หุ่นยนต์เตะฟุตบอลควบคุมโดย PIC

SOCCER ROBOT CONTROLLED BY PIC

นายวรพงษ์ รุ่งวสันติสุข

นายวรวิทย์ มงคลนิตย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

หุ่นยนต์เตะฟุตบอลควบคุมโดย PIC

SOCCER ROBOT CONTROLLED BY PIC

โดย

นายวรพงษ์ รุ่งวสันติสุข เลขประจำตัว 44010412

นายวรวิทย์ มงคลนิตย์ เลขประจำตัว 44010417

อาจารย์ที่ปรึกษา

ผศ.ดร.ชูชาติ ปิณฑวิรุฬห์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง หุ่นยนต์เตะฟุตบอลควบคุมโดย PIC

SOCCER ROBOT CONTROLLED BY PIC

ผู้จัดทำ

1. นายวรพงษ์ รุ่งวสันติสุข เลขประจำตัว 44010412

2. นายวรวิทย์ มงคลนิตย์ เลขประจำตัว 44010417

.....

(ผศ.ดร.ชูชาติ ปิณฑวิรุจน์)

อาจารย์ที่ปรึกษา

หุ่นยนต์เตะฟุตบอลควบคุมโดย PIC

นาย วรพงษ์ รุ่งวสันติสุข
นาย วรวิทย์ มงคลนิตย์
ผศ.ดร.ชูชาติ ปิณฑวิรุจน์ (อาจารย์ที่ปรึกษา)
ภาคเรียนที่ 2 ปีการศึกษา 2547

บทคัดย่อ

รายงานฉบับนี้ได้นำเสนอหุ่นยนต์เตะฟุตบอล โดยใช้การรับส่งคลื่นวิทยุ และใช้คอมพิวเตอร์สั่งงานให้กับวงจรส่งคลื่นวิทยุผ่านพอร์ตอนุกรม โดยใช้โปรแกรมเดลไฟ ส่วนรับคลื่นวิทยุจะรับคำสั่งมาจากส่วนส่งคลื่นวิทยุ และใช้ควบคุมส่วนมอเตอร์ ให้ทำงานโดยรถบังคับสามารถวิ่ง เดินหน้า ถอยหลัง เลี้ยวซ้าย เลี้ยวขวา โดยส่วนมอเตอร์จะใช้ DC มอเตอร์ 2 ตัว ควบคุมล้อฝั่งซ้ายและฝั่งขวา และใช้อิมเมจประมวลผล หาตำแหน่งลูกบอลและประตู เมื่อหาตำแหน่งประตูหลังจากได้บอลแล้ว จะใช้โซลินอยยิงประตู

SOCCER ROBOT CONTROLLED BY PIC

Mr. Worapong Rungwasantisuk
Mr. Worawit Mongkolnit
Assoc. Prof. Chuchat Pintawirut (Advisor)
2nd Semester, Educational Year 2004

Abstract

The system consists of 3 main parts (i) image processing, (ii) RF receiver and transmitter and (iii) soccer robot. The image processing takes the digitized image of soccer robot in the football field, and calculates the position of the goal, robot and the ball. The RF transmitter sends the command to the robot so that it moves to the ball and kicks the ball toward the goal by using solenoids.

สารบัญ

บทที่ 1
บทนำ

1.1 กล่าวนำ

ปัจจุบันนี้เรื่องราวของการเชื่อมต่ออุปกรณ์ภายนอกผ่านทางพอร์ตของคอมพิวเตอร์เป็นสิ่งจำเป็นและต้องมีการเรียนรู้ทั้งฮาร์ดแวร์และซอฟแวร์ไปพร้อมๆกันด้วยความสามารถของระบบปฏิบัติการวินโดว์และโปรแกรมอย่าง เดลไฟ ทำให้การทำงานเกี่ยวกับพอร์ตและหน้าตาของโปรแกรมที่ปรากฏบนจอภาพมีความสวยงามและทรงประสิทธิภาพ

ด้วยเหตุนี้ผู้จัดทำจึงได้ทำการศึกษาถึงการเชื่อมต่ออุปกรณ์ภายนอกผ่านทางพอร์ตของคอมพิวเตอร์โดยใช้โปรแกรม เดลไฟ ในการควบคุมฮาร์ดแวร์ซึ่งในส่วนของฮาร์ดแวร์ ผู้จัดทำได้พยายามประยุกต์รถบังคับวิทยุให้มีประสิทธิภาพเพิ่มมากขึ้นซึ่งจะนำไปสู่การพัฒนาเป็นรถเตะฟุตบอล (Soccer Robot) ต่อไปได้โดยที่อุปกรณ์ทุกอย่างนั้นสามารถ หาได้โดยทั่วไป

1.2 ขอบเขตการทำงาน

ในการศึกษาและการจัดทำโครงงานนี้มีขอบเขตของโครงงานคือ

ศึกษาความรู้เบื้องต้นเกี่ยวกับการเชื่อมต่ออุปกรณ์ภายนอกผ่านทางพอร์ตอนุกรมคอมพิวเตอร์

ศึกษาการทำงานของเครื่องรับ- ส่งวิทยุของ TLP315 กับ RLP315

ศึกษาการใช้งานโปรแกรม เดลไฟ เบื้องต้น

ศึกษาการดึงภาพจากกล้อง VDO ผ่านทางเดลไฟ

ศึกษาการควบคุมการทำงานของ DC มอเตอร์

สามารถควบคุมรถบังคับวิทยุไปตามทิศทางที่ต้องการได้

ใช้เดลไฟประมวลผลทางอิมเมจ เพื่อหาตำแหน่งของวัตถุ

บทที่ 2
ระบบการสื่อสาร

2.1 รูปแบบของการสื่อสารตามหลักใหญ่ๆ

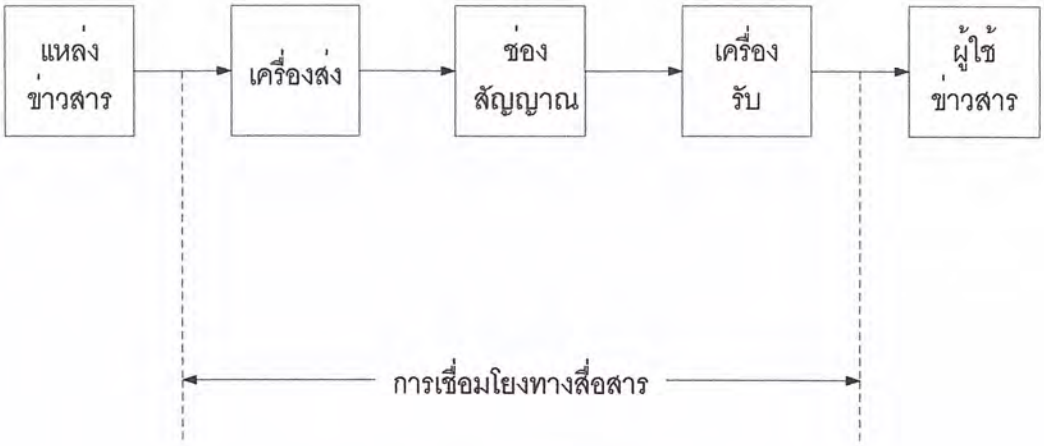
ระบบการสื่อสารปัจจุบันมีด้วยกันหลายรูปแบบ ซึ่งสามารถแยกรูปแบบตามหลักใหญ่ๆ คือ การสื่อสารระหว่างคนกับคน เป็นการสื่อสารโดยใช้ระบบไฟฟ้าที่แปลงจากสัญญาณเสียงเป็นสัญญาณไฟฟ้าเพื่อติดต่อระหว่างคนกับคนนั้นมีใช้กันมานานและยังมีความสำคัญ ยิ่งตราบเท่าทุกวันนี้ ตัวอย่างได้แก่ โทรศัพท์ และวิทยุ เป็นต้น

การสื่อสารระหว่างคนกับเครื่อง การสื่อสารนี้มีความจำเป็นมากสำหรับสังคมที่ทันสมัยที่ต้องการความรวดเร็วในการได้มาซึ่งข้อมูลในการตัดสินใจของนักธุรกิจ และในการควบคุมการทำงานของเครื่องจักรด้วยเครื่องคอมพิวเตอร์โดยมีคนเป็นผู้ออกคำสั่ง

การสื่อสารระหว่างเครื่องกับเครื่อง เป็นการสื่อสารในการย้ายข้อมูลหรือแฟ้มข้อมูลจากเครื่องคอมพิวเตอร์เครื่องหนึ่งไปยังอีกเครื่องหนึ่ง ที่ต้องการความเร็วในการส่งข้อมูลสูง วัตถุประสงค์ของการสื่อสารนี้คือ มุ่งใช้ทรัพยากรร่วมกัน ทั้งทางฮาร์ดแวร์และซอฟต์แวร์ ของศูนย์คอมพิวเตอร์ต่างๆอย่างมีประสิทธิภาพ

2.2 หลักการระบบสื่อสาร

ระบบสื่อสารดังกล่าวข้างต้นสามารถอธิบายการทำงานอย่างๆ ตามรูป 2.1 ดังนี้ ข่าวสารจากแหล่งข่าวสารจะถูกแปลงให้อยู่ในรูปสัญญาณไฟฟ้าที่เรียกว่าสัญญาณมอดูเลตติ้ง (modulating signal) สัญญาณมอดูเลติ้งนี้จะแปลงให้อยู่ในรูปของรหัสตามวิธีการสื่อสารแบบดิจิตอลหรือส่งตรงเข้าเครื่องส่ง (transmitter) ตามวิธีการสื่อสารแบบอนาล็อกก็ได้ ในเครื่องส่งก็จะมีเครื่องโมดูเลต (modulation) ที่ทำหน้าที่โมดูเลตสัญญาณมอดูเลติ้งเข้ากับตัวพาห์ (carrier) ตัวพาห์นี้มีกำลังส่งสูงพอที่จะพาสัญญาณมอดูเลติ้งไปที่ไกลๆได้ด้วยความถี่สูงตามกระบวนการมอดูเลชัน (modulation) จากนั้นสัญญาณจะถูกคัปปลิ้งออกอากาศหรือส่งตามสายส่งก็ได้สัญญาณที่ผ่านช่องสัญญาณ (channel) ซึ่งไม่ว่าจะเป็นอากาศหรือสายส่งก็ตามจะถูกรบกวนจากเสียงรบกวน (noise) หรือสัญญาณแทรกที่ไม่พึงปรารถนา (undesired interference) เมื่อสัญญาณไปถึงเครื่องรับของผู้ใช้ปลายทาง เสาอากาศของเครื่องรับจะแปลงสัญญาณที่เป็นคลื่นแม่เหล็กไฟฟ้า แต่ถ้าส่งตามสายเครื่องรับจะรับสัญญาณในรูปของกระแสไฟฟ้าหรือแรงดันไฟฟ้าได้ทันที จากนั้นเครื่องดีมอดูเลเตอร์ (demodulator) ในเครื่องรับจะแปลงสัญญาณที่มีความถี่สูงให้มีความถี่ต่ำลงและแยกสัญญาณโมดูเลติงออกจากตัวพาห์ตามกระบวนการดีโมดูเลชัน(demodulation) และถูกถอดรหัส (decode) กลับเป็นสัญญาณอนาล็อกตามเดิม ตามวิธีการสื่อสารแบบดิจิตอล



รูป 2.1 แผนภาพของระบบสื่อสาร

2.3 ชนิดของสัญญาณและผลต่างๆที่ต้องคำนึงถึง

สัญญาณที่ใช้ในการส่งข่าวสาร มีตั้งแต่รูปคลื่นไซน์บริสุทธิ์เพียงความถี่เดียวจนถึงสัญญาณคอมเพล็กซ์ที่ประกอบด้วยรูปคลื่นไซน์หลายความถี่มารวมกันจนเป็นรูปพัลส์ เป็นต้น ในบางกรณีอาจมีการจัดรูปพัลส์ให้อยู่ในรูปที่จะลดความเพี้ยน (distortion) ต่างๆที่อาจเกิดขึ้นในวงจรสื่อสารเมื่อการตอบสนอง (response) ของวงจรนั้นไม่คงพอที่จะทำให้พัลส์ ผ่านได้โดยไม่เกิดความผิดเพี้ยน ตัวอย่างเช่น พัลส์ชนิดเร็ส์โคไซน์ (raised cosine) เป็นต้น

การเลือกสัญญาณที่มีความถี่ใดหรือรูปร่างใดนั้น สามารถทำได้โดยให้ผ่านเครื่องกรองความถี่ที่ไม่ต้องการออกไปและเปลี่ยนแปลงรูปร่างของสัญญาณตามต้องการได้ สัญญาณที่มีค่าของขนาด (amplitude) เป็นค่าต่อเนื่องเราเรียกว่าสัญญาณอนาล็อก ส่วนสัญญาณที่มีขนาด (amplitude) เปลี่ยนแปลงเป็นค่าของเลขลงตัวเรียกว่าสัญญาณดิจิตอล

สิ่งที่สำคัญอย่างหนึ่งในระบบสื่อสารคือ ความถี่ซึ่งแน่นอนที่สุดคือความหมายของช่วงกว้าง (bandwidth) ของความถี่ที่ต้องเข้ามาเกี่ยวข้องด้วยช่วงกว้างความถี่นี้จะขึ้นอยู่กับลักษณะการใช้งาน เริ่มตั้งแต่ประมาณ 100 Hz ซึ่งใช้ในระบบโทรเลขจนถึงหลายๆเมกเฮิร์ท(MHz)ในระบบโทรทัศน์ กล่าวโดยสรุปเราจะต้องออกแบบให้ระบบสื่อสารเรามีความกว้างของแบนวิดท์แคบที่สุดโดยที่จะต้องให้มีข้อมูลที่มีความสำคัญอยู่ครบถ้วนนั้นเอง

สิ่งที่สำคัญอีกอย่างหนึ่งคือ กำลัง (power) เครื่องส่งวิทยุอาจมีกำลังส่งเป็นร้อยกิโลวัตต์ในขณะที่กำลังส่งในสายส่งมีแค่มิลลิวัตต์ในกรณีที่กำลังของสัญญาณมีกำลังอ่อนมากๆนั้นการออกแบบวงจรก็จะยุ่งยากขึ้นโดยเฉพาะการถ่ายเทพลังงานจากวงจรหนึ่งไปวงจรหนึ่งเพื่อให้ได้กำลังงานสูงสุดนั้น การทำให้อิมพีแดนซ์ระหว่างวงจรหนึ่งต่อวงจรหนึ่งแมทช์ (match) กันจึงเป็นสิ่งสำคัญ

การลดทอน (attenuation) หรือการสูญเสีย (loss) ในวงจรก็เป็นสิ่งสำคัญ ที่จะต้องคำนึงถึงในการออกแบบวงจร อีกทั้งอัตราส่วนระหว่างสัญญาณต่อสัญญาณรบกวนด้วยที่ได้กล่าวไว้ข้างต้นแล้ว

2.4 ทำไมต้องมีการมอดูเลชัน

มอดูเลชันจะให้ประโยชน์ต่างๆ ดังต่อไปนี้

1. ทำให้สัญญาณมีกำลังสูงทำให้สามารถเดินทางเหนือแพร่กระจาย ไปที่ไกลๆได้
2. ทำให้สัญญาณมีความถี่สูงขึ้น ซึ่งเหมาะสำหรับการรับส่งสัญญาณมากขึ้นเพราะใช้เสาอากาศที่สั้นลงได้
3. สามารถแบ่งความถี่ให้หลายๆสัญญาณส่งพร้อมกันภายใต้ตัวพาห์ตัวเดียวกันได้เรียกว่าการมัลติเพล็กซิง
4. ทำให้สัญญาณมีความต้านทานการรบกวนของเสียงรบกวนหรือของสัญญาณรบกวนได้ดีขึ้น

2.5 RF Transmission

ในการติดต่อระหว่างอุปกรณ์โดยใช้ RF นั้นเราจะใช้ TLP315 และ RLP315 ซึ่งเป็นโมดุลในการรับ - ส่ง RF โดย TLP315 เป็นโมดุลตัวส่งที่มี 4 ขา ประกอบไปด้วย

1. GND
2. Data In
3. V_{cc}
4. Antenna (RF Output)

ดังรูปที่ 2.1 แสดงขา 1, 2, 3 และ 4 เรียงจากซ้ายไปขวาตามลำดับ

รูปที่ 2.1 แสดง TLP315 IC ภาคส่ง RF

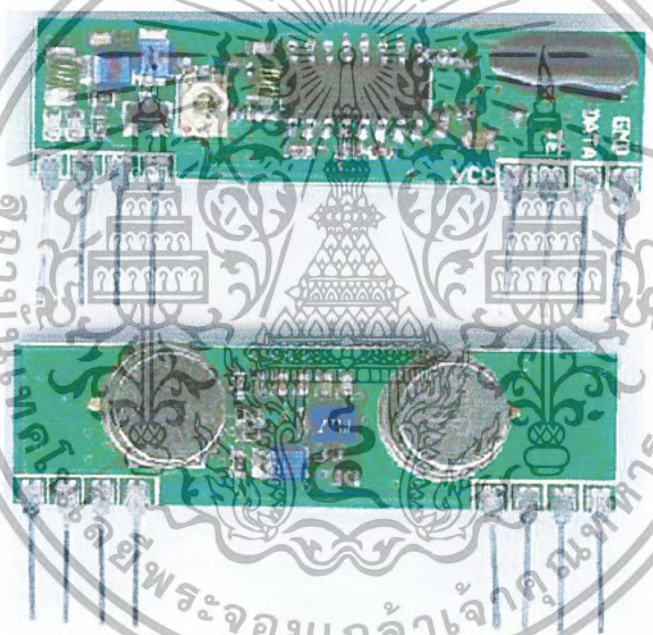
Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Vcc	Operating Supply Voltage	-	2.0	-	12.0	V
Icc 1	Peak Current (2V)	-	-	-	1.64	mA
Icc 2	Peak Current (12V)	-	-	-	19.4	mA
Vh	Input High Voltage	Idata = 100uA (High)	Vcc-0.5	Vcc	Vcc+0.5	V
Vl	Input Low Voltage	Idata = 0uA (Low)	-	-	0.3	V
FO	Absolute Frequency	315 MHz Module	314.8	315	315.2	MHz
PO	RF Output Power _50 ohm	Vcc = 9-12 V	-	16	-	dBm
		Vcc = 5-6 V	-	14	-	dBm
DR	Data Rate	External Encoding	512	4.8K	200K	bps

