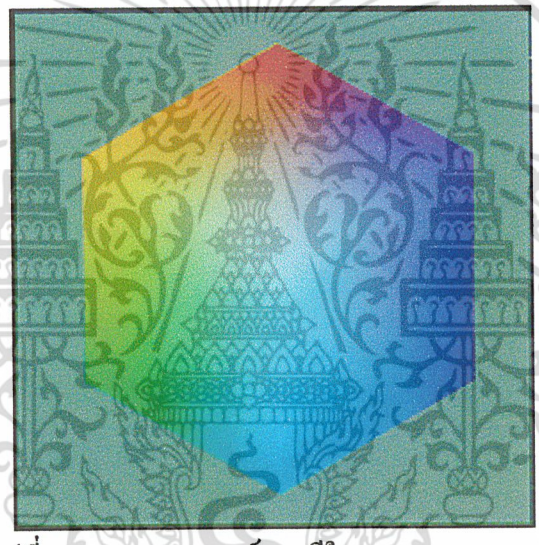


The RGB Cube

รูปที่ 2.1 แสดงลูกบาศก์ของสีใน RGB color Model

สมมุติว่า ค่อของสีถูก normalize ดังรูปที่ 1 ในลูกบาศก์ มันคือค่อทั้งหมดของ RGB ที่ถูกกำหนดให้อยู่ใน rank [0,1]

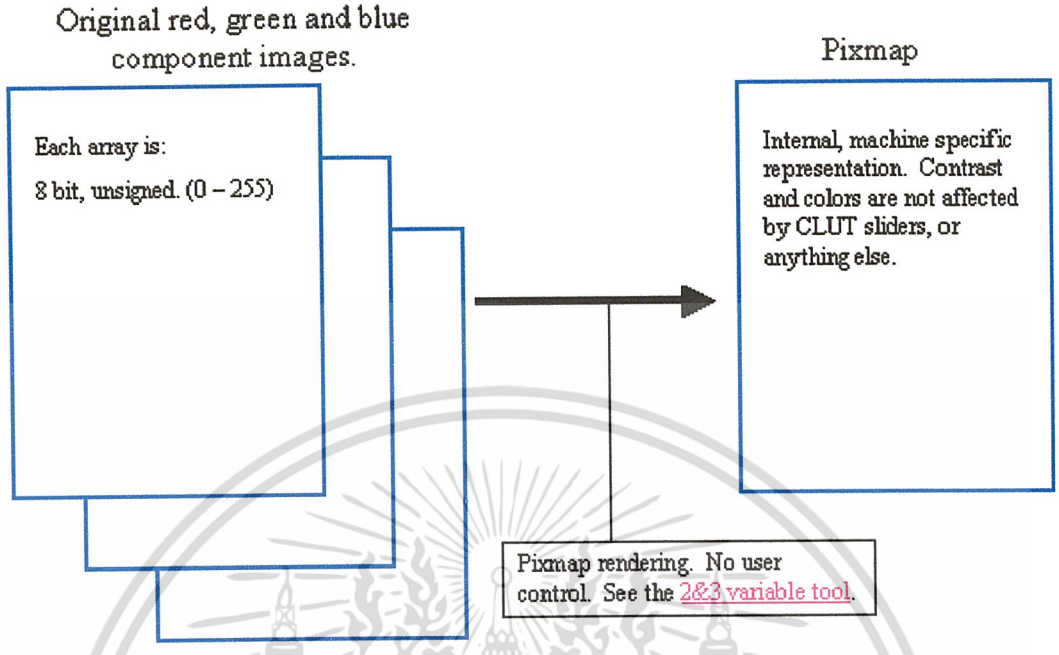
ภาพถูกแสดงใน RGB color model ประกอบด้วย ภาพ 3 ส่วน ซึ่งเป็นส่วนของแม่สีแต่ละสี เมื่อนำมาจัดเพื่อแสดงใน RGB monitor ภาพทั้ง 3 จะรวมกัน บนจอที่เคลือบด้วยสารเรืองแสง จะทำให้เกิดรูปที่สมบูรณ์ขึ้นมา จำนวนของ bit ที่ใช้ในการแสดงแต่ละ pixel ใน RGB เรียกว่า pixel depth พิจารณาภาพ RGB แต่ละแม่สี แดง,เขียวและน้ำเงิน ซึ่งมีค่า 8 bit ต่อ 1 สี ดังนั้นภาพ RGB จึงมีความละเอียดเท่ากับ 24 bit ในโหมดของ full color ภาพจะใช้ความละเอียดที่ 24 bit สีทั้งหมดในโหมด 24 bit นี้มีทั้งหมด 16,777,216 สี รูปที่ 2 แสดง ลูกบาศก์ของ 24 bit RGB color ที่ตรงกับรูปที่ 1



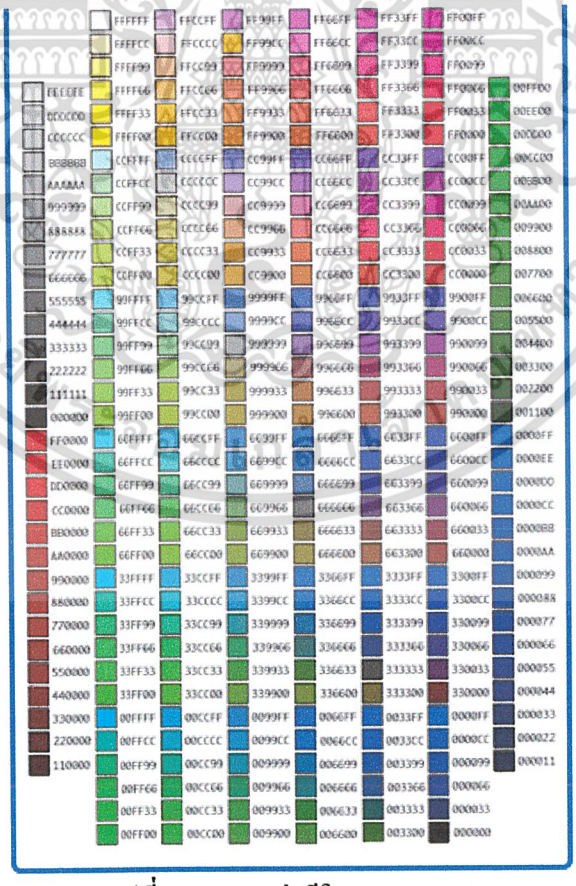
รูปที่ 2.2 แสดงลูกบาศก์ของสีใน RGB color Model

ลูกบาศก์ที่แสดงดังรูป 2.2 เป็นรูปของแข็ง ทางสะดวกที่ดูสี ที่ก่อให้เกิด color plane (พื้นผิวของลูกบาศก์) นี่เป็นทางที่ง่ายและได้ผลคือการกำหนดค่อตายตัวให้กับสีใดสีหนึ่งแล้ว ทำการเปลี่ยนค่าของอีกสองสี ตอนนี้เรากำลังใช้ค่าที่แท้จริงกว่าค่าที่แสดงจากค่าทางคณิตศาสตร์ ซึ่ง normalize ในรูป rank [0,1] เพราะ ค่าแรกถูกใช้ใน computer เพื่อ generate สี รูปที่ 2.3 แสดง รูปของสีแต่ละสี ซึ่งจะเอาแต่ละสีมารวมกันใน color monitor ในส่วนของภาพค่า 0 จะเท่ากับสีดำ และ 255 คือสีขาว (นี่เป็นโหมด gray scale) รูปที่ 2.3 แสดง 3 hidden surface planes ของลูกบาศก์ในรูปที่ 2.2

มันเป็นที่น่าสนใจในเรื่องการได้มาของภาพ ในรูปที่ 2.3 รูปจะได้มาจากการรวมกันของ 3 filter ที่ sensitive ต่อ สีแดง,เขียว และน้ำเงิน ตามลำดับ



รูปที่ 2.3 แสดง 3 hidden surface planes ของลูกบาศก์



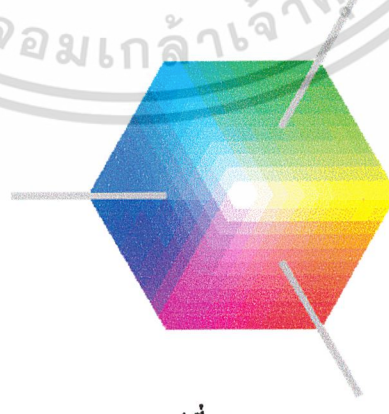
รูปที่ 2.4 แสดงค่าสีในเลขฐาน 16

เมื่อเราใช้ การ์ดจอที่มีคุณภาพดี และ monitor ที่สามารถแสดงภาพ RGB 24 bit ได้อย่างเหมาะสม ระบบหลายๆระบบในปัจจุบันยังแสดงผลในระดับสีเพียง 256 สี แต่ก็ยังมีอีกหลายโปรแกรมที่ต้องใช้สีที่มากกว่าสีเพียงไม่กี่ร้อยสีนั้น และบางครั้งก็น้อยกว่านั้น ตัวอย่างที่ดีของกรณีนี้ คือการแยกสีโดยใช้ Pseudocolor image processing techniques โดยการนำเอาระบบต่างๆ ที่ใช้ตอนนั้นมาพิจารณาถึง subset ของสี ที่คล้ายกันแล้วทำการคัดลอก subset นี้เรียกว่า เซตของ safe RGB color หรือ เซตของ all-system-safe color โปรแกรมใน internet เรียกว่า safe Web color or safe browser color

บนสมมุติฐาน ที่ว่าสี 256 สีนั้นเป็นจำนวนที่น้อยเกินไปที่จะใช้ในการสร้างสีใหม่โดยเคร่งครัด โดยระบบต่างๆ ที่จะใช้สีที่มีค่าใกล้เคียงกัน มันเป็นมาตรฐานที่ใช้อย่างกว้างขวางในการใช้สี สี 40 สีจาก 256 สี จะเป็นที่รู้ว่า จะมีกระบวนการที่แตกต่างกันในแต่ละระบบปฏิบัติการ และสีที่เหลืออีก 216 สีจะเป็นสีที่ใช้ในระบบทั่วไปได้

แต่ละสีใน 216 safe color เป็นรูปแบบจาก สีหลัก 3 สีของ RGB แต่ แต่ละค่าจะมีค่าแค่ 0,51,102,153,204 หรือ 255 RGB triplet จะมีค่า 6^3 = 216 ค่าที่เป็นไปได้ มันเป็นค่าที่อยู่ในเลขฐาน 16 เรียกใหม่ว่าเป็นเลข hex number 0,1,2,3,4,5,6,7,8,9,A,B,C,..,F ซึ่งเท่ากับ 1,2,3,4,5,6,7,8,9,10,11,..,15 หรือจะใช้ (FF)16 = (255)10 = (11111111)2 และเราจะเห็นว่ากลุ่มของเลขฐาน 16 จะอยู่ในรูป เลข 8 bit

รูปที่ 2.4 แสดง 216 safe color โดยได้จากค่าของ RGB สี่เหลี่ยมบนซ้าย มีค่า FFFFFF (white) อันที่สอง ด้านขวา มีค่าเท่ากับ FFFFCC และอันที่สามมีค่า FFFF99 และสี่เหลี่ยมด้านล่างมีค่า FFCCFF,FFCCCC,FFCC99 ตามลำดับ



รูปที่ 2.5

รูปที่ 2.5 แสดงลูกบาศก์ของ RGB safe color ซึ่งไม่เหมือนกับ ลูกบาศก์ของ full color ในรูปที่ 2 ซึ่งเป็นของแข็ง ลูกบาศก์ในรูปที่ 5 เป็นสีที่มีใน surface plane ดังที่แสดงใน รูป 5 แต่ละ plane มีทั้งหมด 36 สี แตในลูกบาศก์นี้จะมีทั้งหมด 216 สี ที่แตกต่างกัน

2.1.2 HSI color model

2.1.2.1ระบบสี HSI

สิ่งที่จำเป็นสำหรับการแสดงผลความคลุมเครืออย่างแรกคือการเลือกระบบการแสดงผลสีเชิงจิตวิทยา เช่นการให้ระบุว่าสีใดมีความขาวที่สุด หรือจางที่สุด ซึ่งได้รับการยืนยันจากตัวอย่างของ เจียง (1996) ที่ได้ทดสอบการรับสัมผัสทางสี โปรแกรมทาง GIS ได้ใช้สีที่มีค่า Saturation และ Intensity ที่คงที่ เรียกว่าสีสมมติ (pseudo-colour) เพื่อการแสดงผลการพยากรณ์ที่ถูกต้อง ในระบบ IL WIS นั่นคือ ระบบสีแบบ HSI เป็นระบบสีที่ส่วนใหญ่เลือกใช้ในการแสดงผลความคลุมเครือ นอกจากนี้ระบบสีนี้มีส่วนสามารถนำเข้ามาแปลงกับระบบสี RGB ได้ ซึ่งนำมาใช้คำนวณรงควัตถุและสีประสมได้

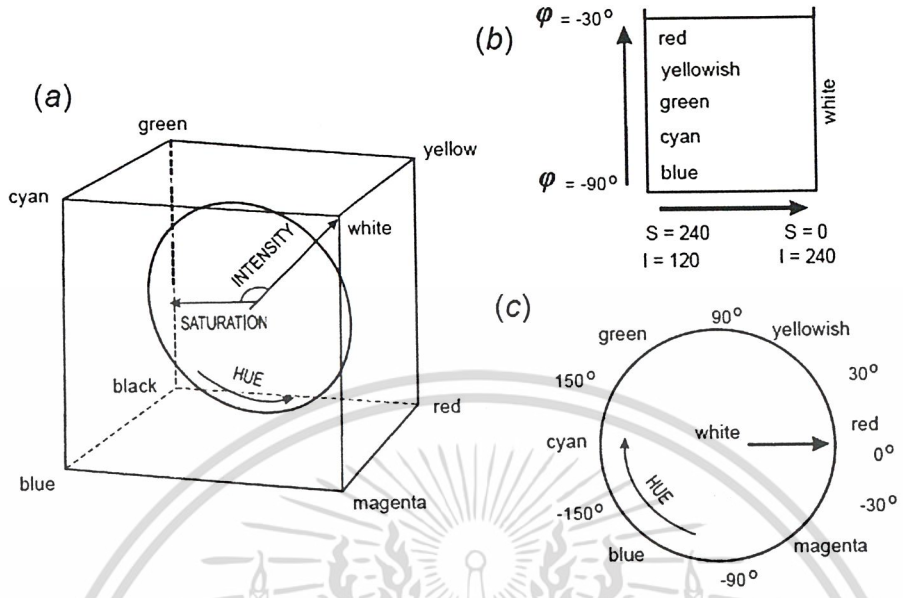
สูตรการแปลงระบบสี HSI และ RGB นั้นแตกต่างกันขึ้นอยู่กับผู้ผลิตซอฟท์แวร์แต่ละราย วิธีของ IL WIS นั้นมีรากฐานมาจากการออกแบบทางเรขาคณิตของลูกบาศ์กสีระบบ RGB ดังภาพที่ 1ก

$$H = \frac{360}{2\pi} \cdot \arctan\left(\frac{\sqrt{3}}{2} \cdot [G - B], R - \frac{[G + B]}{2}\right) \cdot \frac{240}{360} \quad (1)$$

$$S = \sqrt{R^2 + G^2 + B^2 - R \cdot G - R \cdot B - G \cdot B} \cdot \frac{240}{255} \quad (2)$$

$$I = \frac{R + G + B}{3} \cdot \frac{240}{255} \quad (3)$$

การรับค่าแดง (R) เขียว (G) และน้ำเงิน (B) นั้นไล่ระดับจาก 0 ถึง 255 ในขณะที่ H S I นั้นไล่ระดับจาก 0 ถึง 240 เพื่อให้สอดคล้องกับผังสีที่ใช้ในระบบปฏิบัติการไมโครซอฟท์วินโดว์



รูปที่ 2.6 ทฤษฎีสีของ HSI Color Model

2.1.2.2 การออกแบบตัวแสดง

สิ่งที่สำคัญในการแสดงผลความคลุมเครือต่อมาคือการออกแบบตัวแสดงที่เหมาะสมและสอดคล้องกับระบบสี CM ในครั้งนี้ได้มาการออกแบบตัวแสดงสีอยู่สองประเภท ได้แก่

1. ตัวแสดงแบบสองมิติ (Two-dimensional legend) ตัวแสดงนี้ใช้กับตัวแปรต่อเนื่อง ในที่นี้แกนตั้งแสดงค่าที่ทำนายและแกนนอน (ความขาว)ใช้แสดงการพยากรณ์ความคาดเคลื่อน ในแกนตั้ง ไล่ระดับสี (hue) ระดับต่ำสุดที่สีน้ำเงิน (-90°) ไปจนถึงระดับสูงสุดที่สีแดง (-30°) ซึ่งสอดคล้องกับสเกลสีสมมติที่ใช้กันแพร่หลายในชุด GIS สำหรับการแสดงผลค่าตัวแปรต่อเนื่อง (Unit Geo Software Development, 2001) ในแกนนอน การเปลี่ยน saturation-intensity เป็นเชิงเส้นจากความขาวต่ำสุด ไปยังความขาวสูงสุด ดังภาพที่ 1b ในกรณีนี้สีเต็มหมายถึงความคลุมเครือต่ำสุด และความขาวสูงสุดหมายถึงความคลุมเครือสูงสุด เพื่อไม่ให้เกิดความสับสนระหว่างค่าสูงและต่ำจึงได้นะส่วนของวงกลมสีที่แสดงสีบานเย็น (magenta -30° ถึง -90°) ออกไป

2. วงล้อสี (Colourwheel) ตัวแสดงนี้ใช้เพื่อให้สอดคล้องกับผลของการจำแนกความต่อเนื่อง โดยใช้ตัวแสดงแบบวงกลม (HSI Colourwheel) (Niblack 1986, pp. 61) เส้นรอบวงของสี (hue) แสดงถึงพื้นที่จำแนก ระยะทางของรัศมีแสดงค่าความสับสน ดังภาพที่ 1c .ในกรณีนี้ระดับที่ใกล้เคียงกับวงล้อสีนั้นคล้ายคลึงกับการจำแนก ส่วนที่จุดศูนย์กลางของวงกลมจะเริ่มขาวขึ้น นั่นคือสี (hue) เริ่มไม่เด่นชัด แสดงถึงสถานะที่มีความคลุมเครือขององค์ประกอบสีทั้งหมด (Hengl et al., in press-a)

ตัวแสดงที่ได้ออกแบบดังกล่าวนี้ สามารถนำไปใช้ในการแบ่งช่วงต่างๆและการแบ่งระดับที่ต่างๆกันได้

2.1.2.3 ตัวแปรต่อเนื่อง

ในกรณีของตัวแปรต่อเนื่อง เราสนใจการแสดงผลทั้งการทำนายผลและความคลาดเคลื่อนของการทำนายพร้อมๆกัน ความคลุมเครือแสดงด้วยค่าความเบี่ยงเบนมาตรฐานจากความคลาดเคลื่อนของการทำนาย (prediction error) ซึ่งนำไปจับเทียบกับสิ่งที่กำลังศึกษาอยู่ การหาผลเฉลยทั่วไปนั้นกระทำโดยแบ่งการพยากรณ์ของตัวแปรที่แปลงค่าโดย total variance ของตัวอย่างที่สำรวจ ซึ่งเป็นการความคลาดเคลื่อนจากการทำนายที่ normalize หรือ มีความสัมพันธ์กัน โดยแสดงผลเป็นค่าร้อยละ (Hengl et al., in press-b)

$$\sigma_{E,r}(s_o) = \frac{\sigma_E(s_o)}{s_z} \qquad (4)$$

โดยที่ $\sigma_E(s_o)$ เป็นค่าความคลาดเคลื่อนจากการทำนาย และ s เป็นค่าเบี่ยงเบนมาตรฐานของการสำรวจตัวอย่าง การทำนายได้ค่าพึงพอใจเมื่อแบบจำลองแสดงให้เห็น total variation มากกว่าร้อยละ85 จาก rule of thumb เราสามารถพิจารณาค่า $\sigma_{E,r}(s_o)$ ให้เข้าใกล้ร้อยละ 40 ซึ่งเป็นค่าที่ถูกต้องพึงพอใจในการทำนาย ในกรณีอื่นๆ ถ้ามีค่าเกินร้อยละ 80 แบบจำลองจะมีความหลากหลายน้อยกว่าร้อยละ 50 ที่จุดบ่งชี้ ซึ่งการทำนายจะไม่เป็นที่พึงพอใจ

ผังการทำนายและความคลาดเคลื่อนจากการทำนายสามารถแสดงผลได้พร้อมๆกัน โดยแสดงค่าการทำนายด้วยสี (hue) และความคลุมเครือด้วยความขาว (whiteness) ในขั้นแรกการทำนายถูกแปลงเป็นสีดังสมการ

$$\varphi_1 = -90 + z_r \cdot 300 \qquad (5)$$

$$\varphi_2 = \begin{cases} \varphi_1 + 360 & if \quad \varphi_1 \le -360 \\ \varphi_1 & if \quad \varphi_1 > -360 \end{cases} \qquad (6)$$

เมื่อ f_1 เป็นมุมสีหน่วยเป็นองศาตามเข็มนาฬิกา f_2 เป็นค่าที่แปลงไปยังช่วง -360 ถึง 360 และ z_r เป็นค่าทำนาย $(z_r \in [0,1])$ ภาพในระบบ HSI แปลงได้จากสมการ

$$H = (\varphi_2 + 360) \cdot \frac{240}{360} \tag{7}$$

$$S = (1 - u_r) \cdot 240 \tag{8}$$

$$I = (1 + u_r) \cdot 120 \tag{9}$$

โดย u_r เป็นค่าการทำนายความคลุมเครือ $(u_r \in [0,1])$ ซึ่งค่าสำหรับการทำนายและความคลาดเคลื่อนจากการทำนายต้องแผ่กระจายก่อนคำนวณโดยสมการ

$$z_r = \frac{\hat{z} - z_1}{z_2 - z_1}; \quad u_r = \frac{\sigma_{E,r} - u_1}{u_2 - u_1} \tag{10}$$

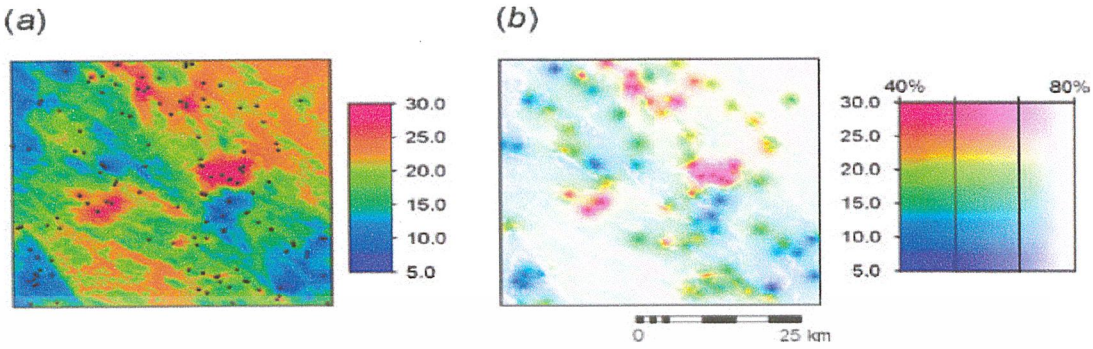
โดย z เป็นผังการทำนาย $\sigma_{E,r}$ เป็นผังความคลาดเคลื่อนการทำนายสัมพันธ์ ทำให้เป็นมาตรฐานเดียวกันโดยใช้สมการที่ 4 ค่า z_1 และ z_2, u_1 และ u_2 เป็นขอบเขตบนและขอบเขตล่างของการ

ตรวจสอบค่าทำนายและความคลาดเคลื่อนการทำนายสัมพันธ์ จากค่าตัวแปรที่ทำนายขอบเขตนั้นจำกัดอยู่ที่ค่าต่ำสุดและค่าสูงสุด สำหรับความคลุมเครือ ค่าแบ่งช่วงที่นำมาใช้อยู่ที่ร้อยละ 40 ถึง 80

จากภาพระบบ HSI สามารถแปลงเป็นภาพระบบ RGB โดยใช้คำสั่งใน IL WIS

$$z_{RGB} = colorhsi[H, S, I] \tag{11}$$

จากสมการที่ 5 ถึง 7 ค่าที่ต่ำกว่าอันได้แก่ค่า H ตั้งแต่ -90° ถึง -150° นั้นแปลงเป็นสีโทนน้ำเงิน ส่วนค่าที่สูงสุดอันได้แก่ค่า H ตั้งแต่ -330° ถึง -30° นั้นแปลงเป็นสีโทนแดง ค่ากลางนั้นแปลงเป็นสีโทนน้ำทะเล เขียว และเหลือง



รูปที่ 2.7แสดงผลภาพแบบรวมกันแสดงถึงความสัมพันธ์ระหว่างความคลุมเครือและข้อมูลที่ได้รับ

การแสดงผลภาพแบบรวมกันแสดงถึงความสัมพันธ์ระหว่างความคลุมเครือและข้อมูลที่ได้รับมาทั้งหมดเพื่อใช้ในการแบ่งช่วง ในตัวอย่างดังภาพที่ 2b การเปรียบเทียบอัตราส่วนของภาพความหนาพื้นผิวดินนั้นทำให้เป็นค่าเชิงปริมาณด้วยความขาว ค่าความสว่างที่ปรับปรุงให้ถูกต้องได้แก่ 1. มีค่าเท่ากับค่า RGB ต้นแบบสำหรับความคลุมเครือสัมพันธ์ที่น้อยกว่าหรือเท่ากับร้อยละ 40 และ 2. สีขาวบริสุทธิ์สำหรับความคลุมเครือสัมพันธ์ที่มีค่ามากกว่าหรือเท่ากับร้อยละ 80 การแสดงผลส่วนใหญ่ที่ห่างจากจุดภาพหลักจะเป็นซีซีดจางลง ซึ่งพื้นที่เหล่านี้อาจจะต้อการข้อมูลเพิ่มเติมเพื่อให้เหมาะแก่การพยากรณ์ ซึ่งสามารถเปิดดูข้อมูลเพิ่มเติมได้ที่ http://www.itc.nl/personal/hengl/GRK/

2.1.2.4 แผนที่แบบแบ่งจำแนก (Categorical maps)

แผนที่แบบแบ่งจำแนกนั้นมีพื้นฐานมาจากการสั่งสมข้อมูลประกอบการคำนวณและจำแนกข้อมูลบนแผนที่ เช่นแผนที่ของพื้นที่ปกคลุมสามารถหาได้โดยการสั่งสมข้อมูลประกอบการคำนวณและจำแนกภาพถ่ายทางดาวเทียมแบบหลายสเปกตรัม ซึ่งปัจจุบันนี้ มีการใช้ระบบการจำแนกแบบต่อเนื่องและสร้างการพยากรณ์เชิงปริมาณกันมาก แผนที่ที่มีค่าสมาชิกมักได้มาจากการใช้การจำแนกแบบ fuzzy k-means (Burrough et al., 1977) ค่าสมาชิกนั้นคำนวณจากระยะทางมาตรฐานในสมบัติของพื้นที่

$$\mu_c(i) = \frac{\left[d_c^2(i)\right]^{-\frac{1}{(q-1)}}}{\sum_{c=1}^{k}\left[d_c^2(i)\right]^{-\frac{1}{(q-1)}}}; \quad c = 1,2,\mathrm{K}\ k; \quad i = 1,2,\mathrm{K}\ n \tag{12}$$

$$\mu_i(c) \in [0,1] \tag{13}$$

โดย $\mu_c(i)$ เป็นค่าสมาชิกของฟัซซี่ ตั้งแต่วัตถุลำดับที่ i ในคลัสเตอร์ลำดับที่ c, d เป็นระยะทาง, k เป็นจำนวนของคลัสเตอร์ และ q เป็น fuzzy exponent ที่ใช้พิจารณาจำนวนความฟัซซี่

ในกรณีนี้ มีเป้าหมายเพื่อแสดงผลสมาชิกหลายตัวและความสับสนระหว่างกัน ซึ่งกระทำได้โดยการผสมสีลงไปในแต่ละจุดภาพ (i) เช่นเดียวกันกับการ average ระดับเทาของระบบสี RGB การได้ผังสี R_i G_i B_i ใหม่ ณ จุด i แต่ละจุดนั้นทำการคำนวณแยกเป็นอิสระต่อกันแนแต่ละแบนด์ของภาพ

$$R_i = \frac{\sum_{c=1}^{k}\left(\mu_{i,c} \cdot R_c\right)}{\sum_{i=1}^{n}\mu_{i,c}} \tag{14}$$

$$G_i = \frac{\sum_{c=1}^{k}\left(\mu_{i,c} \cdot G_c\right)}{\sum_{i=1}^{n}\mu_{i,c}} \tag{15}$$

$$B_i = \frac{\sum_{c=1}^{k}\left(\mu_{i,c} \cdot B_c\right)}{\sum_{i=1}^{n}\mu_{i,c}} \tag{16}$$

โดยค่า R_i G_i B_i เป็นค่าสีที่ได้ใหม่ ค่า R_c G_c B_c ค่าอยู่ในช่วง 0 ถึง 255 สำหรับแต่ละระดับสี และค่า k เป็นจำนวนของระดับ ระบบสี RGB นั้นสามารถแสดงผลเช่นเดียวกับการรวมสี

ก่อนที่จะทำการคำนวณสีประสม จำเป็นที่จะต้องเลือกประเภทสีที่จะแสดงถึงการจำแนกของแต่ละระดับ ปัญหาคือคุณสมบัติมีเป็นจำนวนมาก ดังนั้นการวางตำแหน่งของระดับกึ่งกลางที่วงล้อสีนั้นจึงเป็นสิ่งที่ลำบาก วิธีการหาค่าวิธีหนึ่งคือการใช้การถ่วงค่าแรกและค่าที่สองของการวิเคราะห์ส่วนประกอบหลักเช่นเดียวกับแกนของพื้นที่ที่ลดทอนสมบัติลงไป เรียกว่า biplot display (Gabriel, 1971) ค่าสี H ของแต่ละระดับชั้นนั้นเป็นค่าระยะมุมรอบจุดศูนย์ถ่วง ในกรณีของค่าถ่วงคือจุดศูนย์กลางของพิกัด

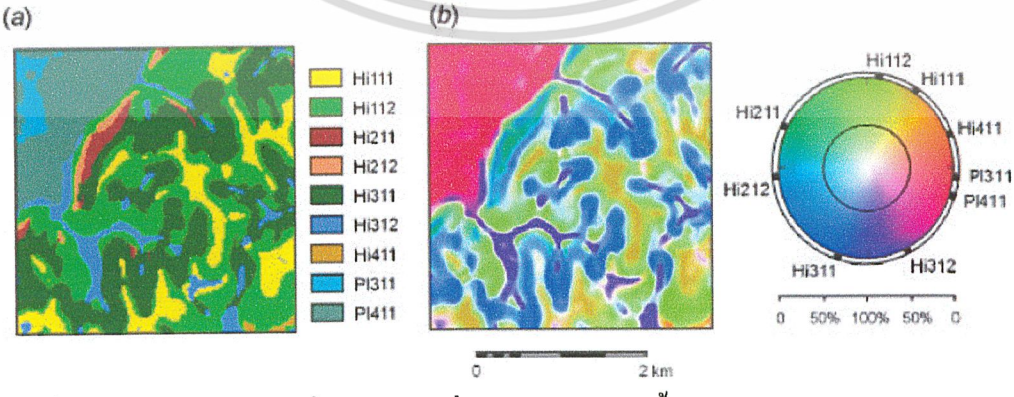
$$H_c = \frac{360}{2\pi} \cdot \arctan(F_{1c}, F_{2c}) \cdot \frac{240}{360} \qquad (17)$$

โดย H_c เป็นค่า H ระดับใหม่ที่ได้มา และ F_{1c} และ F_{2c} เป็นปัจจัยถ่วงในระดับชั้นที่ c ในการแสดงระดับชั้นสีในระบบสารสนเทศทางภูมิศาสตร์ ค่า H_c S_c I_c (ที่ S_c=240 และ I_c=120) ต้องแปลงค่าเป็นระบบสี RGB ก่อนจากนั้นจึงจะสามารถคำนวณค่าการประสมสี (CM) ด้วยสมการที่ 14 ถึงสมการที่ 16 ความสับสนระหว่างระดับชั้นนั้นแก้ไขโดยใช้ความขาว นั่นคือจำนวนจุดภาพสีขาวโดยการแทนด้วยค่าความสว่างคงที่ด้วยค่า Saturation S

$$R_{i*}, G_{i*}, B_{i*} = colorhsi[H_i, S_i, (120 + 0.5 \cdot S_i)] \qquad (18)$$

โดยที่ค่า H_i, S_i เป็นค่า Hue และ Saturation ที่ได้มาจากจากผังสี CM และ ค่า R_{i*}, G_{i*}, B_{i*} เป็นค่าจากผัง RGB ที่ได้รับการปรับปรุงแล้ว ทั้งค่า Saturation และความสว่างเป็นอย่างเป็นมุมรัศมี เช่นความสว่างเปลี่ยนจาก 120 (เส้นรอบรูป) ไปยังความสว่างสูงสุด (255 หรือจุดศูนย์กลางของวงล้อสี) วิธีในการออกแบบนั้นได้กล่าวไว้ใน (Hengl et al., in press-a)

ตัวอย่างในภาพที่ 3 แสดงการแสดงผลการจำแนกผิวหน้ารูปทรงพื้นดินโดยใช้การจำแนกแบบ fuzzy k-means ค่าของสีนั้นพิจารณาเพื่อแสดงความสมพันธ์ในการจำแนกระหว่างระดับชั้นพิกัดของระดับชั้นกลางทั้ง 9 ระดับ (ผิวหน้ารูปทรงพื้นดิน) ได้แปลงค่าจากหลายตัวแปรไปสู่ปริภูมิสองมิติ และได้เทียบเคียงไปบนวงล้อ HSI (Hengl et al., in press-a) ระดับชั้นที่อยู่ใกล้กับปริภูมิจะนำไปรวมกับสีทั่วๆไป เทคนิคการรวมสีจำกัดการหาค่าผังสี CM ไปสู่ จำนวนระดับชั้นสี Hue ทั้งหมด 7 สีแยกกันอย่างอิสระซึ่งเป็นรากฐานสำหรับการตีความอัตโนมัติต่อไป



รูปที่ 2.8 ผังสี CM ดังภาพที่ 3b แสดงสีที่อยู่ระหว่างระดับชั้นเดียวกัน เช่นบริเวณรอบเส้นรอบรูป