ตารางที่ 2-1 แสดงข้อมูลที่จำเป็นของ TLP315

และ RLP315 ซึ่งเป็นโมดุลที่มี 8 ขา ประกอบด้วย

1. GND
2. Digital Data Output
3. Linear Output/Test
4. V_{cc}
5. V_{cc}
6. GND
7. GND
8. Antenna

ดังรูปที่ 2.2 เรียงตามขาจากซ้ายไปขวา 1, 2, 3, 4, 5, 6, 7 และ 8 ตามลำดับ ภาพล่าง



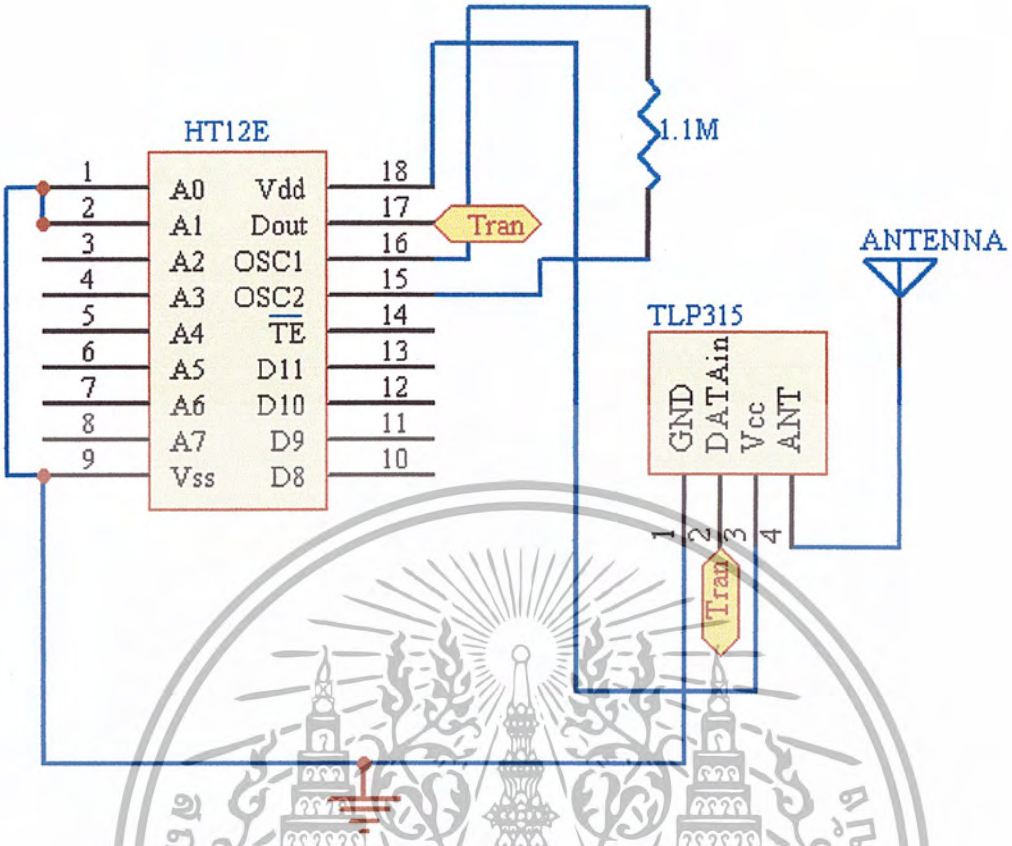
รูปที่ 2.2 แสดง RLP IC ภาครับ RF

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Vcc	Operating Data Voltage	-	3.3	5.0	6.0	V
Itot	Operating Current	-	-	4.5	-	mA
Vdata	Data Out	Idata = +200 uA (High)	Vcc-0.5	-	Vcc	V
		Idata = -10 uA (Low)	-	-	0.3	V

Electrical Characteristics					
Characteristics	SYM	Min	Typ	Max	Unit
Operation Radio Frequency	FC	315,418 and 433.92			MHz
Sensitivity	Pref	-	-110	-	dBm
Channel Width	-	-	+/- 500	-	kHz
Noise Equivalent BW	-	-	4	-	kHz
Receiver Turn On Time	-	-	5	-	ms
Operation Temperature	TOP	-20	-	80	C
Baseboard Data Rate	-	-	4.8	-	Hz

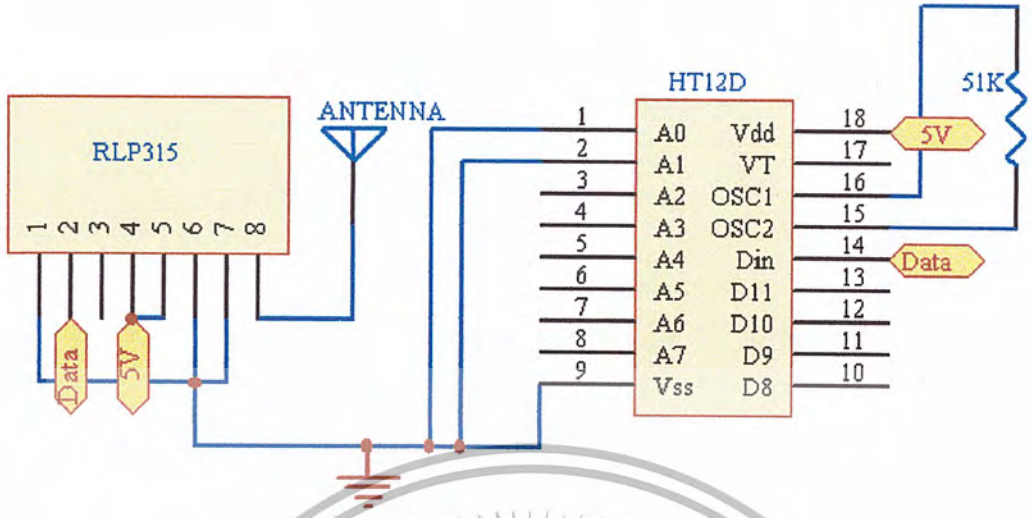
ตารางที่ 2-2 แสดงข้อมูลที่จำเป็นสำหรับ RLP315

การประยุกต์ใช้งานเราจะใช้งานร่วมกับ IC เบอร์ HT12E และ HT12D ซึ่งเป็นตัว Encoder และ Decoder เพื่อที่จะทำให้สามารถเพิ่มช่องสัญญาณในการรับ - ส่งได้



รูปที่ 2.3 แสดงการประยุกต์ใช้งาน TLP315

เมื่อทำการทำการset ค่า A_0 และ A_1 แล้วป้อนข้อมูลเข้าทางขา $D_8 - D_{11}$ และทำการ Set ขา TE ของ HT12E ให้เป็น 0 แล้วข้อมูลจะถูก Encoder มาที่ขา D_{OUT} ของ HT12E และเมื่อเรานำขานี้ไปต่อกับขา Data In ของ TLP315 ก็จะกลายเป็นการส่งข้อมูลออกไปทาง Antenna ของ TLP315

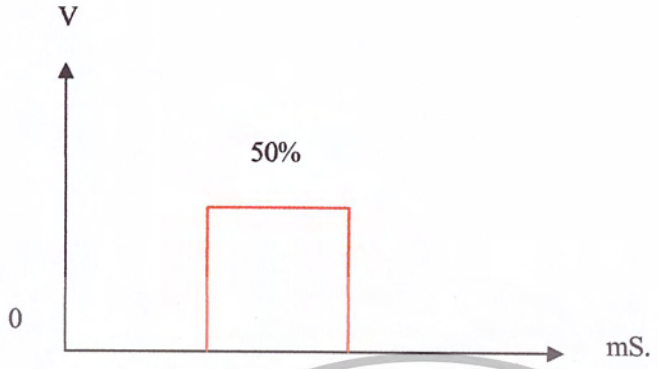


รูปที่ 2.4 แสดงการประยุกต์ใช้งาน RLP315

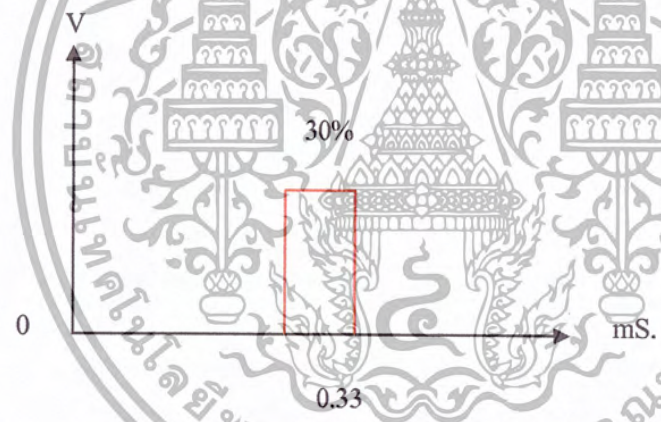
เมื่อข้อมูลไปถึงภาครับ โดยรับจาก Antenna ของ RLP315 ซึ่งข้อมูลนี้ก็คือข้อมูลที่ถูก Encoder มาแล้วเราก็นำข้อมูลนี้ผ่านเข้าไปที่ HT12D (ที่ถูก set ขาA_0และA_1ให้ตรงกับตัวส่งแล้ว) ทางขา Data In ของ HT12D ข้อมูลที่ถูก Decode แล้วก็จะปรากฏอยู่ที่ $D_8 - D_{11}$ ของ HT12D ในขณะที่ขา VT ของ HT12D ก็จะเป็น Logic 1 เพื่อแสดงว่า HT12D สามารถ Decode สัญญาณได้

หมายเหตุ

โมดูล TLP315 และ RLP315 ไม่สามารถรับ–ส่งข้อมูลจาก Serial Port ได้โดยตรงจึงต้องใช้ HT12E และ HT12D เป็นตัวช่วยในการส่งข้อมูลแทน หรือถ้าจำเป็นต้องใช้งานโดยการส่งจาก Serial Port โดยตรงจะต้องทำการวนลูปส่งข้อมูลซ้ำกัน 300 – 600 ครั้ง ตัวรับจะสามารถรับค่าได้ประมาณ 3 – 10 ครั้ง เมื่อทำการส่งคลื่นสี่เหลี่ยมความถี่ 3kHz ที่ขา Data In ของ TLP315 และทำการวัดที่ขา Digital Data Out ของ RLP315 ผลที่ได้เป็นดังรูปที่ 2.5 และรูป 2.6

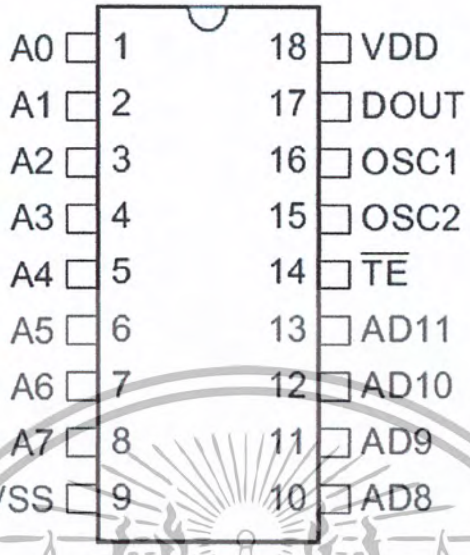


รูปที่ 2.5 แสดงความถี่ Data ที่ขา Data In ของ TLP315



รูปที่ 2.6 แสดงความถี่และ duty cycle ของ Data ที่ขา Digital Data Out ของ RLP315

2.6 HT12E IC Encoder ของภาคส่ง



รูปที่ 2.5 แสดงภาพ IC HT12E

Pin Name	I/O	Internal Connection	Description
A0-A7	I	NMOS Transmission Gate Protection Diode	Input pins for address A0-A7 setting. These pins can be externally set to VSS or left open
AD8-AD11	I	NMOS Transmission Gate Protection Diode	Input pins for address/data AD8-AD11 setting. These pins can be externally set to VSS or left open
DOUT	O	CMOS OUT	Encoder data serial transmission output
TE	I	CMOS IN Pull - high	Transmission enable, active low
OSC 1	I	OSCILLATOR 1	Oscillator input pin
OSC 2	O	OSCILLATOR 1	Oscillator input pin
VSS	I	-	Negative power supply, grounds
VDD	I	-	Positive power supply

ตารางที่ 2-3 แสดงคำบรรยายขาแต่ละ pin ของ HT12E

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	-	-	2.4	5	12	V
I_{STB}	Standby Current	3V	Oscillator Stops	-	0.1	1	uA
		12V		-	2	4	uA
I_{DD}	Operating Current	3V	No load f_{OSC} = 3kHz	-	40	80	uA
		12V		-	150	300	uA
I_{DOUT}	Output drive current	5 V	V_{OH} = 0.9 V_{DD} (Source)	-1	-16	-	mA
			V_{OL} = 0.1 V_{DD} (Sink)	1	16	-	mA
V_{IH}	"H" Input Voltage	-	-	0.8 V_{DD}	-	V_{DD}	V
V_{IL}	"L" Input Voltage	-	-	0	-	0.2 V_{DD}	V
f_{OSC}	Oscillator Frequency	5V	R_{OSC} = 1.1 MΩ	-	3	-	KHz
R_{TE}	TE pull – high resistance	5 V	V_{TE} = 0V	-	1.5	3	MΩ

ตารางที่ 2-4 แสดงข้อมูลที่จำเป็นสำหรับ HT12E

2.6.1 Address/data programming (preset)

สถานะของแต่ละขาของ address/data สามารถกำหนดค่าจำเพาะได้โดยให้ pre-set เป็นลอจิก Low หรือ High ถ้าสัญญาณ transmission-enable นำมาใช้ encoder สแกนและส่งผ่านสถานะของ 12 บิต ของ address/data แบบอนุกรมผ่าน A0 ถึง AD11 สำหรับ HT12E encoder

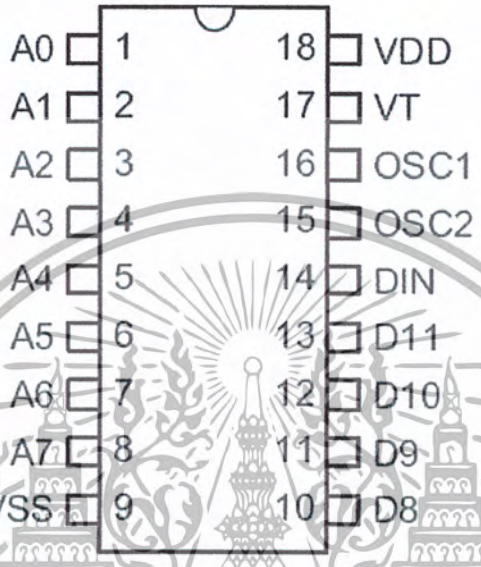
เนื่องจากการส่งผ่านข้อมูล บิตเหล่านี้จะถูกส่งด้วยกระบวนการซิงโครนัส ถ้า trigger signal ไม่ทำงาน IC จะเข้าสู่โหมด standby คือไม่ทำงาน และกระแสจะลดลงจนน้อยกว่า 1 uA สำหรับแหล่งจ่ายแรงดัน 5V

โดยปรกติการประยุกต์ใช้ preset ขา address ด้วยรหัสจำเพาะโดยใช้ DIP หรือ PCB ขณะที่ข้อมูลถูกเลือกโดยการกดสวิตช์

2.6.2 Transmission enable

สำหรับ Encoder HT12E การส่งผ่านจะถูกกำหนดโดย ใช้สัญญาณ Low ให้กับขา TE

2.7 HT12D IC Decoder ของภาครับ



รูปที่ 2.6 แสดงภาพ IC HT12D

Pin Name	I/O	Internal Connection	Description
A0-A11	I	NMOS Transmission Gate	Input pins for address A0-A11 setting. These pins can be externally set to VSS or VDD
D8-D11	O	CMOS OUT	Output data pins
DIN	I	CMOS IN	Serial data input pin
VT	O	CMOS OUT	Valid transmission, active high
OSC 1	I	OSCILLATOR	Oscillator input pin
OSC 2	O	OSCILLATOR	Oscillator output pin
VSS	I	-	Negative power supply, grounds
VDD	I	-	Positive power supply

ตารางที่ 2-5 แสดงคำบรรยายขาแต่ละ pin ของ HT12D

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	-	-	2.4	5	12	V
I_{STB}	Standby Current	3V	Oscillator Stops	-	0.1	1	uA
		12V		-	2	4	uA
I_{DD}	Operating Current	5V	No load f_{OSC} = 150kHz	-	200	400	uA
I_O	Data output source current (D8-D11)	5 V	V_{OH} = 4.5V	-1	-16	-	mA
	Data output silk current (D8-D11)	5V	V_{OL} = 0.5V	1	16	-	mA
I_{VT}	VT output source current	5 V	V_{OH} = 4.5V	-1	-16	-	mA
	VT output silk current		V_{OL} = 0.5V	1	16	-	mA
V_{IH}	"H" Input Voltage	5V	-	3.5	-	5	V
V_{IL}	"L" Input Voltage	5V	-	0	-	1	V
f_{OSC}	Oscillator Frequency	5V	R_{OSC} = 51k Ω	-	150	-	KHz

ตารางที่ 2-6 แสดงข้อมูลที่จำเป็นสำหรับ HT12D

2.7.1 การทำงานของ HT12D

12 บิต อนุกรมของ decoder สามารถแบ่งได้หลายกรณีโดยการรวม ขา Address และขา Data โดยการจับคู่กับ 12 บิต อนุกรมของ encoder

Decoder รับข้อมูลที่ส่งมาจาก encoder แปล N บิตแรกของรหัสเป็น Address และ 12-N บิตสุดท้ายเป็น Data เมื่อ N เป็น รหัส Address สัญญาณที่ขา DIN active จึงจะทำการ decode ข้อมูล

Decoder จะ check ข้อมูลจาก Address ที่ได้รับมาอย่างต่อเนื่อง 3 ครั้ง ถ้า Address Code ตรงกับ Address ของ decoder 12-N บิตของข้อมูลจะถูก decode ไปที่ขา output และขา VT จะเป็น High เพื่อบอกว่าข้อมูลถูกต้อง

ดังนั้นขา VT จะเป็น High เมื่อข้อมูลถูกต้องเท่านั้น โดยปรกติจะเป็น Low

บทที่ 3
ความรู้เบื้องต้นเกี่ยวกับเดลไฟ

3.1 รู้จักกับ Delphi 7

Delphi 7 คือ ซอฟแวที่นำมาใช้เขียนโปรแกรมเพื่อสร้างแอพพลิเคชั่น หรือซอฟแวร์อีกที โดยมันจะประกอบไปด้วยเครื่องมือชนิดต่างๆ ที่ใช้ในการเขียนโปรแกรมทำได้อย่างสะดวก

Delphi 7 จัดเป็นเครื่องมือเขียนโปรแกรมชนิด Visual Programming เช่นเดียวกับ Visual Basic หรือ Visual C++ โดยมีข้อดีคือ สามารถเขียนโปรแกรมได้ง่าย และให้ผลงานออกมารวดเร็ว ซึ่งจะแตกต่างจากเครื่องมือเขียนโปรแกรมรุ่นเดิมๆ ที่มีความยุ่งยากซับซ้อน ดังนั้น Delphi 7 จึงใช้งานได้ง่ายและรวดเร็วกว่า

Delphi 7 ได้รับการพัฒนาให้สามารถใช้แอพพลิเคชั่นแบบข้ามแพล็ตฟอร์มได้นั่นคือ สามารถพัฒนาแอพพลิเคชั่นที่ทำงานได้ทั้งบน Windows และลีนุกซ์นั่นเอง

Delphi 7 นั้นมีความสามารถมากมาย ได้รับการต้นรับเป็นอย่างดีจากนักพัฒนาแอพพลิเคชั่นทั่วโลกรวมทั้งเมืองไทยด้วยสามารถสร้างแอพพลิเคชั่นบน Windows ได้หลายเวอร์ชั่น ซึ่งแอพพลิเคชั่นที่สร้างจาก Delphi ได้รับการยกย่องเป็นอย่างมากในด้านขนาดโปรแกรมที่กะทัดรัด และทำงานได้อย่างรวดเร็ว เมื่อเทียบกับแอพพลิเคชั่นที่สร้างจากเครื่องมือตัวอื่นๆทำให้ Delphi แต่ละเวอร์ชั่นได้รับรางวัลจากสถาบันต่างๆทั่วโลก

สำหรับในประเทศไทย Delphi ถูกนำไปใช้พัฒนาระบบงานด้านฐานข้อมูล โดย Delphi มีจุดเด่นในการสร้างแอพพลิเคชั่นที่ติดต่อกับฐานข้อมูลได้หลายรูปแบบ

3.2 หลักการเขียนโปรแกรมด้วย Delphi 7

3.2.1 ตัวแปรและชนิดของข้อมูล

ก่อนการเขียนโปรแกรมสิ่งแรกที่ต้องรู้จักคือ ข้อมูล เพราะข้อมูลคือ สิ่งที่เราใช้ในการควบคุม และแลกเปลี่ยนกันระหว่างผู้ใช้งานกับตัวโปรแกรม

เมื่อมีข้อมูลแล้วเราจะต้องสิ่งที่เรียกว่า ตัวแปร ขึ้นมาเก็บข้อมูลนั้นไว้ซึ่ง Delphi 7 นั้นสามารถทำงานกับข้อมูลได้หลายชนิด แบ่งข้อมูลได้เป็นกลุ่มๆดังนี้

1. ข้อมูลเบื้องต้น (Simple Type) เป็นชนิดข้อมูลเบื้องต้นที่ Delphi ใช้งานพื้นฐานเช่น ตัวอักษร, เลขจำนวนเต็ม, เลขทศนิยม เป็นต้น
2. ข้อมูลชนิดข้อความ (String) เป็นการนำข้อมูลเบื้องต้นชนิดตัวอักษรมาเรียงต่อกัน
3. ข้อมูลชนิดโครงสร้าง (Structure Type) เป็นชนิดข้อมูลเกิดจากการนำเอาชนิดข้อมูลเบื้องต้นมาประกอบกันเป็นโครงสร้างเพื่อเก็บข้อมูลเป็นชุดๆ เช่น อาร์เรย์

4. ข้อมูลชนิดอ้างอิง (Pointer) เป็นข้อมูลที่ราใช้ในการอ้างอิงกับหน่วยความจำในคอมพิวเตอร์ซึ่งขะใช้บอกตำแหน่งข้อมูล

3.2.1.2 การประกาศตัวแปร

ในการใช้งานข้อมูลนั้นต้องมีตัวแปรมารองรับ ตัวแปรจะถูกใช้งานได้ก็ต่อเมื่อมีการประกาศตัวแปรเสียก่อน ซึ่งก็เหมือนกับต้องมาลงทะเบียนให้ Delphi รู้จักเสียก่อนจึงจะใช้งานได้ ซึ่งมีรูปแบบของการประกาศตัวแปรดังนี้

VAR ชื่อตัวแปร : ชนิดตัวแปร ;

สำหรับการประกาศตัวแปร เรามักจะประกาศไว้ต้นโปรแกรม หรือตอนต้นของโปรแกรมย่อย ซึ่งจะประกาศเป็นตัวแปรชนิดเดียวหรือหลายชนิดพร้อมๆกันก็ได้ โดยมกฎเกณฑ์ดังนี้

1. ตัวแปรต้องขึ้นต้นด้วยตัวอักษรหรือเส้นขีดล่างเท่านั้น (_) แล้วตามด้วยตัวอักษรตัวเลข เครื่องหมาย
2. ห้ามเว้นวรรคในชื่อตัวแปร และในตัวแปรห้ามมีเครื่องหมายต่างๆ เช่น $, %, *, @, +, - เป็นต้น
3. ตัวแปรหนึ่งตัวความยาวไม่เกิน 63 ตัวอักษร
4. ตัวอักษรตัวพิมพ์ใหญ่ พิมพ์เล็กมีความต่างกัน เพราะฉะนั้นตัวแปร Name, name, NAME จึงต่างกัน
5. ตัวแปรที่ตั้งห้ามขึ้นห้ามตรงกับคำที่สงวนไว้ เช่น begin, end, if เป็นต้น

ข้อมูลชนิด Array

อาร์เรย์ เป็นรูปแบบการจัดเก็บข้อมูลแบบเป็นชุด ซึ่งข้อมูลแต่ละตัวในอาร์เรย์ต้องเป็นข้อมูลชนิดเดียวกัน โดยจะใช้ Index ในการอ้างอิงถึงข้อมูลแต่ละตัวในอาร์เรย์ รูปแบบมีดังนี้

VAR ชื่อตัวแปรชนิด_Array : array [indexเริ่มต้น.....indexสุดท้าย] of ชนิดข้อมูลที่เก็บใน_Array ;

Type ชนิดของอาร์เรย์ = array [indexเริ่มต้น.....indexสุดท้าย] of ชนิดข้อมูลที่เก็บใน_Array ;

รู้จักกับ Dynamic Array

ในบางครั้งการทำงานกับอาร์เรย์เราอาจจะพบปัญหาต่อไปนี้

1. ยากที่จะกำหนดขนาดของอาร์เรย์ เช่น การสร้างอาร์เรย์ที่เก็บรายชื่อพนักงานบริษัทแต่ไม่ทราบว่ามีพนักงานกี่คน แถมมีโอกาสที่จะเพิ่มหรือลดได้ตลอดเวลา

2. การประกาศอาร์เรย์ขนาดใหญ่ จากข้อ1. อาจทำให้เราแก้ปัญหาโดยประกาศอาร์เรย์ที่ใหญ่ๆไปเลย ซึ่งทำให้มีการใช้พื้นที่ไม่คุ้มค่า เพราะมีหน่วยความจำที่จองไว้มากแต่ไม่ได้ใช้

3. ประกาศอาร์เรย์ขนาดใหญ่เอาไว้ แต่ไม่ได้ถูกใช้งานทันที แม้บางครั้งต้องใช้อาร์เรย์ขนาดใหญ่ก็จริงแต่กว่าจะได้ใช้งานก็อาจจะเป็นการทำงานในช่วงท้ายๆของโปรแกรม ทำให้หน่วยความจำที่จองไว้ตั้งแต่ต้นไม่ได้นำมาใช้งานทันที เป็นการสิ้นเปลืองทรัพยากรของระบบโดยที่ไม่จำเป็น

จากปัญหาข้างต้นทำให้มีการคิดไดนามิกอาร์เรย์ ขึ้นมา โดยจะเป็นอาร์เรย์ที่มีคุณสมบัติพิเศษคือ ไม่จำเป็นต้องระบุขนาดในตอนประกาศ แต่ไม่สามารถเปลี่ยนขนาดอาร์เรย์ได้ในขณะที่แอพพลิเคชั่นทำงาน ซึ่งเราจะต้องเขียนโค้ดเอง

ในการใช้งานเราจะประกาศตัวแปรแบบอาร์เรย์เอาไว้โดยไม่ระบุขนาด และจะใช้การระบุขนาดของอาร์เรย์ด้วยฟังก์ชัน SetLength ส่วน Index ของไดนามิกอาร์เรย์จะเริ่มนับตั้งแต่ 0 ขึ้นไป

3.2.2 การใช้งานค่าคงที่

ในบางครั้งเราอาจจะต้องใช้ข้อมูลค่าหนึ่งซึ่งมีค่าเหมือนเดิมไปตลอดโดยไม่เปลี่ยนแปลง เราเรียกข้อมูลตัวนั้นว่าค่าคงที่ (Constant)

ค่าคงที่นั้นสามารถกำหนดให้เป็นข้อมูลชนิดใดก็ได้ ไม่จำเป็นต้องเป็นตัวเลขหรือตัวอักษรเท่านั้นซึ่งมีรูปแบบการกำหมดค่าดังนี้

Const ชื่อค่าคงที่ = ค่าคงที่ที่กำหนด ;

ตัวดำเนินการเปรียบเทียบ

ตัวดำเนินการเปรียบเทียบหรือ Comparison Operator นี้ทำหน้าที่เปรียบเทียบค่าของตัวแปร 2 ค่า ในลักษณะมากกว่าหรือน้อยกว่า โดยจะให้ผลลัพธ์เป็นค่าจริง (True) หรือเท็จ (False)

3.3 การควบคุมทิศทางการทำงานของโปรแกรม

การควบคุมทิศทางของโปรแกรม เป็นการเลียนแบบการทำงานของมนุษย์ เพราะในสถานการณ์ต่างๆสมองของเราต้องมีการคิดเพื่อตัดสินทำงานในแบบต่างๆ

ในการเขียนโปรแกรมก็คือ การจำลองตัวเราลงไปเพื่อจัดการกับสถานการณ์ต่างๆ ซึ่งเราต้องมีข้อมูลประกอบการพิจารณา ถ้าข้อมูลบอกเราอย่างหนึ่งเราก็ต้องทำงานแบบหนึ่ง แต่ถ้าบอกข้อมูลอีกอย่างเราก็ต้องทำงานอีกแบบที่แตกต่างกัน ซึ่งการที่ต้องจัดการกับสถานการณ์ต่างๆนี้เราเรียกมันว่า การควบคุมทิศทางของโปรแกรม (Control Flow)

การควบคุมทิศทางของโปรแกรม มีอยู่ด้วยกัน 2 รูปแบบได้แก่

1. การตัดสินใจ คือการเลือกจากตัวเลือกที่มีอยู่
2. การทำงานแบบวนซ้ำ คือวนซ้ำจนกว่าจะครบรอบหรือครบจำนวนหรือถูกกำหนดให้หยุดการวนซ้ำ

การใช้งาน If...Then...Else

จะเป็นการทำงานที่มีการเลือกหนึ่งตัวเลือกจากสองตัวเลือก ซึ่งเราจะต้องมีตัวแปรตัวหนึ่งถูกตรวจสอบว่า มีค่าเป็นจริงหรือเป็นเท็จ ถ้าเป็นจริง (TRUE) ก็จะต้องเลือกตัวเลือกแรกซึ่งอยู่หลังคำว่า Then แต่ถ้าตรวจสอบว่าเป็นเท็จ ก็จะต้องเลือกตัวเลือกอีกตัวที่อยู่หลังคำว่า Else รูปแบบการใช้งานมีดังนี้

If.....then

ชุดคำสั่งเมื่อเงื่อนไขนั้นตรวจสอบแล้วเป็น True

Else

ชุดคำสั่งเมื่อเงื่อนไขนั้นตรวจสอบแล้วว่าเป็น False

การใช้งาน Case...Of

เราจะใช้ Case...of ในการเลือกหนึ่งตัวเลือกจากตัวเลือกที่มีหลายตัวโดยเราจะใช้ตัวแปรหนึ่งตัวตรวจสอบว่า ตรงกับตัวเลือกใด โดยที่ตัวเลือกแต่ละตัวเราก็กำหนดให้มีการทำงานที่แตกต่างกันด้วย รูปแบบมีดังนี้

Case.....of

ชุดคำสั่งเมื่อตัวแปรตรวจสอบตรงกับค่าที่ 1

.

.

.

Else

ชุดคำสั่งเมื่อค่าตัวแปรตรวจสอบไม่ตรงกับค่าใดๆ

End;

การใช้งาน While...do

While...do เป็นการทำงานที่เราต้องการให้มีการวนซ้ำไปเรื่อยๆ โดยทุกๆ ครั้งที่จะวนซ้ำใหม่ ให้มีการตรวจสอบเงื่อนไขการวนซ้ำก่อนทุกครั้ง ถ้าเงื่อนไขเป็นจริง ก็จะวนซ้ำต่อไป แต่ถ้าเป็นเท็จก็ให้หยุดการวนซ้ำ มีรูปแบบการใช้งานดังนี้

While.....do

.....

การใช้ repeat...until

การทำงานของ repeat...until นั้นจะแตกต่างจาก while...do ตรงที่การหยุดการวนซ้ำจะหยุดก็ต่อเมื่อตรวจสอบแล้วว่าเป็นจริงจึงจะหยุด

Repeat

.....

Until .....

การใช้งาน for...to....do

จะใช้เมื่อแอพพลิเคชั่นทำงานซ้ำด้วยจำนวนรอบที่แน่นอน ซึ่งจะใช้ตัวแปรหนึ่งตัวทำหน้าที่ในการนับว่าทำซ้ำครบรอบหรือไม่ โดยแต่ละครั้งของการวนรอบตัวแปรที่นับรอบนั้นจะเพิ่มค่าขึ้นทีละหนึ่งๆ และเพิ่มไปเรื่อยๆ จนกว่าจะมากกว่าค่าสุดท้าย

For [ตัวแปรนับรอบ] := [ค่าเริ่มต้น] to [ค่าสุดท้าย] do

.....

การใช้งาน for...downto...do

สามารถลดค่าตัวแปรในการนับรองลงทีละ 1 โดยใช้งานดังนี้

For [ตัวแปรนับรอบ] := [ค่าเริ่มต้น] downto [ค่าสุดท้าย] do

.....

สร้าง และใช้งาน Procedure

โพรซีเจอร์ คือโปรแกรมย่อยที่ทำงานเสร็จแล้วจะไม่มีการคืนค่าใดๆกลับมายังผู้ที่เรียกใช้งานโพรซีเจอร์นั้น เราจึงใช้โพรซีเจอร์ในงานที่ไม่สนใจ หรือไม่ต้องการผลลัพธ์ที่ได้จากการทำงานของโพรซีเจอร์ เพียงแค่สั่งให้โพรซีเจอร์ทำงานเท่านั้นก็เพียงพอแล้ว

```
Procedure [ชื่อโพรซีเจอร์] ;
Begin
    .....
End;
```

สร้างและใช้งาน Function

ฟังก์ชัน คือโปรแกรมย่อยที่ถูกเรียกใช้งาน และภายหลังที่ทำงานเสร็จแล้วจะคืนค่ากลับมาให้กับผู้ที่เรียกฟังก์ชันนั้นได้นำไปใช้งานต่อ ดังนั้นการใช้งานฟังก์ชันจึงมักจะสนใจที่ผลการทำงานที่ได้รับกลับมา ซึ่งหมายถึงการนำค่านั้นไปใช้ในคำสั่งถัดไปหลังจากเรียกใช้ฟังก์ชันนั้น

```
Function [ชื่อฟังก์ชัน] : [ชนิดของตัวแปร] ;
Begin
        [ชุดคำสั่งในโปรแกรมย่อยแบบ function]
        [ชื่อฟังก์ชัน] := [ค่าที่คืนกลับมา]
end;
```

3.4 การเชื่อมต่อ Serial Port ด้วยเดลไฟ

การเชื่อมต่อจะใช้ Component ที่มีชื่อว่า Comport ในการติดต่อ โดยมีคำสั่งสำคัญดังนี้

Component. Open ; //เป็นการเปิด Serial Port ก่อนการใช้งาน

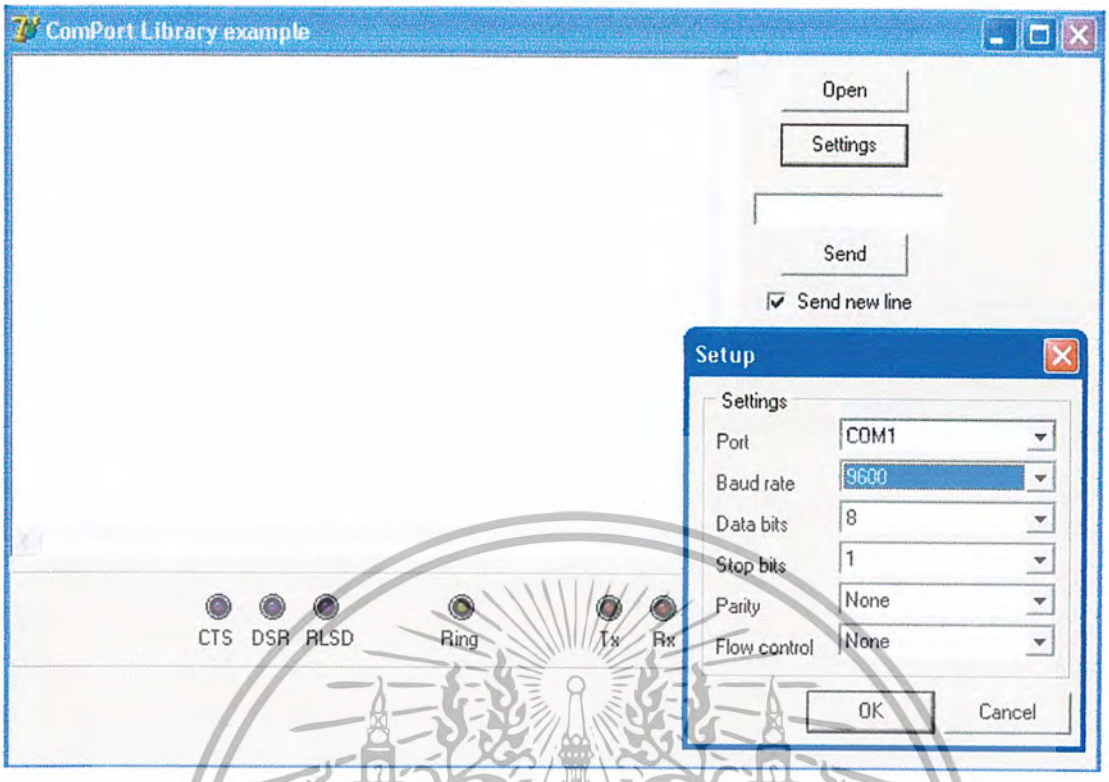
Component. Close ; //ปืนการปิด Serial Port หลังการใช้งาน

Component. WriteStr('B1') ; //เป็นการส่งผ่านข้อมูล (B1) ออกไปจาก Serial Port

Component. ReadStr(Str,Component) ; //เป็นการรับข้อมูลทาง Serial Port

Conponent. ShowSetupDialog ; //เป็นการ Setup Serial Port ว่าใช้ Com1 หรือ Com 2 และใช้ baud rate เท่าไร

ในการใช้งาน Serial Port นั้น ต้องมีการเปิด Port ก่อนที่จะมีการส่ง Data ไม่เช่นนั้นจะเกิด Error ขึ้นได้



รูปที่ 3.1 แสดงโปรแกรมตัวอย่างในการส่งของมูลทาง Serail Port

3.5 การ Capture Picture Setting

สำหรับการเชื่อมต่อกับกล้องเข้า Computer นั้นเราใช้ Card Capture โดยเราจะนำสายสัญญาณจากกล้องต่อเข้ากับสาย AV In ของ Card Capture แล้วทำการดึงภาพด้วย Delphi โดยใช้ Component ที่มีชื่อว่า ImageEnView โดยมีคำสั่งสำคัญดังนี้

ImageEnView.IO.DshowParams.SetCurrentVideoFormat(w,h, '') ; // ตั้งค่าหน้าจอ

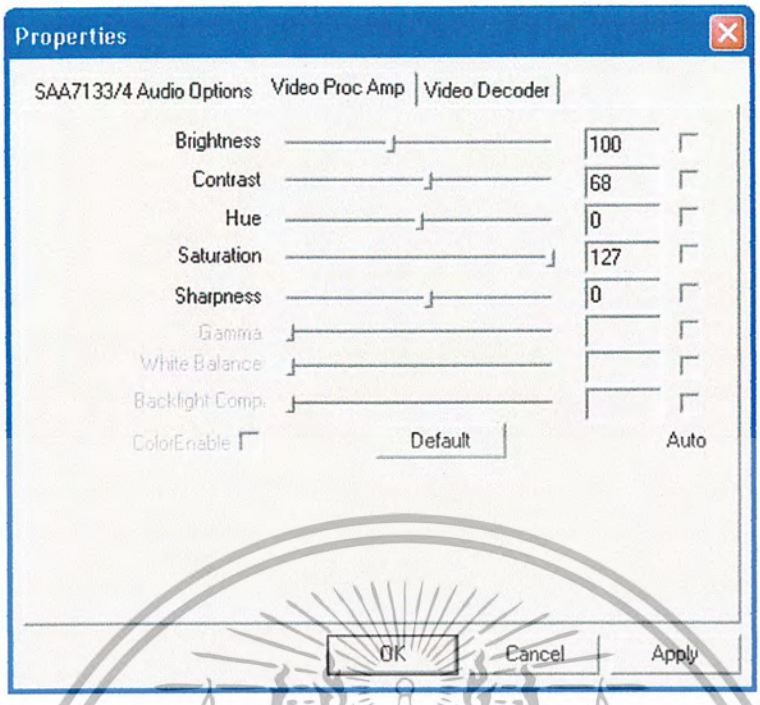
ImageEnView.IO.DshowParams.GetCurrentVideoFormat(w,h,f) ; // แสดงอินโฟของภาพ