แสดงถึงการคาบเกี่ยวในการจำแนกแล้วจำแนกสีให้ ในทางกลับกัน สีที่อยู่ระหว่างชั้นที่ตัดกันอย่างเห็นได้ชัดมีสีไปถึงที่จุดศูนย์กลางของวงกลมนั้นคือเป็นสีโทนขาว เช่นจุดภาพระหว่างเนินกลว้างและชัน (Hi211) และระดับที่ราบทั้งสองจุด (Pl311 และ Pl411) มีลักษณะเป็นสีโทนขาว ซึ่งเป็นการมองทางจิตวิทยา สีขาวบริสุทธิ์แสดงถึงการประสมการตัดกันอย่างเห็นได้ชัดหรือระดับชั้นที่ไม่สามารถนิยามได้ ขณะที่ระดับชั้นที่มีการประสมที่เหมือนๆกันอยู่ในระดับที่ลดหลั่นกันไป

เมื่อเปรียบเทียบกับผังสี CM ที่มีทางลาดขึ้นสองจุด ขณะที่การเลือกสีเป็นแบบตามความรู้สึก ทำให้บางครั้งเกิดความคิดที่ผิดไปเกี่ยวกับความเหมือนกันระหว่างระดับ ดังภาพที่ 3a ในทางกลับกัน ในกรณีของผังสี ระดับชั้นจำแนกนั้นไม่ได้เรียงลำดับลดหลั่นเป็นสัดส่วนกันไปตามแนวคิดทั่วไป เนื่องจากวงกลมไม่มีจุดเริ่มต้นดังนั้นการบ่งชี้ระดับชั้นจึงไม่เป็นลำดับ เช่นเดียวกันกับบางระดับชั้นสามารถวางจุดให้ใกล้ชิดหรือห่างออกจากกันในเส้นรอบรูปได้

Saturation ที่ได้มาจากผังสี CM สามารถแยกแยะอาณาเขตการเปลี่ยนแปลงขั้นปฐมภูมิระหว่างระดับชั้นที่แตกต่างกันพอประมาณ ยิ่งไปกว่านั้น ยังสามารถวางตำแหน่งพื้นที่ที่มีความสับสนในการจำแนก เช่นการออกแบบเพิ่มเติมหรือการเก็บข้อมูลเพิ่มเติมให้ละเอียดขึ้นบนพื้นที่

2.2 ทฤษฎีไมโครคอนโทรลเลอร์ MCS-51

2.2.1 แนะนำไมโครคอนโทรลเลอร์

ก่อนที่จะทำความรู้จักกับไมโครคอนโทรลเลอร์ เราควรจะทำความรู้จักกับที่มาที่ไปของชิปตัวนี้ เพื่อความเข้าใจที่ดีในการกล่าวถึงเกี่ยวกับไมโครคอนโทรลเลอร์กันซะก่อนเดิมทีเดียวนั้น ในงานควบคุมระบบต่างๆ ได้นำไมโครโปรเซสเซอร์หรือไมโครคอมพิวเตอร์เข้ามาใช้ในงานควบคุม โดยในการใช้งานนั้นจุต้องมีการต่อร่วมกับอุปกรณ์ภายนอก จำพวกพอร์ต I/O หน่วยความจำข้อมูล (RAM) หน่วยความจำโปรแกรม (ROM) ทั้งยังมีชิปจำพวก UART (Universal Asychronous Receiver Transmitter) เพื่อให้ในการรับส่งข้อมูลแบบอนุกรม โดยมีหน้าที่เปลี่ยนข้อมูลจากรูปแบบขนาน (Parallel) ไปเป็นข้อมูลในรูปแบบอนุกรม (Series) สำหรับทำการส่งชุดข้อมูล และแปลงข้อมูลจากรูปแบบอนุกรมกลับเป็นรูปแบบขนานในขั้นตอนการรับข้อมูล ซึ่งจะเห็นว่ากว่าจะนำไมโครโปรเซสเซอร์ หรือไมโครคอมพิวเตอร์ดังกล่าวมาใช้ในระบบควบคุมโดยเฉพาะนั้น จะค่อนข้างยุ่งยากสิ้นเปลืองอุปกรณ์ร่วมต่างๆ เพื่อที่จะให้คลอบคลุมการใช้งานควบคุมที่มีประสิทธิภาพ ฉะนั้นจากปัญหาต่างๆที่ผ่านมา จึงทำให้มีบริษัทผู้ผลิตชิปไมโครโปรเซสเซอร์หรือไมโครคอมพิวเตอร์ดังกล่าวได้ทำการพัฒนา และผลิตชิปที่มีการรวมคุณสมบัติต่าง ๆ เอาไว้ครบเพื่องานควบคุมโดยเฉพาะ

โดยในชิปจะประกอบไปด้วย ไมโครโปรเซสเซอร์,พอร์ต, I/O,หน่วยความจำข้อมูลขนาดเกล็ก ตลอดจนอาจจะมีพอร์ตใช้งานในการรับส่งข้อมูล (UART) และหน่วยความจำโปรแกรม (ROM) เข้าไปด้วยภายในชิปเพื่อใช้ในระบบควบคุมอย่างเต็มประสิทธิภาพสามารถประยุกต์ใช้งานง่ายและประหยัดค่าใช้จ่ายเกี่ยวกับอุปกรณ์เสริมต่างๆ ชิปที่รวมคุณสมบัติต่างๆดังที่เกล่าวมาข้างต้นจึงถูกเรียกว่าไมโครคอนโทรลเลอร์ชิปเดี่ยว(Microcontroller Unit; MCU)นั่นเอง โดยบริษัทผู้ผลิตก็ได้ผลิต MCU ออกมาหลายรูปแบบ ซึ่งจะมีคุณสมบัติและความสามารถที่แตกต่างกันเพื่อให้เหมาะสมกับงานในรูปแบบต่างๆ จึงทำให้มีการแยกประเภทของ MCU ออกเป็นตระกูลๆ หรือเบอร์ต่างๆ โดยขึ้นอยู่กับคุณสมบัติของ MCU นั้นและบริษัทผู้ผลิตเอง

โดยตระกูลที่ได้รับความนิยมและเป็นที่รู้จักกันดีในหมู่นิสิตนักศึกษา ตลอดทั้งนักอิเล็คทรอนิกส์สมัครเล่นและมืออาชีพก็คือไมโครคอนโทรลเลอร์ MCS-51 เนื่องจากมีข้อมูลหรือหนังสือที่เกี่ยวกับ MCU ตระกูลนี้อยู่พอสมควร

โดยนำเสนอข้อมูลการประยุกต์ใช้งาน MCU ตระกูลนี้เช่นกัน โดยเนื้อหาจะกล่าวถึงการใช้งานทางด้านฮาร์ดแวร์เป็นหลัก ต่อไปจะเป็นการกล่าวถึง สถาปัตยกรรมและคุณสมบัติของ MCS-51 เพื่อเป็นพื้นฐานในการทำความรู้จักและเข้าใจในระดับต้นดังนี้

2.2.2 คุณสมบัติของ MCS-51

1. ใช้เทคโนโลยีชั้นสูงในการสร้าง โดยมีทั้งประเภท HMOS, CMOS และ CHMOS ทำงานด้วยแหล่งจ่ายไฟ+5 Vdc เพียงแหล่งเดียว

2. มีหน่วยประมวลผลขนาด 8 บิต

3. สามารถติดต่อกับหน่วยความจำภายนอกทั้งหน่วยความจำโปรแกรม และหน่วยความจำข้อมูลได้สูงสุด 64 Kbyte

4. มีพอร์ต I/O แบบขนานสองทิศทางจำนวน 4 พอร์ต พอร์ตละ 8 บิต รวมทั้งหมดเป็น 32 บิต จะใช้ในการเข้าถึงแอดเดรสและข้อมูลสำหรับติดต่อกับหน่วยความจำภายนอก

5. พอร์ตใช้งานทุกพอร์ตและมีลักษณะเป็นพอร์ตแล็ช (Latch) คงสภาวะ

6. มีขาพอร์ตที่ใช้สำหรับการรับส่งข้อมูลแบบอนุกรม

7. หนึ่ง Machine Cycle จะใช้เวลา 1 ไมโครวินาที โดยใช้ X-TAL 12 MH_z

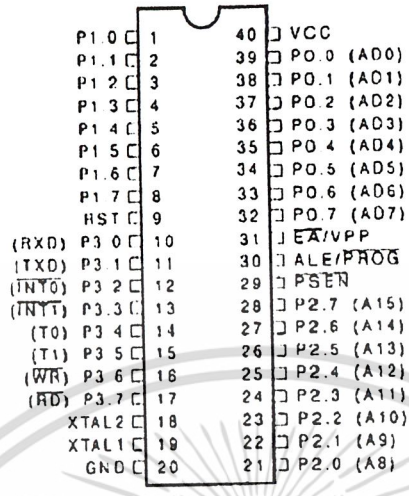
8. สามารถกำหนดการใช้งานพอร์ต อนุกรม I/O ได้ในระดับไบต์หรือบิตได้โดยตรง

9. ตัวเลขทางคณิตศาสตร์ใช้ได้ทั้งระบบฐานสอง และฐานสิบหก

ตระกูลของ MCS-51 จะมีทั้งแบบมี EPROM ในตัวหรือไม่มี EPROM ภายใน (ในการกล่าวถึงหน่วยความจำโปรกแกรมต่อไปนั้นผู้เขียนจะกล่าวถึง EPROM เป็นหลัก เนื่องจากเป็นชนิดประเภทของหน่วยความจำโปรแกรมที่สามารถนำมาประยุกต์ใช้งาน และเป็นที่รู้จักกันดี โดยขอให้ผู้อ่านเข้าใจว่าหน่วยความจำโปรแกรม EEPROM ได้เช่นกัน ข้อแตกต่างหรือรายละเอียดนั้นจะกล่าวไว้ในบทที่ 2) ซึ่งก็จะมีตำแหน่งขาที่เหมือนกัน ตารางที่ 1-1 แสดงถึงรายละเอียดของไมโครคอนโทรลเลอร์ตระกูล MCS-51 เบอร์ 89XX ของบริษัท ATMEL และเบอร์ 80C31,80C32 ของบริษัท INTEL

2.2.3 ลักษณะการจัดขาภายนอกของ MCS-51

รูปที่ 1-1 แสดงการจัดขาตามลักษณะภายนอกของชิป MCS-51 ซึ่งจะมีการแบ่งกลุ่ม จัดขาของ MCS-51 มีอยู่ 4 กลุ่ม คือ 1. กลุ่มขาแหล่งจ่ายไฟเลี้ยง และสัญญาณนาฬิกา 2. กลุ่มขาสำหรับอ้างแอดเดรสและรับส่งข้อมูล 3. กลุ่มขาที่ใช้ในการควบคุม 4. กลุ่มขาพอร์ตใช้งานแบบขนานและอนุกรมพอร์ตใช้งานบางพอร์ตจะทำหน้าที่ได้



รูปที่ 2.9 กลุ่มขาพอร์ตแบบขนาน I/O

รายละเอียดการทำงานตลอดจนโครงสร้างภายใน Microcontroller MCS-51 สามารถศึกษาได้จากคู่มือหรือ Data Sheet การใช้งาน Microcontroller MCS-51 โดยในที่นี้จะแนะนำรายละเอียดและคุณสมบัติในเบื้องต้นสำหรับนำมาประยุกต์ใช้งานต่อไป

2.2.4 ขาที่สำคัญของไมโครคอนโทรลเลอร์ MCS-51

1. ขา VCC เป็นขารับแรงดันไฟกระแสตรง +5 Vdc

2. ขา GND เป็นขากราวด์

3. พอร์ต 0 (Port 0) มี 8 บิต ได้แก่บิต P0.0 – P0.7 เป็นพอร์ตอินพุตเอาต์พุตแบบ 2 ทิศทาง สำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็น ตารางที่ 1-1 แสดงรายละเอียดของไมโครคอนโทรลเลอร์ตระกูล MCS-51

Features	89C1051	89C2051	80C31	80C32	89C51	89C52	89LV51	89LV52	89C55	89S8252
Flash Memore	1K	2K	-	-	4K	8K	4K	8K	20K	8K
Ram	64	128	128	256	128	256	128	256	256	256
EEPROM	-	-	-	-	-	-	-	-	-	2K
Programming	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Timers/Counters	1	2	2	3	2	3	2	3	3	4
Serial UART	1	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Power supply	2.7-6.0	2.7-6.0	4.0-6.0	4.0-6.0	4.0-6.0	4.0-6.0	2.7-6.0	2.7-6.0	2.7-6.0	2.7-6.0
Frequency	0-24	0-24	0-24	0-24	0-24	0-24	0-24	0-24	0-33	0-33
I/O Pins	15	15	32	32	32	32	32	32	32	32
External address/ Data bus	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Pin Count	20	20	40/44	40/44	40/44	40/44	40/44	40/44	40/44	40/44

รูปที่ 2.10 คุณสมบัติต่างๆของ Microcontroller

อินพุตพอร์ตต้องทำการเซ็ตค่า 1 ไปยังพอร์ตเมื่อต้องการใช้งานพอร์ตนั้นทั้งพอร์ตเป็นอินพุตในระดับบิต ก็สามารถกระทำได้โดยการเซ็ตค่า 1 ไปยังแต่ละบิตที่ต้องการใช้งานเป็นพอร์ตอินพุตในระดับ เพื่อกำหนดให้ขาพอร์ตหรือแต่ละบิตเหล่านั้นอยู่ในสถานะปล่อยลอย ซึ่งในสถานะนี้เองที่นำมาใช้เป็นพอร์ตอินพุตอิมพีแดนซ์สูงได้ นอกจากพอร์ตนี้จะใช้งานเป็นพอร์ตอินพุตเอาต์พุตแล้วมันยังถูกใช้งานในการติดต่อกับหน่วยความจำภายนอก (EPROM,RAM) ได้อีกด้วย โดยทำหน้าที่ในการกำหนดตำแหน่งแอดเดรสไบต์ต่ำ (A0-A7) ซึ่งจะใช้งานเป็นแบบมัลติเพล็กซ์สำหรับการรับส่งข้อมูลขนาด 8 บิต (D0-D7)

4. พอร์ต 1 (Port 1) มี 8 บิต ได้แก่บิต P1.0-P1.7 เป็นพอร์ตอินพุตเอาต์พุตแบบ 2 ทิศทาง สำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุตพอร์ตหรืออินพุตในระดับบิต สามารถกระทำได้โดยวิธีเช่นเดียวกันกับพอร์ต 0 ข้างต้น

5. พอร์ต 2 (Port 2) มี 8 บิต ได้แก่ บิต P2.0-P2.7) เป็นพอร์ตอินพุตเอาต์พุตแบบ 2 ทิศทาง สำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุตพอร์ตเหรืออินพุตในระดับบิต สามารถกระทำได้เช่นเดียวกันกับพอร์ต 0 ข้างต้น เช่นเดียวกันกับพอร์ต 0 นอกจากใช้งานเป็นพอร์ตอินพุตเอาต์พุตแล้วมันยังถูกใช้งานในการติดต่อกับหน่วยความจำภายนอก (EPROM,RAM) ได้อีกด้วย โดยทำหน้าที่ในการอ้างตำแหน่งแอดเดรสไบต์สูง (A8-A15)

6. พอร์ต (Port 3) มี 8 บิต ได้แก่บิต P3.0-P3.7 เป็นพอร์ตอินพุตเอาต์พุตแบบ 2 ทิศทาง สำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุตพอร์ตหรืออินพุตในระดับบิต สามารถกระทำได้เช่นเดียวกันกับพอร์ต 0 ข้างต้น นอกจากจะใช้งานเป็นพอร์ตอินพุตเอาต์พุตแล้วมันยังสามารถใช้งานในหน้าที่พิเศษต่างๆ ดังตารางที่ 1-2

7. ขารีเซต (RST) ใช้สำหรับการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยการรีเซตต้องคงสภาวะ high อย่างน้อยนาน 2 Machine cycle ในขณะที่ออสซิลเลเตอร์ยังทำงานอยู่

8. ขา ALE / PROG เป็นขาสัญญาณเพื่อทำหน้าที่ควบคุมการแล็ช (Latch) ค่าตำแหน่งแอดเดรสไบต์ต่ำ (Address Latch Enable) เมื่อต้องการติดต่อกับหน่วยความจำภายนอก นอกจากนี้ขานี้ยังทำหน้าที่เป็นอินพุตรับพัลส์ในการโปรแกรม (Program Pulse Input)) ในส่วนของหน่วยความจำ EPROM สำหรับ

รูปที่ 2.11 แสดงหน้าที่พิเศษของแต่ละขาของพอร์ต 3 ของไมโครคอนโทรลเลอร์

Pin Port	Discription
P3.0	RXD (Serial Input Port)
P3.1	TXD (Serial Output Port)
P3.2	$\overline{\text{INT0}}$ (External Interrupt 0)
P3.3	$\overline{\text{INT1}}$ (External Interrupt 1)
P3.4	T0 (Timer 0 External Input)
P3.5	T1 (Timer 1 External Input)
P3.6	$\overline{\text{RW}}$ (External Data Memory Write Strobe)
P3.7	$\overline{\text{RD}}$ (External Data Memory Read Strobe)

ไมโครคอนโทรลเลอร์ในตระกูล MCS-51 ที่มีหน่วนความจำโปรแกรมภายในเป็น EPROM

9. ขา PSEN (Program Store Enable) ทำหน้าที่เป็นสัญญาณสโตรบเพื่ออ่านคำสั่งจากหน่วยความจำโปรแกมภายนอก เมื่อไมโครคอนโทรลเลอร์ประมวลผลคำสั่งจากหน่วยความจำภายนอก เมื่อไมโครคอนโทรลเลอร์ประมวลผลคำสั่งจากหน่วยความจำภายนอก ขานี้จะส่งสัญญาณสโตรบจำนวน 2 ครั้ง ในแต่ละ Machine Cycle แต่ในขณะที่ติดต่อกับหน่วยความจำข้อมูลภายนอกจะไม่มีการส่งสัญญาณ Strobe แต่อย่างใด

10. ขา EA/Vcc (External Access Enable/Vcc) เป็นขาสำหรับการเลือกใช้หน่วยความจำโปรแกรมจากภายในหรือจากภายนอก โดยมีสถานะเป็น 0 หรือ 1 จะหมายถึงให้ไมโครคอนโทรลเลอร์ รับคำสั่งจากหน่วยความจำภายนอก และภายในตามลำดับ อย่างไรก็ตามถ้าบิตป้องกัน (Security Bit) ในหน่วยความจำ EPROM ถูกโปรแกรมไว้ ไมโครคอนโทรลเลอร์จะไม่รับคำสั่งจากหน่วยความจำภายนอกเลย นอกจากนี้ ขานี้ยังทำหน้าที่รับแรงดันไฟสำหรับการโปรแกรม (Vcc)

ขนาด 12 โวลต์ เพื่อใช้ในระหว่างการโปรแกรมหน่วยความจำโปรแกรม (EPROM) ภายในตัว MCU

11. ขา XTAL1 และขา XTAL2 เป็นขาใช้งานของวงจรอินเวอร์ติ้งออสซิลเลเตอร์แอมปลิไฟเออร์ (Inverting Oscillator Amplifier) สำหรับใช้ต่อร่วมกับคริสตอลภายนอก

2.2.5 Machine Cycle

ในการทำงานของ MCU นั้นจะถูกกำหนดความเร็วในการทำงานหรือการประมวลผลต่างๆ ด้วยสัญญาณนาฬิกา (Clock) ที่ป้อนให้กับ MCU ไม่ว่าจะเป็นการต่อสัญญาณนาฬิกาโดยตรง ซึ่งวิธีการนี้จะต้องคำนึงถึงชนิดหรือประเภทของ MCU นั้นๆ ด้วยว่าเป็น CMOS,CHMOS หรือ HMOS โดยแต่ละชนิดประเภทจะมีการต่อสัญญาณนาฬิกาโดยตรงที่แตกต่างกันไป จากรูปที่ 1-2 (ก) นั้นจะเป็นตัวอย่างการต้อสัญญาณนาฬิกาโดยตรงจากภายนอกสำหรับ MCU ชนิดประเภท CMOS,CHMOS โดยวิธีการต่อสัญญาณนาฬิกาจากภายนอกวิธีนี้ออกจะยุ่งยาก และต้องระมัดระวังในการต่อใช้งาน ผู้เขียนจึงขอแนะนำให้ใช้วิธีที่ 2 ดังแสดงในรูปที่ 1-2 (ข) จะเป็นการใช้ X-TAL ในการกำเนิดสัญญาณนาฬิกาให้กับ MCU ซึ่งจะมีความแน่นอนและมีวงจรการใช้งานที่มีความซับซ้อนน้อยกว่าวิธีแรก แต่ขอให้ผู้อ่านคำนึงถึงการเลือกใช้ความถี่ของ X-TAL ให้เหมาะสมกับประเภทและชนิดของงานด้วย เพราะ X-TAL นั้นยิ่งมีความถี่มากก็จะเป็นตัวกำเนิดสัญญาณรบกวนมากเช่นกัน (การแก้ไขสัญญาณรบกวนในเบื้องต้นนั้นให้ทำการ ชีลด์ (Shield) หรือ ต่อตัวถัง X-TAL ลงกราวด์ก็จะแก้ปัญหาสัญญาณรบกวนได้บ้าง

สัญญาณนาฬิกาจะเป็นตัวควบคุมการทำงาน การประมวลผลคำสั่งต่างๆ ของ MCU โดยในการประมวลผลคำสั่งของ MCS-51 จะทำงานเป็นรอบวัฏจักรของภายเครื่อง หรือที่เรียกว่า Machine Cycle โดย 1 Machine Cycle จะใช้ช่วงเวลาในการทำงานเท่ากับคาบเวลาของสัญญาณนาฬิกาจำนวน 12 ลูก (12 Oscillator period) จากคุณสมบัติข้างต้นเราสามารถคำนวณเวลาการทำงานใน 1 Machine Cycle ของ MCS-51 จากความถี่ของ X-TAL ที่เลือกนำมาใช้งานได้จากสมการ

$$1 \text{ Machine Cycle} = 12 \frac{12}{X-TAL} \text{ oscillator (Second)}$$

โดยเมื่อเรารู้ระยะเวลาในการทำงานใน 1 Machine Cycle ของ CPU แล้วจะช่วยให้สามารถประมาณช่วงเวลาในการทำงานของคำสั่งแต่ละคำสั่งในการเขียนโปรแกรมได้ ซึ่งคำสั่งแต่ละคำสั่งจะถูกกำหนดการทำงานเป็น Machine Cycle เช่นกัน โดยรายละเอียดการทำงานของคำสั่งในรูป

2.2.6 พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ตะกูล MCS-51 มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์ 1 ชุด (วงจรสื่อสารแบบฟูลดูเพล็กซ์ หมายถึง วงจรสื่อสารที่สามารถทำการรับและส่งข้อมูลในลักษณะ 2 ทิศทางได้ในเวลาเดียวกัน) โดยใช้ขาสัญญาณของพอร์ต 3 คือขา P3.0 เป็นขารับข้อมูลเข้าหรือ RxD และขา P3.1 เป็นขาส่งข้อมูลออกหรือ TxD โดยวงจรสื่อสารข้อมูลแบบอนุกรมของไมโครคอนโทรลเลอร์ตระกูล MCS-51 แบบแฟลชเป็นแบบอะซิงโครนัสปกติแล้วพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 จะใช้ในการติดต่อสื่อสารกับพอร์ตอนุกรมของคอมพิวเตอร์ โดยใช้มาตรฐาน RS-232 แต่ในปัจจุบันสามารถติดต่อกันในมาตรฐาน RS-422 หรือ RS-485 ได้แล้วโดยใช้ไอซีพิเศษที่ทำหน้าที่ในการแปลงสัญญาณการสื่อสารดังกล่าว

2.2.7 การสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารข้อมูลแบบอะซิงโครนัสคือการรับและส่งข้อมูลโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วยแต่จะใช้การกำหนดค่าอัตราเร็วในการรับและส่งข้อมูลให้มีค่าเท่ากัน ซึ่งเรียกอัตราเร็วนี้ว่า **อัตราบอด** หรือ **บอดเรต(baud rate)** มีหน่วยเป็น **บิตต่อวินาที (bit per second : bps)**

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกันคือ

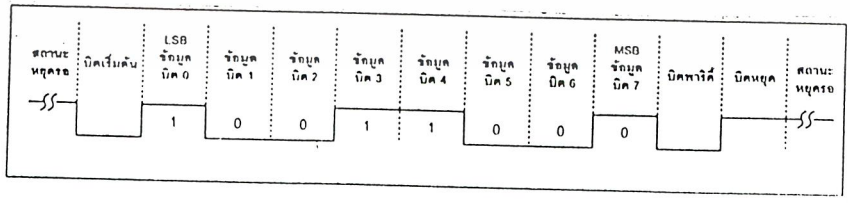
1. บิตเริ่มต้น (start bit) มีขนาด 1 บิต
2. บิตข้อมูลแบบอนุกรม มีขนาด 8 บิต
3. บิตตรวจสอบพาริตี้ (parity bit) มีขนาด 1 บิตหรือไม่มี
4. บิตปิดท้ายหรือบิตหยุด (stop bit) มีขนาด 1 บิต

รูปที่ 9-1 แสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัสเมื่อไม่มีการส่งข้อมูล ขา DATA จะมีสถานะลอจิก “1” เรียกสถานะนี้ว่า **สถานะหยุดรอ(waiting stage)** การเริ่มส่งข้อมูบจะเริ่มจากการให้ขา DATA มีลอจิก “0” ด้วยช่วงระยะเวลา 1 บิต เรียกบิตนี้ว่า **บิตเริ่มต้น (start bit)** จากนั้นบิตข้อมูลจะถูกส่งออกไป โดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุดหรือบิต LSB ก่อน ซึ่งข้อมูลที่ต้องการส่งมีจำนวน 8 บิต จากนั้นตามด้วย **บิตพาริตี้(parity bit)** ซึ่งใช้ในการตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูล บิตสุดท้ายที่จะส่งคือ **บิตปิดท้าย** หรือ **บิตหยุด(stop bit)** โดยจะเป็นากรทำให้ขา DATA มีสถานะลอจิก “1” อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิต, 1.5 บิต หรือ 2 บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว

อัตราความเร็วในการรับและส่งข้อมูลแบบอะซิงโครนัสหรืออัตราบอดหรือบอดเรตที่ใช้สำหรับพอร์ตอนุกรม RS-232 มีด้วยกันหลายค่าตั้งแต่ 110 ถึง 19,200 บิตต่อวินาที โดยมีค่าเพิ่มมากขึ้นตามเทคโนโลยีของคอมพิวเตอร์ เนื่องจากอัตราบอดคือค่าของจำนวนบิตที่สามารถส่งได้ใน 1 วินาที สมมติว่า ข้อมูลอนุกรมมีขนาด 8 บิต ไม่มีการตรวจสอบพาริตี้ มีบิตเริ่มต้น 1 บิต และบิตปิดท้าย 1 บิตมีความยาวของข้อมูล 1 ไบต์จะมีความยาวเท่ากับ 10 บิต ถ้าใช้บอดเรตในการส่งข้อมูลเท่ากับ 9,600 บิตต่อวินาที ก็จะสามารถรับส่งข้อมูลได้ด้วยความเร็ว 960 ไบต์ต่อวินาที

การตรวจสอบพาริตี้สามาถกำหนดให้เป็นแบบคี่(odd), แบบคู่(even) หรือม่มีการตรวจสอบพาริตี้ก็ได้พาริตี้คี่หรือพาริตี้คู่แสดงถึงจำนวนลอจิก “1” ทั้งหมดภายในข้อมุลที่ส่งไป 1 ไบต์รวมบิตพาริตี้ว่ามีจำนวนเป็นเลขคู่หรือเลขคี่ ยกตัวอย่าง ข้อมูลที่จะทำการส่งมีขนาด 8 บิต มีค่าเท่ากับ 99H หรือ 10011001B จะเห็นว่าข้อมูลในไบต์นี้มีจำนวนลอจิก “1” จำนวน 4 ตัวซึ่งเป็นเลขคู่ ดังนั้นถ้ากำหนดค่าพาริตี้เป็นคู่ ค่าของบิตพาริตี้จะต้องมีลอจิกเป็น “0” แต่ถ้ากำหนดพาริตี้เป็นคี่ ค่าของบิตพาริตี้จะต้องเป็น “1” เพื่อให้ข้อมูล 1 ไบต์รวมทั้งบิตพาริตี้เป็นคี่

บิตพาริตี้ถูกสร้างขึ้นจากภาคส่งข้อมูลของ UART(Universal Asynchronous Receiver Transmitter: เป็นอุปกรณ์ที่ใช้ในการรับและส่งข้อมูลอนุกรม) ซึ่งทางภาครับต้องกำหนดการตรวจส่อบพาริตี้ที่ตรงกันเอาไว้ว่าจะตรวจสอบพาริตี้คู่หรือคี่ จากนั้นภาครับของ UART จะทำการตรวจสอบค่าพาริตี้ที่เกิดขึ้นว่าเป็นคู่หรือคี่ โดยการนับจำนวนลอจิก “1” ทั้งหมดรวมทั้งบิตพาริตี้ด้วย ถ้ากำหนดพาริตี้ไว้เป็นคู่แต่อ่านค่าตัวเลขในการนับออกมาได้ตัวเลขเป็นคี่ทางภาครับจะแสดงข้อผิดพลาดออกมาให้ผู้ใช้ทาบ กระบวนการดังกล่าวเป็นวิธีการตรวจสอบความผิดพลาดที่เกิดขึ้นในการรับส่งข้อมูลที่ง่ายที่สุด แต่มันสามารถตรวจสอบได้เมื่อมีบิตข้อมูลที่ทำการรับส่งผิดพลาดเพียงบิตเดียวเท่านั้น ถ้าข้อมูลที่ทำการส่งมีบิตผิดพลาดมากกว่า 1 บิต การตรวจสอบด้วยวิธีนี้จะไม่ได้ผลดสำหรับการตั้งพาริตี้เป็น NONE นั้นทั้งภาครับและส่งจะไม่มีการตรวจสอบพาริตี้



รูปที่2.12 รูปแบบข้อมูลอนุกรมแบบอซิงโครนัส

2.2.8 รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของพอร์ตอนุกรมใน MCS-51

ในการทำงานของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 มีรีจิสเตอร์ที่เกี่ยวข้องอยู่ 2 ตัวดังนี้

รีจิสเตอร์บัฟเฟอร์ของพอร์ตอนุกรมหรือ SBUF (Serial data buffer register)

มีแอดเดรสอยู่ที่ 99H ในพื้นที่ของรีจิสเตอร์ฟังก์ชั่นพิเศษหรือ SFR มีขนาด 8 บิต แบ่งเป็น 2 ส่วนคือรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล (transmit buffer register) และรับข้อมูล(receive buffer register) เมื่อมีการเขียนข้อมูลมายังรีจิสเตอร์ SBUF ข้อมูลนั้นจะถูกส่งต่อไปยังบัฟเฟอร์สำหรับส่งข้อมูล เพื่อส่งออกจากไมโครคอนโทรลเลอร์ผ่านทางขา TxD หรือขา P3.1 ในกรณีที่มีการอ่านข้อมูลจากรีจิสเตอร์ SBUF ข้อมูลจะถูกส่งผ่านไปยังรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูลเพื่อส่งต่อไปยังไมโครคอนโทรลเลอร์ต่อไป สำหรับการรับส่งข้อมูลอนุกรมจากภายนอกนั้นจะผ่านมาทางขา RxD หรือ P3.0 ของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

รีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรมหรือ SCON (Serial port Control Register)

เป็นรีจิสเตอร์ขนาด 8 บิตมีแอดเรสอยู่ที่ 98H ในพื้นที่ของรีจิสเตอร์ SFR สามารถเข้าถึงได้ในระดับบิตมีรายละเอียดการทำงานดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

รูปที่ 2.13 Register ขนาด 8 บิต

SM0-SM1(Serial port mode bit 0-1) : ใช้ในการเลือกโหมดการทำงานของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51 ดังมีรายละเอียดต่อไปนี้

SM2 : ใช้ในการเอ็นเอเลการสื่อสารในแบบมัลติโปรเซสเซอร์(multiprocessor) ในการทำงานของ

โหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 ถ้าบิตนี้เป็น “1” บิต RI จะไม่แอกติฟพ้าบิตที่ 9 ที่รับเข้ามาเป็น “0” (ข้อมูลบิตที่ 9 เก็บไว้ที่บิต RB8) ในการทำงานโหมด 1 ถ้าบิตนี้เซต บิต RI จะไม่แอกติฟพ้ายงไม่ได้รับบิตหยุด ส่วนในโหมด 0 บิตนี้ไม่มีการใช้งาน

REN(Enable serial reception) : ใช้ในการเอ็นเอเบิลการรับข้อมูลของพอร์ตอนุกรม ทำการเซตและเคลียร์ด้วยการบวนการทางซอฟต์แวร์ ถ้าต้องการให้มีการรรับข้อมูลต้องเซตบิตนี้ให้เป็น “1”

TB8 : ใช้สำหรับเก็บข้อมูลบิตที่ 9 ที่ต้องการส่งออกไปในการทำงานโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 เซตและเคลียร์ด้วยกระบวนการทางซอฟแวร์

RB8 : ใช้สำหรับรับข้อมูลบิตที่ 9 ที่เข้ามาในการทำงานโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 แต่ถ้าหากพอร์ตอนุกรมทำงานอยู่ในโหมด 1 และบิต SM2 เป็น “0” ข้อมูลที่บิต RB8 คือข้อมูลของบิตหยุด (STOP bit) สำหรับในการทำงานโหมด 0 บิตนี้จะไม่ใช้งานบิต RB8 นี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟแวร์

TI (Transmit Interrupt flag) : ใช้ในการแสดงการเกิดอินเตอร์รัปต์เมื่อมีการส่งข้อมูลออกจากพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์เมื่อทำการส่งข้อมูลบิตที่ 8 ไปเรียบร้อยแล้วในการทำงานโหมด 0 ส่วนในการทำงานโหมดอื่น บิตนี้จะเซตเมื่อมการเริ่มต้นส่งบิตหยุดออกไป การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟต์แวร์เท่านั้น

RI (Receive Interrupt flag) : ใช้แสดงการเกิดอินเตอร์รัปต์เมื่อมีการรับข้อมูลเข้าสู่พอร์ตอนุกรม สามารถเซตได้ด้วยการะบวนการทางฮาร์ดแวร์ เมื่อทำการรับข้อมูลบิตที่ 8 เรียบร้อยแล้วในการทำงานโหมด 0 ส่วนในการทำงานโหมดอื่น บิตนี้จะเซตเมื่อมีสามารถรับบิตหยุดของข้อมูลอนุกรมไปได้ครึ่งทางแล้ว ยกเว้นในกรณีที่บิต SM2 มีการเซตบิตนี้จะเซตได้ก็ต่อเมื่อการรับบิตหยุดหรือบิตที่ 9 เกิดขึ้นอย่างสมบูรณ์แล้ว การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟต์แวร์เท่านั้น