ImageEnView.IO.DshowParams.Run. // เริ่มการจับภาพ

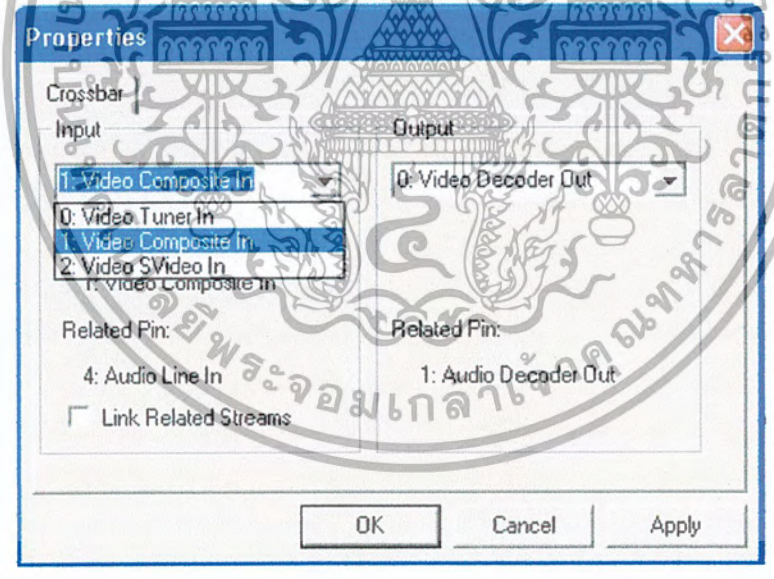
ImageEnView.IO.DshowParams.ShowPropertyPages(iepVideoInput, ietfilter) ; //ใช้ ตั้งค่าไดอะล็อกของวีดีโอ

ImageEnview1.IO.DshowParams.ShowPropertyPages(iepVideoInputSource, ietFilter) ; // ตั้งค่าที่มาของไดอะล็อกวีดีโอ

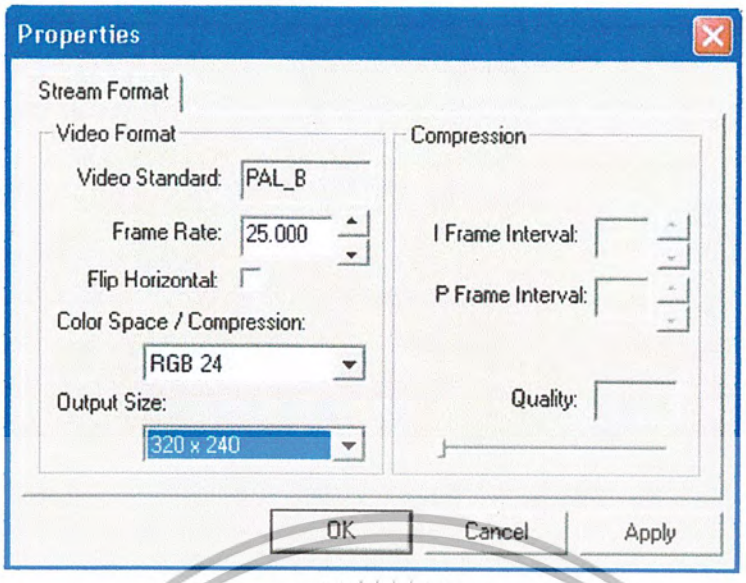
ImageEnview1.IO.DshowParams.ShowPropertyPages(iepVedioInput, ietOutput) ; // ตั้งรูปแบบของไดอะล็อก



รูปที่ 3.2 แสดงวีดีโอ ไดอะล็อก



รูปที่ 3.3 แสดง Input Source Dialog



รูปที่ 3.4 แสดง Format Dialog

ในการใช้งานกับโครงงานนี้ให้ Set Video Source Dialog ตรง Input เป็น Video Composite In แล้วกด OK ในส่วน Video Dialog. ให้ไปที่ Video Proc.Amp ให้ทำการปรับค่า

Brightness 80

Contrast 68

Hue 0

Saturation 78

Sharpness 0

ในส่วนสุดท้ายคือส่วนของ Format Dialog ให้ปรับ Color Space Compression เป็น RGB 24 และปรับ Output Size เป็น 320*240 Pixels

3.6 การเขียนโปรแกรม Delphi เพื่อใช้ในการประมวลผลภาพ Bitmap

รูปที่ 3.5 ภาพแสดงพิกัดของจุดแต่ละ Pixel

ในระบบภาพ Bitmap จะมองภาพเป็น 2 แกน ซึ่งแสดงไว้ดังภาพที่ 3.5 ซึ่งในแต่ละ pixels จะประกอบด้วยสี Red, Green ,Blue ในอัตราส่วนที่แตกต่างกันไปในแต่ละสี ซึ่งคำสั่งสำคัญสำหรับการดึงค่าสีออกจาก Pixels ต่างๆ คือ

X : = Image2.canvas.Pixels[I,j]; // ใช้กับ Image Conponent ใน Addition

X : = ImageEnview1.IEbitmap.Canvas.Pixels[I,j]; // ใช้กับ Image Enview Component ใน ImageEn

3.7 การส่งข้อมูลผ่าน Serial Port โดยใช้เดลไฟ

สำหรับโครงงานการส่งข้อมูลออกจาก Serial Port จะส่งเป็น 2 ตัวอักษรติดกันเช่น 'A1', 'A2', 'B1', 'B2' ซึ่งตัวอักษรตัวแรกนั้นใช้ในการเลือกหุ่นยนต์ที่จะควบคุม

ส่วนตัวอักษรตัวที่ 2 นั้นใช้เป็นคำสั่งควบคุมหุ่นยนต์ว่าจะให้หุ่นยนต์เลี้ยวซ้าย, เลี้ยวขวา, เดินหน้า, ถอยหลัง เป็นต้น สำหรับโครงงานนี้มีรูปแบบข้อมูลดังนี้

'A1' // ให้หุ่นยนต์ A เลี้ยวซ้าย	'B1' // ให้หุ่นยนต์ B เลี้ยวซ้าย
'A2' // ให้หุ่นยนต์ A เลี้ยวขวา	'B2' // ให้หุ่นยนต์ B เลี้ยวขวา
'A3' // ให้หุ่นยนต์ A เดินหน้า	'B3' // ให้หุ่นยนต์ B เดินหน้า
'A4' // ให้หุ่นยนต์ A ถอยหลัง	'B4' // ให้หุ่นยนต์ B ถอยหลัง
'A5' // ให้หุ่นยนต์ A หยุด	'B5' // ให้หุ่นยนต์ B หยุด
'A7' // ให้หุ่นยนต์ A วิ่งช้า	'B7' // ให้หุ่นยนต์ B วิ่งช้า
'A8' // ให้หุ่นยนต์ A เตะ	'B8' // ให้หุ่นยนต์ B เตะ

บทที่ 4
พื้นฐานภาษา C กับ CCS C คอมไพเลอร์

4.1 โครงสร้างภาษา C

โครงสร้างของภาษา C ในรูปแบบมาตรฐาน (ANSI Standard C) จะประกอบไปด้วยรายละเอียดดังนี้

พรีโปรเซสเซอร์ไดเร็กตีฟ (Preprocessor directives)

ชุดคำสั่งที่ใช้ในการจัดการเตรียมข้อมูลสำหรับการประมวลผล โดยก่อนที่จะมีการคอมไพล์ (Compile) หรือแปลตัวโค้ดโปรแกรมให้เป็นภาษาเครื่องนั้น จะมาทำงานในส่วนนี้ก่อน จึงเรียกว่าพรีโปรเซสเซอร์ (ก่อนทำการคอมไพล์) สำหรับ CCS C คอมไพเลอร์จะมีไดเร็กตีฟทั้งที่เป็นมาตรฐานเดียวกับ ANSI C และที่เพิ่มเติมเฉพาะสำหรับ CCS C คอมไพเลอร์

การประกาศ (Declarations)

ก่อนการใช้งานตัวแปรหรือฟังก์ชั่น ต้องมีการประกาศและสร้างตัวแปรหรือฟังก์ชั่นขึ้นมาก่อน

การกำหนดค่า (Definitions)

การประกาศและจองหน่วยความจำหรือกำหนดค่าให้กับตัวแปรหรือฟังก์ชั่น

นิพจน์ (Expressions)

นิพจน์คือ การกระทำระหว่างตัวดำเนินการ (Operators) กับตัวถูกดำเนินการหรือ โอเปอแรนด์ (Operands) เพื่อให้เกิดค่าใดค่าหนึ่ง

สเตตเมนต์ (Statements)

คำสั่งการทำงานคือคำสั่งที่ใช้ในการทำงานตามความต้องการของผู้เขียนโปรแกรม

ฟังก์ชั่น (Functions)

ฟังก์ชั่นคือ ส่วนประกอบของโปรแกรมที่กำหนดให้ทำงานอย่างใดอย่างหนึ่งจนเสร็จสิ้นโดยที่ฟังก์ชั่นจะประกอบไปด้วย การประกาศใช้งานตัวแปร การกำหนดค่าให้กับตัวแปร นิพจน์และคำสั่งการทำงาน

ฟังก์ชั่น main() (main function)

เป็นฟังก์ชั่นที่จะต้องมีการประกาศทุกครั้ง เมื่อมีการเขียนโปรแกรมด้วยภาษา C เพราะ การทำงานของโปรแกรมจะเริ่มต้นที่ฟังก์ชั่นนี้ ปละเป็นฟังก์ชั่นที่ใช้ในการเรียกฟังก์ชั่นอื่นๆ ในการทำงาน

ตัวแปร (Variable)

ตัวแปรคือ ชื่อที่ผู้เขียนโปรแกรมกำหนดขึ้นมาเพื่อใช้ในการจองหน่วยความจำหรือแทนหน่วยความจำ เพื่อการทำงาน ก่อนที่จะประกาศใช้งานตัวแปรได้ต้องกำหนดชนิดข้อมูลของตัวแปร ก่อนการกำหนดการใช้งานตัวแปร ชนิดข้อมูลของตัวแปรเช่น char, int, float และ long เป็นต้นการประกาศใช้งานตัวแปรโดยทั่งไปมีรูปแบบดังนี้

```
Type    variable_name;
```

ตัวอย่างที่ 4-1

```
Char ch;          // ประกาศตัวแปร ch มีชนิดข้อมูลเป็น char
Int I ;           // ประกาศตัวแปร I มีชนิดข้อมูลเป็น int
```

ค่าคงที่ (Constants)

การประกาศใช้งานค่าคงที่ในภาษา C ใช้ไดเร็กตีฟ #define ในการกำหนดค่าคงที่ แต่ถ้าเป็นการกำหนดค่าคงที่ในหน่วยความจำรอม (ROM) ของไมโครคอนโทรลเลอร์ จะใช้คีย์เวิร์ด const

1. การประกาศค่าในหน่วยความจำ

```
#define TRUE   1
#define FALSE  0
#define PI 3.142
```

2. การประกาศค่าคงที่ในหน่วยความจำรอม

```
char const name[16] = {"CCS C Compiler"};
```

คอมเมนต์ (Comments)

คอมเมนต์คือ ข้อความหมายเหตุที่คอมไพเลอร์จะไม่นำมาคอมไพล์ร่วมด้วย เป็นหมายเหตุสำหรับผู้เขียนโปรแกรม ใน CCS C คอมไพเลอร์ จะมีรูปแบบการใช้งานอยู่ 2 แบบคือ คอมเมนต์แบบ บรรทัดเดียว และแบบหลายบรรทัด

1. แบบบรรทัดเดียว จะใช้เครื่องหมาย // เช่น

```
// CCS C Compiler for PIC Microcontroller
```

2. แบบหลายบรรทัด จะใช้เครื่องหมาย /**/ เช่น

```
/*  CCS C Compiler for PIC Microcontroller
OR PIC C Compiler for PIC Microcontroller  */
```

ฟังก์ชั่น (Functions)

ภาษา C จะประกอบไปด้วยฟังก์ชั่นในการทำงาน และจะต้องมีฟังก์ชั่น main() เป็นฟังก์ชั่นหลักที่ใช้เรียกฟังก์ชั่นย่อยๆต่างๆ ทำงาน รูปแบบในการเขียนฟังก์ชั่นจะมีดังนี้

```
Type  function_name (parameter)  {
        Statements;
}
```

โดย type : ประเภทข้อมูลที่ส่งหลับ

Function_name : ชื่อฟังก์ชั่น

Parameter : พารามิเตอร์ที่ส่งให้กับฟังก์ชั่นถ้ามี

คีย์เวิร์ด (Keywords) หรือคำสงวนสำหรับโปรแกรมภาษา C

คีย์เวิร์ดคือ คำสงวนสำหรับไว้ใช้ใน โปรแกรมภาษา C โดยเฉพาะห้ามนำไปใช้สร้างชื่อฟังก์ชั่นหรือตัวแปร เพราะจำทำให้เกิดข้อผิดพลาดในขณะคอมไพล์โปรแกรม คีย์เวิร์ดของ C มีดังนี้

Auto	double	int	struct	break
Long	switch	case	enum	register
typedef	else	void	if	float
Char	extern	return	union	const
Short	unsigned	continue	for	signed
Default	goto	sizeof	volatile	do
Static	which			

4.2 ตัวแปรและชนิดข้อมูลในโปรแกรมภาษา C ของไมโครคอนโทรลเลอร์ PIC

นอกจากรายละเอียดโครงสร้างของภาษา C ที่ต้องทำความเข้าใจ ตัวแปรและชนิดข้อมูลที่ใช้ในการสร้างตัวแปร ก็เป็นส่วนสำคัญที่ต้องทำความเข้าใจ ก่อนที่จะเริ่มต้นเขียนโปรแกรมด้วยภาษา C โดยเฉพาะการเลือกใช้งานชนิดของข้อมูล เนื่องจากการใช้งานชนิดของข้อมูล ก็คือการเลือกใช้งานหน่อยความจำแรมของไมโครคอนโทรลเลอร์เช่นเดียวกัน จึงต้องเลือกใช้งานชนิดของข้อมูลให้เหมาะสม เพื่อนำมาสร้างตัวแปร ไม่ว่าจะเป็นจำนวนทศนิยม (floating point) จำนวนเต็ม (integer) ตัวอักขระ (character) เพราะหน่วยความจำข้อมูลแรมของไมโครคอนโทรลเลอร์มีขีดจำกัด

4.2.1 ชนิดของข้อมูล (Data Type)

ในการสร้างตัวแปรขึ้นมาใช้งาน จะต้องมีการระบุชนิดข้อมูลให้กับตัวแปร โดยชนิดข้อมูลก็จะแบ่งออกเป็น 2 ชนิดใหญ่ๆ คือ ชนิดข้อมูลจำนวนตัวเลข (number) และชนิดข้อมูลตัวอักษร (string) รายละเอียดของชนิดข้อมูล CCS C คอมไพเลอร์ แสดงดังตารางที่ 4-1

ชนิดข้อมูล	ขนาด	ค่าของข้อมูล
int1	ตัวเลข 1 บิต	0 หรือ 1
int 8	ตัวเลข 8 บิต	0 ถึง 255
int 16	ตัวเลข 16 บิต	0 ถึง 65,535
int 32	ตัวเลข 32 บิต	0 ถึง 4,294,967,295
char	ตัวอักขระ 8 บิต	ตัวอักขระรหัสแอสกี้
float	ตัวเลขทศนิยม 32 บิต	$3.4 * 10^{-38}$ ถึง $3.4 * 10^{38}$
Short	ตัวเลข 1 บิต	0 หรือ 1
Int	ตัวเลขจำนวนเต็ม 8 บิต	0 ถึง 255
Long	ตัวเลข 16 บิต	0 ถึง 65,535
Void	ไม่กำหนด	-

ตารางที่ 4-1 รายละเอียดชนิดของข้อมูล CCS C คอมไพเลอร์รองรับ

ชนิดของข้อมูลในตารางที่ 3-1 ค่าเริ่มต้น (default) จะเป็นค่าที่ไม่คิดเครื่องหมาย หรือ unsigned ยกเว้น float แต่สามารถกำหนดค่าด้วย unsigned หรือ signed ได้เช่น unsigned int , signed int เป็นต้น

เพิ่มเติม

ผู้พัฒนาโปรแกรมสามารถกำหนดขนาดของชนิดข้อมูลได้ใหม่ โดยการใช้พรีโปรเซสเซอร์ #TYPE ตัวอย่างเช่น

#TYPE SHORT=8, INT=16, LONG=32

เป็นการเปลี่ยนขนาดของชนิดข้อมูล shot จาก 1 บิต เป็น 8 บิต, int จาก 8 บิต เป็น 16 บิต และ long จาก 16 บิต เป็น 32 บิต เนื่องจากค่าปรกติที่คอมไพเลอร์กำหนดให้ short เท่ากับ 1 บิต int เท่ากับ 8 บิต และ long เท่ากับ 16 บิต

4.2.2 ประเภทของการเก็บข้อมูลตัวแปร (Variable Storage Class)

ตัวแปรและฟังก์ชั่นที่ใช้งานในภาษา C จะมีรายละเอียดอยู่ 2 ส่วน คือ ชนิดของข้อมูล (data type) และการเก็บข้อมูล (storage class)

โดยชนิดของข้อมูลจะเป็นการระบุว่าตัวแปรที่สร้างขึ้นมานั้น มีชนิดข้อมูลเป็นอย่างไร เช่น char, int float เป็นต้น และอีกส่วน โดยปรกติแล้วจะมีค่าเริ่มต้นเป็น auto (นั่นคือสามารถปรับเปลี่ยนได้อัตโนมัติ) จึงไม่เขียนระบุไว้ตอนเริ่มต้นสร้างตัวแปร แต่เมื่อต้องการระบุการจัดเก็บและใช้งานหน่วยความจำว่าต้องการใช้งานบนหน่วยความจำโปรแกรมหรือบนรีจิสเตอร์ จึงต้องมีการระบุลงไปในตอนสร้างตัวแปรใช้งาน ตัวกำหนดประเภทหน่วยความจำแสดงรายละเอียดดังตารางที่ 4-2

คุณสมบัติ	รายละเอียด
auto (automatic)	เป็นการระบุให้ตัวแปรเป็นประเภทหน่วยความจำอัตโนมัติและสามารถใช้ได้ภายในฟังก์ชั่นเท่านั้น
static	ตัวแปรที่ประกาศจะเป็นแบบทั่วไป (โกลบอล : global) มีค่าเริ่มต้นเท่ากับ 0 จะใช้ภายในหรือภายนอกฟังก์ชั่นก็ได้ เป็นตัวแปรแบบสถิตย์
extern (external)	เป็นการบอกให้รู้ว่าตัวแปรที่ถูกประกาศมีการใช้งานแล้วในฟังก์ชั่นอื่น (นิยมใช้ในการประกาศข้ามไฟล์ของซอร์สโปรแกรม)
register	จะเหมือนกับการประกาศแบบ auto แต่จะใช้กับตัวแปรที่มีการใช้งานบ่อยๆ นิยมประกาศใช้กับตัวแปรที่เกี่ยวข้องกับการวนลูป เนื่องจากเรียกใช้งานได้อย่างรวดเร็ว เพราะเป็นการประกาศในหน่วยความจำความเร็วสูง