2.2.9 โหมดการทำงานของพอร์ตอนุกรมใน MCS-51

พอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 สามารถเลือกการทำงานได้ถึง 4 โหมด คือ

1. โหมด 0 เป็นการกำหนดให้พอร์ตอนุกรมทำงานในลักษณะชิฟต์รีจิสเตอร์
2. โหมด 1 เป็นการกำหนดให้เป็น UART ขนาด 8 บิต สามารถเลือกอัตราบอดได้
3. โหมด 2 เป็นการกำหนดให้เป็น UART ขนาด 9 บิต โดยมีอัตราบอดคงที่
4. โหมด 3 เป็นการกำหนดให้เป็น UART ขนาด 9 บิต สามารถเลือกอัตราบอดได้

การเลือกโหมดทำได้ด้วยการกำหนดข้อมูลให้แก่บิต SM0 และ SM1 ในริจิสเตอร์ SCON

2.2.10 การทำงานในโหมด 0 ของวงจรพอร์ตอนุกรม

มีไดอะแกรมกรทำงานและไดอะแกรมเวลาแสดงในรูปที่ 9-2 ข้อมูลอนุกรมจะผ่านเข้าและออกทางขา RxD ส่วนขา TxD ทำหน้าที่เป็นสัญญาณนาฬิกาของการเลื่อนข้อมูล(shift clock) ในโหมดนี้มีจำนวนข้อมูล 8 บิตโดยทำการรับและส่งข้อมูลในบิต LSB ก่อนอัตราในการรับส่งข้อมูลหรืออัตราบอดถูกกำหนดไว้คงที่ที่1/12 ของความถี่สัญญาณนาฬิกา

เริ่มต้นการส่งข้อมูลด้วยการเขียนข้อมูลที่ต้องการส่งมายังรีจิสเตอร์ SBUF สัญญาณเขียนข้อมูล SBUF แอกติฟเป็น “1” ที่สเตต 6 เฟส 2 (S6P2) ของแมชีนไซเกิล ส่งมายังวงจรควบคุมการส่ง (TX control) ทำให้วงจรควบคุมเริ่มต้นส่งข้อมูล สัญญาณ SEND จะแอกติฟเป็น “1” ตลอดการส่งข้อมูล

ข้อมูลจากรีจิสเตอร์ SBUF จะถูกเลื่อนออกที่ขา P 3.0 หรือ ขา RxD ครั้งละบิต ตามจังหวะของสัญญาณนาฬิกาที่ส่งออกมาทางขา P3.1 หรือ TxD โดยสัญญาณนาฬิกาของการเลื่อนข้อมูลจะมีขอบขาลงของสัญญาณที่สเตต 3 เฟสและมีขอบขาขึ้นของสัญญาณที่สเตต 6 เฟส 1 ของแต่ละเมชีนไซเกิลในกระบวนการส่งข้อมูลจะกระทั่งเมื่อส่งข้อมูลครบ 8 บิตแล้ว บิต TI ในรีจิสเตอร์ SCON จะเกิดการเซต เป็นการแจ้งให้ทราบว่าส่งข้อมูลครบแล้ว หากการอินเตอร์รัปต์จากพอร์ตอนุกรมได้รับการเอ็นเอเบิลไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบเมื่อเสร็จสิ้นกระบวนการรับข้อมูล สัญญาณ SEND จะกลายเป็น “0” จนกว่าจะเริ่มต้นกระบวนการรับข้อมูลใหม่

ในกระบวนการรับข้อมูลเริ่มต้นด้วยการเซต REN ให้เป็น “1” และเคลียร์บิต RI ในรีจิสเตอร์ SCON ก่อนที่สเตต 6 เฟส 2 ของแมชีนไซเกิลถัดไป วงจรควบคุมการรับ (RX control) จะทำการเขียนข้อมูล 11111110 ไปยังชิฟตีรีจิสเตอร์สำหรับรับข้อมูลและทำการแอกติฟสัญญาณ RECEIVE ให้เป็น “1” ในสัญญานนาฬิกาลูกถัดไป

เมื่อสัญญาณRECEIVE แอกติฟก็จะเกิดการส่งสัญญาณนาฬิการของการเลื่อนข้อมูล(Shift clock) ขึ้นผ่านทางขา P3.1 หรือ TxD เพื่อทำการกำหนดจังหวะการรับข้อมูลครั้งละบิต โดยสัญญาณนาฬิกานี้จะเกิดขึ้นในช่วงสเตต 3 เฟส 1 ถึง สเตต 6 เฟส 1 ของแต่ละแมชีนไซเกิล การรับข้อมูลเข้ามาทางขา P3.0 หรือ RxD จะเกิดขึ้นที่สเตต 5 เฟส 2 ในแมชีนไซเกิลเดียวกับสัญญาณนาฬิการของการเลื่อนข้อทมูล จนกระทั่งรับข้อมูลครบทั้ง 8 บิต บิตRI จะได้รับการเซตเพื่อแจ้งการเสร็จสิ้นกระบวนการรับข้อมูล หากการอินเตอร์รัปต์จากพอร์ตอนุกรมได้รับการเอ็นเอเบิลไว้ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ เมื่อเสร็จสิ้นกระบวนการรับข้อมูล สัญญาณRECIEVE จะกลายเป็น “0” จนกว่าจะเริ่มต้นกระบวนการรับข้อมูลใหม่

การทำงานในโหมดนี้ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 จะใช้ในการเชื่อมต่อกับไอซีรีจิสเตอร์ภายนอกเพื่อทำการขยายจำนวนพอร์ตอินพุตหรือเอาต์พุต แต่ไม่เป็นที่นิยมใช้งานมากนัก เนื่องจากในไมโครคอนโทรเลอร์ MCS-51 เองมีพอร์ตอยู่ค่อนข้างมากและติดต่อกับพอร์ตเหล่านั้นได้ง่ายและเร็วกว่ามาก

2.2.11 การทำงานในโหมด 1 ของวงจรอนุกรม

มีไดอะแกรมในรูปที่ 9-3 ในโหมดนี้ใช้ในการส่งข้อมูลรวม 10 บิต โดยส่งข้อมูลออกทางขา P3.1 หรือ TxD และรับข้อมูลเข้าทางขา P3.0 หรือ RxD ข้อมูลทั้ง 10 บิต ประกอบด้วย บิตเริ่มต้น (มีค่าเป็น "0") 1 บิต บิตข้อมูล 8 บิต โดยรับข้อมูลหรือส่งในบิต LSB ก่อน และบิตหยุดหรือบิตปิดท้าย (มีค่าเป็น "1") ในการรับข้อมูล บิตหยุดจะถูกเก็บในบิต RB8 ในรีจิสเตอร์ SCON อัตราบอดในโหมดนี้ได้รับการกำหนดโดยอัตราการเกิด overflow ของ timer 1 ใน AT89C51 ส่วนใน Microcontroller เบอร์ AT89C52 และในอนุกรม AT89Sxx สามารถเลือกให้อัตราการเกิด overflow ของ timer หรือ timer 2 ในการกำหนดอัตราการบอดได้

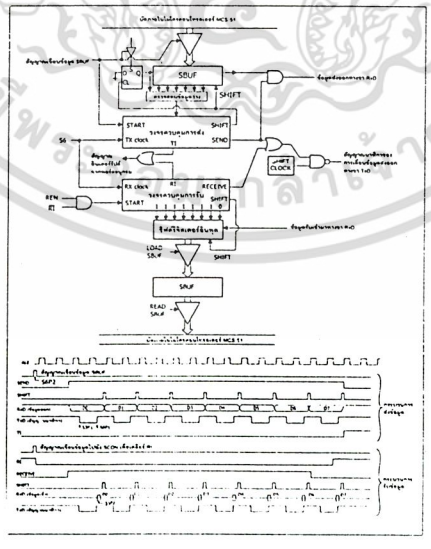
กระบวนการส่งข้อมูลเริ่มต้นด้วยการ active สัญญาณเขียนข้อมูล มายังรีจิสเตอร์ SBUF ส่งมายังวงจรควบคุมการส่ง (TX control) จากนั้นวงจรควบคุมจะทำการ active สัญญาณ SEND ที่สเตต 1 เฟส 1 ของ machine cycle ต่อมา โดยสัญญาณ SEND จะเป็น 0 ตลอดการส่งข้อมูล เมื่อสัญญาณ SEND active จะทำการส่งบิตเริ่มต้นก่อนเป็นบิตแรก โดยมีคาบเวลาของบิตเริ่มต้นเท่ากับ 1 machine cycle จากนั้นตามด้วยการส่งบิตข้อมูล 8 บิต เรียงลำดับจากบิต LSB โดยข้อมูลที่ทำการส่งถูกเรียกออกมาจากรีจิสเตอร์บัฟเฟอร์ สำหรับการส่งข้อมูลในทุกๆบิตข้อมูลที่ทำการส่งออกไปจะเกิดสัญญาณพัลส์ SHIFT ขึ้น เพื่อให้เรียกข้อมูลในแต่ละบิต จากรีจิสเตอร์บัฟเฟอร์ การกำหนดจังหวะในการส่งข้อมูล ใช้สัญญาณนาฬิกาการส่ง (Tx clock) เป็นตัวกำหนด โดยสัญญาณนาฬิกานี้ได้มาจากการหารสัญญาณ TCLK จาก timer 1 ด้วย 16 หลังจากการส่งบิตข้อมูล ก็จะทำการส่งบิตหยุด หรือบิตปิดท้าย 1 บิต ดังนั้นการส่งข้อมูลจะใช้สัญญาณนาฬิกาทั้งหมด 10 ลูก เมื่อทำการส่งข้อมูลครบเรียบร้อยแล้ว จะทำการเซตบิต TI ในรีจิสเตอร์ SCON หากการ Interrupt จากพอร์ตอนุกรม ได้รับการ Enable ไว้ ก็จะเกิดการ Interrupt ขึ้นในระบบ หลังจากที่ทำการบริการ Interrupt หรือส่งข้อมูลเรียบร้อยแล้ว ต้องทำการ clear บิต TI ก่อนเป็นอันดับแรก เพื่อให้การรับส่งข้อมูลทางพอร์ตอนุกรมดำเนินต่อไปได้

ด้านการรับข้อมูล จะทำการตรวจจับการเปลี่ยนแปลงระดับลอจิกจาก 1 เป็น 0 ที่ขา RxD โดยใช้อัตราการสุ่มเท่ากับ 1/16 เท่าของอัตราบอด เมื่อตรวจจับพบ timer/counter ที่ใช้ในการกำหนดอัตราบอดจะรีเซตและทำการเขียนข้อมูล 1FFH ไปยังชิฟต์รีจิสเตอร์ ข้อมูลจะเริ่มเดิน

ทางเข้าสู่พอร์ตอนุกรมของ Microcontroller ผ่านทางขา RxD ในการตีความว่าบิตที่เข้ามาเป็น 0 หรือ 1 จะใช้ผลการสุ่มข้างมาก โดยบิตของข้อมูลที่เข้ามาได้รับการแบ่งออกเป็น 16 state การสุ่มข้อมูลจะทำการสุ่ม state ที่ 7,8 และ 9 หาก 2 ใน 3 ของการสุ่มพบว่าข้อมูลเป็นลอจิกใด จะตีความข้อมูลในบิตนั้น เป็นตามเสียงข้างมาก ยกตัวอย่าง สุ่มพบลอจิก 1 2 ใน 3 ครั้ง จะตีความว่าบิตของข้อมูลเป็น 1

ลำดับของการรับข้อมูลมีลักษณะเดียวกับการส่งคือ เริ่มด้วยบิตเริ่มต้นก่อน ตามด้วยบิตข้อมูล และบิตปิดท้าย ในทุกๆการรับข้อมูลได้ 1 บิต จะมีพัลส์ SHIFT เกิดขึ้น เพื่อทำการเลื่อนข้อมูล เข้าสู่ รีจิสเตอร์บัฟเฟอร์ การรับข้อมูล การกำหนดจังหวะการรับข้อมูลใช้สัญญาณนาฬิกาการรับข้อมูล (Rx clock) หลังจากสัญญาณนาฬิกา ลูกสุดท้าย อันหมายถึงสามารถรับข้อมูลได้ครบแล้ว วงจรควบคุมการรับข้อมูลจะทำการส่งข้อมูลจากรีจิสเตอร์บัฟเฟอร์ ไปยังรีจิสเตอร์ SBUF และบิต RB8 ในรีจิสเตอร์ SCON โดยข้อมูลในบิต RB8 ก็คือข้อมูลของบิตหยุดนั่นเอง พร้อมกันนั้นยังทำการเซตบิต R1 ในรีจิสเตอร์ SCON ด้วย หากการ Interrupt จากพอร์ตอนุกรมได้รับการ Enable ก็จะเกิดการ Interrupt ขึ้นในระบบ หลังจากการบริการ Interrupt หรือรับข้อมูลเรียบร้อยแล้ว ต้องทำการ clear บิต R1 ก่อน เพื่อให้การรับข้อมูลทางพอร์ตอนุกรมดำเนินต่อไป

การทำงานในโหมดนี้ได้รับความนิยมสูงสุด เนื่องจากมีกระบวนการที่ไม่ซับซ้อน และสามารถทำการรับส่ง ข้อมูลกับ computer ได้อย่างมีประสิทธิภาพ



รูปที่ 2.13 ไดอะแกรมการทำงานในโหมด 0 ของพอร์ตอนุกรม

2.2.12 การทำงานในโหมด 2 และ 3 ของวงจรพอร์ตอนุกรม

ในทั้งสองโหมดนี้จะใช้รูปแบบข้อมูลรวม 11 บิต ประกอบด้วยบิตเริ่มต้น มีค่าเป็น 0 และ 1 บิต บิตข้อมูล 8 บิต โดยทำการรับและส่งบิต LSB ก่อน บิตข้อมูลบิตที่ 9 และบิตปิดท้ายมีค่าเป็น 1 จำนวน 1 บิต ในการส่งข้อมูล ข้อมูลบิตที่ 9 จะทำการเก็บไว้ในบิต TB8 ในรีจิสเตอร์ SCON และในการรับข้อมูล ข้อมูลบิตที่ 9 จะทำการเก็บไว้ในบิต RB8 ในรีจิสเตอร์ SCON สำหรับอัตราบอดในโหมด 2 จะคงที่โดยเลือกได้สองค่า คือ 1/32 และ 1/64 ของความถี่สัญญาณนาฬิกา สำหรับในโหมด 3 อัตราบอดสามารถปรับได้เหมือนกับในโหมด 1

ในรูปที่ 9-4 และ 9-5 เป็นไดอะแกรมการทำงาน และไดอะแกรมการทำงานของโหมด 2 และ 3 ของพอร์ตอนุกรม การทำงานโดยรวมจะทำงานคล้ายกับในโหมด 1 ส่วนที่แตกต่างกันคือ จำนวนบิตของข้อมูลที่ในโหมด 2 และ 3 จะมีเพิ่มมาอีก 1 บิต โดยส่วนใหญ่จะเป็นบิตตรวจสอบพาริตี้

2.2.13 อัตราบอดของพอร์ตอนุกรมใน microcontroller MCS-51

โหมด 0

อัตรบอดของโหมดนี้มีค่าคงที่ โดยสามารถคำนวณได้จากสูตร

อัตรบอดของโหมด 0 = ความถี่ของสัญญาณนาฬิกา/12 หน่วยเป็น bit/sec

โหมด 1 และ 3

เนื่องจากสองโหมดนี้สามารถเลือกแหล่งอัตราบอดได้ 2 แหล่ง คือ จากอัตรา overflow ของ timer 1 และ 2 สำหรับอัตราบอดที่ใช้การ overflow ของ timer 1 จะต้องใช้ค่าของบิต SMOD ในรีจิสเตอร์ PCON มาพิจารณาประกอบด้วย สามารถคำนวณอัตราบอดได้จาก

อัตราบอด = (2 ยกกำลังค่าบิต SMOD / 32) * อัตรา overflow ของ timer 1

ถ้าหากใน timer 1 ไม่ได้ Enable interrupt ไว้ สามารถคำนวณค่าอัตราบอดได้จาก

อัตราบอด = (2 ยกกำลังค่าบิต SMOD / 32) * (ความถี่สัญญาณนาฬิกา/{12*[256-(TH1)]})

ในตารางที่ 9-1 แสดงการกำหนดอัตราบอดโดยใช้ timer 1

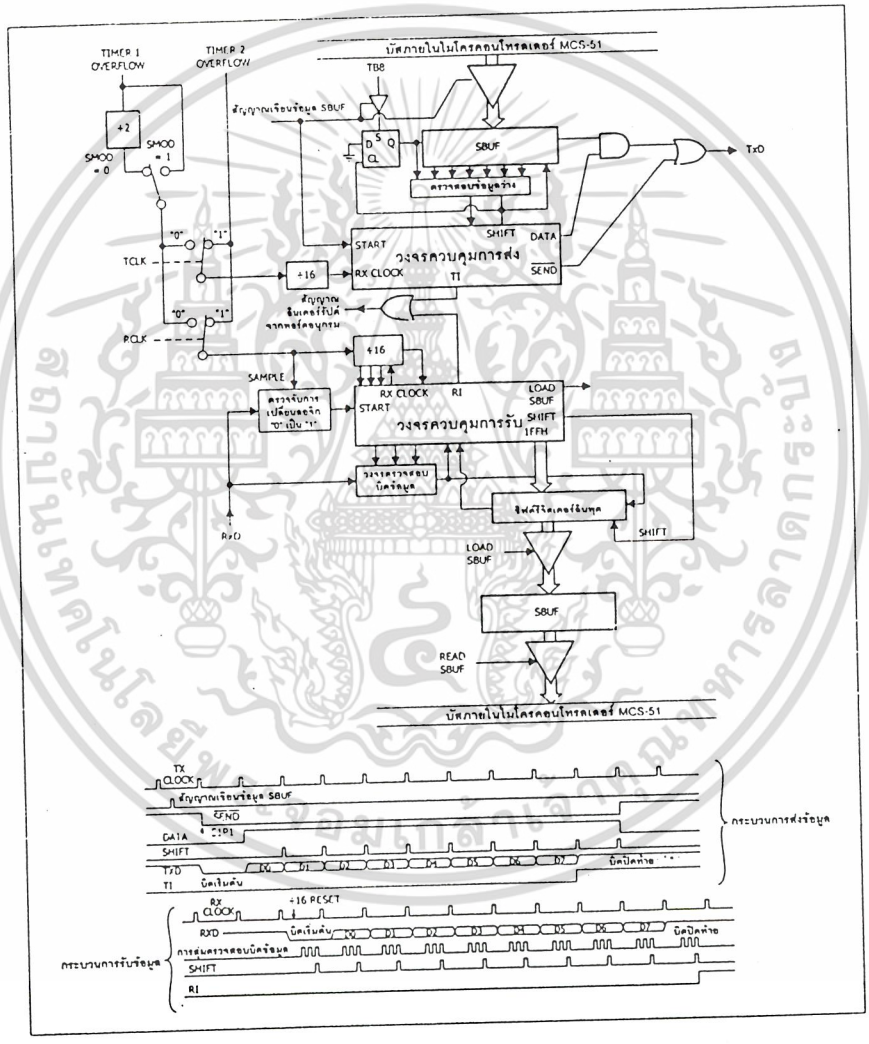
ในกรณีที่ใช้ timer 2 ในการกกำหนดอัตราบอด โดยกำหนดให้ timer 2 ทำงานในโหมดกำหนดอัตราบอด สามารถคำนวณหาอัตราบอดได้จาก

อัตราบอด = อัตรา overflow ของ timer 2/16 หน่วยเป็น bit/sec

ถ้ากำหนดให้ timer 2 ทำงานในโหมดปกติ สามารถคำนวณหาค่าอัตราบอดได้จาก

อัตราบอด = ความถี่ของสัญญาณนาฬิกา /(32 * (65535-(RCAP2H,RCAP2L)))

โดยที่ (RCAP2H,RCAP2L) เป็นค่าของรีจิสเตอร์ RCAP2H และ RCAP2L มีขนาด 16 บิต ไม่คิดเครื่องหมาย



รูปที่ 2.14 ไดอะแกรมการทำงานในโหมด 1

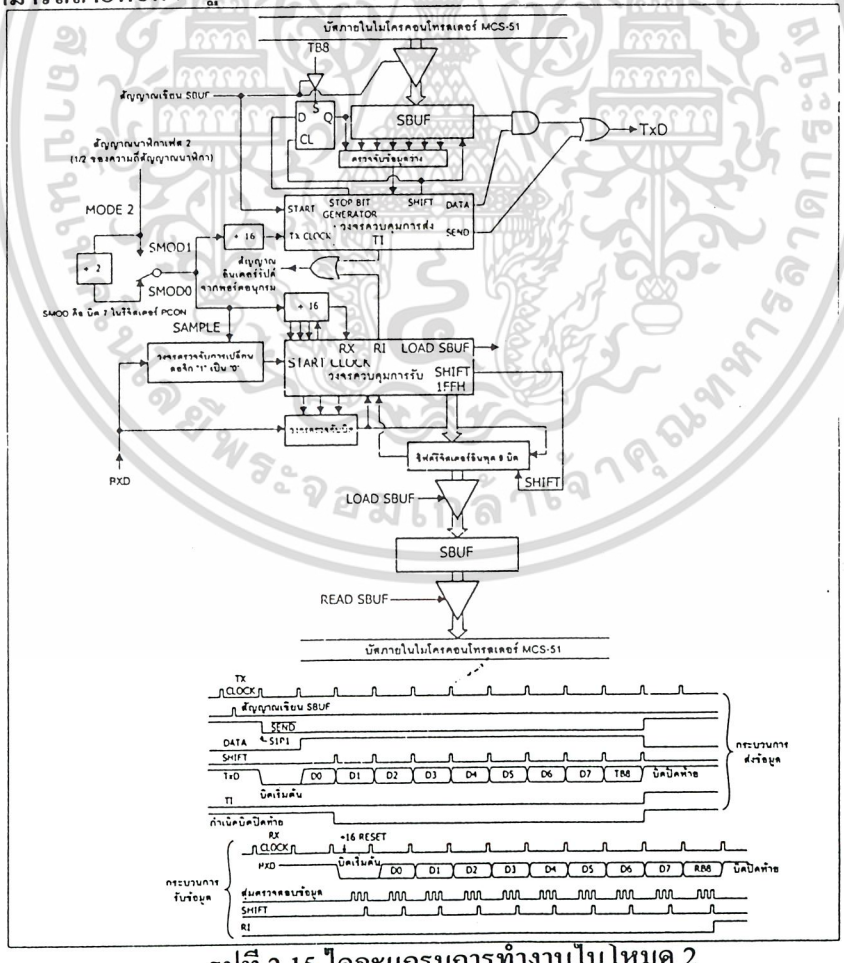
2.2.14 โหมด 2

ในโหมดนี้อัตราบอดจะขึ้นอยู่กับค่าของบิต SMOD ในรีจิสเตอร์ PCON เป็น 0 อัตราบอดจะเท่ากับ 1/64 ของความถี่สัญญาณนาฬิกา ในกรณีที่ SMOD เป็น 1 อัตราบอดจะเท่ากับ 1/32 ของความถี่สัญญาณนาฬิกาสามารถแสดงเป็นสูตร คำนวณทางคณิตศาสตร์ได้ดังนี้

อัตราบอด = (2 ยกกำลังของบิต SMOD /64) * ความถี่สัญญาณนาฬิกา

2.2.15 การกำหนดค่าของ Timer เพื่อเลือกอัตราบอด

ในการใช้งานพอร์ตอนุกรมของ microcontroller MCS-51 สิ่งที่ต้องให้ความสนใจมากที่สุดประการหนึ่งคือ อัตราการถ่ายข้อมูล หรือ อัตราบอด ซึ่งการกำหนดอัตราบอดนั้นจะขึ้นอยู่กับค่าความถี่ของสัญญาณนาฬิกา เป็นหลัก สำหรับโหมดการทำงานของพอร์ตอนุกรมที่สามารถเลือกอัตราบอดได้อย่างอิสระคือโหมด 1 และ 3 โดยกำหนดได้จากอัตราการเกิด over flow ในอัตราที่สูงมากเท่าใด อัตราบอดก็จะมีค่าสูงมากขึ้นตามไปด้วย นั่นหมายความว่า อัตราในการถ่ายทอดข้อมูลจะสูงมากสามารถถ่ายทอดข้อมูลได้รวดเร็ว



รูปที่ 2.15 ไดอะแกรมการทำงานในโหมด 2

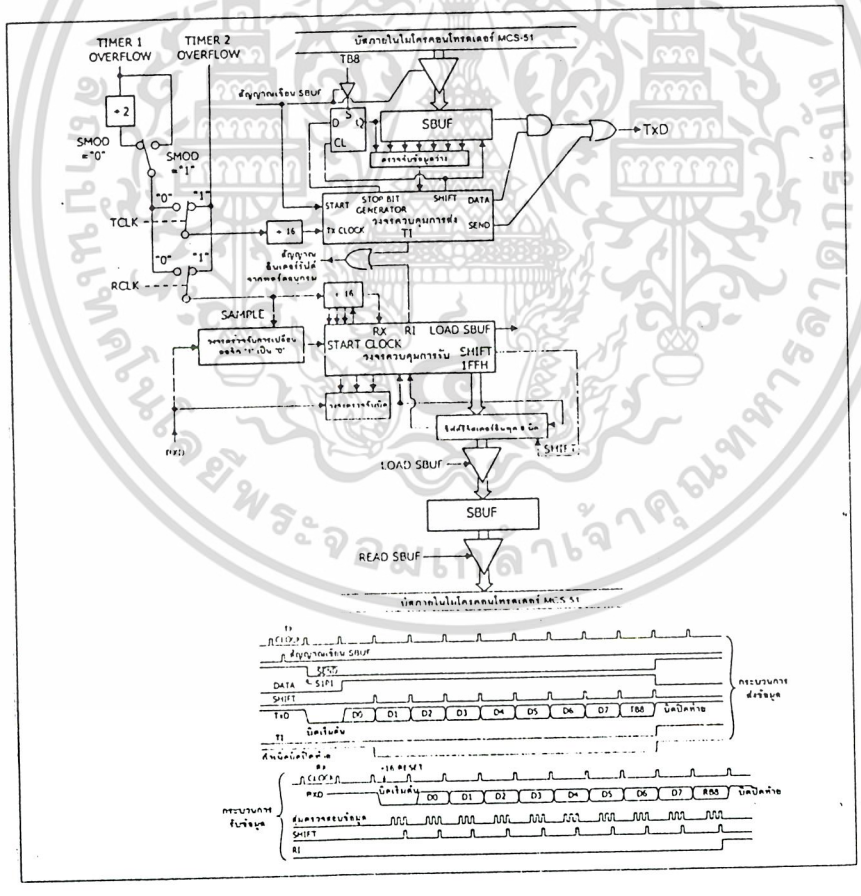
ในการใช้ timer 1 เพื่อกำหนดอัตราบอดในโหมด 1 และ 3 ของพอร์ตอนุกรมจะต้องกำหนดให้ timer 1 ทำงานในโหมด 8 บิต แบบตั้งค่าอัตโนมัติ และการกำหนดค่ารีโหลดให้แก่รีจิสเตอร์ TH11 จึงเป็นตัวแปรหลักในการกำหนดอัตราบอดให้แก่พอร์ตอนุกรม ของ microcontroller MCS-51

เริ่มต้นด้วยการ clear บิต SMOD ซึ่งเป็นบิต 7 ของรีจิสเตอร์ PCON ให้เป็น 0 ค่าของการรีโหลดได้แก่ TH1 สามารถคำนวณได้จาก

TH1 = 256-((ค่าความถี่ของคริสตอล/384)/อัตราบอด)

แต่ถ้าบิต SMOD เกิดการเซต จะเป็นการ Enable การทวีคูณของอัตราบอด ดังนั้นการกำหนดค่าให้แก่ TH1 จึงต้องคำนวณจาก

TH1 = 256-((ค่าความถี่ของคริสตอล/192)/อัตราบอด)



รูปที่ 2.16 ไดอะแกรมการทำงานในโหมด 3

ยกตัวอย่าง ถ้าหากใน microcontroller AT89C51 ใช้คริสตอล 11.0592 MHz ต้องการกำหนดอัตราบอด ของพอร์ตอนุกรมของ microcontroller ไว้ที่ 19,200 bit/sec ในกรณีที่ไม่ Enable การทวีคูณของอัตราบอด ค่ารีโหลดของ microcontroller จะเท่ากับ

TH1 = 256-((ความถี่ของคริสตอล/384)/อัตราบอด)

= 256-((11059200/384)/19200)

= 254.5

เนื่องจากผลลัพธ์ที่ได้เป็นค่าที่ไม่ใช่จำนวณเต็ม ถ้าหากกำหนดค่าของ TH1 เป็น 254 เมื่อทำการแทนค่าเพื่อคำนวณหาอัตราบอด จะได้อัตราบอดเท่ากับ 14,400 bit/sec และถ้าหากกำหนดค่าของ TH1 ของ 255 อัตราบอดจะมีค่าเท่ากับ 28,800 bit/sec ดังนั้นจะเห็นได้ว่าค่าของ TH1 ที่ไม่เป็นจำนวนเต็มจะไม่สามารถทำให้เกิดอัตราบอดตามที่ต้องการ

ทางแก้ไขคือ ให้ทำการ Enable การทวีคูณค่าอัตราบอด โดยการเซตบิต SMOD ในรีจิสเตอร์ PCON ให้เป็น 1 จากนั้นแทนค่าลงในสมการหาค่า TH1 เมื่อมีการเซตบิต SMOD ได้ผลดังนี้

TH1 = 256-((ค่าความถี่ของคริสตอล/192)/อัตราบอด)

= 256-((11059200/192)/19200)

= 253

นำค่า TH1 ที่ได้ทำการแทนค่าคำนวณหาค่าอัตราบอดจะได้เท่ากับ 19,200 bit/sec สามารถสรุปขั้นตอนในการเลือกอัตราบอดโดยการกำหนดค่าของ timer 1 ได้ดังนี้

อัตราบอด (บิตต่อวินาที : bps)	ความถี่สัญญาณนาฬิกา	SMOD	ไทเมอร์ 1		
			C/T	โหมด	ค่ารีโหลด
โหมด 0 : สูงสุด 1 MHz	12 MHz	X	X	X	X
โหมด 2 : สูงสุด 375K	12 MHz	1	X	X	X
โหมด 1,3 : 62.5K	12 MHz	1	0	2	FFH
19.2K (19,200)	11.0592 MHz	1	0	2	FDH
9.6K (9,600)	11.0592 MHz	0	0	2	FDH
4.8K (4,800)	11.0592 MHz	0	0	2	FAH
2.4K (2,400)	11.0592 MHz	0	0	2	F4H
1.2K (1,200)	11.0592 MHz	0	0	2	E8H
137.5	11.0592 MHz	0	0	2	1DH
110	6 MHz	0	0	2	72H
110	12 MHz	0	0	1	FEEBH

รูปที่ 2.17 ตารางการเลือกอัตราบอดของวงจรอนุกรม

1. กำหนดให้พอร์ตอนุกรมของ microcontroller MCS 51 ทำงานในโหมด 1 และ 3

2. กำหนดให้ timer 1 ทำงานในโหมด 2 หรือ โหมด 8 บิต ตั้งค่าอัตโนมัติ

3. กำหนดข้อมูลให้แก่ TH1 เท่ากับ 253 เพื่อให้สามารถกำเนิดอัตราบอดได้ 19,200 bit/sec ตามต้องการ

4. ทำการเซตบิต SMOD ซึ่งเป็นบิต 7 ของรีจิสเตอร์ PCON เพื่อ Enable การทวีคูณอัตราบอด

2.2.16 การเขียนหรือส่งข้อมูลออกจากพอร์ตอนุกรม

ข้อมูลที่ต้องการส่งออกทุกค่าต้องนำไปเก็บไว้ใน รีจิสเตอร์บัฟเฟอร์ ของพอร์ตอนุกรม ซึ่งก็คือ รีจิสเตอร์ SBUF ดังตัวอย่าง

```
MOV  SBUF,#'A'
```

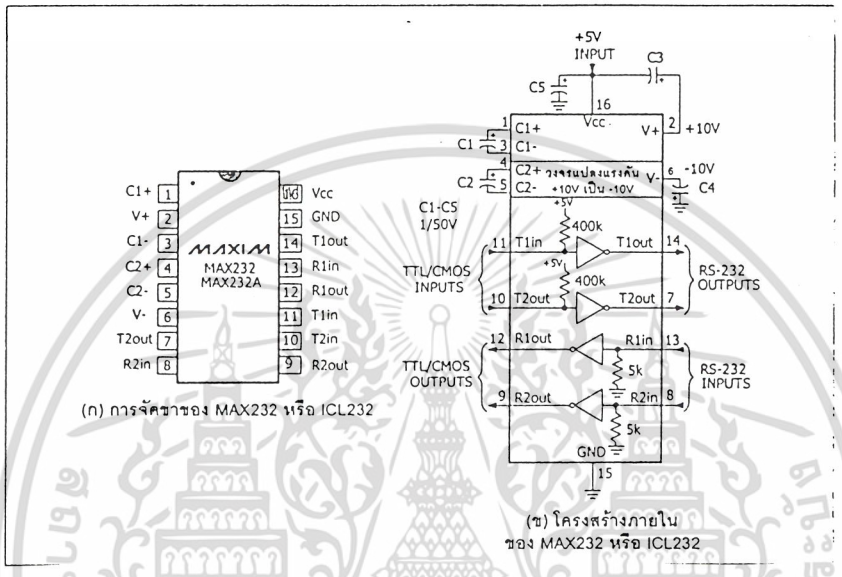
จากคำสั่งข้างต้นเป็นการส่งข้อมูลของตัวอักษร A ออกไปยังพอร์ตอนุกรมของ microcontroller อย่างไรก็ตามก่อนทำการส่งข้อมูลทุกครั้ง ต้องแน่ใจว่าบิต T1 clear หรือว่ามีค่า 0 และเมื่อทำการส่งข้อมูลเรียบร้อยแล้ว จะเกิดการเซตบิต T1 เพื่อแจ้งให้ทราบ ดังตัวอย่างโปรแกรมต่อไปนี้

```
CLR   RI          ; clear ค่าบิต T1 เพื่อเตรียมการส่งข้อมูลออก
JNB   RI,$        ; รอคอยการเซตของบิต RI อันเป็นการแจ้งให้ทราบว่า การรับ
                  ; ข้อมูลเสร็จสมบูรณ์ และมีข้อมูลเกิดขึ้นที่รีจิสเตอร์ SBUF
MOV   A,SBUF      ; อ่านค่าจากรีจิสเตอร์ โดยการโอนย้ายข้อมูลผ่านทางรีจิสเตอร์A
CLR   RI          ; หลังจากการทำการอ่านข้อมูลเรียบร้อยแล้ว ต้องทำการ clear bit
                  ; RI เสมอ
```