ตารางที่ 4-2 การระบุรูปแบบของการจัดเก็บข้อมูลของตัวแปรในโปรแกรมภาษา C ที่ CCS C คอมไพเลอร์กำหนด

ตัวอย่างการใช้งาน

```
Register int8  I;        // กำหนดตัวแปร I เป็นชนิด int 8 เก็บอยู่ในหน่วยความจำ
                         ความเร็วสูง
Extern int I ;           // ตัวแปร I เป็นตัวแปรชนิด int และมีการประกาศไว้แล้วใน
                         ไฟล์ใดไฟล์หนึ่ง
```

เพิ่มเติม

ถ้าสร้างตัวแปรขึ้นมาใช้งานโดยไม่ได้ระบุประเภทของหน่วยความจำ ค่าเริ่มต้นจะเป็น auto ตัวอย่างเช่น

```
Char  ch ;               // มีความหมายเท่ากับ auto char ch ;
Int  I,  j,  k ;         // มีความหมายเท่ากับ auto int I,  j,  k  ;
```

4.2.3 ตัวแปร (variable)

ตัวแปรคือชื่อที่ผู้พัฒนาโปรแกรมประกาศขึ้นมาเพื่อใช้เก็บข้อมูลชั่วคราว การประกาศใช้งานตัวแปรก็คือการประกาศใช้งานหน่วยความจำ การประกาศใช้งานตัวแปรต้องเลือกชนิดของข้อมูลให้เหมาะสมกับข้อมูลที่จะใช้และไม่ควรที่จะประกาศใช้งานตัวแปรมากเกินความจำเป็น เนื่องจากเมื่อมีการประกาศใช้งานตัวแปรหน่วยความจำแรมจะถูกจองไว้ใช้งานสำหรับตัวแปรนั้นทันที

ในการประกาศใช้งานตัวแปรในภาษา C จะต้องกำหนดชนิดของตัวแปรนั้นว่าเป็นชนิดใด เช่น เป็นตัวแปรชนิด Integer, Character หรือ Floating เป็นต้น นอกจากนี้ยังใช้คีย์เวิร์ด typedef (ตั้งชื่อใหม่ใช้อ้างอิงแทนชื่อเก่า) เพื่อใช้ในการอ้างอิงถึงชนิดของข้อมูลที่ได้กำหนดขึ้นมา เช่น ถ้าต้องการนำค่าตัวเลข 2 ตัวมาบวกกันแล้วให้ผลลัพธ์ไปเก็บไว้ จะต้องกระทำตามขั้นตอนดังนี้

1. สร้างตัวแปรขึ้นมารองรับ 3 ตัวสำหรับเก็บค่า 3 ค่า
2. นำค่าที่ต้องการบวกกันมาเก็บไว้ในตัวแปรทั้งสองตัวที่สร้างขึ้น
3. นำค่าตัวแปรทั้งสองตัวที่ได้เก็บค่าไว้แล้วมากระทำกันด้วยเครื่องหมายบวก แล้วเก็บไว้ในอีกตัวแปร ก็จะได้ค่า 2 ค่าบวกกันและสามารถนำไปใช้งานได้

เช่น

```
Int  I, j,  k ;          // สร้างตัวแปรขึ้นมา 3 ตัวเป็นชนิดข้อมูล
integer
I = 10;                  // กำหนดค่า I เท่ากับ 10
```

```
J = 15 ;                    // กำหนดค่า j เท่ากับ 15
K = I + j ;                 // นำค่า I บวกกับ j แล้วนำไปเก็บไว้ใน k
```

ก่อนที่จะสร้างตัวแปรขึ้นมาใช้งานจะต้องทราบถึงกฎของการตั้งชื่อตัวแปรกันก่อน

4.2.4 กฎในการตั้งชื่อตัวแปร

ในการสร้างตัวแปรขึ้นมาใช้งานจะต้องมีกฎที่เกี่ยวข้องกับการตั้งชื่อ เพื่อป้องกันไม่ให้ไปซ้ำกับค่าที่ถูกกำหนดไว้ในตัวคอมไพเลอร์แล้ว หรือสร้างชื่อตัวแปรที่คอมไพเลอร์ไม่สามารถตีความได้ กฎในการตั้งชื่อตัวแปรมีดังนี้

1. ไม่มีตัวอักษรพิเศษประกอบอยู่ที่ตัวแปร เช่น !,@,#,$,%,^,&,* เป็นต้น
2. ชื่อตัวแปรต้องไม่ขึ้นต้นด้วยตัวเลข
3. ไม่มีช่องว่างระหว่างชื่อของตัวแปร
4. ใช้อักษรตัวเล็กใหญ่ได้ แต่จะให้ความหมายเป็นคนละตัวแปร เช่นตัวแปร Abc กับ abc เป็นคนละตัวแปรกัน เนื่องจากภาษา C จะคำนึงถึง case sensitive
5. ห้ามตั้งชื่อตัวแปร ซ้ำกับคำสงวน (keyword) หรือคำสั่งที่มีในภาษา C และ Build-in functions เช่น ไม่ตั้งชื่อตัวแปรดังนี้ char, float, int, output_b, printf เป็นต้น

4.2.5 การประกาศใช้งานตัวแปร (Variable Declaration)

ในการประกาศใช้งานตัวแปรหรือการประกาศใช้งานข้อมูลจะมีรูปแบบในการประกาศใช้งานดังนี้

1. การประกาศใช้ตัวแปรที่ระบุประเภทหน่วยความจำ

รูปแบบ

```
[type-qualifier]  [type-specifier]  [declarator] ;
```

<u>ตัวอย่างที่ 4-2</u>

```
Register int8 I 8 ;         // ประกาศให้ตัวแปร i8 เป็นตัวแปรชนิด int8 ใช้
                               งานในหน่วยความจำ register
```

2. การประกาศใช้งานตัวแปรที่ไม่ระบุประเภทหน่วยความจำ

รูปแบบ

[type-specifier] [declaration] ;

ตัวอย่างที่ 4-3

```
Int1    i1 ;            // ประกาศให้ตัวแปร i1 เป็นตัวแปรชนิด int1
Int a, b, c, d ;        // ประกาศให้ตัวแปร a, b, c, d เป็นตัวแปรชนิด int
```

3. การประกาศใช้ตัวแปรประเภทอีนัม (enumeration)

รูปแบบ

Enum [id] { [id [= cexpr] }

ตัวอย่างที่ 4-4

```
Enum tasks      { taskA, taskB, taskC } ;
```

เป็นการประกาศใช้ตัวแปรแบบลำดับ คือ taskA คือ 0, taskB คือ 1, taskC คือ 2

4. การประกาศใช้งานตัวแปรประเภทโครงสร้างและยูเนียน (structure and union)

รูปแบบ

Struct or union [id] {[type-qualifier [[*] id cexpr [cexpr]]] }

ตัวอย่างที่ 4-5

```
Struct  { byte time ;   // ตัวแปรขนาด 8 บิต
Byte command : 4;       // ตัวแปรขนาด 4 บิต
Byte source : 4 ;       // ตัวแปรขนาก 4 บิต
} msg ;
```

5. การประกาศใช้งานตัวแปรประเภทตัวระบุชื่อประเภทข้อมูล

รูปแบบ

Typedef [type-qualifier] [type-specifier] [declarator] ;

ตัวอย่างที่ 4-6

```
Typedef register int8   regi8 ;         // ใช้ regi8 แทน register int8 ได้
```

การใช้งาน

```
Regi8   ri8;            // ตัวแปร ri8 มีชนิดข้อมูลเป็น register int8
```

6. การประกาศใช้งานตัวแปรชุดหรือตัวแปรอาร์เรย์

รูปแบบ

```
[type-specifier]  [declarator]  [n]          // ขนาด 1 มิติ
[type-specifier]  [declarator]  [n] [m]      // ขนาด 2 มิติ
```

ตัวอย่างที่ 4-7

```
Int a[10] ;                  // ตัวแปร a เป็นตัวแปรอาร์เรย์มีขนาด 10
สมาชิก
In tab [10] [10] ;           // ตัวแปร ab เป็นตัวแปรอาร์เรย์มีขนาด
10*10 สมาชิก
```

7. การประกาศใช้งานตัวแปรพอยน์เตอร์

รูปแบบ

```
[type-specifier]  * [declarator] ;
```

ตัวอย่างที่ 4-8

```
Char *chptr ;                // ตัวแปร chptr เป็นตัวแปรพอยน์เตอร์
                             ชนิด character
Int *iptr ;                  // ตัวแปร iptr เป็นตัวแปรพอยน์เตอร์ชนิด
                             integer
```

ตัวอย่างที่ 4-9 การประกาศใช้งานตัวแปร

```
Int a,b,c,d ;                // ประกาศตัวแปร a,b,c,d เป็นตัวแปร
                             ประเภทข้อมูล integer
Typedef int byte ;           // ประกาศให้ตัวแปร byte เป็นตัวแปรแทน
                             ชนิดข้อมูล int
Typedef short bit ;          // ประกาศให้ตัวแปร bit เป็นตัวแปรแทน
                             ชนิดข้อมูล short
Bit e,f ;                    // ประกาศตัวแปร e,f เป็นตัวแปรประเภท
                             ข้อมูล bit (short)
Byte g[3] [2] ;              // ประกาศตัวแปรอาร์เรย์ g เป็นตัวแปร
                             ประเภทข้อมูล byte (int)
Char *h ;                    // ประกาศตัวแปร h เป็นตัวแปรประเภท
                             ตัวชี้ไปยังชนิดข้อมูล char
```

```
Enum Boolean {false, true } ;  // ประกาศตัวแปร Boolean เป็นตัวแปร
                               ประเภท enum
Boolean j ;                    // ประกาศตัวแปร j เป็นตัวแปรชนิดข้อมูล
                               enum
Byte k = 5 ;                   // ประกาศตัวแปร k มีค่าเท่ากับ 5
Byte const WEEKS = 52 ;        // ประกาศตัวแปร WEEKS เป็นค่าคงที่ มี
                               ค่าเท่ากับ 52
Byte cinst FACTORS [4] = {8, 16, 64, 128 } ;   // ประกาศตัวแปรอาร์เรย์
                                               FACTORS เป็นค่าคงที่มีค่า
                                               8,16,64,128
Struct data_record     {       // ประกาศตัวแปร data_record
เป็นตัวแปร                                   ประเภทโครงสร้าง
Byte a [2] ;
Byte b : 2 ;                   // มีขนาด 2 บิต
Byte c : 3 ;                   // มีขนาด 3 บิต
Int d ;
}
```

4.2.6 การกำหนดค่าให้กับตัวแปร (Variable Assignment)

เมื่อสร้างตัวแปรขึ้นมาใช้งานแล้ว สามารถกำหนดค่าให้กับตัวแปรได้ด้วยขั้นตอนง่ายๆคือ

```
Variable_name = value          // ตัวแปรที่สร้างขึ้น = ค่าที่กำหนดให้กับ
                               ตัวแปร
```

ตัวอย่างที่ 4-10

```
Int I = 10 ;                   // ประกาศใช้งานตัวแปร I และกำหนดค่า
                               ให้เท่ากับ 10 ทันที
Int abc ;                      // ประกาศใช้งานตัวแปร abc
Int a,b,c ;                    // ประกาศใช้งานตัวแปร a,b,c เป็นชนิด int
abc = 12 ;                     // กำหนดค่าให้กับตัวแปร abc เท่ากับ 12
```

ในการกำหนดค่าให้กับตัวแปรจะต้องขึ้นต้นด้วยชื่อของตัวแปรก่อน จากนั้นตามด้วยเครื่องหมายเท่ากับและค่าของตัวแปรที่ต้องการกำหนดให้กับตัวแปรนั้น ปิดท้ายด้วยเครื่องหมายเซมิโคลอน จุดที่สำคัญคือ ค่าที่กำหนดให้กับตัวแปรต้องสัมพันธ์

กับชนิดข้อมูลของตัวแปรนั้นด้วย ถ้าสร้างตัวแปรมีชนิดข้อมูลเป็น integer ค่าที่กำหนดให้กับตัวแปรต้องเป็นจำนวน integer ด้วย

ตัวอย่างที่ 4-11

```
Int I ;              // ตัวแปร I มีชนิดข้อมูลเป็น integer
Float f ;            // ตัวแปร f มีชนิดข้อมูลเป็น float
Char c ;             // ตัวแปร c มีชนิดข้อมูลเป็น character
i = 20 ;             // กำหนดค่าให้กับตัวแปร i มีค่าเท่ากับ 20
f = 2.5 ;            // กำหนดค่าให้กับตัวแปร f มีค่าเท่ากับ 2.5
c = 'A' ;            // กำหนดค่าให้กับตัวแปร c มีค่าเท่ากับตัวอักษร A
```

4.2.7 การเปลี่ยนชนิดข้อมูล (Type conversations)

ในภาษา C สามารถที่จะเปลี่ยนแปลงชนิดของข้อมูลได้ด้วยหลักการง่ายๆ คือ การกำหนดชนิดข้อมูลใหม่หน้าข้อมูลเดิมที่ต้องการเปลี่ยนแปลง ซึ่งมีรูปแบบดังนี้

รูปแบบ

```
(type-specifier)  value
```

ตัวอย่างที่ 4-12

```
(int)  12.5 ;        // เปลี่ยนข้อมูลจากจำนวนทศนิยมเป็นจำนวนเต็ม
                     การใช้งาน
Float  f =12.5
Printf ("%d", (int)  f) ;        // ข้อมูลที่แสดงจะมีค่าเท่ากับ 12
```

ในการเปลี่ยนแปลงข้อมูลจะต้องให้ความสำคัญในเรื่องของขนาดในการเก็บข้อมูลของแต่ละชนิดข้อมูลด้วยเพราะถ้าเปลี่ยนแปลงข้อมูลจากขนาดใหญ่ไปขนาดเล็กข้อมูลจะถูกตัดทิ้งโดยอัตโนมัติ (เว้นแต่มีความต้องการในการเปลี่ยนแปลงแบบนั้น) นอกจากการเปลี่ยนแปลงชนิดข้อมูลด้วยวิธีดังกล่าวแล้ว CCS C คอมไพเลอร์ ยังมี Build-in functions ที่ใช้ในการเปลี่ยนแปลงข้อมูลเพิ่มเติมด้วยไม่ว่าจะเป็นฟังก์ชั่น atoi(), atoi32(), atoll() และ atof() ที่ใช้ในการจัดการกับข้อมูลเช่นเดียวกัน

รหัสควบคุมการพิมพ์	ความหมาย
%c	พิมพ์ตัวอักขระ
%s	พิมพ์ข้อความตัวอักระ
%u	พิมพ์เลขจำนวนเต็มไม่คิดเครื่องหมาย
%x	พิมพ์เลขฐานสิบหกแบบ int ในอักษรพิมพ์เล็ก

รหัสควบคุมการพิมพ์	ความหมาย
%X	พิมพ์เลขฐานสิบหกแบบ int ในอักษรพิมพ์ใหญ่
%d	พิมพ์เลขจำนวนเต็ม
%e	พิมพ์เลขจำนวนจริงแบบทางวิทยาศาสตร์ รูปแบบ exp หรือเอ็กโปรแนนเชียล
%f	พิมพ์เลขจำนวนจริง
%lx	พิมพ์เลขฐานสิบหกแบบ long int ในอักษรพิมพ์เล็ก
%lX	พิมพ์เลขฐานสิบหกแบบ long int ในอักษรพิมพ์ใหญ่
%lu	พิมพ์เลขฐานสิบหกแบบ unsigned long
%ld	พิมพ์เลขฐานสิบหกแบบ signed long
\n	ขึ้นบรรทัดใหม่ (line feed)
\r	การรีเทิร์น (return fee)
\t	เลื่อนแท็บ
\b	เลื่อนกลับ (backspace
\f	ขึ้นหน้าใหม่ (form feed)
\a	ส่งเสียงบี๊บ (bell)
\v	เลื่อนแท็บแนวตั้ง (vertical space
\?	แสดงเครื่องหมาย ? (question mark
\'	แสดงเครื่องหมาย * (single quote)
\"	แสดงเครื่องหมาย * (double quote)
\\	แสดงเครื่องหมาย \ (a single backslash)

ตารางที่ 4-3 รหัสควบคุมการพิมพ์ที่ใช้ในฟังก์ชั่น printf ของ CCS C คอมไพเลอร์

4.3 ฟังก์ชั่นและคำสั่งควบคุมในโปรแกรมภาษา C ของไมโครคอนโทรลเลอร์ PIC

ฟังก์ชั่น (function) เป็นส่วนหนึ่งของโปรแกรมภาษา C ที่ใช้ในการรวบรวมชุดคำสั่งต่างๆ มีหน้าที่ในการทำงานเฉพาะและเสร็จสิ้นภายในฟังก์ชั่น ฟังก์ชั่นใน CCS C คอมไพเลอร์ จะมีรูปแบบการเขียนและการใช้งานฟังก์ชั่นเช่นเดียวกับฟังก์ชั่นในภาษา C มาตรฐาน ซึ่งมีฟังก์ชั่นใช้งานอยู่ 2 รูปแบบหลักๆ คือ

1. ฟังก์ชั่นที่ผ่านค่าข้อมูลไปให้กับฟังก์ชั่นอื่น ฟังก์ชั่นในลักษณะนี้เรียกว่า pass by value คือ ผ่านค่าเฉพาะข้อมูลไปให้กับฟังก์ชั่นเท่านั้น เมื่อฟังก์ชั่นทำงานเสร็จแล้วจะไม่มีผลใดๆ กับตัวแปรที่ผ่านค่าไปให้กับฟังก์ชั่น ซึ่งการใช้งานฟังก์ชั่นในลักษณะนี้จะมีทั้งที่มีการผ่านค่าไปให้และไม่ผ่านค่า
2. ฟังก์ชั่นที่ผ่านค่าอ้างอิงหรือแอดเดรส ฟังก์ชั่นในลักษณะนี้เรียกว่า pass by reference คือเป็นการผ่านค่าอ้างอิงหรือแอดเดรสไปให้กับฟังก์ชั่น เมื่อฟังก์ชั่นทำงานเสร็จสิ้นแล้วจะมีผลกับตัวแปรที่ผ่านค่าให้กับฟังก์ชั่น ซึ่งการเรียกใช้งานฟังก์ชั่นนี้จะมีการเพิ่มเติมเครื่องหมาย “&” เข้ามาเกี่ยวข้องในขณะเรียกใช้งานฟังก์ชั่น และการประกาศตัวแปรของฟังก์ชั่นที่ใช้ในการผ่านค่าจะเป็นตัวแปรชนิดพอยน์เตอร์ การเรียกใช้งานจะเหมือนกับการเรียกใช้งานฟังก์ชั่นทั่วไป

รูปแบบการประกาศใช้งานฟังก์ชั่นทั่วไป

Qualifier function_name ([args]) { stmt }

โดย Qualifier : ชนิดหรือค่าที่ส่งกลับ (return) มาจากฟังก์ชั่น

Function_name : ชื่อของฟังก์ชั่น

Args : อาร์กิวเมนต์ของฟังก์ชั่น (ค่าที่ส่งให้กับฟังก์ชั่น)