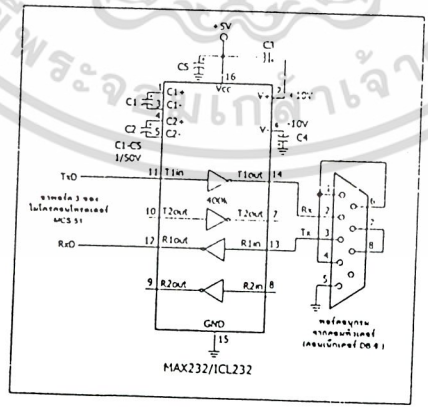
2.2.17 การเชื่อมต่อกับพอร์ตอนุกรมของคอมพิวเตอร์

การใช้งานวงจรพอร์ตอนุกรมของ microcontroller MCS-51 มักนิยมใช้ในการติดต่อแลกเปลี่ยนข้อมูลกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมในมาตรฐาน RS-232 เป็นส่วนใหญ่ แต่เนื่องจากระดับสัญญาณของพอร์ตอนุกรม RS-232 มีระดับตั้งแต่ +3,-3 ถึง +12,-12 ในขณะที่ระดับ

สัญญาณของ microcontroller MCS-51 อยู่ในระดับ TTL ดังนั้นจึงไม่สามารถเชื่อมต่อกับพอร์ตอนุกรม microcontroller MCS-51 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์ได้โดยตรง จึงต้องอาศัยการเชื่อมต่อผ่านไอซีพิเศษที่ทำหน้าที่ในการแปลงสัญญาณ



รูปที่2.18รายละเอียดเบื่องต้น ของไอซีแปลงแปลงสัญญาณเพื่อเชื่อมกับพอร์ตอนุกรมของคอมพิวเตอร์



รูปที่2.19 วงจรเชื่อมต่อ MAX232 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์และ MCS-51

ไอซีที่ทำหน้าที่ในการแปลงระดับสัญญาณนี้ต้องทำการแปลงข้อมูลส่งของ microcontroller MCS-51 จากระดับ TTL ไปเป็นระดับ RS-232 และทำการแปลงข้อมูลรับจากคอมพิวเตอร์จากระดับของ RS-232 เป็นระดับ TTL เพื่อทำให้สามารถถ่ายทอดไปให้ microcontroller MCS-51 ได้อย่างสมบูรณ์ไอซีดังกล่าวมีด้วยกันหลายเบอร์จากหลายผู้ผลิต อาทิ MAX232 จาก MAXIM

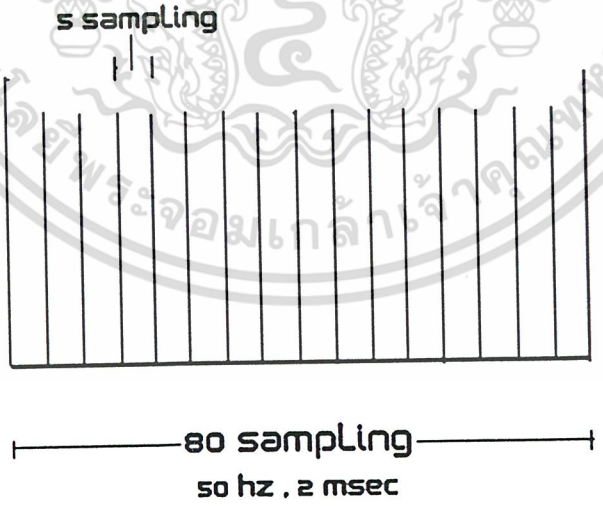
บทที่ 3
การออกแบบโครงงาน

3.1 หุ่นยนต์

ในขั้นตอนแรกของโครงงาน เราได้ทำการออกแบบตัวหุ่น เกี่ยวกับรูปแบบการเคลื่อนที่ของหุ่น การดึงบอลเข้ามา การใช้มอเตอร์ที่เหมาะสม ต่อมาได้ทำการออกแบบวงจรที่ต้องการ เมื่อได้บทสรุปแล้ว เราก็ได้ทำการต่อวงจรขึ้นมา ต่อมา ได้ทำการออกแบบตัวหุ่นยนต์เกี่ยวกับรูปทรง โดยส่วนล้อทำจากพลาสติกและนำโรเลอร์ซึ่งเป็นล้อขนาดเล็กมาต่อซึ่งทำให้ช่วยในการเคลื่อนที่ได้ดีขึ้น และนำส่วนต่างๆมาประกอบกัน

ในส่วนของโปรแกรม แบ่งออกเป็น 2 ส่วน คือโปรแกรมส่วนคอนโทรลเลอร์ และโปรแกรมส่วนของVB ที่ใช้ในการควบคุมผ่านเครื่องคอมพิวเตอร์ โดยจะอธิบายดังนี้

- โปรแกรมในส่วนคอนโทรเลอร์จะใช้ภาษาCสำหรับไมโครคอนโทรเลอร์ในการเขียน ดังรูป โดยโปรแกรมจะใช้ความถี่ 500 Hz คือเท่ากันเวลา 1 ใน 500 หรือ 2 มิลลิวินาที เราได้ทำให้ภายใน 25 microsecond จะเกิดสัญญาณ sampling ขึ้นมา 1 ครั้งดังนั้นจะเกิดสัญญาณ sampling ขึ้นมา80 ครั้ง เราต้องการความเร็วทั้งหมด 16 ระดับ เพราะฉะนั้น ใน1 ระดับจะมีสัญญาณ 5 sampling โดยจะแสดงดังรูปต่อไปนี้



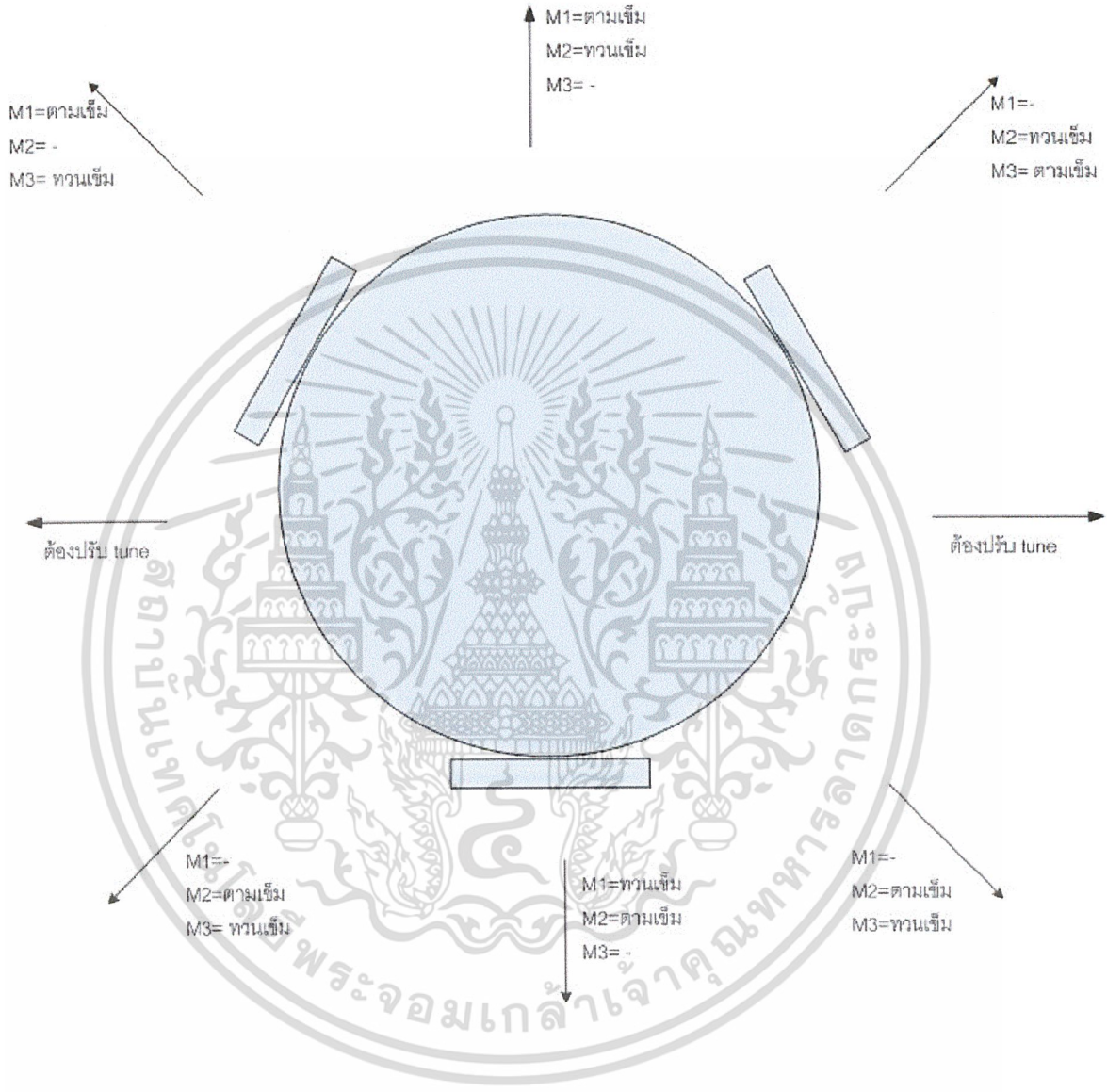
รูปที่ 3.1รูป สัญญาณ sampling ทั้งหมดถูกแบ่งออก16 pulse เท่าๆกัน

เมื่อได้ pulse มาแล้ว การขับเคลื่อนมอเตอร์เราได้จัดให้มีทั้งหมด 16 gears โดยจะแสดงรูปการขับเคลื่อนโดยแบ่งเป็นเกียร์ ต่างๆดังนี้

รูปที่ 3.2 รูป การขับเคลื่อนโดยใช้pulse 1 pulse ในการขับเคลื่อนโดยให้เป็นเกียร์ 1

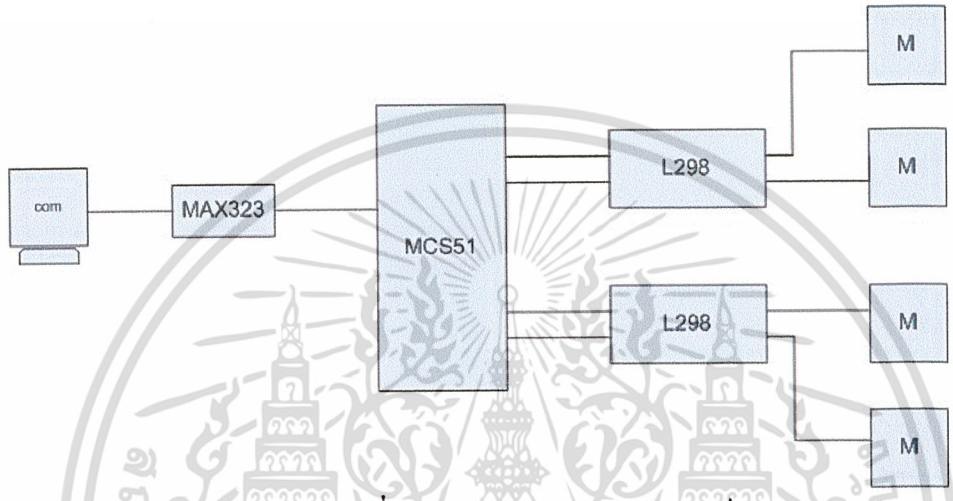
รูปที่ 3.3 รูปการใช้ pulse 5 pulse ให้เป็นเกียร์ 5

3.1.1หลักการเคลื่อนที่ของหุ่นยนต์



รูปที่ 3.4 รูปการหมุนของมอเตอร์ ในทิศทางของหุ่นยนต์ทิศต่างๆ โดย m1=มอเตอร์บนขวา m2= มอร์เตอร์บนซ้าย m3=มอเตอร์ตัวล่าง

ลักษณะของวงจรที่ใช้ภายในตัวหุ่นยนต์มีลักษณะดังรูป โดยจะทำการติดต่อจากคอมพิวเตอร์ผ่านทาง serial port เข้ามายัง MAX323 และผ่านต่อไปยัง ไมโครคอนโทรเลอร์ MCS51 หลังจากนั้นมันจะไปคอนโทรมอเตอร์ผ่านทางไอซี L298 ตามลำดับ



รูปที่ 3.5 ลักษณะวงจรของหุ่นยนต์

3.1.2 ลักษณะของหุ่นยนต์

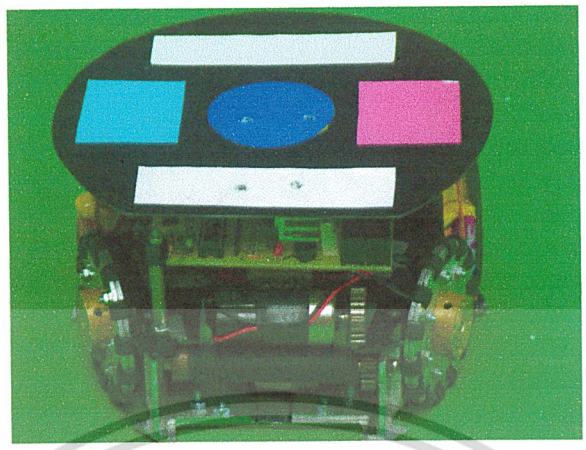
รูปร่าง

- หุ่นยนต์ต้องสามารถบรรจุลงในทรงกระบอกขนาดเส้นผ่านศูนย์กลาง 18 เซนติเมตรได้
- ถ้าหุ่นยนต์ไม่ได้ติดกล้องที่ตัวหุ่นยนต์ หุ่นยนต์ต้องสูงไม่เกิน 15 เซนติเมตร

สี และ สัญลักษณ์

- การแข่งขันแบ่งออกเป็นสองฝ่าย โดยแบ่งเป็นฝ่ายสีเหลือง และ ฝ่ายสีน้ำเงินโดยที่ฝ่ายสีเหลืองต้องยิงประตูสีเหลือง และ ฝ่ายสีน้ำเงินต้องยิงประตูสีน้ำเงิน
- หุ่นยนต์ฝ่ายสีเหลือง และ ฝ่ายสีน้ำเงิน จะถูกติดตั้งสัญลักษณ์วงกลมขนาดเส้นผ่านศูนย์กลาง 4 เซนติเมตรที่มีสีของฝ่ายตนเอง ในตำแหน่งที่มองเห็นได้จากด้านบน
- หุ่นยนต์สามารถใช้สีขาว สีดำ ได้ไม่จำกัด
- หุ่นยนต์สามารถใช้สีอื่น ๆ ที่ได้รับอนุญาตจากกรรมการได้

ตัวอย่างสีที่ได้รับอนุญาตจะพิมพ์บนกระดาษแข็งและแจกให้ผู้แข่งขันล่วงหน้า



รูปที่ 3.6 แสดงส่วนประกอบทั้งหมดของหุ่น

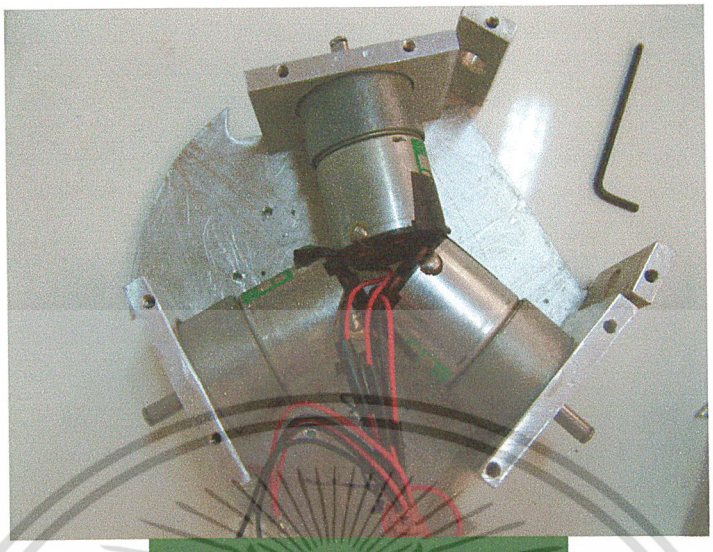
ส่วนประกอบหลักๆของ board ประกอบด้วย

1. MCS-51
2. IC สำเร็จรูปสำหรับการส่งแบบไร้สาย

IC สำเร็จรูป L298

รูปที่ 3.7 แสดงส่วนประกอบของวงจรที่ใช้

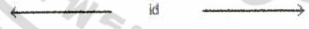
ล้อที่ใช้จะเป็นแบบ omni-direction เพื่อการเคลื่อนที่ ที่เป็นอิสระโดยที่หุ่น 1 ตัวมีล้อทั้งหมด 3 ล้อ โดยที่แต่ละล้อจะวางห่างกันเป็นมุม 120 องศา



รูปที่ 3.8 แสดงส่วนของล้อแบบ omni-direction และตำแหน่งการวาง motor

3.1.3 Protocol

Protocol ที่ใช้ส่งข้อมูลจากคอมพิวเตอร์ ไปยังตัวหุ่นยนต์นั้น มีลักษณะดังนี้

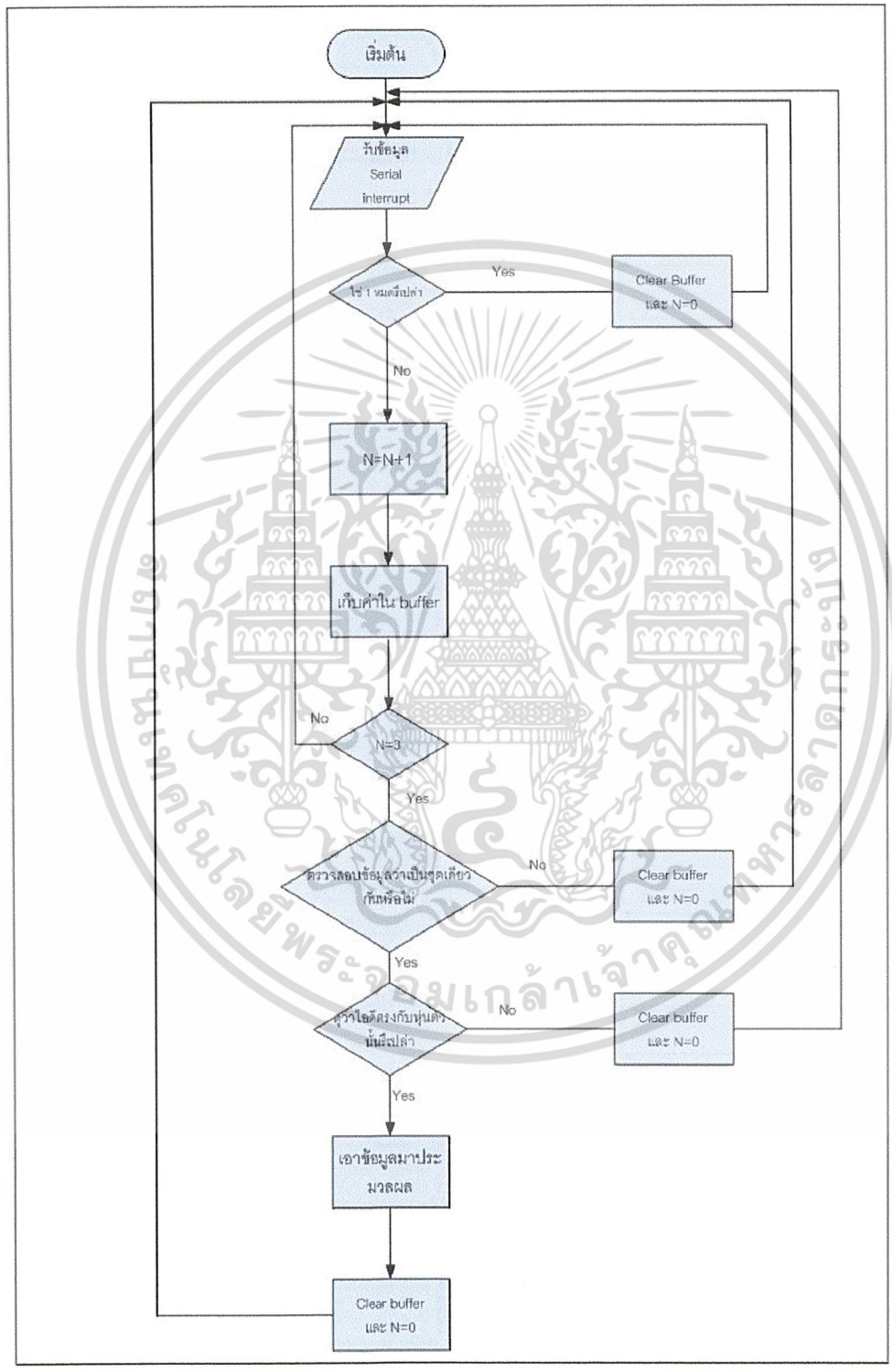


				d1	d2	d3	dribble
V1				V2			
V3				Shoot	id	id	id
1111				1111			

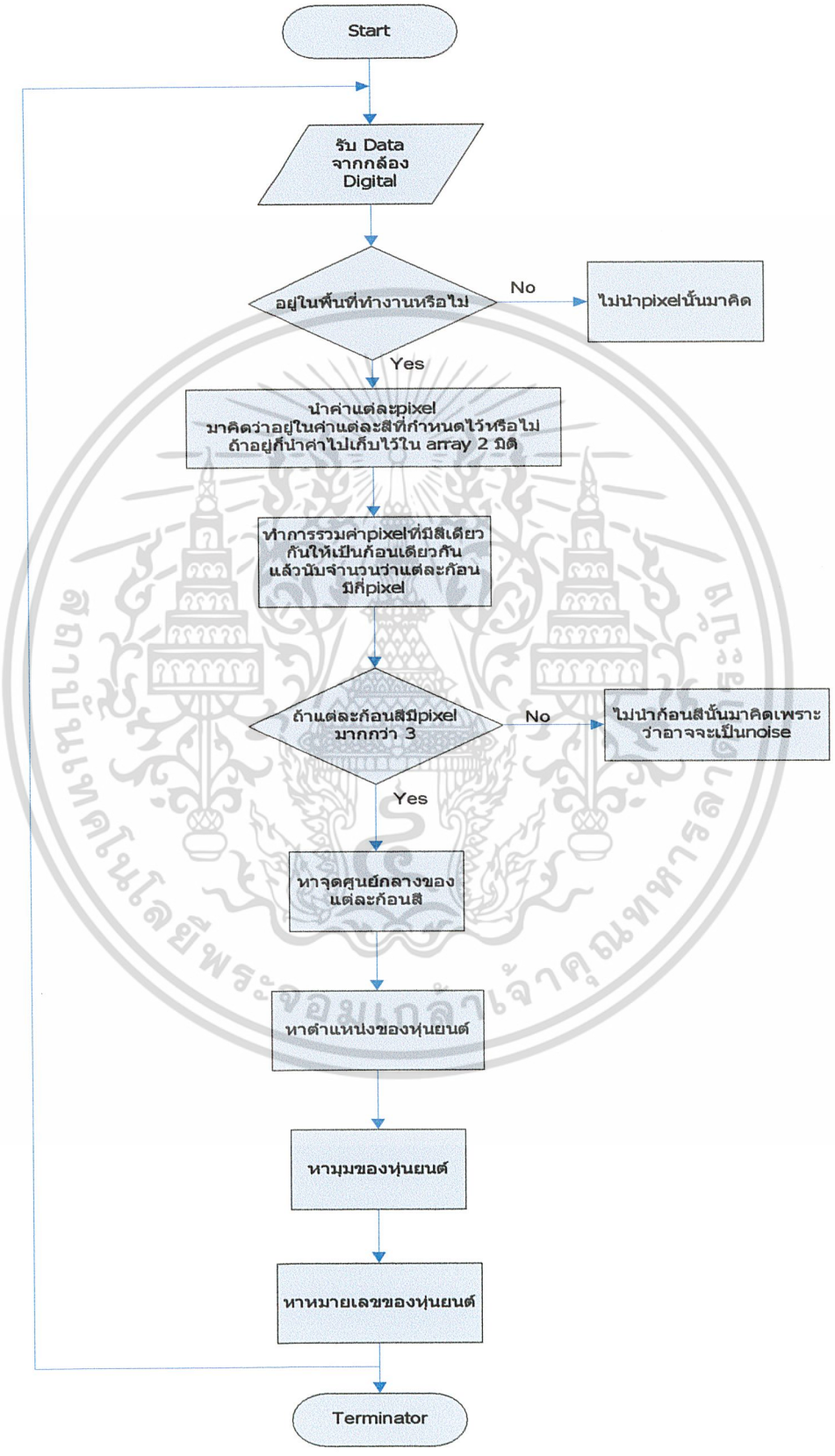
รูปที่ 3.9 แสดง protocol ที่ใช้ในการติดต่อหุ่นยนต์

จากรูป bit แรกจะเป็นฟรีบิต ซึ่งไม่ได้นำมาใช้ ส่วน 3 bit ต่อมา จะเป็น bit ที่บ่งบอกถึงหมายเลขของหุ่นยนต์ และ 3 bit ถัดมาที่เป็น d1,d2,d3 เป็นบิตที่บอกว่าล้อหมายเลขอะไรที่จะให้หมุนบ้าง v1,v2,v3 คือความเร็วของล้อแต่ละล้อ bit shoot เป็น bit ที่ใช้บอกว่าจะให้ยิงรึเปล่า และทำการส่ง id ของหุ่นยนต์เพื่อยืนยันอีกครั้ง และปิดด้วย bit ที่เป็น 1 หมด

Flow Chart แสดงอัลกอริทึมในกระบวนการของ robot



Flow Chart แสดงอัลกอริทึมในกระบวนการ Image Processing



บทที่ 4

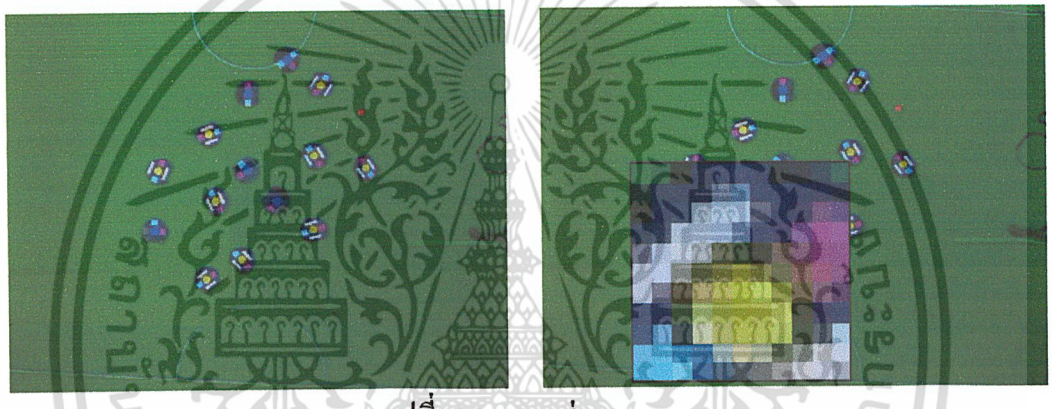
การทดลอง

4.1 Image Processing

ขั้นตอนของกระบวนการ Image Processing ประกอบด้วย

- ขั้นแรกเป็นการนำภาพที่รับมาจากกล้อง Digital มา ซึ่งภาพที่รับเข้ามาจะเป็นภาพใน RGB Model ซึ่งการต้องนำมาแปลงให้อยู่ใน HSI Model

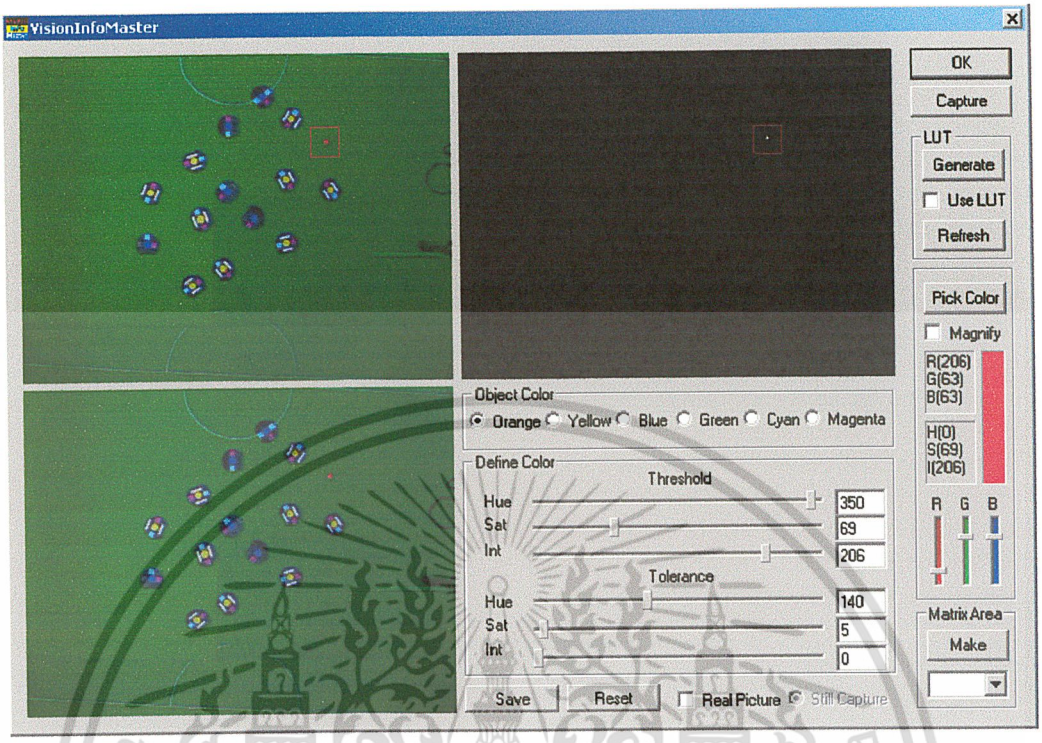
- ต่อมาในส่วนของการกำหนดค่าของแต่ละสีที่ใช้เนื่องจากว่าค่าสีที่ได้รับมาจากกล้องสีที่ได้ไม่ได้เป็นสีเดียวกันต้องมีการกำหนดค่าช่วงของค่าสีที่ยอมรับได้



รูปที่ 4.1 แสดงส่วนขยาย

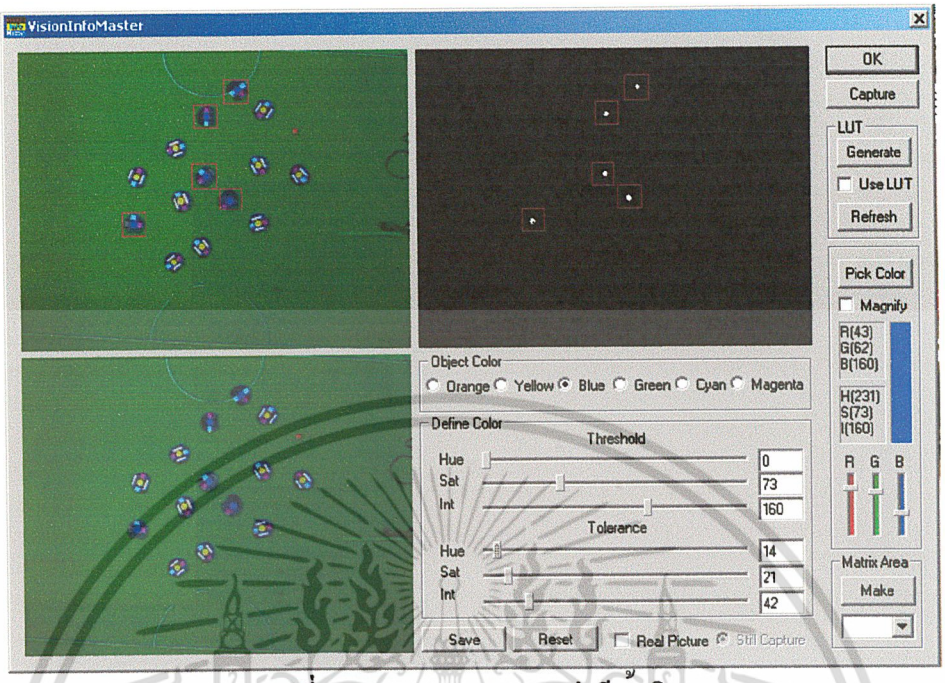
ดังรูปแสดงให้เห็นว่าเมื่อให้ขยายภาพออกมาจะเห็นว่าสีเหลืองที่เห็นจะไม่ได้เป็นสีเหลืองเดียวกันหมด และสีอื่นๆด้วย เพราะฉะนั้นการที่จะกำหนดค่าจะกำหนดแบบตายตัวไม่ได้ต้องกำหนดเป็นช่วง ส่วนสีที่ใช้ในการคำนวณมีด้วยกัน 5 สีด้วยกันคือ

1. สีส้ม เป็นสีของลูกบอล
2. สีเหลือง เป็นสีที่ใช้ในการแบ่งฝ่าย
3. สีน้ำเงิน เป็นสีที่ใช้ในการแบ่งฝ่าย
4. สีม่วง เป็นสีที่ใช้ในการหามุมของหุ่นยนต์
5. สีฟ้า เป็นสีที่ใช้ในการหาหมายเลขของหุ่นยนต์



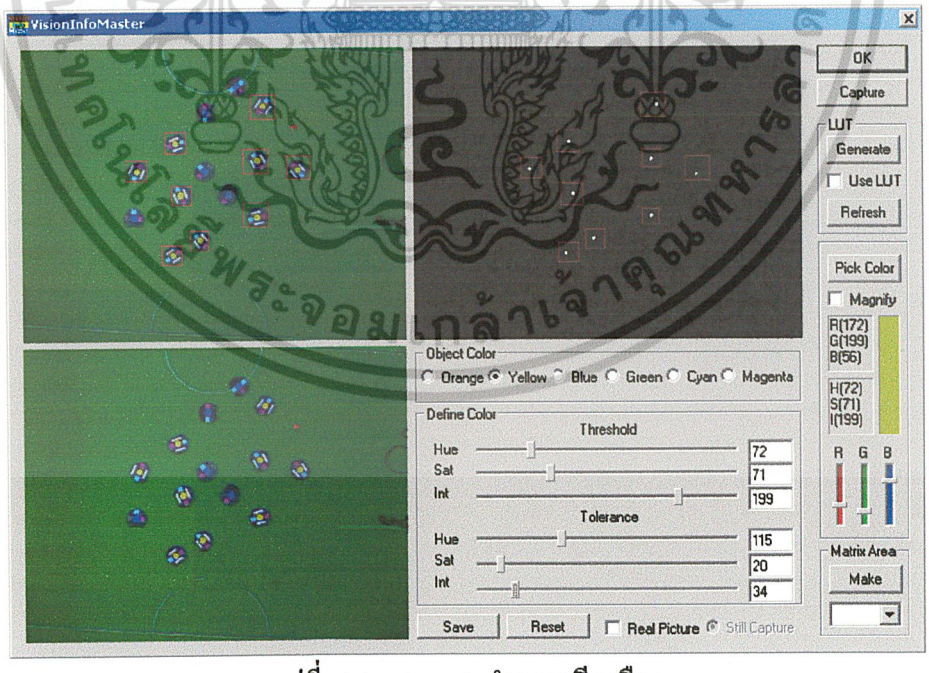
รูปที่ 4.2 การกำหนดสี

ภาพนี้เป็นการแสดงการกำหนดค่าของสีส้มซึ่งเป็นสีของลูกบอลที่ใช้ ในพื้นที่ที่เป็นสีดำแสดงให้เห็นว่าค่าที่เราเลือกมานี้ตรงกับค่า จุด ไหนในรูป จากค่าที่ได้แสดงให้เห็นค่า Threshold ของสีส้มโดยมีค่า Hue = 350, Saturation = 69, Intensity = 206 และค่า Tolerance ของสีส้มโดยมีค่า Hue = 140, Saturation = 5, Intensity = 0 จากค่าที่ได้ทั้งหมดนี้เราสามารถสรุปได้ว่าช่วงของค่าสีส้มคือค่า Hue ที่มีค่าตั้งแต่ 210 – 490 และ ค่า Saturation ที่มีค่าตั้งแต่ 64 – 74 และค่า Intensity ที่มีค่าตั้งแต่ 206 – 206



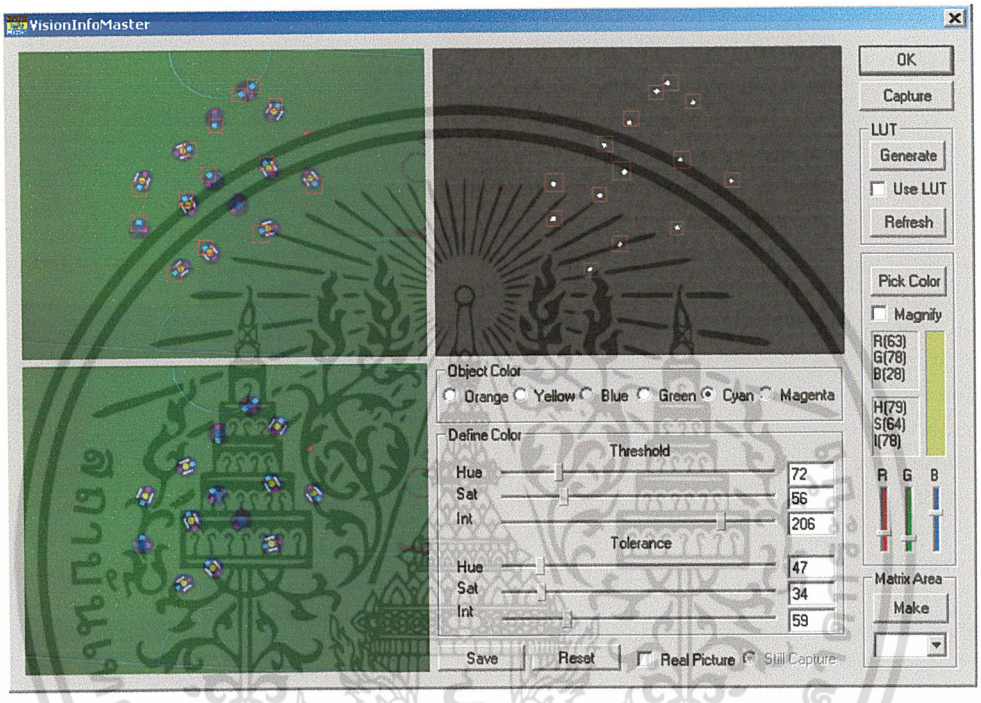
รูปที่ 4.3 แสดงการกำหนดค่าสีน้ำเงิน

ภาพแสดงการกำหนดค่าสีน้ำเงินโดยมีค่า Hue = -14 ถึง 14, Saturation = 52 ถึง 94, Intensity = 118 ถึง 202

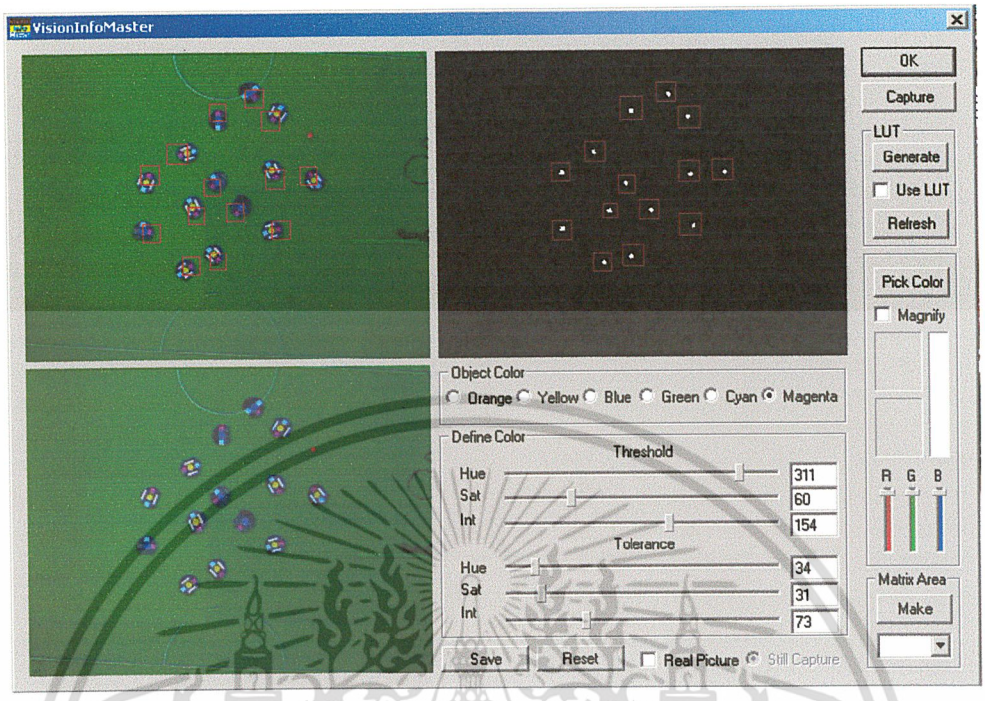


รูปที่ 4.4 แสดงการกำหนดสีเหลือง

ภาพแสดงการกำหนดค่าสีเหลืองโดยมีค่า Hue = -43 ถึง 187, Saturation = 51 ถึง 91, Intensity = 165 ถึง 233

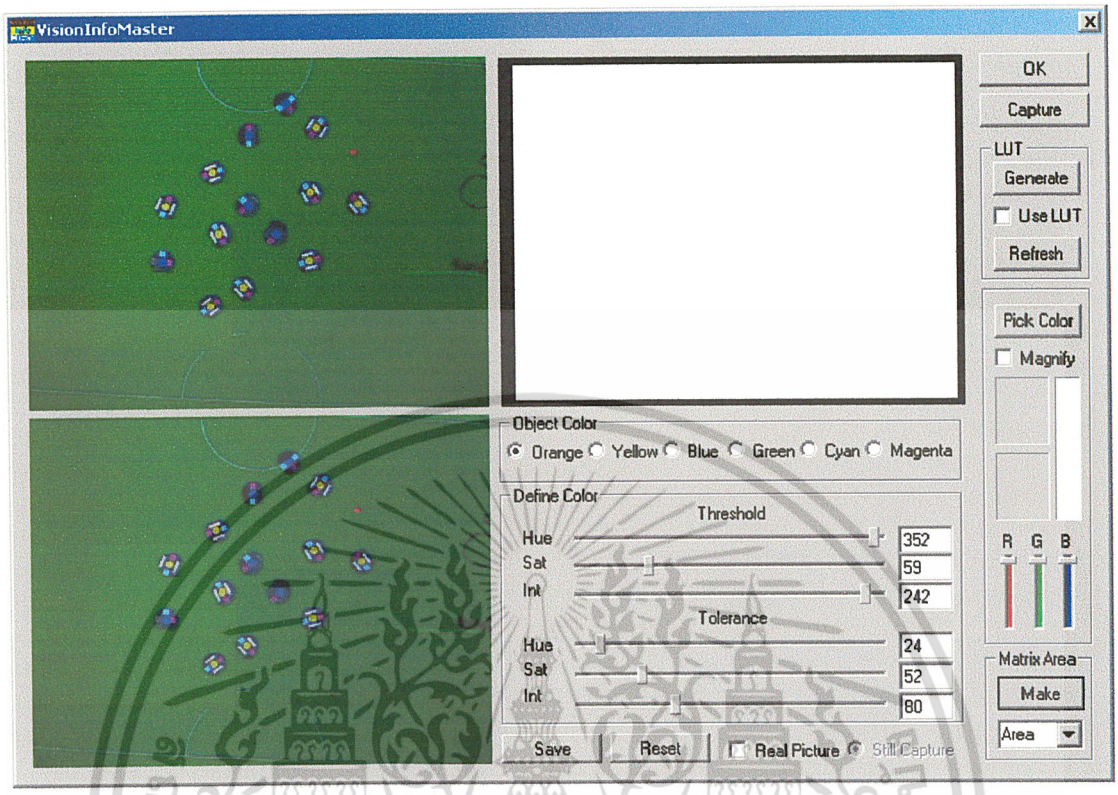


รูปที่ 4.5 ภาพแสดงการกำหนดค่าสีฟ้าโดยมีค่า Hue = 25 ถึง 119, Saturation = 32 ถึง 90, Intensity = 147 ถึง 265



รูปที่ 4.6 ภาพแสดงการกำหนดค่าสีม่วงโดยมีค่า Hue = 277 ถึง 345, Saturation = 29 ถึง 91, Intensity = 81 ถึง 227

- ขั้นต่อมาเราต้องกำหนดพื้นที่ของการทำงานของโปรแกรมเพราะว่าในบางครั้งบริเวณขอบของภาพอาจมีสิ่งรบกวนได้ทำให้การคำนวณผิดพลาดได้

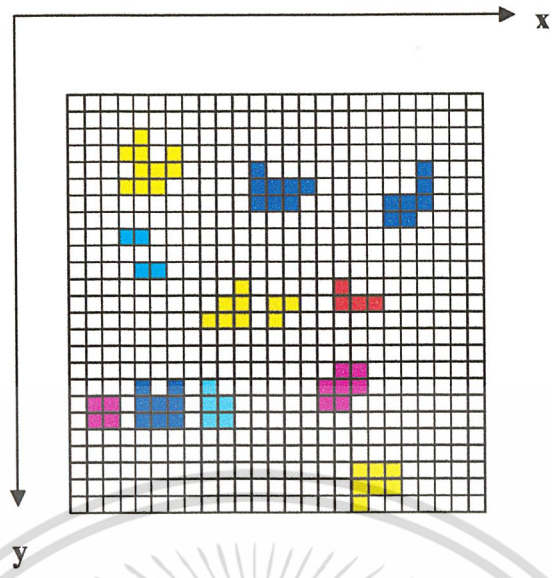


รูปที่ 4.7 แสดงส่วนที่ใช้ในการคำนวณพิกัดของภาพที่รับมา

โดยบริเวณที่เป็นสีขาวก็จะใช้ในการคำนวณแต่ถ้าเป็นบริเวณที่เป็นสีดำก็จะไม่นำค่า จุดของบริเวณนั้นมาใช้ในการคำนวณ

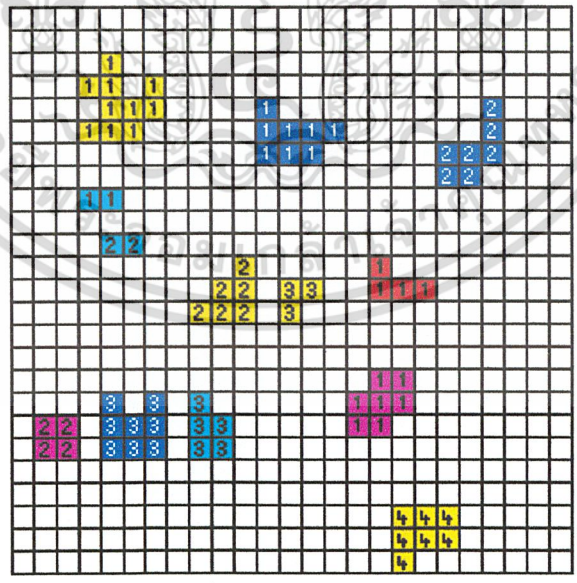
-ในส่วนของการคำนวณหลังจากที่ได้กำหนดค่าของสีและบริเวณที่จะคำนวณแล้ว คือการหาตำแหน่งของหุ่นยนต์ การหามุมของหุ่นยนต์ และการหาหมายเลขของหุ่นยนต์

วิธีการรวมก้อนของสีทำโดยต้องทำการประมวลผลทุกจุด ไล่ตั้งแต่จุดแรกไปถึงจุด สุดท้าย โดยแต่ละจุด จะตรวจดูจุดรอบข้าง4จุดคือ (x-1, y), (x-1, y-1), (x, y-1), (x+1, y-1) ว่าได้มีการกำหนดค่าให้หรือยังถ้ายังไม่ได้กำหนดค่าก็ต้องกำหนดให้เช่นถ้าจุดแรกที่พบเป็นสีเหลือง(5, 3)ยังไม่ได้กำหนดค่าเพราะฉะนั้นต้องกำหนดให้



รูปที่ 4.8 การจัดกลุ่มสี

เป็น 1 หลังจากนั้นสีที่พบต่อมาคือสีเหลืองที่พิกัด(4, 4) แต่เมื่อได้ดูจุดรอบข้าง4จุดแล้ว พบว่าที่จุด(5, 3) ซึ่งเป็นสีเหลืองเหมือนกันมีค่าเป็น 1 เพราะฉะนั้นค่าในตำแหน่งนี้มีค่าเป็น 1 ด้วย จากรูปภาพนี้ถ้าทำการประมวลผลครบจะได้ค่าดังรูป ซึ่งจะเห็นได้ว่าสามารถบอกได้ว่ามีสีเหลืองอยู่ 4 ก้อน สีน้ำเงิน 3 ก้อน สีส้ม 1 ก้อน สีฟ้า 3 ก้อน และ สีม่วง 2 ก้อน

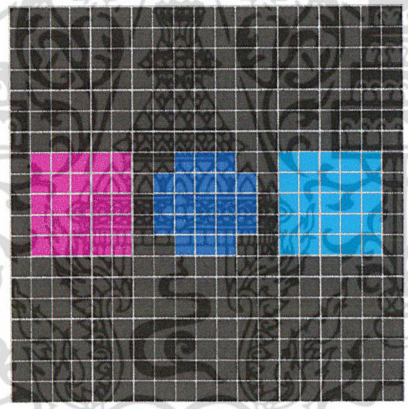


รูปที่ 4.9 การจัดกลุ่มสี

หลังจากที่ได้ทำการรวมก้อนของสีแล้วก็มาทำการหาค่าจุดศูนย์กลางของแต่ละก้อน โดยการหาผลบวกของค่าในแนวแกน x ทั้งหมดของก้อนแล้วหารด้วยจำนวนจุดทั้งหมดในก้อน และในแนวแกน y ก็ทำเหมือนกัน เช่น ตำแหน่งศูนย์กลางของสีเหลืองก้อนที่1 ค่า x = (5+4+5+7+5+6+7+4+5+6) / 10 = 5.4 และค่า

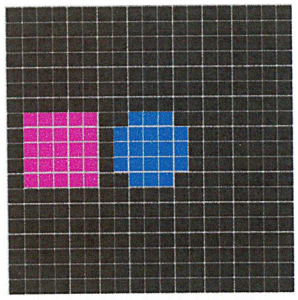
y = (3+4+4+4+5+5+5+6+6+6) / 10 = 5.2 เพราะฉะนั้นตำแหน่งศูนย์กลางของสีเหลืองก้อนที่ 1 เท่ากับ (5.4, 5.2)

การหามุมของหุ่นยนต์ทำได้โดยการใช้สีม่วงกับสีน้ำเงิน ด้วยการยึดการพิจารณาจากสีน้ำเงิน(เหลือง)เป็นหลักโดยจะพิจารณาเป็นสี่เหลี่ยมโดยวัดจากจุดศูนย์กลางสีน้ำเงิน(เหลือง)ไปเป็นรัศมี 10 หน่วย เพราะว่าเป็นบริเวณของตัวหุ่นยนต์ หลังจากที่หาค่าจุดศูนย์กลางของทั้งสีน้ำเงินและสีม่วงแล้วโดยให้เป็นค่า (x1, y1) และ (x2, y2) ตามลำดับ นำค่าที่ได้มาทำการหามุมโดยใช้สูตร มุม = arctan((y2 – y1) / (x2 – x1))

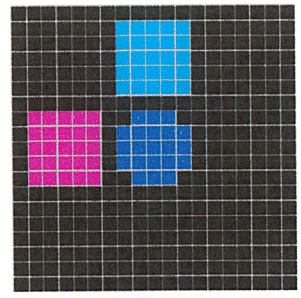


รูปที่ 4.10 แสดงหมายเลขของหุ่นยนต์

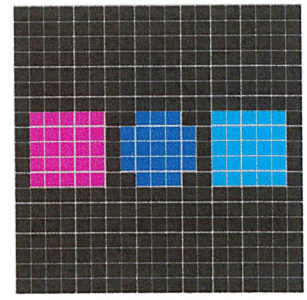
การหาหมายเลขของหุ่นยนต์เนื่องมาจากว่าหุ่นยนต์มีมากกว่า 1 ตัว เพราะฉะนั้นต้องบอกด้วยว่าหุ่นนั้นๆเป็นหุ่นหมายเลขใดเพื่อใช้ในการสั่งการ โดยสีที่ใช้ในการแบ่งแยกหมายเลขหุ่นคือสีฟ้าโดยการกำหนดตำแหน่งของสีฟ้าดังนี้



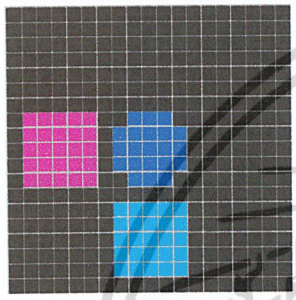
หุ่นยนต์หมายเลข 0



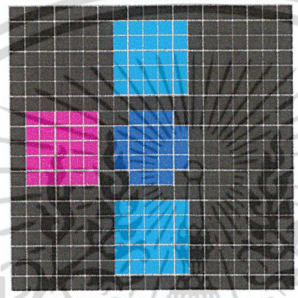
หุ่นยนต์หมายเลข 1



หุ่นยนต์หมายเลข 2



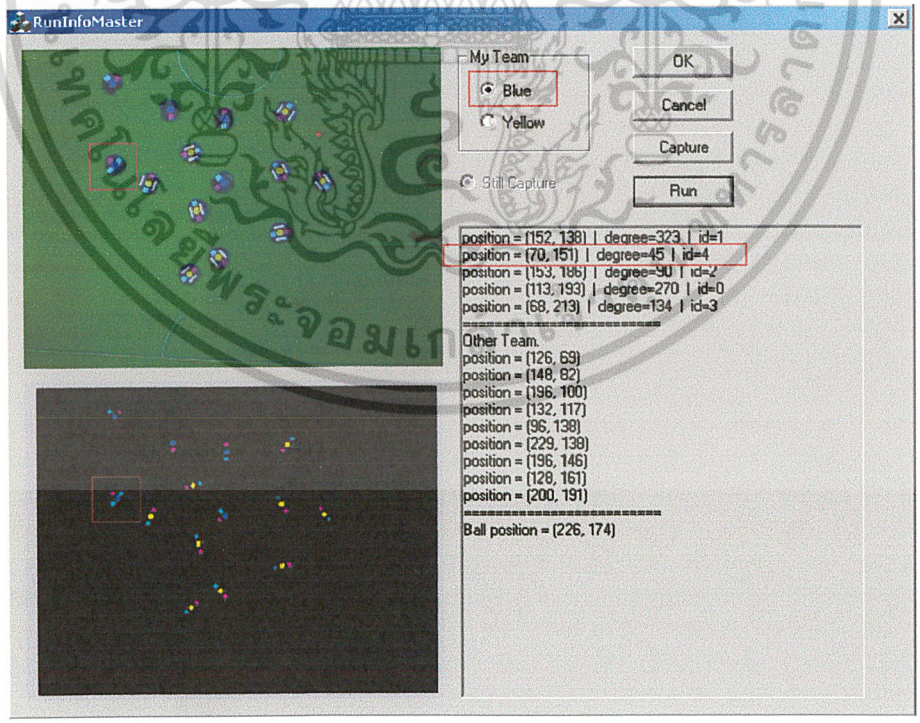
หุ่นยนต์หมายเลข 3



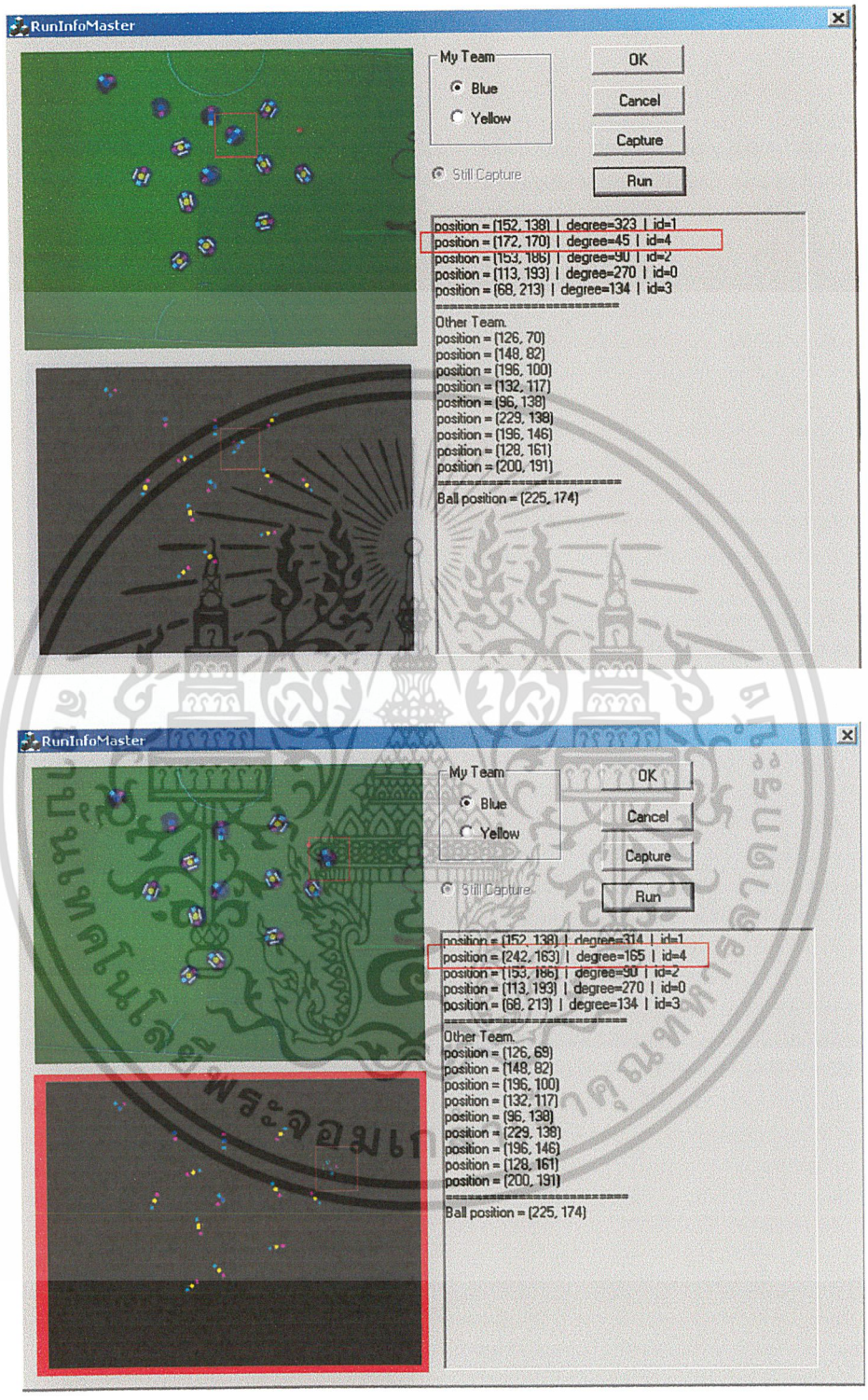
หุ่นยนต์หมายเลข 4

รูปที่ 4.11 แสดงหมายเลขต่างๆของหุ่นยนต์

ตัวอย่างโปรแกรม

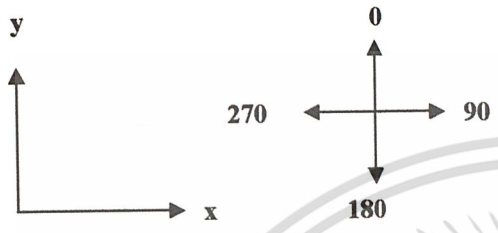


รูปที่ 4.12 ตัวอย่างโปรแกรม

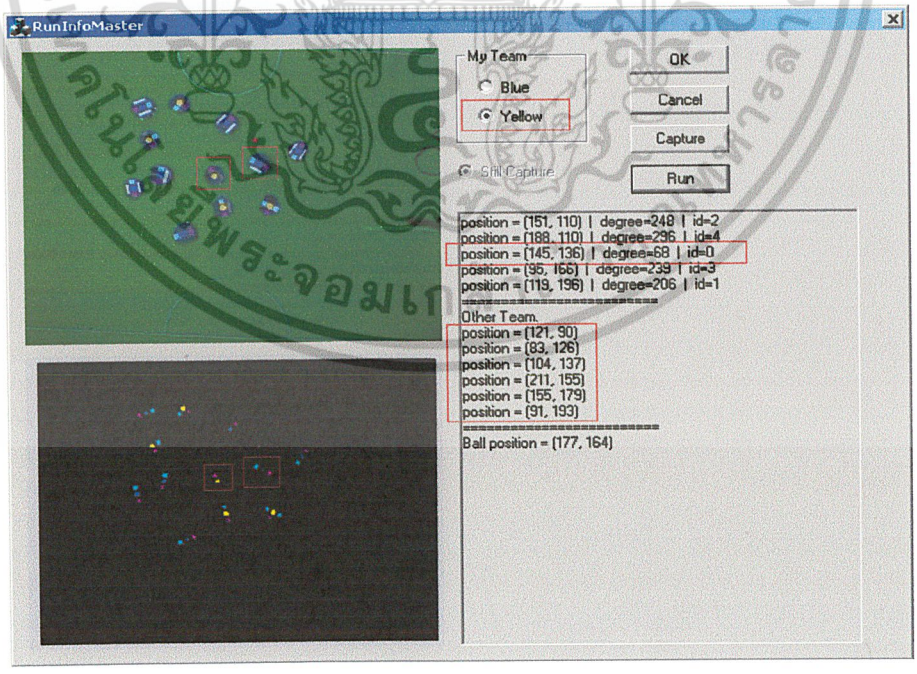


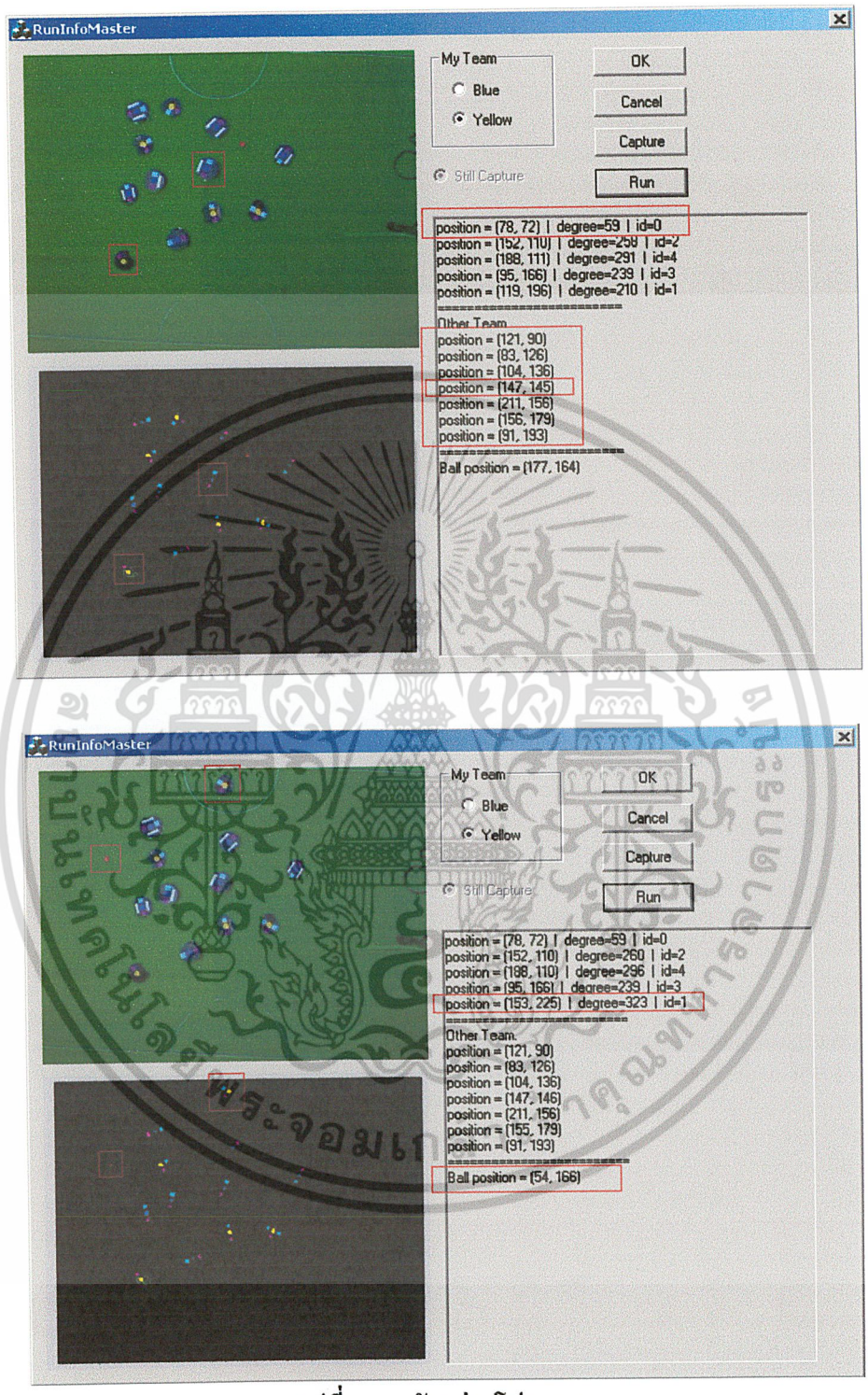
รูปที่ 4.13 ตัวอย่างโปรแกรม

สามภาพด้านบนแสดงให้เห็นการเปลี่ยนแปลงตำแหน่งของหุ่นยนต์โดยที่สามารถเลือกได้ว่าจะใช้สีใดระหว่างสีน้ำเงินกับสีเหลือง ในที่นี้เลือกสีน้ำเงินจะเห็นว่าภาพแรกหุ่นยนต์หมายเลข 4 มีตำแหน่งอยู่ที่



(70, 150) มุมเท่ากับ 45 องศา โดยที่ตำแหน่ง(0, 0) อยู่ที่มุมซ้ายล่าง และ มุมที่ได้จะเป็นมุมที่ทำกับแกนดังรูปซ้ายมือ ภาพที่สอง แสดงให้เห็นการเปลี่ยนแปลงตำแหน่งโดยหุ่นยนต์หมายเลข 4 ได้ย้ายไปอยู่ที่ตำแหน่ง (172, 170) ทำมุม 45 องศา ภาพที่สามตำแหน่งของหุ่นยนต์หมายเลข 4 อยู่ที่ (242, 163) และทำมุม 165 องศา ภาพสุดท้ายที่แสดงเป็นสีแดงเพื่อให้เห็นว่าเกิดจากตอนแรกที่ต้องกำหนดพื้นที่ของการทำกระบวนการ Image Processing พื้นที่บริเวณสีแดงเป็นพื้นที่ที่กำหนดว่าไม่ต้องทำการประมวลผลเพราะฉะนั้นพื้นที่บริเวณนี้จะไม่ได้รับการประมวลผลเลยเห็นได้จากภาพที่ 1 และ ภาพที่ 2 ส่วนที่แสดงตำแหน่งของสีจะเล็กว่าขนานภาพจริงที่ได้รับมาจากกล้อง เพราะฉะนั้นไม่ว่าบริเวณนั้นจะมีการเกิด noise มากเท่าไรก็ไม่มีผลกับการประมวลผล





รูปที่ 4.14 ตัวอย่างโปรแกรม

สามภาพถัดมาแสดงถึงกรณีที่เราเปลี่ยนมาใช้สีเหลืองเป็นสีของทีมจะเห็นว่า หุ่นยนต์หมายเลข 0 จะอยู่ในตำแหน่ง (145,136) มุม 68 องศา และจะเห็นว่า ตรงตำแหน่งของ other team เราจะเห็นหุ่นยนต์ฝ่ายตรงข้าม มี 6 ตัว ซึ่งจากหุ่นยนต์ที่มีจริงๆนั้น จะมีหุ่นยนต์ฝ่ายสีน้ำเงินอยู่ 7 ตัว

ซึ่งเกิดจากปัญหาบางประการของกล้อง เช่นการที่แสงไม่เพียงพอ ทำให้กล้องจับภาพได้ไม่ดีเท่าที่ควร และเมื่อเราลองย้ายหุ่นยนต์สีน้ำเงินที่ตอนแรก ไม่สามารถจับภาพได้ ไปไว้ในตำแหน่งอื่น จะเห็นได้ดงรูปที่ 2 คือ other team จะมีหุ่นยนต์ฝ่ายตรงข้ามเพิ่มขึ้นมาอีก 1 ตัว เป็น 7 ตัว และรูปที่ 3 แสดง การเปลี่ยนตำแหน่งของลูกบอล จะเห็นว่าตำแหน่งลูกบอลมีการเปลี่ยนแปลง