Stmt : ชุดคำสั่งที่ใช้งานหรือ Statements

4.3.1 แบบของฟังก์ชั่น (Function prototypes)

เมื่อมีการเขียนฟังก์ชั่นขึ้นมาใช้งาน แล้วฟังก์ชั่นที่ถูกเรียกใช้งานได้ถูกเขียนเป็นลำดับหลังฟังก์ชั่นที่เรียกใช้งาน คอมไพเลอร์จะแจ้งข้อความเตือนหรือไม่ก็จะแจ้งข้อความผิดพลาดว่าฟังก์ชั่นที่เรียกใช้งานนั้นยังไม่ได้ถูกประกาศหรือยังไม่ได้ถูกสร้างขึ้นมาใช้งาน ดังนั้นจึงต้องประกาศแบบของฟังก์ชั่นไว้ล่วงหน้า ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 4-13

```
Void main (void)                {
        Int I ;
        I = add_num (4,5) ;
        Printf (“Result %d”, i) ;
}
int add_num (int I , int j) {
        return (i+j) ;
}
```

จากตัวอย่างข้างต้นคอมไพเลอร์จะแจ้งข้อผิดพลาดดังนี้ Undefined identifier add_num เนื่องจากเมื่อโปรแกรมถูกคอมไพล์ ฟังก์ชั่น main () จะถูกคอมไพล์และค้นหาฟังก์ชั่น add_num () แต่ไม่พบเนื่องจากโค้ดดังกล่าวได้ถูกประกาศหลังฟังก์ชั่น main () การแก้ไขทำได้โดยประกาศแบบฟังก์ชั่นหรือฟังก์ชั่นโปรโตไทป์ก่อน ดังนี้

ตัวอย่างที่ 4-14

```
Int add_num (int I, int j) ;                    // This is function prototype
Void main (void)                    {
        Int I ;
        I = add_num (4,5) ;
        Printf ("Result %d", i) ;
}
int add_num (int I , int j) {
        return (i+j) ;
}
```

หรือไม่จะต้องประกาศฟังก์ชั่น add_num () ก่อนฟังก์ชั่น main ดังนี้

```
int add_num (int I , int j) {
        return (i+j) ;
}
        Void main (void)                    {
                Int I ;
                I = add_num (4,5) ;
                        Printf ("Result %d", i) ;
}
```

จากเงื่อนไขดังกล่าวทำให้ทราบว่า เมื่อมีการประกาศใช้งานฟังก์ชั่นใดก็ตาม ฟังก์ชั่นที่ถูกเรียกใช้งานต้องมาก่อนฟังก์ชั่นที่เรียกใช้งาน หรือไม่ก็ต้องประกาศฟังก์ชั่นโปรโตไทป์ก่อนเรียกใช้งานทุกครั้ง

4.3.2 อาร์กิวเมนต์ของฟังก์ชั่น (Function Arguments)

อาร์กิวเมนต์ของฟังก์ชั่นคือค่าที่จะผ่านให้กับฟังก์ชั่นเมื่อมีการเรียกใช้งานฟังก์ชั่น ตามมาตรฐาน ANSI C จะมีอาร์กิวเมนต์ได้สูงสุด 31 อาร์กิวเมนต์ ตัวอย่างฟังก์ชั่นที่มีอาร์กิวเมนต์ เช่น

ตัวอย่างที่ 4-15

```
int add_num (int I , int j) {                    // ฟังก์ชั่นนี้มีพารามิเตอร์ 2 ตัว หรือมีอากิวเมนต์ 2 ตัว

        return (i+j) ;
}
void show_msg (voidj)   {                        // ฟังก์ชั่นนี้ไม่มีพารามิเตอร์หรือไม่มีอาร์กิวเมนต์
        printf ("CCS C Compiler") ;
}
```

4.3.3 การผ่านค่าสตริงให้กับฟังก์ชั่น (Passing Constant Strings)

รายละเอียดเกี่ยวกับการผ่านค่าให้กับฟังก์ชั่นที่ผ่านมาเป็นการผ่านค่าจำนวนเต็มทั้งผ่านค่าโดยตรงหรือผ่านทางตัวแปร แต่ถ้าเป็นการผ่านค่าข้อมูลที่เป็นสตริงสำหรับ CCS C คอมไพเลอร์สามารถที่จะผ่านค่าได้โดยตรง เนื่องจาก CCS C คอมไพเลอร์ส่งค่าสตริงผ่านฟังก์ชั่นในแบบไม่เป็นไปตามมาตรฐาน ANSI C เช่น

ตัวอย่างที่ 4-16

```
Void print_msg (char msg)    {
                ....
}
void main (void)             {
        print_msg ("PIC C") ;
}
```

จะมีความหมายเช่นเดียวกับ

```
print_msg ("P") ;
print_msg ("I") ;
print_msg ("C") ;
print_msg ("  ") ;
print_msg ("C") ;
```

นอกจากนี้ยังสามารถผ่านค่าสตริงไปยังฟังก์ชั่นได้ด้วยการใช้งานตัวแปรพอยเตอร์หรือตัวชี้ในการจัดการข้อมูลสตริงได้ ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 4-17

```
Void msgString (char  *s)                {
        While (*s!=0)                    {
```

```
            Putc (*s++) ;
            Delay_ms (100) ;
        }
}
void main (void)        {
        msgString ("CCS C Compiler") ;
}
```

4.3.4 คำสั่งควบคุมโปรแกรม (Program Control Statements)

จุดมุ่งหมายในการเขียนโปรแกรมคือ ต้องการให้โปรแกรมทำงานตามเงื่อนไขที่ต้องการไม่ว่าจะเป็นการทำงานในส่วนที่กำหนด การวนลูปชุดคำสั่งที่ต้องการ การทำงานในลักษณะนี้จะต้องมีชุดคำสั่งที่ใช้ในการควบคุมการทำงาน นั่นคือ ชุดคำสั่งควบคุมที่มีมาพร้อมกับตัวภาษาและทุกๆ ภาษาที่ใช้ในการเขียนโปรแกรมก็จะมีชุดคำสั่งเหล่านี้ โดยแตกต่างกันไปในรายละเอียดเล็กน้อยขึ้นอยู่กับภาษานั้น สำหรับ ANSI C จะประกอบไปด้วยชุดคำสั่งควบคุมดังนี้

1. ชุดคำสั่งกำหนดเงื่อนไข (Conditional statements)
2. ชุดคำสั่งการวนลูป (loop statements)

4.3.4.1 ชุดคำสั่งกำหนดเงื่อนไข

คำสั่งกำหนดเงื่อนไขในลักษณะนี้จะเป็นไปในรูปแบบมีทางเลือกอย่างใดอย่างหนึ่งในการทำงาน โดยขึ้นอยู่กับเงื่อนไขที่ได้กำหนดไว้ รายละเอียดของแต่ละคำสั่งมีดังนี้

คำสั่ง if…else…

คำสั่งการทำงานกำหนดเงื่อนไข จะแบ่งออกเป็น 3 รูปแบบคือ

1. if condition then statement จะตรวจสอบเงื่อนไข ถ้าเป็นจริง จะเข้าไปทำงานในส่วนของ statement ถ้าเงื่อนไขเป็นเท็จก็จะข้ามการทำงานในส่วนของ statement ไปทำงานในลำดับถัดไปของโปรแกรม

รูปแบบการใช้งาน

```
If (expr) stmt ;
```

<u>ตัวอย่างที่ 4-18</u>

```
If (x>25)
x = 1;
```

ถ้ามีมากกว่าหนึ่งสเต็ตเมนต์ จะกำหนดขอบเขตด้วย {…}

```
If (x>25)          {
        x = 1;
        y = 2 ;
}
```

2. if condition then statement1 else statement2 จะตรวจสอบเงื่อนไขในแบบถ้าเงื่อนไขเป็นจริงจะเข้าไปทำงานในส่วนของ statement1 ถ้าเงื่อนไขเป็นเท็จก็จะข้ามเข้าไปทำงานในส่วนของ statement2

รูปแบบการใช้งาน

```
        If (expr) stmt ; else stmt ;
```

ตัวอย่างที่ 4-19

```
If (x>25)          {
        x = 1;
        y = 1;
        }       else    {
        x = x+1 ;
        y = y+1 ;
}
```

3. if condition1 then statement1 elseif condition2 else statement2 ในรูปแบบนี้จะตรวจสอบเงื่อนไขที่ 1 ก่อน ถ้าเป็นจริงก็จะเข้าไปทำงานในส่วนของ statement1 ถ้าเป็นเท็จก็จะข้ามเข้าไปตรวจสอบเงื่อนไขที่ 2 ถ้าเงื่อนไขที่ 2 เป็นจริงก็จะทำงานในส่วนของ statement2 แต่ถ้าเป็นเท็จก็จะข้ามการทำงานในส่วนของ statement2 ไปทำงานในลำดับถัดไปของโปรแกรม

รูปแบบการใช้งาน

```
        If (expr) stmt ; elseif (expr) stmt ;
```

ตัวอย่างที่ 4-20

```
If (x<25)
        x = 1;
        Elseif (x>30)
        x= x+1 ;
```

ถ้ามากกว่าหนึ่งสเตตเมนต์ เขียนโปรแกรมได้ดังนี้

```
If (x<25)                    {
        x = 1 ;
        y = 1 ;
        }          else if (x>30)     {
        x = x+1 ;
        y = y+1 ;
}
```

คำสั่ง switch

คำสั่งการทำงานที่ประกอบไปด้วยหลายๆ เงื่อนไข ถ้าใช้ชุดคำสั่งกำหนดเงื่อนไขด้วย if...els... อาจจะไม่สะดวกเท่ากับการใช้คำสั่ง switch แทน

รูปแบบการใช้งานคำสั่ง

```
Switch  (expr)   {
        Case cexpr : stmt ;        // one or more case
        [default : stmt]
        }
```

ตัวอย่างที่ 4-21

```
Switch  (cmd)    {
        case 0 : printf ("command 0") ;
        break ;
        case 1 : printf ("command 1") ;
        break ;
        default : printf ("bad command") ;
        break ;
}
```

4.3.4.2 ชุดคำสั่งการวนลูป

คำสั่งการวนลูปเป็นคำสั่งเงื่อนไขที่มีการทำงานซ้ำๆอยู่ในลูปของเงื่อนไขจนกว่าเงื่อนไขจะเป็นเท็จ จึงออกจากการทำงานของลูป รายละเอียดของแต่ละคำสั่งมีดังนี้

คำสั่งลูป while ()

การทำงานของคำสั่ง จะเริ่มต้นตรวจสอบเงื่อนไขคำสั่ง while ว่าเป็นจริงหรือไม่ ถ้าเป็นจริงจะทำคำสั่งที่อยู่ในลูป while () และจะทำซ้ำไปเรื่อยๆจนกว่าเงื่อนไขจะเป็นเท็จ จากนั้นจึงจะหลุดจากคำสั่งลูป while

รูปแบบการทำงานใช้งานคำสั่ง

```
While (expr) stmt ;
หรือ
While (expr)    {
        Stmt1 :
        Stmt2 :
        …
    }
```

ตัวอย่างที่ 4-22

```
a = 2;
while (a < 10) a++;
b = 3;
while (b > 2  &&  b < 10)          {
    b++ ;
    printf ("abc\n\r") ;
}
```

คำสั่งลูป do … while ()

คำสั่งการวนลูปด้วย do … while () จะแตกต่างกับคำสั่ง while () ตรงที่คำสั่งลูปนี้จะมีการทำงานในลูปอย่างน้อย 1 ครั้งก่อนที่จะตรวจสอบเงื่อนไขในการวนลูป

รูปแบบการใช้งานคำสั่ง

```
Do { stmts ; }  while (expr) ;
```

ตัวอย่างที่ 4-23

```
Do      {
        a = a + 1 ;
}  while (a < 10) ;
```

คำสั่ง for

เป็นคำสั่งการวนลูปที่มีจำนวนแน่นอนและได้กำหนดไว้แล้วในส่วนของการวนลูปการกำหนดเงื่อนไขจะกำหนดค่าใน expr1 ซึ่งเป็นค่าเริ่มต้น expr2 เป็นส่วนเปรียบเทียบเงื่อนไข และ expr3 เป็นส่วนเพิ่มหรือลดค่าข้อมูล

รูปแบบการใช้งานคำสั่ง

```
For (expr1 ; expr2; expr3)        stmt;
หรือ
For (expr1; expr2; expr3){
        Stmt1 ;
        Stmt2 ;
}
```

ตัวอย่างที่ 4-24

```
For (i=0; i<10; i++)
        a = a+1 ;
for (j=0; i<10; i++)      {
        a++ ;
        b++ ;
}
```

4.3.4.3 คำสั่งที่เกี่ยวข้องกับลูป

เป็นคำสั่งที่ใช้งานร่วมกับคำสั่งวนลูป เพื่อช่วยกำหนดเงื่อนไขการทำงานในลูปได้มากขึ้น คำสั่งที่เกี่ยวกับลูปมีดังนี้

คำสั่ง goto

เป็นคำสั่งที่ใช้ในการกระโดดไปในตำแหน่งที่ต้องการ แต่มีเงื่อนไขอยู่ที่การใช้งานคำสั่ง goto จะต้องควบคู่กับคำสั่ง label ด้วยทุกครั้ง ตัวอย่างการใช้งาน เช่น

ตัวอย่างที่ 4-25

```
For (i=0; i<10; i++)      {          จากตัวอย่างจะเห็นว่าคำสั่งให้วนลูป 10 ครั้งแต่
        Printf ("if i>5 goto label") ;          เนื่องจากมีการตรวจสอบเงื่อนไขถ้า i มากกว่า 5
        If ( i>5)                               คำสั่ง goto จะถูกเรียกใช้งานโดยการกระโดดไปทำที่ตำแหน่ง
```

```
        goto label ;                    label ทันที
    }
    label ;
```

คำสั่ง break

เป็นคำสั่งที่ใช้เพื่อการหยุดทำงานในลูปและออกจากลูป แตกต่างกับคำสั่ง goto ตรงที่เมื่อคำสั่ง break ถูกเรียกใช้งานจะออกจากลูปและทำงานในตำแหน่งต่อจากคำสั่งลูป แต่คำสั่ง goto จะไปทำงานในตำแหน่ง label แทน ซึ่งขึ้นอยู่กับผู้เขียนว่าจะกำหนดตำแหน่ง label ไว้ที่ไหน เช่น

ตัวอย่างที่ 4-26

```
For ( I = 0; i< 10; i++)    {
        Printf ("if i > 5 break") ;
        If (i > 5)  break ;
}
a = 10;
```

คำสั่ง continue

เป็นคำสั่งการทำงานที่ตรงข้ามกับคำสั่ง break การทำงานของคำสั่ง continue จะบังคับให้โปรแกรมกระโดดข้ามการทำงานไปในรอบต่อไป เช่น

ตัวอย่างที่ 4-27

```
For (i=0; i<10 ;  i++)    {          ในตัวอย่างนี้จะไม่มีการพิมพ์ i = 5 ออกมาเนื่องจาก
        Printf ("if  i > 5  continue\n\r") ;     เมื่อ i เท่ากับ 5 คำสั่ง continue จะถูกเรียกใช้งาน
        If ( i = = 5 ) continue ;                ทำให้โปรแกรมกระโดดข้ามไปทำงานในรอบต่อไปทันที
        Printf ("I = %d\n\r", i) ;
```

4.4 นิพจน์และตัวดำเนินการในภาษา C ของไมโครคอนโทรลเลอร์ PIC

การเขียนโปรแกรมไม่ว่าจะเป็นภาษา C,BASIC หรือ PASCAL จะต้องเกี่ยวข้องกับนิพจน์และตัวดำเนินการ ซึ่งจะอยู่ในรูปของชุดคำสั่ง การเขียนโปรแกรมจึงหนีไม่พ้นที่จะต้องทำความเข้าใจกับนิพจน์และตัวดำเนินการ โดยเฉพาะกับการเขียนโปรแกรมด้วยภาษา C แล้ว การใช้งาน

นิพจน์และตัวดำเนินการใน CCS C คอมไพเลอร์ จะมีรูปแบบและลักษณะการใช้งานเช่นเดียวกับ ANSI C จึงสามารถศึกษารายละเอียดเพิ่มเติมได้จากหนังสือภาษา C ที่เป็น ANSI Standard C

4.4.1 นิพจน์ (Expressions)

นิพจน์คือการนำเอาตัวแปร ค่าคงที่ หรือตัวเลข ที่มากระทำด้วยกันด้วยเครื่องหมายดำเนินการ ด้วยเงื่อนไขหนึ่ง ตัวอย่างนิพจน์ เช่น

```
a + b ;          // นำค่าในตัวแปร a และ b มาบวกกัน
a = = b ;        // นำค่าในตัวแปร a และ b มาเปรียบเทียบกันด้วยเครื่องหมายเท่ากับ (เขียน
                 // เครื่องหมาย "=" สองครั้งติดกัน)
a+b/(a*b) ;      // นำค่าในตัวแปร b หารด้วย a คูณกับ b (การกระทำการคูณก่อนให้ได้ค่า)
                 // และนำค่าที่ได้มาบวกกับ a
```

โดยการเขียนนิพจน์จะต้องมีเครื่องหมายตัวดำเนินการเข้ามาเกี่ยวข้องด้วยและมีระดับความสำคัญของการกระทำในตัวดำเนินการตามหลักระดับความสำคัญของตัวดำเนินการ ซึ่งจะกล่าวรายละเอียดต่อไป

4.4.2 ตัวดำเนินการ (Operators)

เครื่องหมายดำเนินการคือ ตัวกระทำหรือเครื่องหมายกระทำที่มีผลต่อ ข้อมูลหรือตัวแปร ในภาษา C จะแบ่งเครื่องหมายดำเนินการออกเป็น 5 ประเภทใหญ่ คือ

1. เครื่องหมายดำเนินการทางคณิตศาสตร์ (Arithmetic & Unary Operators)
2. เครื่องหมายดำเนินการสัมพันธ์และลอจิก (Relational and Logical Operators)
3. เครื่องหมายดำเนินการกำหนดค่าหรือให้ค่า (Assignment Operators)
4. เครื่องหมายดำเนินการเงื่อนไข (Conditional Operators)
5. เครื่องหมายดำเนินการทางบิต (Bitwise Operators)

4.4.3 เครื่องหมายดำเนินการทางคณิตศาสตร์

ในภาษา C จะมีเครื่องหมายที่ใช้ดำเนินการเกี่ยวกับคณิตศาสตร์ดังนี้

เครื่องหมายดำเนินการ	การกระทำ	ตัวอย่าง
+	การบวกค่า	a + b = 13*
-	การลบค่า	a – b = 7*

*	การคูณค่า	a * b = 30*
/	การหารค่า	a / b = 3*
%	การหารค่าเอาเฉพาะเศษ	a % b = 1*
-	จำนวนลบ	-a
- -	ลดค่า	-a , a- -
+ +	เพิ่มค่า	++a, a++

(*ค่าในตัวอย่างเกิดจากการกำหนดให้ a=10 และ b=3)

โดยเครื่องหมาย +, -, *, / และ % จะเกี่ยวข้องกับการดำเนินการทางด้านคณิตศาสตร์ เครื่องหมาย – เมื่ออยู่หน้าตัวเลขคือการกำหนดค่าเป็นจำนวนลบ ส่วนเครื่องหมาย - - และ + + เป็นเครื่องหมายเกี่ยวกับการลดและการเพิ่มค่าให้กับตัวแปร เช่น

```
a = 10 ;        // กำหนดค่า 10 ให้กับตัวแปร a
a++ ;           // หลังจากทำงานตามคำสั่งนี้แล้ว a จะมีค่าเท่ากับ 11
a - - ;         // หลังจากทำงานตามคำสั่งนี้แล้ว a จะมีค่าเท่ากับ 10
```