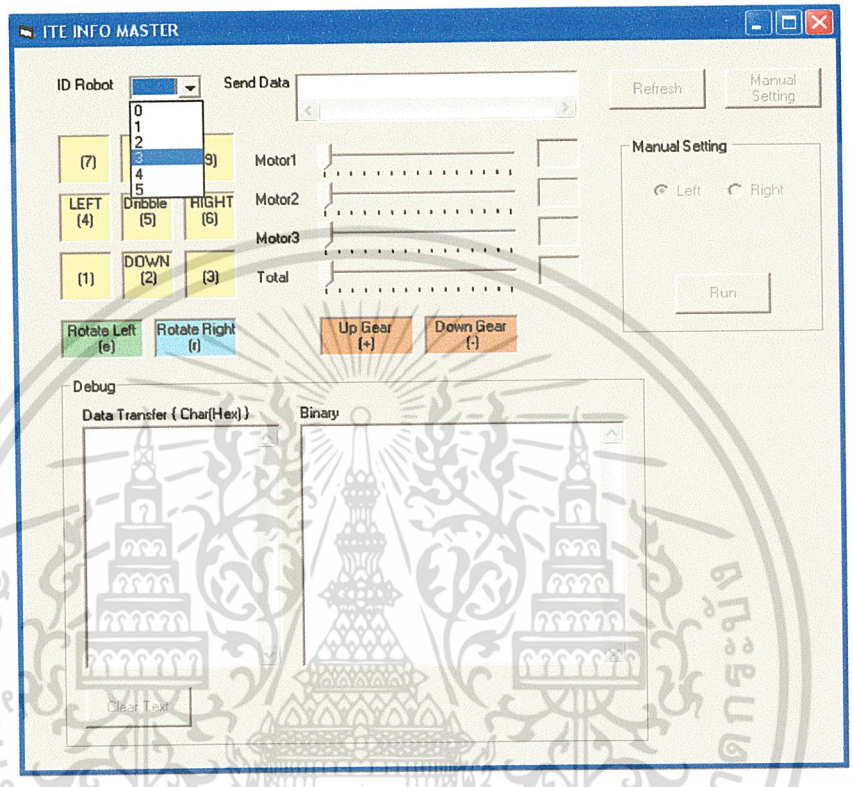
4.2 การทดลองในส่วนของหุ่นยนต์

วิธีการทดลองนั้นเราจะนำโปรแกรมที่เขียนจาก VB ซึ่งเป็นโปรแกรมควบคุมหุ่นยนต์ผ่านทาง keyboard ติดต่อทาง serial port และใช้สัญญาณ wireless ในการส่งสัญญาณไปยังตัวหุ่นให้ทำงาน การทดลองนี้จะทำการจับเวลาการเคลื่อนที่ของหุ่นยนต์ในระยะทาง 1 เมตร โดยจะเปลี่ยนไปทีละเกียร์ จากเกียร์ 1 ถึง 14

ผลการทดลองออกมาดังนี้

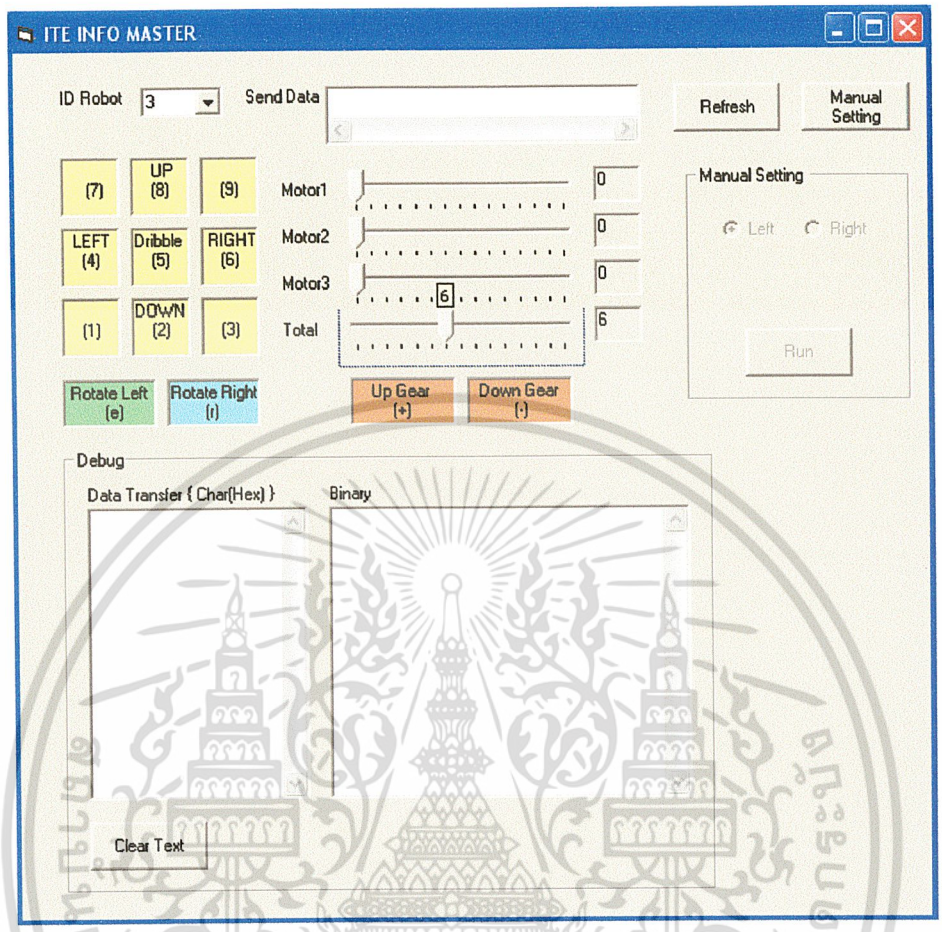
เกียร์ 1 ไม่สามารถเคลื่อนที่ได้
เกียร์ 2 ไม่สามารถเคลื่อนที่ได้
เกียร์ 3 ใช้เวลา 9.27 วินาที
เกียร์ 4 ใช้เวลา 6.57 วินาที
เกียร์ 5 ใช้เวลา 4.95 วินาที
เกียร์ 6 ใช้เวลา 4.1 วินาที
เกียร์ 7 ใช้เวลา 3.65 วินาที
เกียร์ 8 ใช้เวลา 3.14 วินาที
เกียร์ 9 ใช้เวลา 2.87 วินาที
เกียร์ 10 ใช้เวลา 2.43 วินาที
เกียร์ 11 ใช้เวลา 2.21 วินาที
เกียร์ 12 ใช้เวลา 1.89 วินาที
เกียร์ 13 ใช้เวลา 1.72 วินาที
เกียร์ 14 ใช้เวลา 1.58 วินาที

โปรแกรม VB ที่ใช้ในการควบคุมในคอมพิวเตอร์นั้นใช้ในการควบคุมหุ่นโดยใช้ Keyboardในการบังคับหุ่นยนต์ โดยตัวโปรแกรมจะติดต่อผ่านทาง serial port ดังรูป



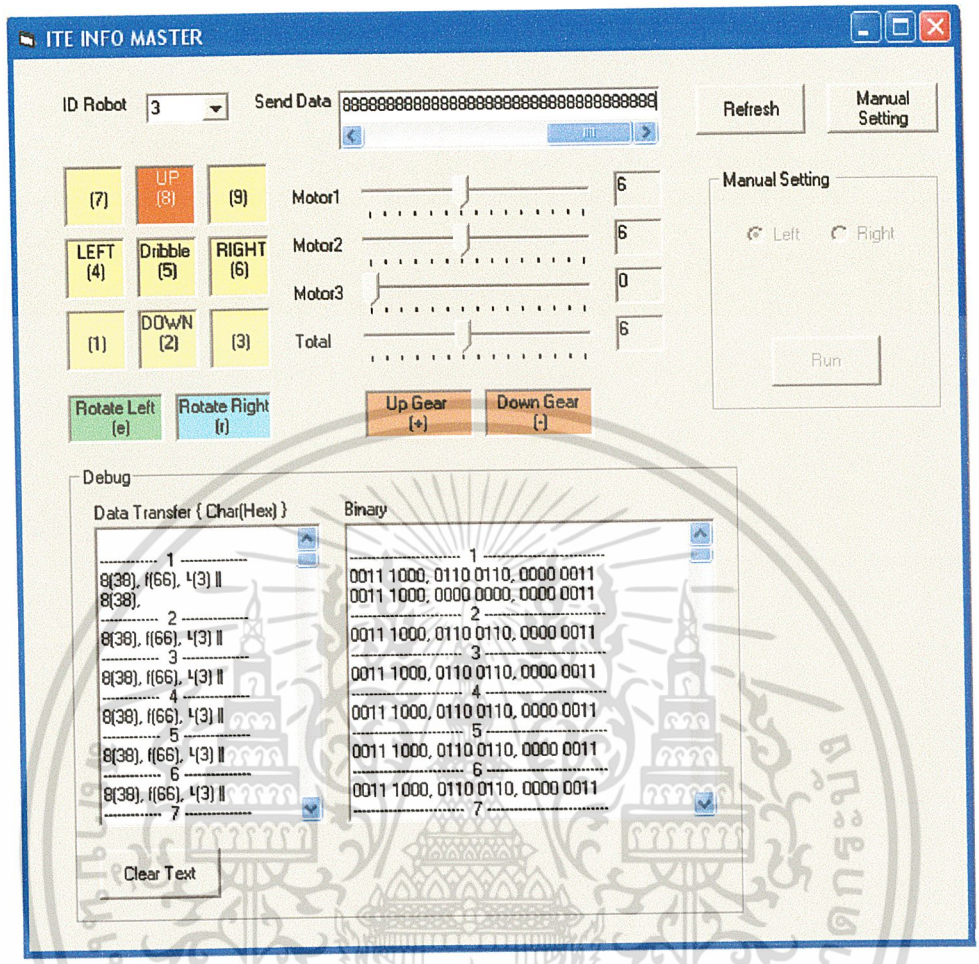
รูปที่ 4.15 ตัวอย่างโปรแกรม

- ขั้นแรก เราต้องเลือก ID robot ว่าเราจะควบคุมหุ่นตัวไหน



รูปที่ 4.16 ตัวอย่างโปรแกรม

- ต่อมาเลือกเกียร์ที่ต้องการที่ตำแหน่ง total หรือเราอาจจะเปลี่ยนโดยทาง keyboard ก็ได้โดยกดปุ่ม + เพื่อเพิ่มเกียร์ ปุ่ม – เพื่อลดเกียร์



รูปที่ 4.17 ตัวอย่างโปรแกรม

- เมื่อเซตทุกอย่างแล้ว เราสามารถกดบังคับหุ่นได้ตามต้องการโดยใช้ปุ่ม ตัวเลขดังนี้ 8(เดินหน้า),4(ซ้าย),6(ขวา),2(ถอยหลัง)

บทที่ 5
สรุปผลการทดลอง

5.1 สรุปผลการทดลอง

การทดลองการทำงานของหุ่นยนต์เบื้องต้นเพื่อใช้เป็นพื้นฐานที่สำคัญ ในการทำหุ่นยนต์เตะฟุตบอล(Soccer Robot) ต่อไปซึ่งการทำงานของหุ่นดีขึ้นกว่าครั้งแรกที่ได้ทำการทดสอบแต่การส่งสัญญาณ wireless ที่ยังมีการส่งที่ผิดพลาดเล็กน้อย ทั้งยังมีปัญหาในการเคลื่อนที่ของหุ่นยนต์ที่จำเป็นจะต้องมีการ tune การเคลื่อนที่ที่เหทาะสมซึ่งในบางครั้งการที่เราสวมล้อไม่แน่นจะทำให้หุ่นเคลื่อนที่ผิดไปจากที่เคย tune ไว้

ในส่วนของโปรแกรม VB นั้นได้มีการพัฒนาจากการควบคุมโดย keyboard เราได้พัฒนาให้ใช้งานได้กับ joystick ซึ่งจะทำให้การเคลื่อนที่ของหุ่นยนต์สามารถควบคุมได้ง่ายขึ้น

ในส่วนของโปรแกรม Image Processing นั้น จากการทดลองจะเห็นได้ว่าปัญหาเดียวที่เกิดขึ้นคือปัญหาเรื่องของแสงสว่าง ที่มีแสงสว่างไม่ทั่วทั้งสนามซึ่งในการแข่งขันจริงๆ การที่แสงไม่มีความสม่ำเสมอทั้งสนามนั้น จะทำให้การจับภาพของโปรแกรม เกิดการผิดพลาดขึ้นได้ ซึ่งถ้า Image Processing ทำงานผิดพลาด จะทำให้ระบบของ robocup ถึงกับทำงานไม่ได้เลยทีเดียว แต่ถ้าแสงมีความสม่ำเสมอ จะเห็นได้ว่า การรับภาพจากกล้องเข้ามาประมวลผล โปรแกรมก็สามารถทำงานได้ในเวลาที่ค่อนข้างจะดีเลยทีเดียวนั่นคือ 20 frame/sec ซึ่งถือว่าเพียงพอที่จะบอกว่าเป็นการจับภาพแบบ real time

5.2 ปัญหาที่พบในการดำเนินการทดลอง

1. ช่วงแรกของการเขียนโปรแกรม Image Processing ซึ่งต้องมีการศึกษาภาษา C++ กันใหม่ ทำให้ต้องใช้เวลาในการศึกษานานพอสมควร

2. ปัญหาที่เกิดจากเครื่องส่งสัญญาณวิทยุ (RF) ซึ่งเครื่องส่งมักมีปัญหาในการส่งทำให้ข้อมูลที่ส่งไปให้หุ่นยนต์เกิดความผิดพลาด

3. แสงสว่างที่ใช้ในการทดลอง Image Processing กับในการแข่งขันจริงมีความสม่ำเสมอไม่เท่ากัน ซึ่งแน่นอนว่า การที่แสงสว่างไม่สม่ำเสมอทำให้การจับภาพของ Image Processing จะมีความผิดพลาดเกิดขึ้นได้

4. หุ่นยนต์ที่ใช้มีความแตกต่างของมอร์เตอร์แต่ละตัวที่แตกต่างกันในด้านความเร็ว ทำให้เวลาที่จะกำหนดความเร็วของหุ่นยนต์จะต้องทำการมากำหนดความเร็วของหุนยนต์แต่ละตัวซึ่งทำให้เกิดการเสียเวลา และมีความไม่แน่นอนสูง

5.3 แนวทางในการพัฒนาโครงงาน

1. จะต้องมีการปรับเปลี่ยนตัวส่งสัญญาณวิทยุที่มีคุณภาพกว่านี้

2. จะต้องมีการปรับปรุงโปรแกรม Image Processing ให้มีความยืดหยุ่นในการจับภาพที่ดีกว่านี้ ซึ่งก็เป็นการยากอีกเช่นกัน ดังนั้นจะต้องมีการควบคุม ความเข้มของแสงในสนามให้มีความคงที่

3. จะต้องทำการปรับปรุงตัวหุ่นยนต์ ให้มีคุณภาพที่ดีขึ้น

บรรณานุกรม

1. Maher .A.SidAhhmed,"Image Processing theory algorithms&architecture", McGraw-Hill International , 1992
2. http://www.st.com
3. http://www.geocities.com/phoj2002/microcontroller.pdf
4. Omni-drive dynamics http://www.cs.oz.au

ภาคผนวก

```
1  //*****************************************************************************
2  //*****************************************************************************
3
4  // modify use sync fac "1111 1111" and use buffer 4 buffer
5  // if buffer3 = "1111 1111" => if buffer0 != "0000 0000" (empty buffer) => if id = IDROBOT =
6  // by surachai      date 5/01/48   time 14.41 P.M.
7  // use new ic AT89C4051 (20 Pin) output port0
8
9  // new edit because direction motor change( motor 1, motor  direction convert) for new robot
10 // by surachai  date 16/1/48 time 2.56 A.M.
11
12 // new edit because use old direction for old robot and id change from 0 to 2
13 // by surachai date 3/2/48 time 23.25 P.M.
14 //*****************************************************************************
15 //*****************************************************************************
16 #include <reg52.h>
17
18 #define COUNTER_CLOCKWISE   0   // define direction motor counter clockwise = 0
19 #define CLOCKWISE           1   // define direction motor clockwise = 1
20
21 #define STOP                0   // define event for motor shoot and dribble and stop
22 #define DRIBBLE             1   // define event for motor shoot and dribble and stop
23 #define SHOOT               2   // define event for motor shoot and dribble and stop
24
25 #define TIME_INVERT_M_1     2   // 1 gear = 2 pulse sampling (15 gear)
26 #define TIME_INVERT_M_2     2   // 1 gear = 2 pulse sampling (15 gear)
27 #define TIME_INVERT_M_3     2   // 1 gear = 2 pulse sampling (15 gear)
28 #define TIME_INVERT_M_4     2   // 1 gear = 2 pulse sampling (15 gear)
29 #define ALL_PULSE           30  // all have 30 pulse sampling
30
31 #define ID_ROBOT            2   // define this robot is id = 1
32 //#define DEFAULTDRIBBLE    13
33 // motor1 = port0.0 - 0.1 ** motor2 = port 0.2 - 0.3 **
34 // motor3 = port 0.4 - 0.5 ** motordribble = port 0.6 - 0.7
35 sbit    MOTOR_1_1 = P1^0;
36 sbit    MOTOR_1_2 = P1^1;
37 sbit    MOTOR_2_1 = P1^2;
38 sbit    MOTOR_2_2 = P1^3;
39 sbit    MOTOR_3_1 = P1^4;
40 sbit    MOTOR_3_2 = P1^5;
41 sbit    MOTOR_4_1 = P1^6;
42 sbit    MOTOR_4_2 = P1^7;
43 unsigned int ts_counter=0;          //The counter for time sampling (0.01sec)
44 //********************* initial ****************************************
45 unsigned int gear_m_1 = 0;
46 unsigned int gear_m_2 = 0;
47 unsigned int gear_m_3 = 0;
48 unsigned int gear_m_4 = 0;       // default speed motor dribble
49
50 unsigned char di_m_1 = CLOCKWISE;
51 unsigned char di_m_2 = CLOCKWISE;
52 unsigned char di_m_3 = CLOCKWISE;
53 unsigned char di_m_4 = STOP;     // motor dribble
54 // ***************************************************************
55 unsigned char BufferData[4];  // buffer for data from serial port
56 //unsigned char n_Buffer = 0;     // number buffer
57
58 void open_intr_service(void)
59  {
60     IE=0x93;     //open Series,Timer0 and INT0 interrupt
61     IP=0x02;     //define high priority level is Timer0 interrupt
62  }
63 void open_timer_service(void)
64  {
65     TMOD=0x22;  //Timer0 mode2 and Timer1 mode2
66     TCON=0x01;  //Active INT0 interrutp at down time
67     //TH0=0xa1; //47 microsec of one sampling and 42 pulse sampling in pwm(500Hz)
68     //TH0 = 0x9c; // 50 microsec of one sampling and 40 pulse sampling in pwn(500Hz)
69     //TH0 = 0xce;    // 25 microsec of one sampling and 80 pulse sampling in pwn(500Hz)
70     TH0 = 0x79;  // 67 microsec of one sampling and 30 pulse sampling ing pwn(500Hz)
71  }
72 void open_series_service(void)
73  {
74     SCON=0x50;  //Mode1 8 bit
75     PCON=0x00;
76     TH1=0xf3;   //4800 bps 24 MHz
```

```
77      TR1=1;      //Strat timer1
78   }
79   void timer0_int() interrupt 1
80   {
81      if(ts_counter <= ALL_PULSE)
82      {
83          ts_counter++;       //counter for time sampling
84      }
85      else
86      {
87          ts_counter = 1;
88      }
89   }
90
91   void controlmotor(unsigned int n_motor, unsigned int gear, unsigned char direction)
92   {
93      switch(n_motor)
94      {
95          case 1:
96          {
97              if(direction == COUNTER_CLOCKWISE)
98              {
99                  if(ts_counter <= TIME_INVERT_M_1*gear)
100                 {
101                     MOTOR_1_1 = 1;
102                     MOTOR_1_2 = 0;
103                 }
104                 else
105                 {
106                     MOTOR_1_1 = 0;
107                     MOTOR_1_2 = 0;
108                 }
109             }
110             else
111             {
112                 if(ts_counter <= TIME_INVERT_M_1*gear)
113                 {
114                     MOTOR_1_1 = 0;
115                     MOTOR_1_2 = 1;
116                 }
117                 else
118                 {
119                     MOTOR_1_1 = 0;
120                     MOTOR_1_2 = 0;
121                 }
122             }
123             break;
124         }
125
126         case 2:
127         {
128             if(direction == COUNTER_CLOCKWISE)
129             {
130                 if(ts_counter <= TIME_INVERT_M_2*gear)
131                 {
132                     MOTOR_2_1 = 0;
133                     MOTOR_2_2 = 1;
134                 }
135                 else
136                 {
137                     MOTOR_2_1 = 0;
138                     MOTOR_2_2 = 0;
139                 }
140             }
141             else
142             {
143                 if(ts_counter <= TIME_INVERT_M_2*gear)
144                 {
145                     MOTOR_2_1 = 1;
146                     MOTOR_2_2 = 0;
147                 }
148                 else
149                 {
150                     MOTOR_2_1 = 0;
151                     MOTOR_2_2 = 0;
152                 }
```

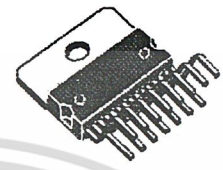
```
153             }
154             break;
155         }
156 .
157         case 3:
158         {
159             if(direction == COUNTER_CLOCKWISE)
160             {
161                 if(ts_counter <= TIME_INVERT_M_3*gear)
162                 {
163                     MOTOR_3_1 = 1;
164                     MOTOR_3_2 = 0;
165                 }
166                 else
167                 {
168                     MOTOR_3_1 = 0;
169                     MOTOR_3_2 = 0;
170                 }
171             }
172             else
173             {
174                 if(ts_counter <= TIME_INVERT_M_3*gear)
175                 {
176                     MOTOR_3_1 = 0;
177                     MOTOR_3_2 = 1;
178                 }
179                 else
180                 {
181                     MOTOR_3_1 = 0;
182                     MOTOR_3_2 = 0;
183                 }
184             }
185             break;
186         }
187
188         case 4:   // out no pulse
189         {
190             switch (direction)
191             {
192                 case 0:
193                 {
194                     MOTOR_4_1 = 0;
195                     MOTOR_4_2 = 0;
196                     break;
197                 }
198                 case 1:
199                 {
200                     MOTOR_4_1 = 1;
201                     MOTOR_4_2 = 0;
202                     break;
203                 }
204                 case 2:
205                 {
206                     MOTOR_4_1 = 0;
207                     MOTOR_4_2 = 1;
209                     break;
209                 }
210                 break;
211             }
212             break;
213         }
214     }
215 }
216 void OperateData()
217 {
218     // *************** id ***************************
219     unsigned char Id_e = BufferData[2] & 0x07;
220     unsigned char Id_s = (BufferData[0] & 0x70) >> 4;
221     // *************** dribble **********************
222     unsigned char Dribble = BufferData[0] & 0x01;
223     // *************** shoot ************************
224     unsigned char Shoot = (BufferData[2] & 0x08) >> 3;
225     if(Id_s == Id_e)
226     {
227         // ****** direction all motor *********
228         di_m_1 = (BufferData[0] & 0x08) >> 3;
```

```
229         di_m_2 = (BufferData[0] & 0x04) >> 2;
230         di_m_3 = (BufferData[0] & 0x02) >> 1;
231
232         if((Dribble == 0) && (Shoot == 0))
233             di_m_4 = STOP;
234         else
235         {
236             if((Shoot == 1) && (Dribble ==0))
237                 di_m_4 = SHOOT;
238             else
239                 di_m_4 = DRIBBLE;
240         }
241         //di_m_4 = BufferData[0] & 0x01; // dribble = '1' , if not = '0'
242
243         // ****** velocity all motor **********
244         gear_m_1 = (BufferData[1] & 0xF0) >> 4;
245         gear_m_2 = BufferData[1] & 0x0F;
246         gear_m_3 = (BufferData[2] & 0xF0) >> 4;
247         //gear_m_4 = 14;
248
249         // ******** clear buffer *************
250         BufferData[0] = 0;
251         BufferData[1] = 0;
252         BufferData[2] = 0;
253         BufferData[3] = 0;
254     }
255 }
256 void VarifyData(unsigned char InData)
257 {
258     unsigned char id = (BufferData[0] & 0x70) >> 4;
259     if(InData == 255)
260     {
261         if(BufferData[0] != 0)
262         {
263             if (id == ID_ROBOT)
264             {
265                 OperateData();
266             }
267         }
268     }
269 }
270 void series_int() interrupt 4
271 {
272     BufferData[0] = BufferData[1];
273     BufferData[1] = BufferData[2];
274     BufferData[2] = BufferData[3];
275     BufferData[3] = SBUF;
276     VarifyData(BufferData[3]);
277     RI=0;
278 }
279 void main(void)
280 {   open_intr_service();
281     open_timer_service();
282     open_series_service();
283     TR0 = 1;
284     // clear buffer
285     BufferData[0] = 0;
286     BufferData[1] = 0;
287     BufferData[2] = 0;
288     BufferData[3] = 0;
289     while(1)
290     {
291         controlmotor(1, gear_m_1, di_m_1);
292         controlmotor(2, gear_m_2, di_m_2);
293         controlmotor(3, gear_m_3, di_m_3);
294         controlmotor(4, gear_m_4, di_m_4); // if direction clockwise = dribble ** if counter
295     }
296 }
297
```

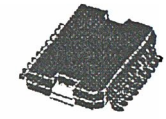
L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)



Multiwatt15



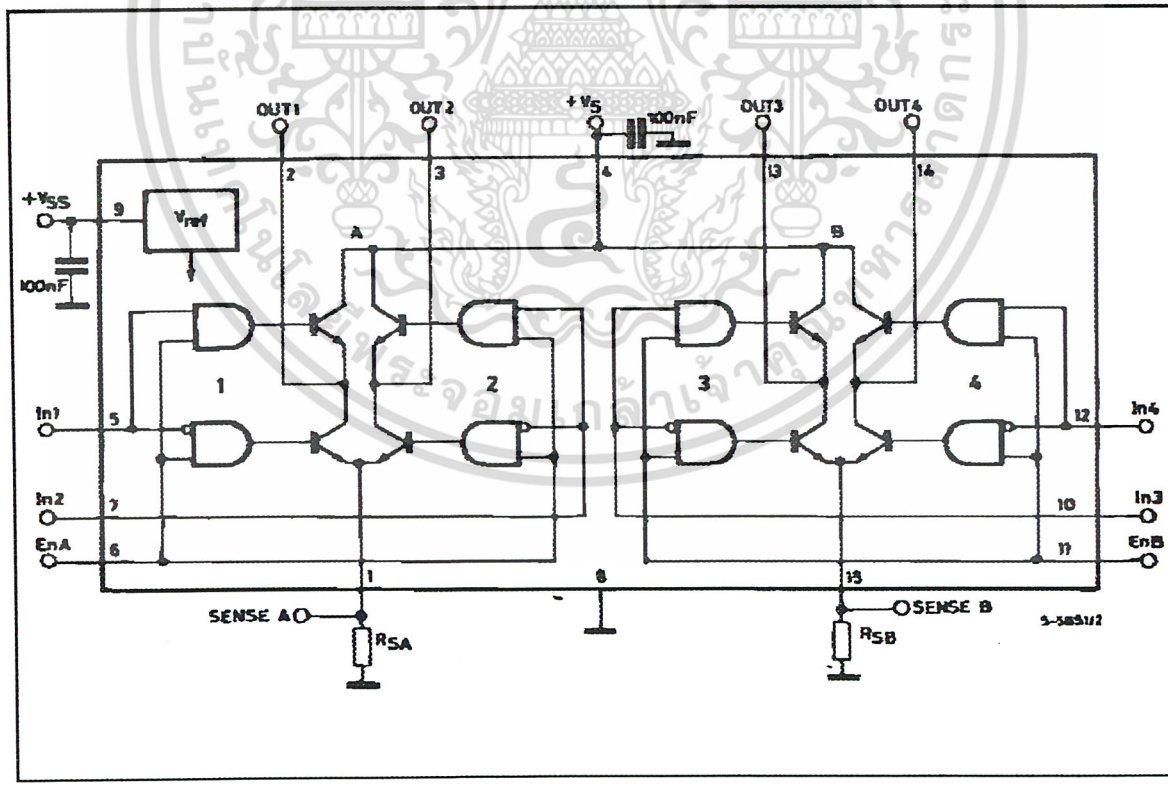
PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

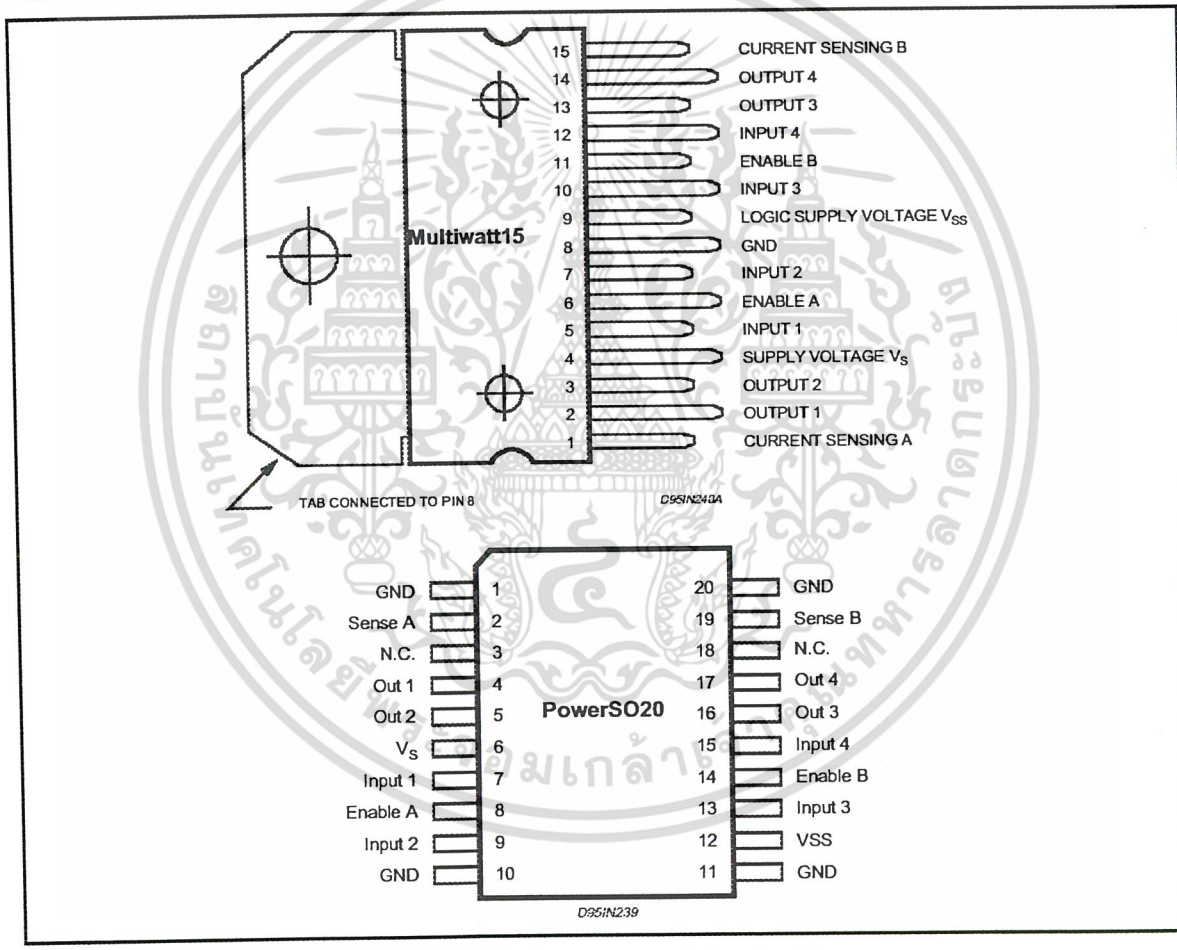
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I, V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

PIN CONNECTIONS (top view)



THERMAL DATA

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

PIN FUNCTIONS (refer to the block diagram)

MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V_S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	VSS	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

ELECTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V, T_j = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 4)	Operative Condition	V_{IH} +2.5		46	V
V_{SS}	Logic Supply Voltage (pin 9)		4.5	5	7	V
I_S	Quiescent Supply Current (pin 4)	V_{en} = H; I_L = 0 V_i = L		13	22	mA
		V_i = H		50	70	mA
		V_{en} = L V_i = X			4	mA
I_{SS}	Quiescent Current from V_{SS} (pin 9)	V_{en} = H; I_L = 0 V_i = L		24	36	mA
		V_i = H		7	12	mA
		V_{en} = L V_i = X			6	mA
V_{iL}	Input Low Voltage (pins 5, 7, 10, 12)		–0.3		1.5	V
V_{iH}	Input High Voltage (pins 5, 7, 10, 12)		2.3		VSS	V
I_{iL}	Low Voltage Input Current (pins 5, 7, 10, 12)	V_i = L			–10	µA
I_{iH}	High Voltage Input Current (pins 5, 7, 10, 12)	V_i = H ≤ V_{SS} –0.6V		30	100	µA
V_{en} = L	Enable Low Voltage (pins 6, 11)		–0.3		1.5	V
V_{en} = H	Enable High Voltage (pins 6, 11)		2.3		V_{SS}	V
I_{en} = L	Low Voltage Enable Current (pins 6, 11)	V_{en} = L			–10	µA
I_{en} = H	High Voltage Enable Current (pins 6, 11)	V_{en} = H ≤ V_{SS} –0.6V		30	100	µA
$V_{CEsat(H)}$	Source Saturation Voltage	I_L = 1A	0.95	1.35	1.7	V
		I_L = 2A		2	2.7	V
$V_{CEsat(L)}$	Sink Saturation Voltage	I_L = 1A (5)	0.85	1.2	1.6	V
		I_L = 2A (5)		1.7	2.3	V
V_{CEsat}	Total Drop	I_L = 1A (5)	1.80		3.2	V
		I_L = 2A (5)			4.9	V
V_{sens}	Sensing Voltage (pins 1, 15)		–1 (1)		2	V

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T_1 (V_i)	Source Current Turn-off Delay	0.5 V_i to 0.9 I_L (2); (4)		1.5		µs
T_2 (V_i)	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		0.2		µs
T_3 (V_i)	Source Current Turn-on Delay	0.5 V_i to 0.1 I_L (2); (4)		2		µs
T_4 (V_i)	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.7		µs
T_5 (V_i)	Sink Current Turn-off Delay	0.5 V_i to 0.9 I_L (3); (4)		0.7		µs
T_6 (V_i)	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.25		µs
T_7 (V_i)	Sink Current Turn-on Delay	0.5 V_i to 0.9 I_L (3); (4)		1.6		µs
T_8 (V_i)	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.2		µs
fc (V_i)	Commutation Frequency	I_L = 2A		25	40	KHz
T_1 (V_{en})	Source Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (2); (4)		3		µs
T_2 (V_{en})	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		1		µs
T_3 (V_{en})	Source Current Turn-on Delay	0.5 V_{en} to 0.1 I_L (2); (4)		0.3		µs
T_4 (V_{en})	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.4		µs
T_5 (V_{en})	Sink Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (3); (4)		2.2		µs
T_6 (V_{en})	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.35		µs
T_7 (V_{en})	Sink Current Turn-on Delay	0.5 V_{en} to 0.9 I_L (3); (4)		0.25		µs
T_8 (V_{en})	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.1		µs

1) 1)Sensing voltage can be –1 V for t ≤ 50 µsec; in steady state V_{sens} min ≥ –0.5 V.
2) See fig. 2.
3) See fig. 4.
4) The load must be a pure resistor.