4.4.4 เครื่องหมายดำเนินการสัมพันธ์และลอจิก

เครื่องหมายดำเนินการในลักษณะนี้จะเกี่ยวข้องกับเรื่องของการเปรียบเทียบค่าเพื่อแสดงผลลัพธ์ของการเปรียบเทียบว่าเป็นจริง (true) หรือเป็นเท็จ (false) โดยที่เครื่องหมาย && , || และ ! จะเกี่ยวข้องกับตัวดำเนินการทางลอจิก ดังมีรายละเอียดดังนี้

เครื่องหมาย	การกระทำ	ตัวอย่าง
<	น้อยกว่า	a < b

ผลลัพธ์เป็นเท็จ (false หรือ 0)

>	มากกว่า	a > b

ผลลัพธ์เป็นเท็จ (true หรือ 1)

<=	น้อยกว่าหรือเท่ากับ	a <= b

ผลลัพธ์เป็นเท็จ (false หรือ 0)

>=	มากกว่าหรือเท่ากับ	a >= b

ผลลัพธ์เป็นเท็จ (true หรือ 1)

= =	เท่ากับ	a = = b

ผลลัพธ์เป็นเท็จ (false หรือ 0)

!=	ไม่เท่ากับ	a != b

ผลลัพธ์เป็นจริง (true หรือ 1)

&&	และ (and)	(a>b) && (b<a) ผลลัพธ์เป็นจริง (true หรือ 1)
||	หรือ (or)	(a>b) || (b>a) ผลลัพธ์เป็นจริง (true หรือ 1)
!	ไม่ใช่ (not)	!(12>a) ผลลัพธ์เป็นเท็จ (false หรือ 0) (เมื่อกำหนดให้ a = 10 และ b = 3)

4.4.5 เครื่องหมายดำเนินการกำหนดค่าหรือให้ค่า

เครื่องหมายดำเนินการที่ใช้ในการกำหนดค่าหรือให้ค่าของข้อมูลหรือตัวแปรหรือการหระทำด้วยนิพจน์จากทางขวาของเครื่องหมาย = (Assignment Operators) ให้กับตัวแปรที่อยู่ทางซ้าย ตัวอย่างเช่น

```
a = 20 ;          // กำหนดค่า 20 ให้กับตัวแปร a
a = a + b ;       // นำค่าใน a บวกกับค่าใน b แล้วกำหนดให้กับ a
```

4.4.6 เครื่องหมายดำเนินการเงื่อนไข

เครื่องหมายดำเนินการเงื่อนไข ? : นี้เป็นเครื่องหมายดำเนินการเลือกอย่างใดอย่างหนึ่งโดยดูจากเงื่อนไขของการดำเนินการลอจิกเป็นหลักตามตัวอย่างต่อไปนี้

<u>ตัวอย่างที่ 4-28</u>

กำหนดให้ a = 10 และ b = 5

```
x = (a > b) ? 0 : 150 ;          // ค่าของ x จะเท่ากับ 150
x = (a < b) ? 0 : 150 ;          // ค่าของ y จะเท่ากับ 0
```

4.4.7 ความสัมพันธ์ของเครื่องหมายดำเนินการ

เครื่องหมายดำเนินการทั้งหมดมีการกำหนดระดับความสำคัญ ทั้งนี้เพื่อประโยชน์ในการลำดับความสำคัญในการทำงานในกรณีที่มีเครื่องหมายดำเนินการหลายตัวในเงื่อนไข สามารถสรุปได้ดังนี้

เครื่องหมายดำเนินการ	ระดับความสำคัญในการกระทำ
-,++,- -,!,sizeof, (type)	ขวาไปซ้าย
*, / , %	ซ้ายไปขวา
+ , -	ซ้ายไปขวา
<,<=,>,>=	ซ้ายไปขวา

= =, ! =	ซ้ายไปขวา
&&	ซ้ายไปขวา
||	ซ้ายไปขวา
? :	ขวาไปซ้าย
=, +=, -=, *=, /=, %=	ขวาไปซ้าย

ตัวอย่างที่ 4-28

c + = (a > 10 && a <= 50) ? ++a : a/b

จากตัวอย่างข้างต้นจะมีการตรวจสอบเงื่อนไขดังนี้

1. ตรวจสอบเงื่อนไขในวงเล็บเป็นอันดับแรก	(a > 10 && a <= 50)
2. ตรวจสอบเงื่อนไขในลำดับที่สอง	a <= 50
3. ตรวจสอบเงื่อนไขในลำดับที่สาม	a > 10
4. ตรวจสอบเงื่อนไขในลำดับที่สี่	&&
5. ตรวจสอบเงื่อนไขในลำดับที่ห้า	? :
ถ้าเงื่อนไขในวงเล็บเป็นจริงจะทำตามเงื่อนไข	++a
ถ้าเงื่อนไขในวงเล็บเป็นเท็จจะทำตามเงื่อนไข	a / b
6. ตรวจสอบเงื่อนไขในลำดับที่หก	+ =

4.4.8 เครื่องหมายดำเนินการทางบิต

เครื่องหมายดำเนินการทางบิตข้อมูลจัดเป็นเครื่องหมายที่มีความสำคัญมากโดยเฉพาะกับการเขียนโปรแกรมด้วยภาษา C กับงานไมโครคอนโทรลเลอร์ ซึ่งหนีไม่พ้นเรื่องของบิตและไบต์ ข้อมูลรายละเอียดของตัวดำเนินการบิตข้อมูลมีดังนี้

เครื่องหมาย	การกระทำ	ตัวอย่าง
&	การแอนด์ข้อมูล	a & b ผลลัพธ์ คือ
|	การออร์ข้อมูล	a | b ผลลัพธ์ คือ 11
^	การเอ็กคลูซีฟ-ออร์ข้อมูล	a ^ b ผลลัพธ์
<<	เลื่อนบิตไปทางซ้าย	a << 1 ผลลัพธ์
>>	เลื่อนบิตไปทางขวา	a >> 1 ผลลัพธ์ คือ 5
~	การกลับค่าบิต	~ a ผลลัพธ์ คือ -11

(เมื่อกำหนดให้ a = 10 และ b = 3)

4.5 การใช้งานพอร์ตอินพุตเอาต์พุตดิจิตอลของไมโครคอนโทรลเลอร์ PIC ด้วยโปรแกรมภาษา C

พอร์ตของไมโครคอนโทรลเลอร์ PIC สามารถทำงานเป็นได้ทั้งพอร์ตอินพุตเอาต์พุตดิจิตอลและเป็นพอร์ตอินพุตวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล โดยที่การเปลี่ยนแปลงสัญญาณอนาลอกนี้จะขึ้นอยู่กับคุณสมบัติของขาพอร์ตนั้น ซึ่งกำหนดไว้แล้วในแต่ละเบอร์ของไมโครคอนโทรลเลอร์ PIC

ไมโครคอนโทรลเลอร์ PIC16F877 มีขาพอร์ตให้ใช้งานมากถึง 34 บิต โดยแบ่งพอร์ต A จำนวน 6 ช่อง พอร์ต B, C และ D อย่างละ 8 ช่อง และพอร์ต E จำนวน 3 ช่องสำหรับ PIC16F628 นั้นมีจำนวนพอร์ตให้ใช้งานเพียง 2 พอร์ต คือ พอร์ต A และ B อย่างละ 8 ช่อง โดยแต่ละขาพอร์ตของแต่ละเบอร์จะมีความสามารถที่แตกต่างกันไป ในตัวอย่างนี้จะใช้ PIC16F877 เป็นหลัก

4.5.1 คำสั่งควบคุมพอร์ตไมโครคอนโทรลเลอร์ PIC

CCSC คอมไพเลอร์ได้เตรียมชุดคำสั่งที่ใช้ในการควบคุมการทำงานของขาพอร์ตไม่ว่าจะเป็นอินพุตหรือเอาต์พุตด้วยฟังก์ชั่นดังต่อไปนี้ คือ

1. คำสั่งไดเร็กตีฟกำหนดคุณสมบัติของพอร์ต ได้แก่

```
#use fast_io
#use fixed_io
#use standard_io
```

2. ฟังก์ชั่นกำหนดพอร์ตอินพุตเอาต์พุต ได้แก่

```
set_tris_x ()
```

3. ฟังก์ชั่นควบคุมพอร์ตเอาต์พุต ได้แก่

```
output_x ()
output_low ()
output_high ()
output_float ()
output_bit ()
port_b_pullups ()
```

4. ฟังก์ชั่นควบคุมพอร์ตอินพุต ได้แก่

```
input_a ()
input_b ()
input_c ()
input_d ()
```

input_e ()

input ()

4.5.2 คำสั่งไดเร็กตีฟ

การควบคุมขาพอร์ตสามารถกำหนดคุณสมบัติในการใช้งานได้ด้วยไดเร็กตีฟดังต่อไปนี้

1. #use fast_io (port)

การใช้งานไดเร็กตีฟนี้จะมีผลโดยตรงกับรีจิสเตอร์ จึงต้องกำหนดการใช้งานพอร์ตด้วยฟังก์ชั่น set_tris_x () ก่อนที่จะเรียกใช้งานคำสั่งควบคุมการทำงานของพอร์ต (x แทนพอร์ต a,b,c,d,e) ว่าจะทำหน้าที่เป็นพอร์ตอินพุตหรือเอาต์พุตในระหว่างที่ใช้งานพอร์ต

ตัวอย่างการใช้งาน

#use fast_io (A)

2. #use fixed_io (port_output=pin,pin?)

การใช้งานไดเร็กตีฟนี้ เป็นการบอกให้คอมไพเลอร์รับรู้ว่า จะใช้พอร์ตดังกล่าวเป็นพอร์ตอินพุตหรือเอาต์พุตตลอดการใช้งานจนกว่าจะมีการเปลี่ยนแปลงเป็นอย่างอื่น

ตัวอย่างการใช้งาน

#use fixed_io (a_outputs=PIN_A2, PIN_A3)

3. #use standard_io (port)

การใช้งานไดเร็กตีฟนี้ เป็นการกำหนดให้พอร์ตเป็นอินพุตหรือเอาต์พุตตลอดการใช้งานขาพอร์ต ซึ่งเป็นค่าปรกติ (default I/O) ของการใช้งาน

ตัวอย่างการใช้งาน

#use standard_io (A)

4.5.3 ฟังก์ชั่นกำหนดพอร์ตอินพุตเอาต์พุต

ก่อนการใช้งานขาพอร์ตของไมโครคอนโทรลเลอร์ PIC จะต้องกำหนดสถานะของขาพอร์ตว่า ต้องการให้ขาใดเป็นอินพุตหรือเอาต์พุตด้วยฟังก์ชั่น set_tris_x () เมื่อมีการใช้งานไดเร็กตีฟ #use fast_io ซึ่งมีรายละเอียดดังนี้

รูปแบบการใช้งาน

Set_tris_a (value)

Set_tris_b (value)

Set_tris_c (value)

Set_tris_d (value)

Set_tris_e (value)

โดยที่ value คือข้อมูล 8 บิตแบบ int ใช้กำหนดบิตของพอร์ตว่า เป็นอินพุตหรือเอาต์พุต

ตัวอย่างการใช้งาน

Set_tris_b (0xF0) ; // RB4-RB7 เป็นขาอินพุตและ RB0-RB3 เป็นขาเอาต์พุต

4.5.4 ฟังก์ชั่นควบคุมพอร์ตเอาต์พุต

1. การใช้งาน output_x ()

output_x () เป็นชื่อเรียกรวม เพื่อแทนความหมายของฟังก์ชั่นดังต่อไปนี้

output_a (), output_b (), output_c (), output_d (), output_e ()

เพราะฉะนั้นเมื่อมีการใช้งานจะต้องระบุว่า ต้องการเอาต์พุตไปที่ขาของพอร์ตใด ไม่ว่าจะเป็น A,B,C,D หรือ E ด้วยฟังก์ชั่น output_a () หรือ output_b () เป็นต้น

รูปแบบการใช้งาน

Output_a (value) // พอร์ต A

Output_b (value) // พอร์ต B

Output_c (value) // พอร์ต C

Output_d (value) // พอร์ต D

Output_e (value) // พอร์ต E

โดยที่ value คือ จำนวนเต็ม 8 บิต

2. การใช้งาน output_low () และ output_high ()

เป็นฟังก์ชั่นที่ใช้ในการควบคุมเอาต์พุตในระดับลอจิกสูงหรือ high และลอจิกต่ำหรือ low ของขาพอร์ตที่ต้องการใช้งาน

รูปแบบการใช้งาน

Output_low (pin)

Output_high (pin)

โดยที่ pin คือ ค่าที่ถูกกำหนดไว้แล้วในเฮดเดอร์ไฟล์ของไมโครคอนโทรลเลอร์ PIC ที่ใช้งาน เช่น PIC16F877 ใช้ไหล์ 16f877.h ซึ่งกำหนดให้แต่ละขาพอร์ตของ A เป็นดังนี้

#define PIN_A0 40

#define PIN_A1 41

#define PIN_A2 42

```
#define PIN_A3          43
#define PIN_A4          44
#define PIN_A5          45
```

ตัวอย่างการใช้งาน

```
Output_low (PIN_A1) ;
Output_high (PIN_A0) ;
```

3. การใช้งาน output_float ()

ฟังก์ชั่น outout_float () จะทำให้ขาพอร์ตเป็นแบบคอลเล็กเตอร์เปิด (open collector) หรือแบบเดรนเปิด (open drain) นั่นคือ เมื่อมีการใช้งานขาพอร์ตที่เป็นคอลเล็กเตอร์เปิด จึงต้องต่อตัวต้านทานพูลอัปไว้ ดังนั้นในการใช้งานในลักษณะนี้ต้องให้ความระมัดระวังเป็นพิเศษ

รูปแบบการใช้งาน

```
Output_float (pin)
```

ตัวอย่างการใช้งาน

```
If ( (data & 0x80) =0 )
        Output_low (pin_A0) ;
Else
        Output_float (pin_A0) ;
```

4. การใช้งาน output_bit ()

เป็นฟังก์ชั่นที่ใช้ในการควบคุมเอาต์พุตในระดับบิตของขาพอร์ต

รูปแบบการใช้งาน

```
Output_bit (pin, value)
```

โดยที่ value มีค่าเป็น 0 หรือ 1

ตัวอย่างการใช้งาน

```
Output_bit( PIN_B0,0) ;                    //เทียบได้กับ  output_low (pin_B0) ;
Output_bit ( PIN_B0,     input ( PIN_B1) ) ; // PIN_B0 เท่ากับ PIN_B1
```

5. การใช้งาน Port_b_pullups()

เป็นการกำหนดให้พอร์ต B เป็นอินพุตที่มีการพูลอัป โดยกำหนดให้เป็น TRUE เมื่อต้องการใช้งาน หรือ FALSE เมื่อยกเลิกการใช้งาน คำสั่งนี้จะใช้ได้กับไมโครคอนโทรลเลอร์

PIC อนุกรม 14 และ 16 บิต เท่านั้น ในขณะที่ในอนุกรม 12 บิต จะใช้ SETUP_COUNTERS แทน

รูปแบบการใช้งาน

Port_b_pullups (value) โดยที่ value จะมีค่าเป็น TRUE หรือ FALSE

ตัวอย่าง

```
Port_b_pullups (FALSE) ;
```

4.5.4 ฟังก์ชั่นควบคุมพอร์ตอินพุต

1. การใช้งาน input_x ()

เป็นกลุ่มฟังก์ชั่นที่เกี่ยวข้องกับการอ่านค่าของขาพอร์ต ได้แก่ input_a (), input_b (), input_c ()

รูปแบบการใช้งาน

```
Value = input_a ()
Value = input_b ()
Value = input_c ()
Value = input_d ()
Value = input_e ()
```

โดยที่ ค่าส่งกลับหรือ value เป็นค่าข้อมูลขนาด 8 บิต

ตัวอย่าง

```
Data = input_b () ;
```

2. การใช้งาน input ()

ฟังก์ชั่นในการอ่านค่าอินพุตขากขาพอร์ตที่ต้องการ

รูปแบบการใช้งาน

Value = input (pin)

โดยที่ pin คือ ขาพอร์ตที่ต้องการอ่านค่า

ค่าที่ส่งกลับ value คือ “0” (FALSE) เมื่อขาพอร์ตมีสถานะลอจิก “0”

“1” (TRUE) เมื่อขาพอร์ตมีสถานะลอจิก “1”

<u>ตัวอย่าง</u>

```
While ( !input (PIN_B1) ) ;  // waits for B1 to go high
If ( input (PIN_A0) )
Printf (“A0 is now high\r\n”) ;
```

4.5.5 การใช้งานขาพอร์ต A ของไมโครคอนโทรลเลอร์ PIC ด้วยโปรแกรมภาษา C

พอร์ต A ของไมโครคอนโทรเลอร์ PIC16F877 เป็นพอร์ต 2 ทิศทาง คือเป็นได้ทั้งอินพุตและเอาต์พุต มีจำนวน 6 ช่อง คือ RA0-RA5 โดยที่ขา RA0/AN0 – RA3/AN3 เป็นได้ทั้งขาพอร์ตอินพุตเอาต์พุตดิจิตอลและอินพุตวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล RA4/T0CKI เป็นขาพอร์ตและอินพุตสัญญาณนาฬิกาของไทเมอร์ 0 ส่วน RA5/AN4/SS ขาพอร์ต อินพุตวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลและขาสัญญาณ Slave Select สำหรับการติดต่ออนุกรมแบบซิงโครนัส ในการเริ่มต้นทำงานครั้งแรกของไมโครคอนโทรลเลอร์จะกำหนดให้พอร์ต A ทั้งหมดเป็นอินพุตอนาลอก (ยกเว้น RA4 ไม่ได้ใช้งานเป็นขาพอร์ตอนาลอกอินพุต)

ชื่อขา	ตำแหน่งขา	ชนิดของขา	ชนิดของวงจรบัฟเฟอร์	รายละเอียดการทำงาน
RA0/AN0	2	อินพุต/เอาต์พุต	ทีทีแอล/อนาลอก	-ขาพอร์ต RA0 -อินพุตวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล ช่อง 0
RA1/AN1	3	อินพุต/เอาต์พุต	ทีทีแอล/อนาลอก	-ขาพอร์ต RA1 -อินพุตวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล ช่อง 1
RA2/AN2/V_{REF}-/CV_{REF*}	4	อินพุต/เอาต์พุต	ทีทีแอล/อนาลอก	-ขาพอร์ต RA2 -อินพุตวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล ช่อง 2 -อินพุตแรงดันอ้างอิงลบของวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล
RA3/AN3/V_{REF+}	5	อินพุต/เอาต์พุต	ทีทีแอล/อนาลอก	-ขาพอร์ต RA3 -อินพุตวงจรแปลงสัญญาณอนลอกเป็นดิจิตอล ช่อง 3