Figure 1 : Typical Saturation Voltage vs. Output Current.

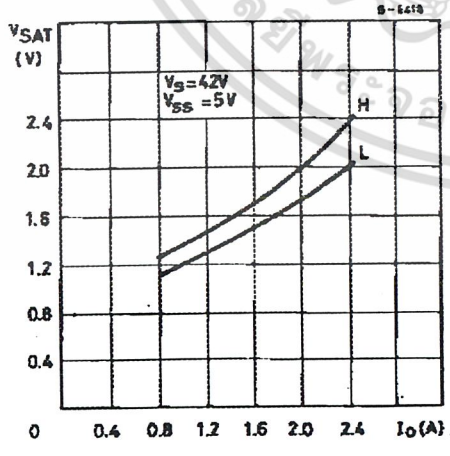
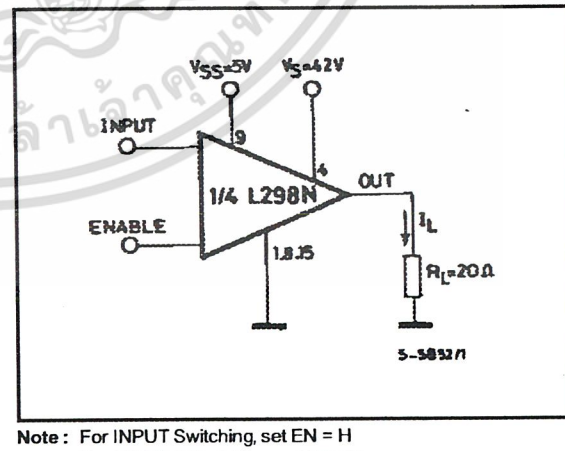


Figure 2 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = H

Figure 3 : Source Current Delay Times vs. Input or Enable Switching.

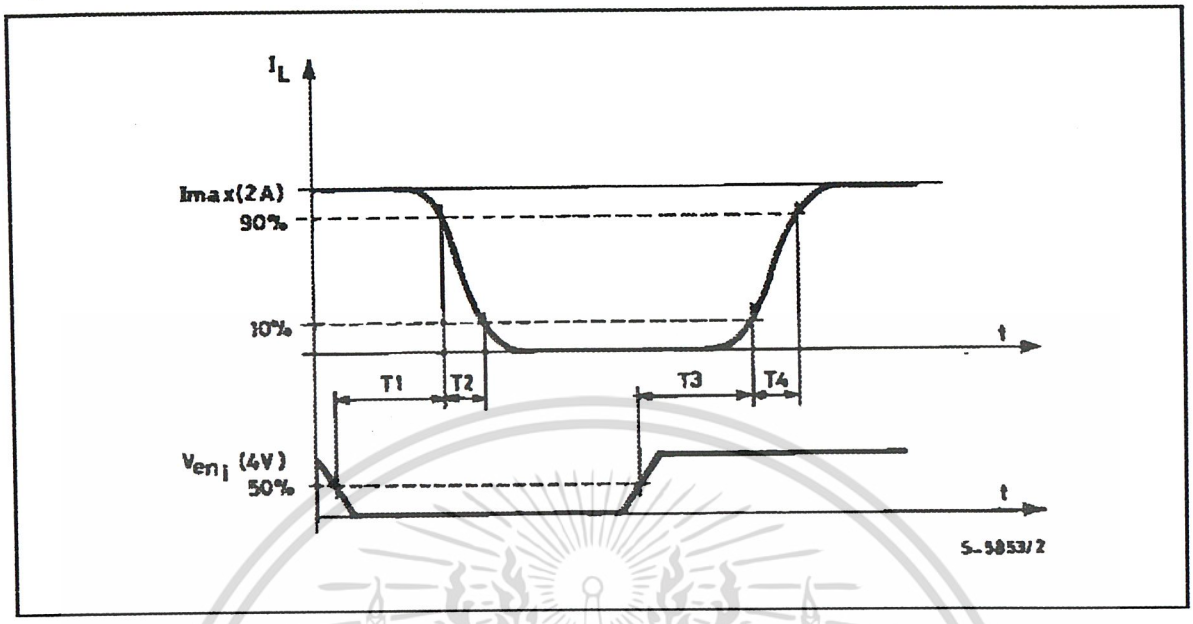
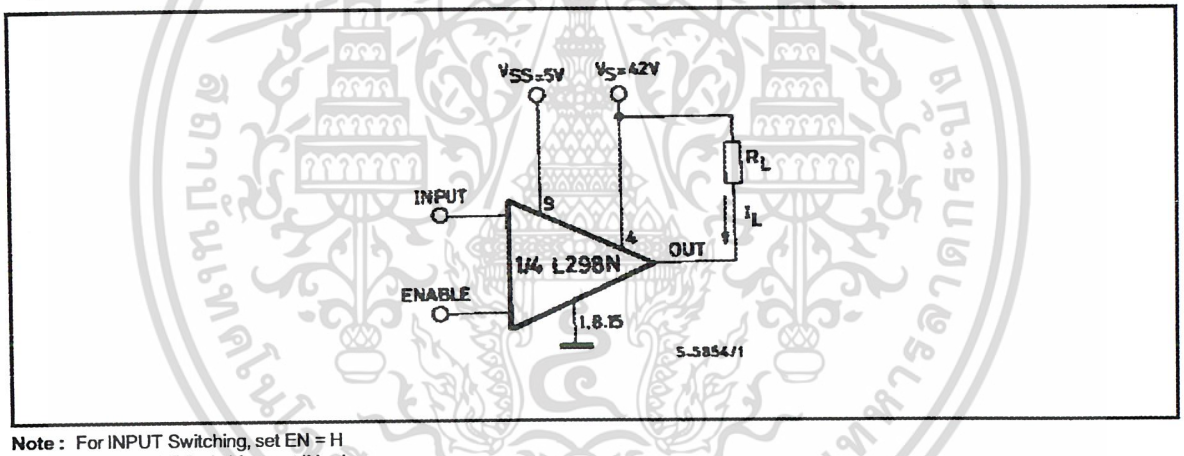


Figure 4 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = L

Figure 5 : Sink Current Delay Times vs. Input 0 V Enable Switching.

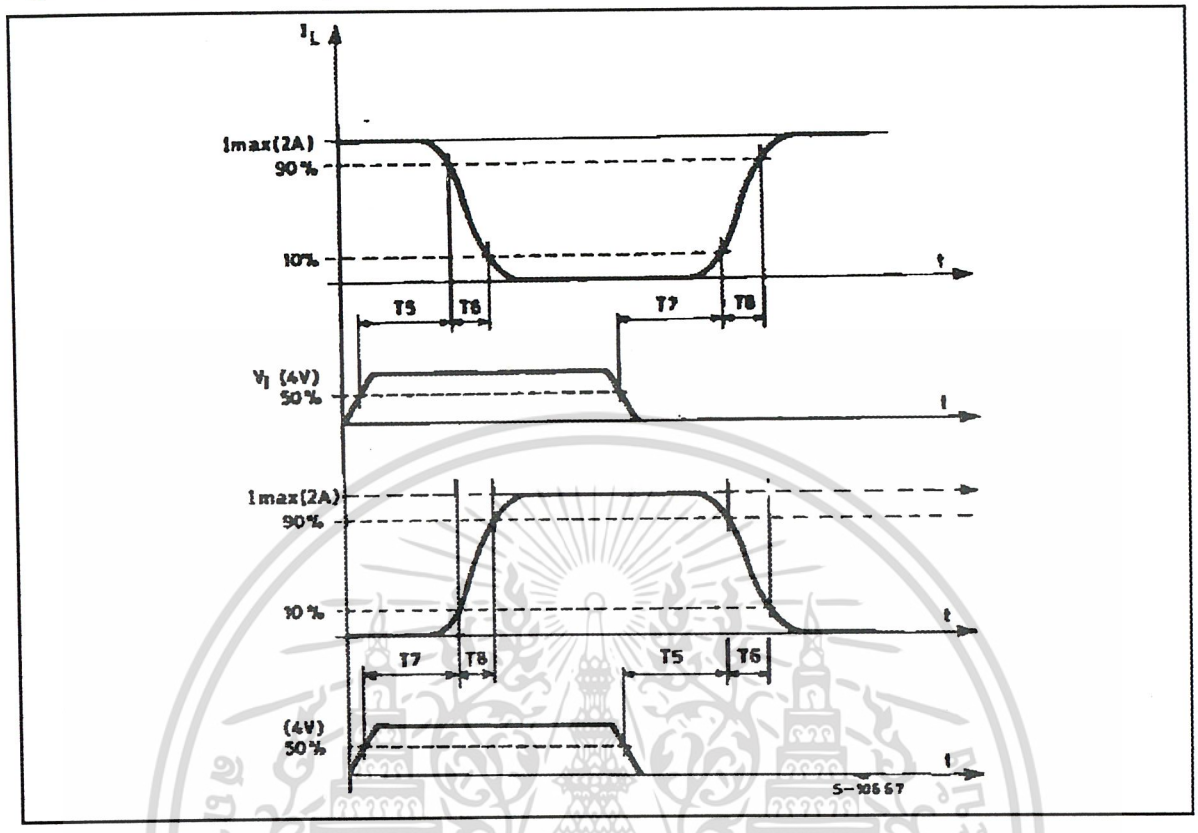
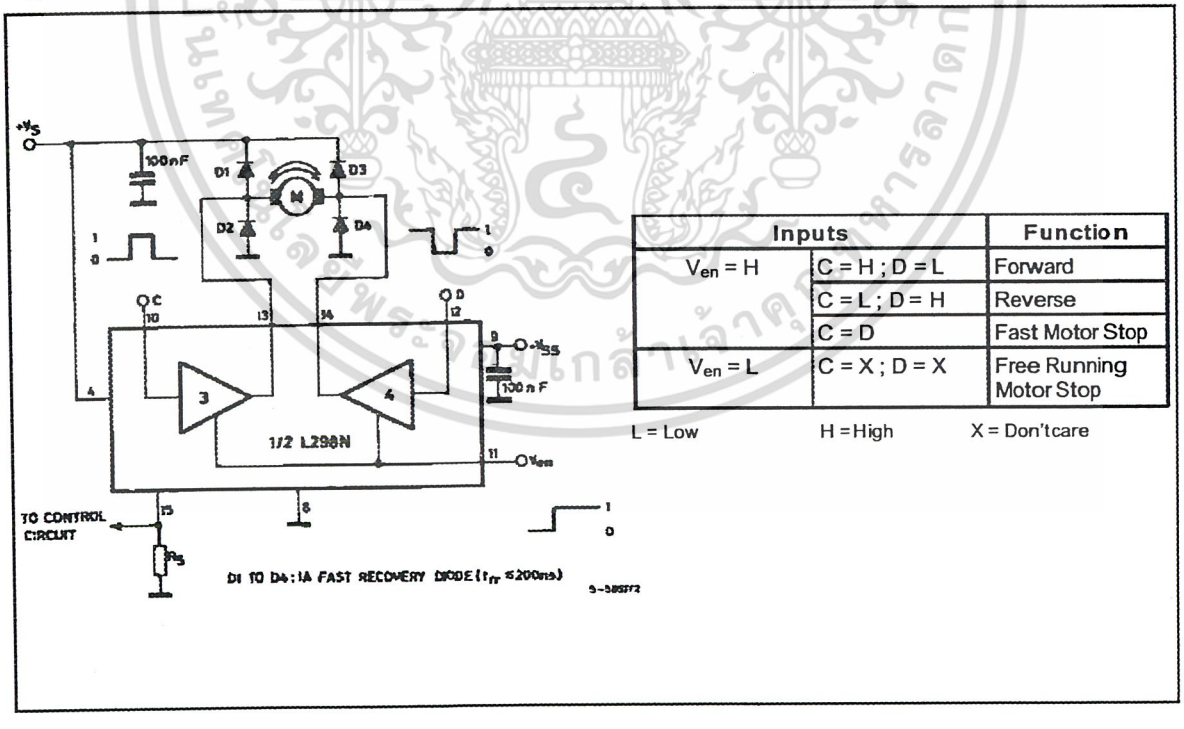


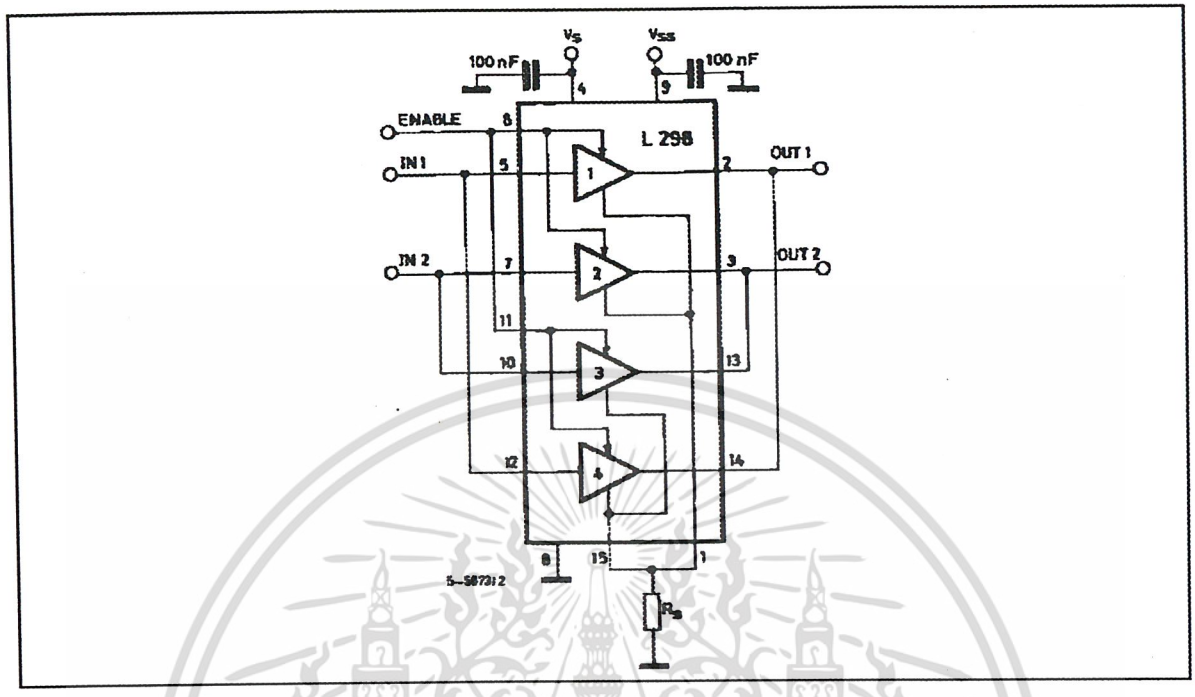
Figure 6 : Bidirectional DC Motor Control.



Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

Figure 7 : For higher currents, outputs can be paralleled. Take care to parallel channel 1 with channel 4 and channel 2 with channel 3.



APPLICATION INFORMATION (Refer to the block diagram)

1.1. POWER OUTPUT STAGE

The L298 integrates two power output stages (A ; B). The power output stage is a bridge configuration and its outputs can drive an inductive load in common or differenzial mode, depending on the state of the inputs. The current that flows through the load comes out from the bridge at the sense output : an external resistor (R_{SA} ; R_{SB}.) allows to detect the intensity of this current.

1.2. INPUT STAGE

Each bridge is driven by means of four gates the input of which are In1 ; In2 ; EnA and In3 ; In4 ; EnB. The In inputs set the bridge state when The En input is high ; a low state of the En input inhibits the bridge. All the inputs are TTL compatible.

2. SUGGESTIONS

A non inductive capacitor, usually of 100 nF, must be foreseen between both Vs and Vss, to ground, as near as possible to GND pin. When the large capacitor of the power supply is too far from the IC, a second smaller one must be foreseen near the L298.

The sense resistor, not of a wire wound type, must be grounded near the negative pole of Vs that must be near the GND pin of the I.C.

Each input must be connected to the source of the driving signals by means of a very short path.

Turn-On and Turn-Off : Before to Turn-ON the Supply Voltage and before to Turn it OFF, the Enable input must be driven to the Low state.

3. APPLICATIONS

Fig 6 shows a bidirectional DC motor control Schematic Diagram for which only one bridge is needed. The external bridge of diodes D1 to D4 is made by four fast recovery elements ($t_{rr} \leq 200$ nsec) that must be chosen of a VF as low as possible at the worst case of the load current.

The sense output voltage can be used to control the current amplitude by chopping the inputs, or to provide overcurrent protection by switching low the enable input.

The brake function (Fast motor stop) requires that the Absolute Maximum Rating of 2 Amps must never be overcome.

When the repetitive peak current needed from the load is higher than 2 Amps, a paralleled configuration can be chosen (See Fig.7).

An external bridge of diodes are required when inductive loads are driven and when the inputs of the IC are chopped ; Shottky diodes would be preferred.

This solution can drive until 3 Amps In DC operation and until 3.5 Amps of a repetitive peak current.

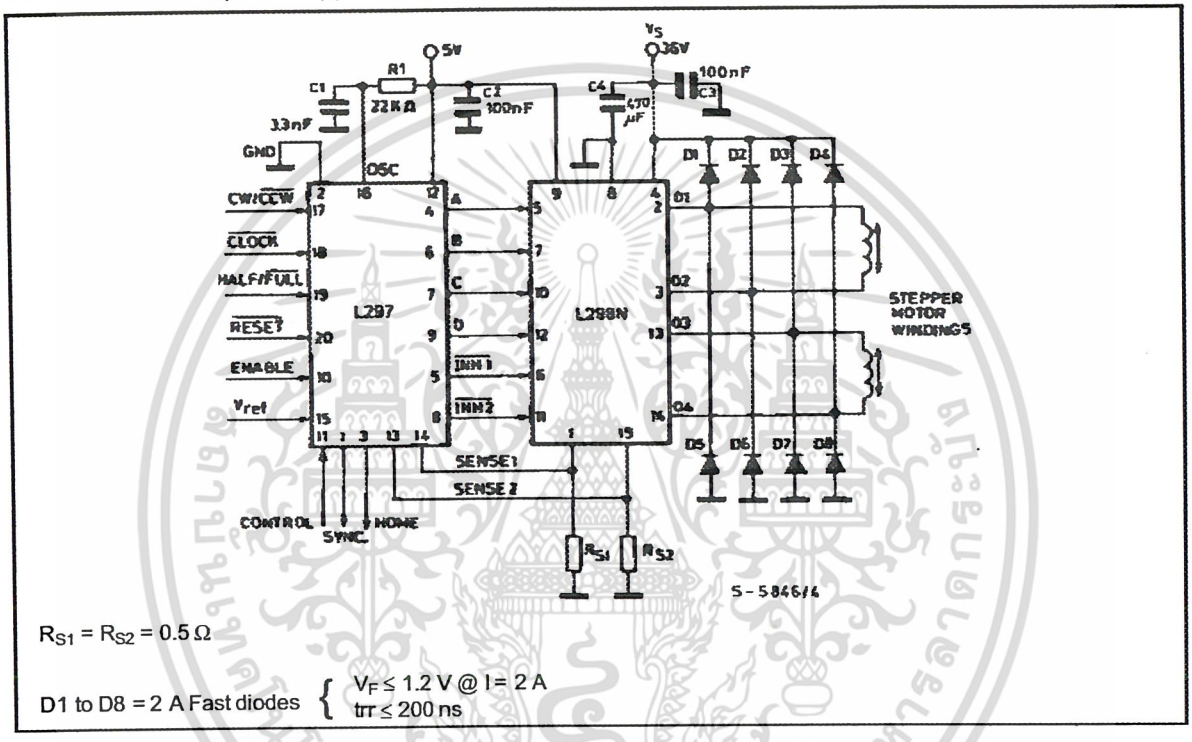
On Fig 8 it is shown the driving of a two phase bipolar stepper motor ; the needed signals to drive the inputs of the L298 are generated, in this example, from the IC L297.

Fig 9 shows an example of P.C.B. designed for the application of Fig 8.

Fig 10 shows a second two phase bipolar stepper motor control circuit where the current is controlled by the I.C. L6506.

Figure 8 : Two Phase Bipolar Stepper Motor Circuit.

This circuit drives bipolar stepper motors with winding currents up to 2 A. The diodes are fast 2 A types.



$R_{S1} = R_{S2} = 0.5\,\Omega$

D1 to D8 = 2 A Fast diodes { $V_F \leq 1.2$ V @ I = 2 A ; $t_{rr} \leq 200$ ns

Figure 9 : Suggested Printed Circuit Board Layout for the Circuit of fig. 8 (1:1 scale).

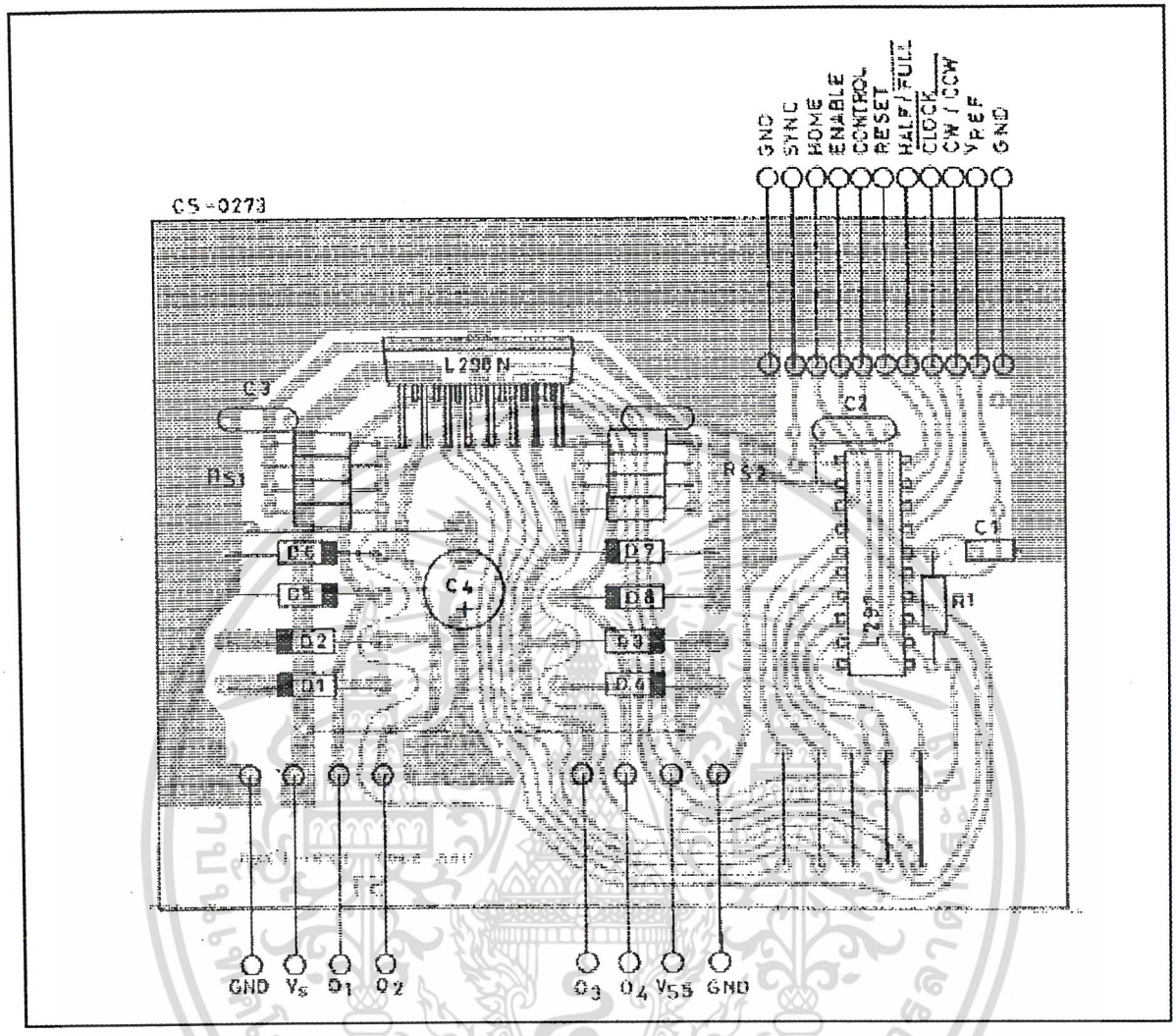
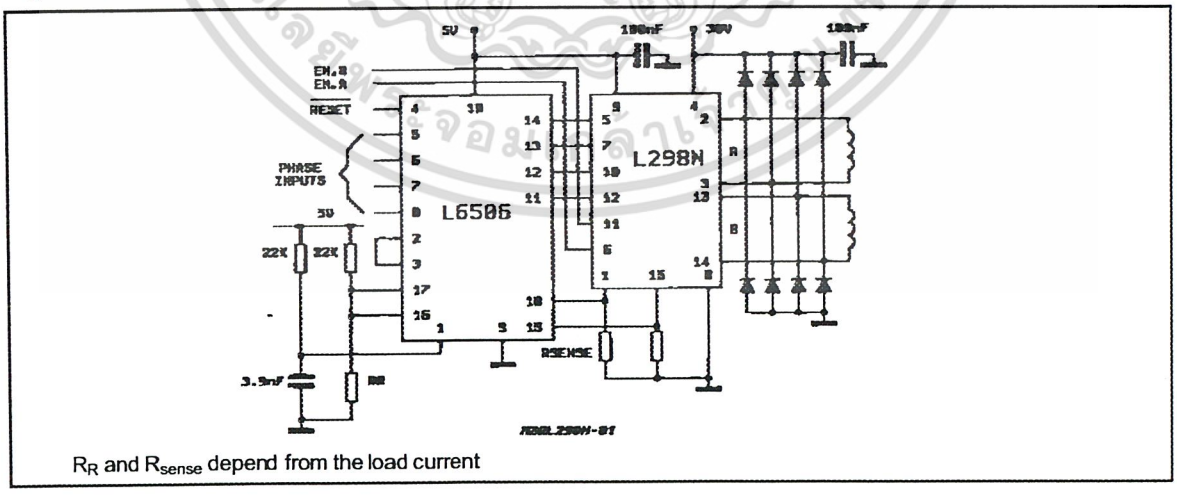


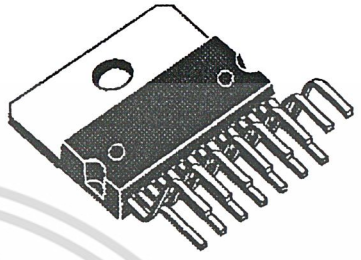
Figure 10 : Two Phase Bipolar Stepper Motor Control Circuit by Using the Current Controller L6506.



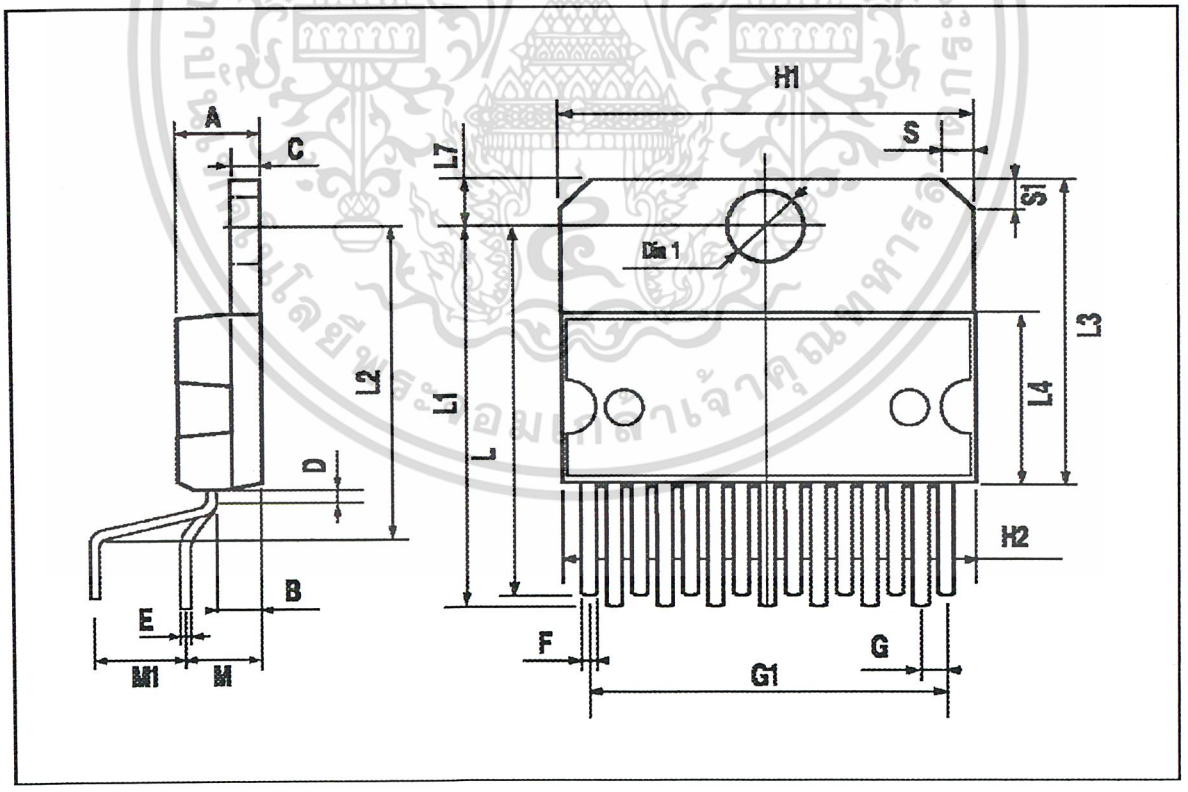
R_R and R_{sense} depend from the load current

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
D		1			0.039	
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.02	1.27	1.52	0.040	0.050	0.060
G1	17.53	17.78	18.03	0.690	0.700	0.710
H1	19.6			0.772		
H2			20.2			0.795
L	21.9	22.2	22.5	0.862	0.874	0.886
L1	21.7	22.1	22.5	0.854	0.870	0.886
L2	17.65		18.1	0.695		0.713
L3	17.25	17.5	17.75	0.679	0.689	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L7	2.65		2.9	0.104		0.114
M	4.25	4.55	4.85	0.167	0.179	0.191
M1	4.63	5.08	5.53	0.182	0.200	0.218
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
Dia1	3.65		3.85	0.144		0.152

OUTLINE AND MECHANICAL DATA

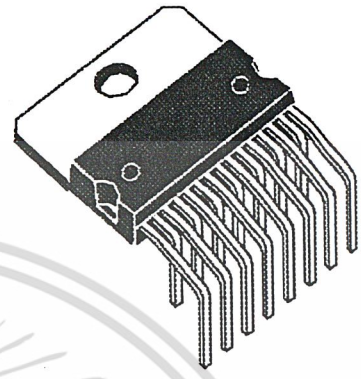


Multiwatt15 V

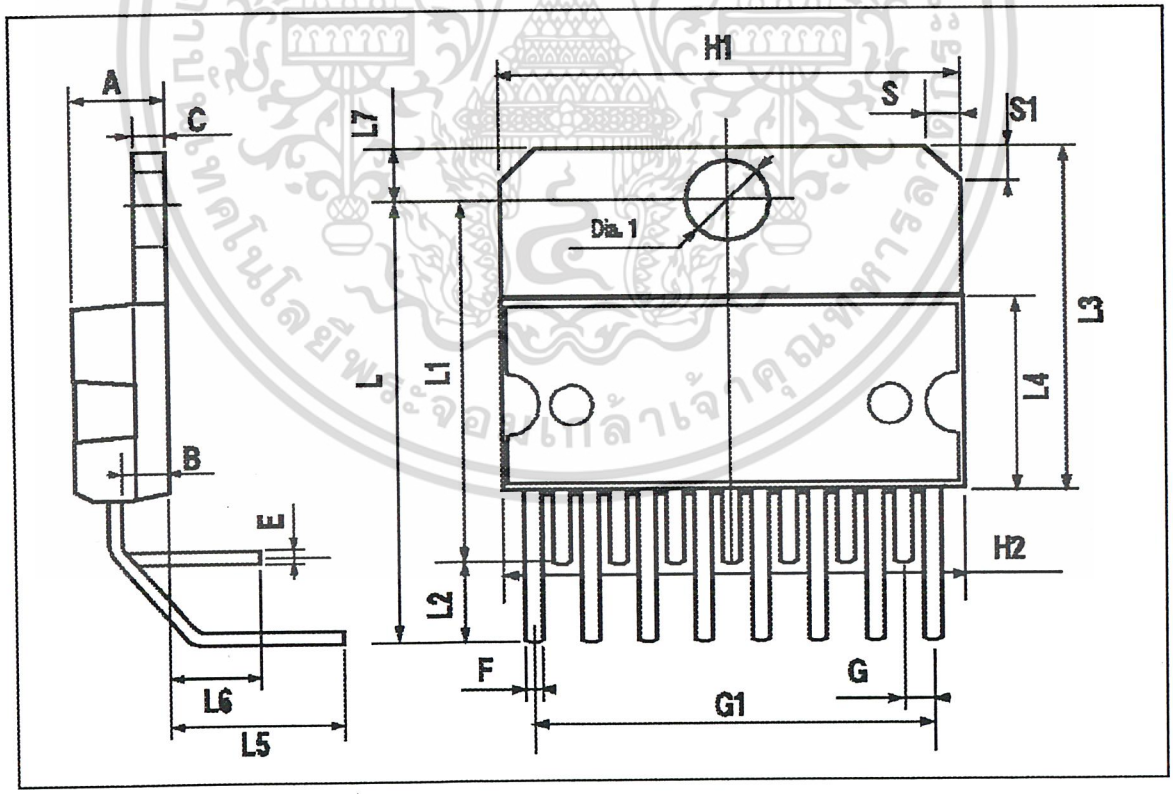


DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.14	1.27	1.4	0.045	0.050	0.055
G1	17.57	17.78	17.91	0.692	0.700	0.705
H1	19.6			0.772		
H2			20.2			0.795
L		20.57			0.810	
L1		18.03			0.710	
L2		2.54			0.100	
L3	17.25	17.5	17.75	0.679	0.689	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L5		5.28			0.208	
L6		2.38			0.094	
L7	2.65		2.9	0.104		0.114
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
Dia1	3.65		3.85	0.144		0.152

OUTLINE AND MECHANICAL DATA



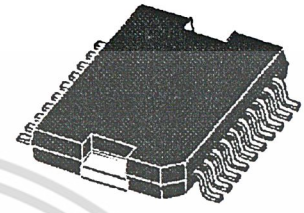
Multiwatt15 H



DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			3.6			0.142
a1	0.1		0.3	0.004		0.012
a2			3.3			0.130
a3	0		0.1	0.000		0.004
b	0.4		0.53	0.016		0.021
c	0.23		0.32	0.009		0.013
D (1)	15.8		16	0.622		0.630
D1	9.4		9.8	0.370		0.386
E	13.9		14.5	0.547		0.570
e		1.27			0.050	
e3		11.43			0.450	
E1 (1)	10.9		11.1	0.429		0.437
E2			2.9			0.114
E3	5.8		6.2	0.228		0.244
G	0		0.1	0.000		0.004
H	15.5		15.9	0.610		0.626
h			1.1			0.043
L	0.8		1.1	0.031		0.043
N	10° (max.)					
S	8° (max.)					
T		10			0.394	

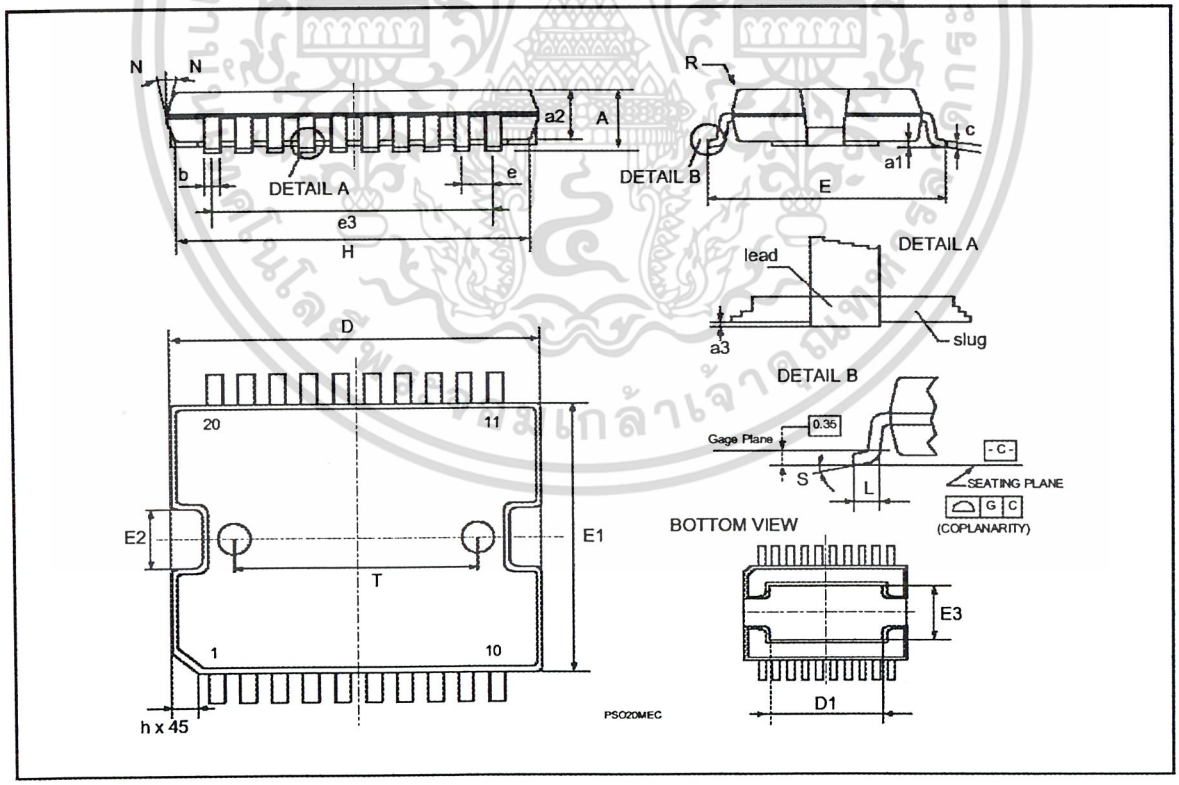
(1) "D and F" do not include mold flash or protrusions.
- Mold flash or protrusions shall not exceed 0.15 mm (0.006").
- Critical dimensions: "E", "G" and "a3"

OUTLINE AND MECHANICAL DATA

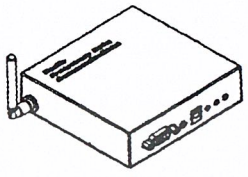


JEDEC MO-166

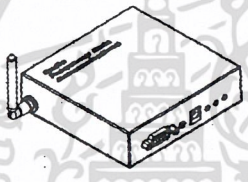
PowerSO20



Information

Machine type	Function
 RDL401A	AUTO SWITCH TX RX DISPLAY RS232 CHANNEL SELECT

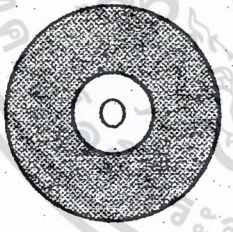
Standard Package

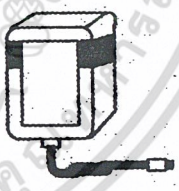

RDL401A


WAS-1297


CN5-OTT0006

Optional


DISK5146

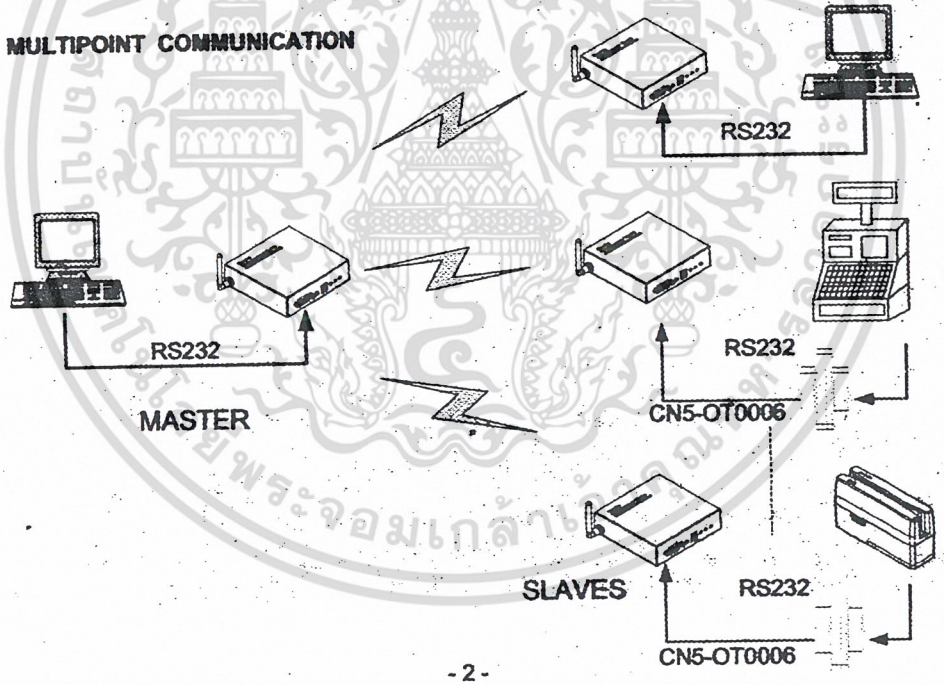

DC9V/120VAC
DC9V/230VAC

Application

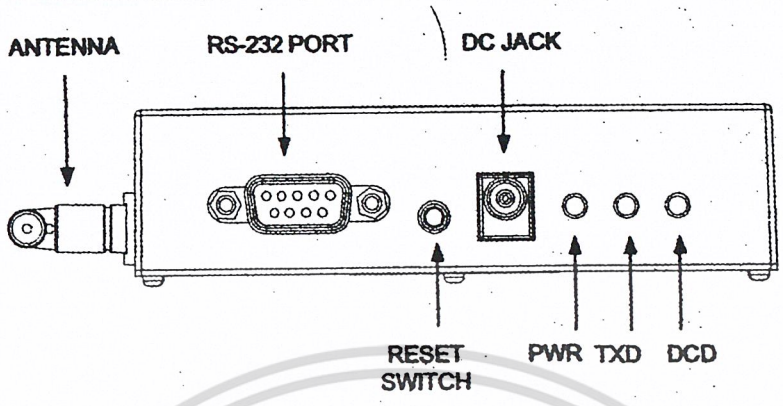
- To replace the RS-232 cable between Computer and Computer (Terminal) .
- Remote control
- Remote monitoring
- Data acquisition
- Computer aided integrated manufacturing
- Point-of-sale system
- Data transmission of movable station

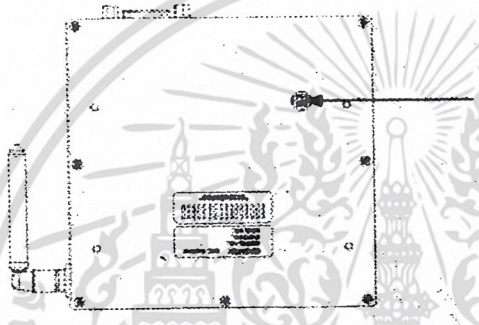
POINT TO POINT COMMUNICATION

MULTIPOINT COMMUNICATION



Configuration





CHANNEL Select Switch :

Switch 1 : ON : 433.92MHz
OFF : 434.33MHz
Switch 2 : ON : 4800 BPS Disable
OFF : 4800 BPS Enable
Switch 3 : ON : 9600 BPS Enable
OFF : 19200 BPS Enable
Switch 4 : Reserved

RS-232 Port -- To be connected with the Computer (Terminal) .

REST Switch -- Press this REST switch to reset radio Channel to the new setting . Before pressing this switch , the radio Channel will not change although the CHANNEL Select Switch has been changed. (The CHANNEL Select Switch is located on the bottom of the RDL401A unit)

Power Jack -- For DC 9V power input .

PWR Power indicator -- Turn on when the power is applied .

TXD Indicator -- Green LED turns on when the data is sent from the connected Computer (Terminal) to RDL401A unit.

DCD Indicator -- Green LED turns on when it detects a radio transmission from the air (Data Carrier Detect).

CHANNEL Select Switch -- DIP switch located on the bottom of the RDL401A unit is for changing radio channel 433.92MHz or 434.33MHz. After you make Channel change , you must press RESET switch or power off and then reboot power . Then the RDL401A will be reset to the new radio Channel .

Installation

1. Install the RS-232 cable between the RDL401A and Computer or Terminal .
   (The pin connection is as PIN ASSIGNMENT)

2. Set up one pair of RDL401 , one for Master Computer (terminal) and the other for Remote Computer (terminal) . Set both RDL401A units at the same channel .

3. Connect the DC power supply to the DC Power Jack on RDL401A unit , the PWR LED will turn on.

4. The DCD LED must light green constantly for the receiving unit when detecting a radio transmission from the other unit . If not ,reinstall them to have a constant green DCD light for the receiving unit .
   Please note that you must not have both units become transmitting at the same time because RDL401A is for half-duplex communication only . At one time you must have one unit become transmitting and the other unit become receiving .

Note :

To get connection with RDL401A , you must set Computer (terminal) as the following parameters under the operation software you are using :

Standard :	Standard RS-232
Baud Rate :	4800/9600/19200 bps
Parity :	None
Data Bits :	8
Stop BIT :	1
Flow control :	None

PIN ASSIGNMENT

9 Pin D Female Connect

Pin Number	Signal
1	X
2	TXD (Out)
3	RXD (In)
4	X
5	Ground
6	X
7	X
8	RTS (Out)
9	X

CN5-OTT0006 Connect

Male A Pin Number	Male B Pin Number
1	1
2	3
3	2
4	6
5	5
6	4
7	8
8	7
9	9

Specification

Frequency :	433.92/434.33MHz
Antenna Impedance :	50 Ohms , unbalanced
Number of Channels :	2 (Frequency Synthesized]
Frequency Stability :	+/- 10 ppm (at 0~60 Deg. C)
Data Rate :	4800/9600/19200 Baud
Data format :	None Parity , 8 Data Bits , 1 Stop Bit , Half-duplex over a single channel Asynchronous , Serial

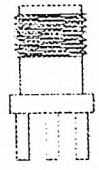
Communication Distance : Approx. 50~150 meters (prospective)

Power Supply :	9 VDC
Environment :	Operating Temperature 0 ~+60 Deg. C Storage Temperature -10 ~+70 Deg. C Humidity : 10 % ~ 95 % , non-condensing
Dimensions :	W 107 * D 102 * H 28 mm
Weight :	APPROX. 0.38 KGs
Transmitter Output Power :	5mW maximum
Modulation Deviation :	35 +/- 5 KHz
PLL Programming Time :	5 m sec , maximum
Current Drain :	60 mA maximun at 9 VDC nominal
Sensitivty :	- 102 dBm
Selectivity :	85 KHz maximum
Receiver Current :	Typical 27 mA

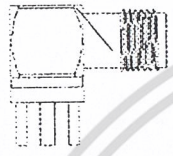
OPTIONAL ANTENNA SOCKET AND ANTENNA

1. Antenna Socket

   P/N: **CN5-OTG0007** (Angle 180)



   P/N: **CN5-OTG0008** (Angle 90)

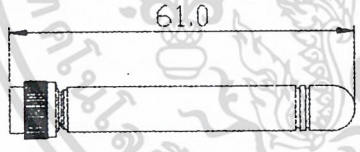


2. Antenna Extension Cable

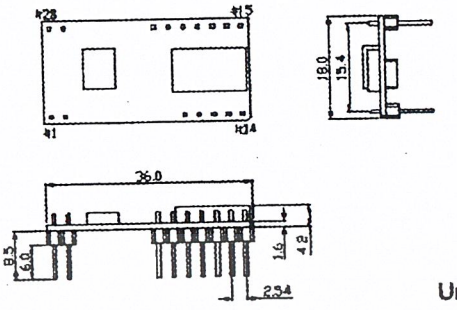
   P/N: **WAS-1529**

3. Antenna

   P/N: **ANT-T001** (50 ohms, 434+-5 MHz)



PACKAGE OUTLINE (DIP28)



Unit: mm

PIN ASSIGNMENT

Pin	Name	I/O	Description
1	ANT	I	*Antenna
2	ANTG	I	*Antenna Signal GND
3,4,5,6,7,8,9	NC		No Connect
10	/RST	I	Active Low Reset
11	RXD	I	UART Data Input
12	TXD	O	UART Data Output
13	GND	I	GND
14	TRD/CMD		Reserved
15	BPS0		4800bps 0:disable 1:enable
16	BPS1		0:9600bps 1:19200bps(*short data pack)
17	CH	I	0:433.92MHz 1:434.33MHz
18	RLED	O	RXD Status LED
19	TLED	O	TXD Status LED
20	GND		GND
21	VCC		Power Input +3.3Vdc~2.7Vdc
22,23,24,25,26	NC		No Connect
27	GND		GND
28	GND		GND

***Antenna Signal GND: Connect to antenna cable GND.**

***Antenna : 50 ohm / 430~435MHz / --25db.**

***19200bps for short data pack (max 70 bytes) only.**

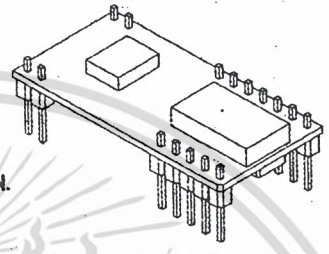
Radio Data Link RF Module with UART Interface

INTRODUCTIONS:

RDL200T (Radio Data Link) RF Module is designed for wireless RS232 serial data communication. It is suitable for most of your wireless data link applications. RDL200T RF Module can be built into your equipment through RS232 interface to replace the cable to become wireless communication. It is a low cost and high quality radio data communication solution.

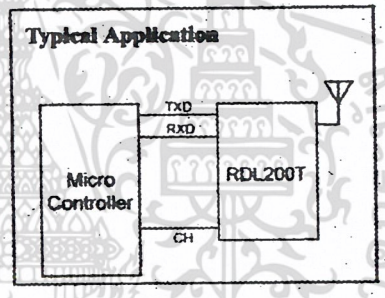
FEATURES:

1. ISM frequency 433.92MHz / 434.33MHz
2. Two channel selectable.
3. Low power consumption with 3.3Vdc
4. RS232 interface (Serial TTL level)
5. Speed up to 9600bps with automatic flow control.
6. Half-duplex with automatic Receiving/Transmitting control.
7. Small package size.
8. Distance: 50~150 meters with appropriate antenna



APPLICATION:

1. Hand-held terminals
2. Computer to computer
3. Computer to terminal
4. Data collection terminal
5. Mobile-around data communications
6. Remote control
7. Remote monitoring
8. POS systems.



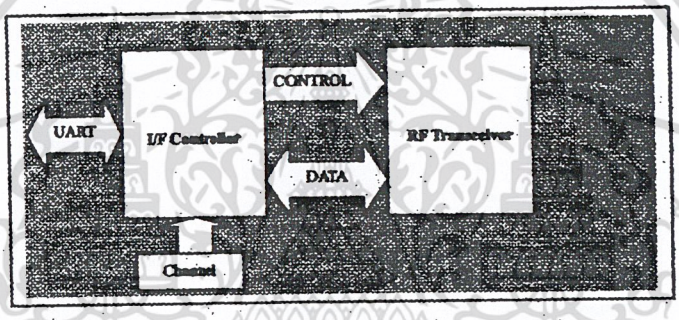
SPECIFICATIONS:

1. Frequency: CH1 433.92MHz / CH2 434.33MHz
2. Modulation: FSK
3. Output power: 7dbm (5mw)
4. Receiving Sensibility: -102dbm
5. Operating voltages: 2.7Vdc~3.3Vdc
6. Power consumption: Tx:60mA / Rx:30mA
7. Interface : RS232 (TTL Level: TXD/RXD)
8. Data Rate : 19200bps/9600bps / 4800bps , Half-duplex
9. Operating temperature: 0°C~55°C (Storage Temperature: -10°C ~60°C)

QUICK REFERENCE DATA

Parameter	Value	Unit
Frequency, Channel#1/Channel#2	433.92/434.33	MHz
Modulation	FSK	
Frequency deviation	+/-30	KHz
Max. RF output power	7	dBm
Sensitivity	-102	dBm
UART Baudrate	19200/9600/4800	bps
Supply voltage	2.7 – 3.3	Vdc
Max. Receive current	30	mA
Max. Transmit current	60	mA

BLOCK DIAGRAM

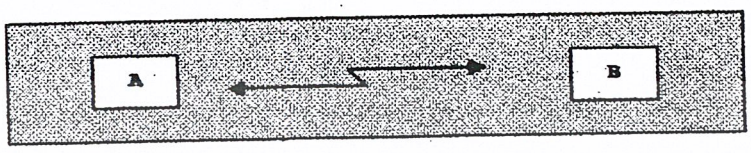


I/F Controller: UART and Radio interface unit
RF Transceiver: A long-range radio transceiver for wireless links operating in the globally available ISM band.

RDL200T APPLICATION

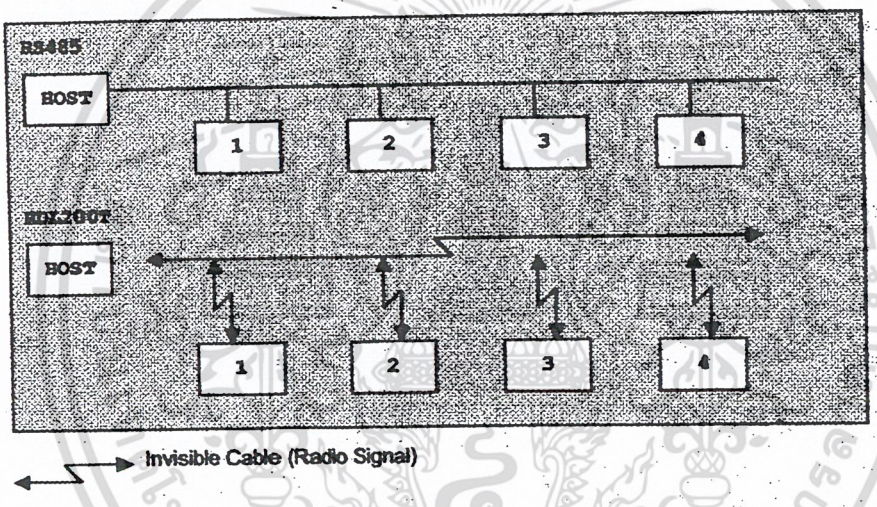
1. Point to Point Radio Data Link

   For point to point radio data link, just change your RS232 transceiver IC to RDL200T in your RS232 product. You don't need to change any firmware. RDL200T serves as a invisible RS232 cable in your product.

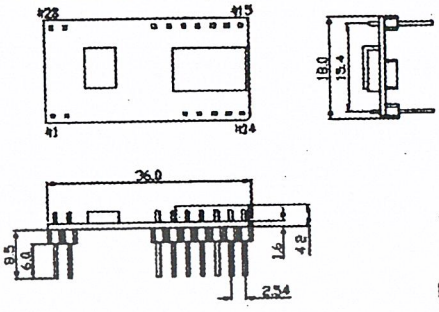


2. Multi-Station Radio Data Link

   For Multi-Station radio data link, you have to build a protocol for Multi-Station that's similar to RS485 to assign machine ID for each station.



PACKAGE OUTLINE (DIP28)



Unit: mm

PIN ASSIGNMENT

PIN	NAME	I/O	DESCRIPTION
1	ANT	I	*Antenna
2	ANTG	I	*Antenna Signal GND
3,4,5,6,7,8,9	NC		No Connect
10	/RST	I	Active Low Reset
11	RXD	I	UART Data Input
12	TXD	O	UART Data Output
13	GND	I	GND
14	TRD/CMD		Reserved
15	BPS0		4800bps 0:disable 1:enable
16	BPS1		0:9600bps 1:19200bps(*short data pack)
17	CH	I	0:433.92MHz 1:434.33MHz
18	RLED	O	RXD Status LED
19	TLED	O	TXD Status LED
20	GND		GND
21	VCC		Power Input +3.3Vdc~2.7Vdc
22,23,24,25,26	NC		No Connect
27	GND		GND
28	GND		GND

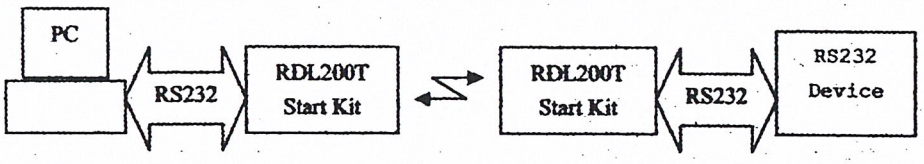
***Antenna Signal GND: Connect to antenna cable GND.**

***Antenna : 50 ohm / 430~435MHz / -25db.**

***19200bps for short data pack (max 70 bytes) only.**

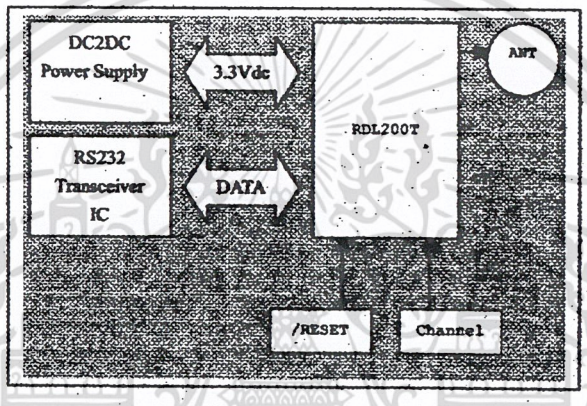
START KIT PACKAGE APPLICATION

Example: Point to Point radio data link

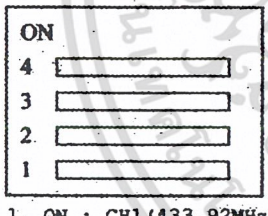


Keep your communication way and easy to upgrade your product to become wireless product with RDL200T.

ASY-RDL200TSK BLOCK DIAGRAM

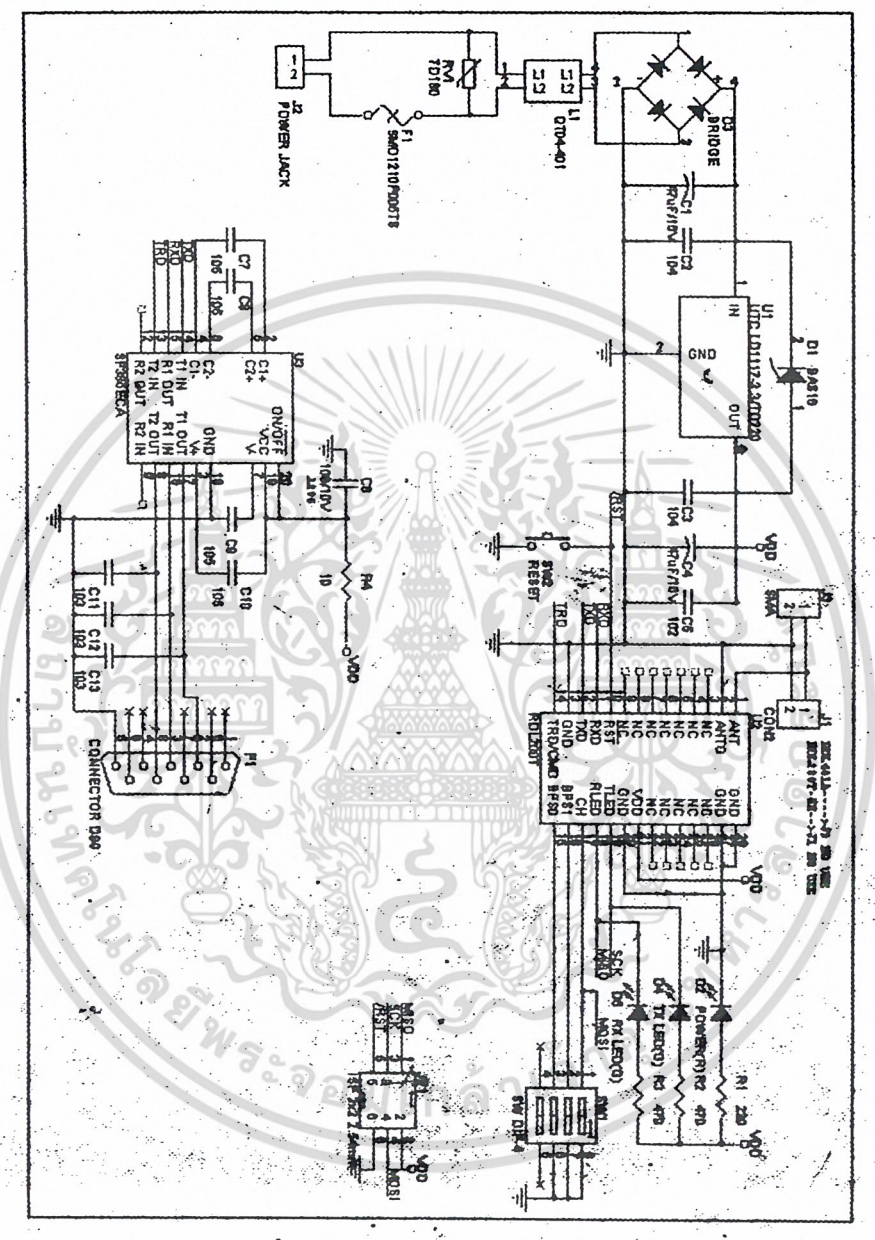


DIP SWITCH



1. ON : CH1(433.92MHz) / OFF:CH2(434.33MHz)
2. ON : 4800 bps DISABLE / OFF : 4800 bps ENABLE
3. ON : 9600 bps ENABLE / OFF : 19200 bps ENABLE
4. Reserved

ASY-RDL200TSK CIRCUIT DIAGRAM



ASY-RDL200TSK BILL OF MATERIALS

UTCLD1117-3.3J	3.3Vdc DC2DC Power supply	U1	1
*RDL200T-00	Radio Data Link Module	U2	1
SP385ACA	RS232 Transceiver IC	U3	1
BAS16	DIODE	D1	1
DBS104G	DIODE BRIDGE SMT DBS104G	D3	
7D180	Variable Resister	RV1	1
CEC45-476016W	47UF/16V	C1,C4	2
CMC5X102K50	1000PF/50V	C5	1
CMC5X103K50	0.01UF/50V	C11,C12,C13	3
C2012X7R1H104K	0.1UF/50V	C2,C3	2
CMC5X105KCH	1UF/25V	C7,C8,C9,C10	4
CMC6Y106Z16	10UF/25V	C6	1
0805-5-470R	470 ohm	R2,R3	2
0805-5-220R	220 ohm	R1	1
0805-5-10R	10 ohm	R4	1
*CN5-OTG007	Antenna Socket (Angle 180)	J3	1
*ANT-T001	Antenna 50 ohms, 434+-5 MHz		1

*PROVIDED BY GIGA-TMS INC.

ORDERING INFORMATION

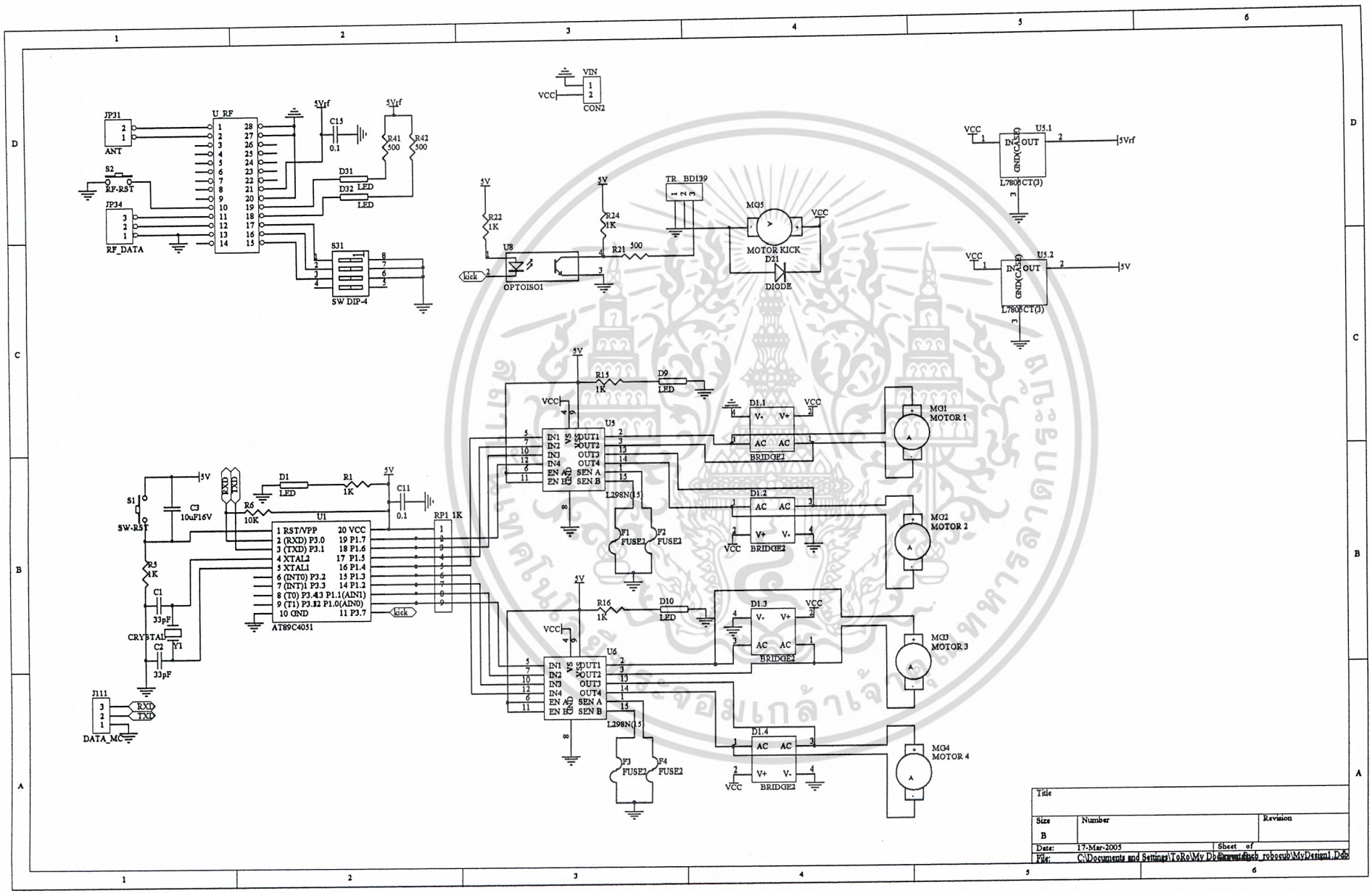
RDL200T-00: (Radio Data Link Module)
P/N: RDL200T-00

RDL200T Start Kit Package: (ASY-RDL200TSK x 2PCS)
P/N: RDL200T-SKE

- E: Include 230VAC/9VDC Adaptor
- U: Include 120VAC/9VDC Adaptor

RDL401A-00 (Stand-Alone Radio Data Link)
P/N: RDL401AE

- E: Include 230VAC/9VDC Adaptor
- U: Include 120VAC/9VDC Adaptor


VIN
CON2
VCC
U_RF
JP31
ANT
S2
RF-RST
JP34
RF_DATA
5Vrf
C15
0.1
R41 500
R42 500
D31
D32 LED
LED
S31
SW DIP-4
5V
R22 1K
U8
kick
OPTOISO1
R24 1K
TR_BD139
R21 500
MG5
MOTOR KICK
D21
DIODE
U5.1
L7805CT(3)
U5.2
R15 1K
D9
LED
U5
L298N(15)
F1 FUSE2
F2 FUSE2
D1.1
D1.2
BRIDGE2
MG1 MOTOR 1
MG2 MOTOR 2
R16 1K
D10
U6
F3 FUSE2
F4 FUSE2
D1.3
D1.4
MG3 MOTOR 3
MG4 MOTOR 4
S1
SW-RST
C3 10uF16V
D1
R1 1K
C11 0.1
R6 10K
U1
AT89C4051
RP1 1K
R5 4K
C1 33pF
CRYSTAL
Y1
C2 33pF
J111
DATA_MC
RXD
TXD
Title
Size B
Number
Revision
Date: 17-Mar-2005
Sheet of
File: C:\Documents and Settings\ToRo\My Documents\pcb_robocup\MyDesign1.Ddb