				-อินพุตแรงดันอ้างอิงบวกของวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล
RA4/T0CHI/C1OUT*	6	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์	-ขาพอร์ต RA4 กรณีใช้พอร์ตเอาต์พุตมีโครงสร้างแบบเดรนเปิด -อินพุตสัญญาณนาฬิกาของไทเมอร์ 0 -เอาต์พุตวงจรเปรียบเทียบแรงดันอนาลอกช่อง 1 (PIC16F87xA
RA5/AN4/SS/C2OUT*	7	อินพุต/เอาต์พุต	ทีทีแอล/อนาลอก	-ขาพอร์ต RA5 -อินพุตวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล ช่อง 4 -ขาสัญญาณ Slave Select ใช้ในการสื่อสารข้อมูลอนุกรมแบบซิงโครนัส -เอาต์พุตวงจรเปรียบเทียบแรงดันอนาลอกช่อง 2 (PIC16F87xA)

*มีเฉพาะในเบอร์ PICF87xA

ตารางที่ 4-5 รายละเอียดพอร์ต A ของไมโครคอนโทรลเลอร์ PIC16F877 (A)

4.5.6 การควบคุมใช้งานขาพอร์ต A

ในการเขียนโปรแกรมควบคุมการใช้งานขาพอร์ตจะต้องกำหนดสถานะของการใช้งานขาพอร์ตก่อนว่าต้องการใช้งานเป็นขาพอร์ตอินพุตหรือเป็นขาพอร์ตเอาต์พุต โดยกำหนดว่าค่าในรีจิสเตอร์ TRISA มีขนาด 8 บิตควบคุมแต่ละขาของพอร์ต A ซึ่งมีรายละเอียดดังนี้

รีจิสเตอร์	บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
TRISA	TRISA	TRISA	TRISA	TRISA	TRISA	TRISA	TRISA	TRISA
PORTA	-	-	RA5	RA4	RA3	RA2	RA1	RA0

ตารางที่ 4-6 แสดงรีจิสเตอร์ 8 บิต ที่ควบคุมขาของพอร์ต A

ถ้าบิตใดในรีจิสเตอร์ TRISA เป็น “0” แสดงว่า กำหนดให้ขาพอร์ตนั้นเป็นเอาต์พุต (“0” เท่ากับเอาต์พุต) และถ้ากำหนดให้เป็น “1” แสดงว่าเป็นขาพอร์ตอินพุต (“1” เท่ากับ อินพุต) CCS C คอมไพเลอร์ จะใช้ฟังก์ชั่น SET_TRIS_A () หรือ set_tris_a () ในการกำหนดค่าในรีจิสเตอร์ TRISA เพื่อกำหนด หน้าที่ของขาพอร์ตให้เป็นอินพุตหรือเอาต์พุตดังนี้

```
SET_TRIS_A ( 0xF0 ) ;        // A5,A4 เป็นขาอินพุต A3,A2,A1,A0 เป็นขาเอาต์พุต
```

4.6 การใช้งานอินเตอร์รัปต์ของไมโครคอนโทรลเลอร์ PIC ด้วยโปรแกรมภาษา C

อินเตอร์รัปต์ (interrupt) คือการขัดจังหวะการทำงานของซีพียูหรือ โปรแกรมที่กำลังทำงานอยู่เพื่อมาทำงานในส่วนโปรแกรมบริการอินเตอร์รัปต์ที่ได้กำหนดไว้ล่วงหน้าแล้ว การใช้อินเตอร์รัปต์นั้นจะช่วยประหยัดเวลาในการทำงานของโปรแกรมที่ไม่ต้องไปคอยตรวจสอบเงื่อนไขใดเงื่อนไขหนึ่งตลอดเวลา โดยส่งหน้าที่การตรวจสอบนี้ให้กับการบริการอินเตอร์รัปต์แทน ในไมโครคอนโทรลเลอร์ PIC จะมีบริการตอบสนองการใช้งานอินเตอร์รัปต์ที่ค่อนข้างมากและหลากหลายประเภท เช่น

1.การอินเตอร์รัปต์จากไทร์เมอร์โอเวอร์โฟลว์

2.การอินเตอร์รัปต์จากการเปลี่ยนแปลงสัญญาณอะนาลอกเป็นดิจิตอล

3.การอินเตอร์รัปต์จากโมดูล CCP1 และ CCP2

4.การอินเตอร์รัปต์เนื่องจากการเปลี่ยนแปลงสัญญาณของพอร์ต B และ C

5.การอินเตอร์รัปต์เนื่องจากการรับส่งข้อมูลอนุกรม RS232 (UART)

การอินเตอร์รัปต์ที่กล่าวมาข้างต้นเป็นเพียงบางส่วนเท่านั้น เนื่องจากการบริการอินเตอร์รัปต์ขึ้นอยู่กับเบอร์ของไมโครคอนโทรลเลอร์ที่ใช้งาน เช่น ถ้าเป็น PIC16F877 มีแหล่งกำเนิดสัญญาณการอินเตอร์รัปต์ถึง 15 แหล่ง ในขณะที่ PIC16F628 มี 10 แหล่ง เป็นต้น

ในการเขียนโปรแกรมควบคุมการอินเตอร์รัปต์ของไมโครคอนโทรลเลอร์ PIC ด้วยภาษา C สำหรับ VVS C คอมไพเลอร์นั้นกำหนดการเขียนในรูปแบบไดเร็กตีฟ เพื่อกำหนดให้ฟังก์ชั่นในบรรทัดถัดมาคือ ฟังก์ชั่นที่เกี่ยวข้องกับการอินเตอร์รัปต์ ไดเร็กตีฟที่ใช้กำหนดการอินเตอร์รัปต์ มีรายละเอียดดังนี้

ไดเร็กตีฟการอินเตอร์รัปต์	ความหมายของการเกิดอินเตอร์รัปต์
#INT_AD	การเปลี่ยนแปลงสัญญาณจากอนาลอกเป็นดิจิตอลเสร็จสมบูรณ์
#INT_ADOF	เกิดไทม์เอาต์ในการเปลี่ยนแปลงสัญญาณจากอนาลอกเป็นดิจิตอล
#INT_BUSCOL	เกิด Bus collision หรือการชนกันของข้อมูลในระบบบัส
#INT_BUTTON	มีการกดปุ่มเกิดขึ้น
#INT_CCP1	จากโมดูลตรวจจับหรือเปรียบเทียบ CPP1
#INT_CCP2	จากโมดูลตรวจจับหรือเปรียบเทียบ CPP2
#INT_COMP	จากการเปรียบเทียบแรงดันอนาลอกเป็นจริง
#INT_EEPROM	เขียนข้อมูลลงในหน่อยความจำอีอีพรอมเสร็จสมบูรณ์
#INT_EXT	จากสัญญาณภายนอก RB0/INT
#INT_EXT1	จากสัญญาณภายนอก #1
#INT_EXT2	จากสัญญาณภายนอก #2
#INT_I2C	อินเตอร์รัปต์ของบัส I^2C (ใช้ได้เฉพาะเบอร์ PIC14000)
#INT_LCD	เกิดจากการใช้งานโมดูล LCD
#INT_LOWVOLT	เมื่อตรวจพบแรงดันไฟเลี้ยงต่ำกว่าที่กำหนด
#INT_PSP	มีข้อมูลเข้ามาที่พอร์ตขนานเสริมหรือ PSP
#INT_RB	เมื่อพอร์ต RB4-RB7 มีการเปลี่ยนแปลงลอจิก
#INT_RC	เมื่อพอร์ต RC4-RC7 มีการเปลี่ยนแปลงลอจิก
#INT_RDA	เมื่อมีการรับข้อมูลจากพอร์ตอนุกรม RS-232
INT_RTCC	จากการโอเวอร์โฟลว์ของไทเมอร์ (RTCC)
#INT_SSP	เมื่อมีการเริ่มต้นใช้งานระบบบัส SPI หรือ I^2C
#INT_TBE	เมื่อบัฟเฟอร์ส่งข้อมูลของโมดูลสื่อสารพอร์ตอนุกรมว่าง
#INT_TIMER0	จากการโอเวอร์โฟลว์ของไทเมอร์ 0 (RTCC)
#INT_TIMER1	จากการโอเวอร์โฟลว์ของไทเมอร์ 1

#INT_TIMER2	จากการโอเวอร์โฟลว์ของไทเมอร์ 2
#INT_TIMER3	จากการโอเวอร์โฟลว์ของไทเมอร์ 3

กรอบแยกที่ 4.1 รายละเอียดแหล่งกำเนิดอินเตอร์รัปต์สำหรับ PIC16F877

```
//////////////////////////////////////////////////////////////////////////////// INT
// Interrupt Functions: ENABLE_INTERRUPTS (), DISABLE_INTERRUPTS (),
// EXT_INT_EDGE ()
//
// Constants used in EXT_INT_EDGE () are:
#define L_TO_H                  0x40
#define H_TO_L                  0       // Constants used in
                                        ENABLE/DISABLE_INTERRUPTS () are :
#define GLOBAL                  0x0BC0
#define INT_RTCC                0x0B20
#define INT_RB                  0x0B08
#define INT_EXT                 0x0B10
#define INT_AD                  0x8C40
#define INT_TBE                 0x8C10
#define INT_RDA                 0x8C20
#define INT_TIMER1              0x8C01
#define INT_TIMER2              0x8C02
#define INT_CCP1                0x8C04
#define INT_CCP2                0x8D01
#define INT_SSP                 0x8C08
#define INT_PSP                 0x8C80
#define INT_BUSCOL              0x8D08
#define INTLOWVOLT              0x8D80
#define INT_EEPROM              0x8D10
```

สำหรับรายละเอียดเกี่ยวกับความสามารถในการตรวจสนองการอินเตอร์รัปต์ของไมโครคอนโทรเลอร์ PIC แต่ละเบอร์ดูได้จากเอกสารของไมโครคอนโทรเลอร์เบอร์นั้นๆ หรือดู

รายละเอียดจากไฟล์เฮดเดอร์ของเบอร์นั้น เช่น เบอร์ PIC16F877 ในโฟลเดอร์ Devices จะมีการบริการอินเตอร์รัปต์ดังแสดงในกรอบแยกที่ 4.1

4.6.1 การใช้งานไดเร็กตีฟที่เกี่ยวข้องกับอินเตอร์รัปต์

1. #int_default อินเตอร์รัปต์ดีฟอลต์ (default)

กำหนดให้ฟังก์ชั่นที่ต่อจากไดเร็กตีฟ #int_default เป็นฟังก์ชั่นอินเตอร์รัปต์เริ่มต้น ถ้าอินเตอร์รัปต์ถูกการกระตุ้นและไม่มีการกำหนดฟังก์ชั่นอินเตอร์รัปต์

รูปแบบการใช้งาน

```
#int_default
```

ตัวอย่าง

```
#int_default
default_isr ()    {
                printf ("Unexplained interrupt\r\n") ;
}
```

2. #int_global กำหนดให้ฟังก์ชั่นที่ต่อจากไดเร็กตีฟ #int_global เป็นฟังก์ชั่นอินเตอร์รัปต์แบบโกบอล

โดยปรกติแล้วไม่นิยมใช้ แต่ถ้าใช้ต้องระวังเนื่องจากคอมไพเลอร์จะไม่สร้างโค้ดเริ่มต้นและเคลียร์ค่าให้และไม่มีการเก็บค่าใดๆในรีจิสเตอร์ (ใช้สร้างอินเตอร์รัปต์ด้วยตนเอง)

รูปแบบการใช้งาน

```
#int_global
```

ตัวอย่าง

```
#int_global
isr ()    {                    // Will be located at location 4
#asm
bsf       isr_flag
retfie
#endasm
}
```

3. #priority กำหนดลำดับความสำคัญของอินเตอร์รัปต์

รูปแบบการใช้งาน

#priority ints

โดยที่ ints คือ รายละเอียดของอินเตอร์รัปต์

ตัวอย่าง

```
#priority rtcc,    rb
```

4.6.2 การใช้งาน Built-in functions ที่เกี่ยวข้องกับอินเตอร์รัปต์

นอกจากไดเร็กตีฟที่เกี่ยวข้องกับการใช้งานอินเตอร์รัปต์แล้ว ในการใช้งานอินเตอร์รัปต์ยังประกอบไปด้วย Built-in functions ที่เกี่ยวข้องกับการอินเตอร์รัปต์มีดังนี้

ฟังก์ชั่น enable_interrupts() กำหนดให้มีการอินเตอร์รัปต์เกิดขึ้น

รูปแบบการใช้งาน

Enable_interrupts (level)

โดยที่ level คือลำดับค่าคงที่ของการอินเตอร์รัปต์ที่กำหนดในดีไวน์ของโฟลเฮดเดอร์ (devices .h file) ของไมโครคอนโทรเลอร์ PIC แต่ละเบอร์

ตัวอย่าง

```
Enable_interrupts (GLOBAL) ;        // เปิดการใช้งานอินเตอร์รัปต์ทั้งหมด
Enable_interrupts (INT_TIMER0) ;    // เปิดการใช้งานอินเตอร์รัปต์ ไทเมอร์ 0
Enable_interrupts (INT_TIMER1) ;    // เปิดการใช้งานอินเตอร์รัปต์ ไทเมอร์ 1
```

4.7 การใช้งานพอร์ตอนุกรม USART ในไมโครคอนโทรลเลอร์ PIC ด้วยโปรแกรมภาษา C

พอร์ตอนุกรมจัดเป็นพอร์ตสื่อสารข้อมูลที่นิยมใช้งานทั้งบนเครื่องคอมพิวเตอร์และไมโครคอนโทรลเลอร์ โดยเฉพาะนำมาใช้งานในการเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์กับอุปกรณ์ไมโครคอนโทรลเลอร์ สำหรับไมโครคอนโทรลเลอร์ PIC ในหลายๆเบอร์ได้เตรียมโมดูลการเชื่อมต่อพอร์ตอนุกรมไว้ให้แล้วที่เรียกว่า USART (Universal Synchronous Asynchronous Receiver Transmitter) ที่มีคุณสมบัติในการทำงานเป็นตัวรับหรือตัวส่งข้อมูลในแบบอะซิงโครนัสหรือซิงโครนัสก็ได้ นอกจากนั้นยังสามารถกำหนดความเร็วในการรับส่งข้อมูลได้ด้วย

ใน CCSC คอมไพเลอร์ ได้เตรียมส่วนติดต่อและใช้งานพอร์ตอนุกรมไว้ให้แล้วทั้งในส่วนของการประกาศไดเร็กตีฟเพื่อเรียกใช้งานพอร์ตอนุกรมและฟังก์ชั่นพร้อมใช้งาน (Built-in functions) เพื่อใช้ในการจัดการข้อมูลอนุกรม ซึ่งมีการทำงานในลักษณะโพลลิ่ง (polling) คือ การเขียนโปรแกรมดักจับข้อมูลที่ผ่านเข้ามาทางพอร์ตอนุกรมตลอดเวลาในลูปของโปรแกรมหลัก และแบบที่สองคือการอินเตอร์รัปต์ของพอร์ตอนุกรม โดยอาศัยฟังก์ชั่นที่เกี่ยวข้องกับการอินเตอร์รัปต์

วิธีที่สองนี้เป็นการใช้อินเตอร์รัปต์ในการดักจับสัญญาณข้อมูลที่ผ่านเข้ามาทางพอร์ตอนุกรมแทน ทำให้ไม่ต้องเขียนโปรแกรมจับข้อมูลทางพอร์ตตลอดเวลาเหมือนวิธีแรก

4.7.1 ไดเร็กตีฟสำหรับพอร์ตอนุกรม

ในการใช้งานไดเร็กตีฟ #use rs232() นั้นจะต้องประกาศใช้งานไดเร็กตีฟ #use delay() ก่อนทุกครั้ง และเมื่อประกาศใช้งานไดเร็กตีฟ rs232 แล้วก็สามารถที่จะเรียกใช้งาน RS232 I/O Built-in functions เพื่อใช้ในการจัดการข้อมูลที่เกี่ยวกับการรับส่งข้อมูลผ่านทางพอร์ตอนุกรม ประกาศใช้งาน #use rs232() ซึ่งมีรายละเอียดการใช้งานไดเร็กตีฟดังนี้

รูปแบบการใช้งาน

#use rs232(options)

โดย

Options มีรายละเอียดการใช้งานดังนี้

Options	**ความหมาย**
BAUD = X	กำหนดอัตราบอด
XMIT = pin	กำหนดขาพอร์ตส่งข้อมูล
RCV = pin	กำหนดขารับข้อมูล
RESTART_WDT	เคลียร์ค่า WDT เมื่อมีการใช้งาน ฟังก์ชั่น getc()
INVERT	กลับขั้วของสัญญาณ
PARITY = X-	กำหนด Parity เมื่อค่า X คือ N,E หรือ O
BITS = X-	กำหนด Bit เมื่อค่า X คือ 5-9
FLOAT_HIGH-	ใช้เอาต์พุตคอลเล็กเตอร์เปิด เมื่อไม่สามารถขับกระแสสูงได้
ERRORS-	เก็บข้อผิดพลาดในตัวแปร RS232_ERRORS และรีเซต ข้อผิดพลาดนั้น
BRGHIOK-	ยอมให้เกิดข้อผิดพลาดเมื่อมีปัญหากับอัตราบอด
ENABLE = pin-	กำหนดให้ขาพอร์ตเป็นลอจิก “1” ระหว่างส่งข้อมูล
	ใช้เอ็นเอเบิลการติดต่อแบบ RS-485

STREAM = streamed-	กำหนดพอร์ต RS232 แบบสตรีม สำหรับฟังก์ชั่น fputc

ตัวอย่าง

```
#use delay (clock=20000000)
    // ความถี่ที่ใช้คือ 20MHz
#use rs232 (baud=9600, xmit=PIN_A2, rcv=PIN_A3)
    // กำหนดบอดเรตที่ 9600, ขาพอร์ต RA2 เป็นขาส่งข้อมูล และ RA3 เป็นขาพอร์ตรับข้อมูล
```

4.7.2 การใช้ฟังก์ชั่นพร้อมใช้งานหรือ Built-in function สำหรับพอร์ตอนุกรม

ในส่วนของ Built-in function แบ่งออกเป็น 2 ส่วนคือ ฟังก์ชั่นที่เกี่ยวข้องกับการรับส่งข้อมูลและ ฟังก์ชั่นที่เกี่ยวข้องกับการใช้งาน RS232 I/O ซึ่งมีรายละเอียดดังนี้

4.7.3 ฟังก์ชั่นพร้อมใช้งานที่เกี่ยวกับการรับข้อมูลจากพอร์ตอนุกรมมาตรฐาน

ฟังก์ชั่นที่เกี่ยวข้องกับการรับข้อมูลผ่านทางพอร์ตอนุกรม RS232 ประกอบไปด้วย

ฟังก์ชั่นรับข้อมูลอักขระขนาด 8 บิต

รูปแบบการใช้งาน

```
value = getc() ;
value = fgetc(stream) ;
value = getch() ;
value = getchar() ;
```

โดยที่

Stream คือค่าคงที่ stream ที่ประกาศด้วยออปชั่น stream ในไดเร็กตีฟ #use rs232()

ค่าที่ส่งกลับ

Value คือ ข้อมูล character ขนาด 8 บิต

ตัวอย่าง

```
#use delay (clock=20000000)
    // use built-in function: delay_ms() & delay_us()
#use rs232 (baud=9600, xmit=PIN_C6, rcv=PIN_C7, stream=PC_COM)
    // use stream io RS232 port
void main (void) {
```

```
        char ch;
        printf ("Input a character :") ;
        ch = getc() ;
        printf ("The character input was : '%c'\r\n" ,ch) ;

while (TRUE)   {
        ch = fgetc (PC_COM) ;
        fputc (ch, PC_COM) ;
        if (ch = = 13)
                printf ( "\nGot a CR\r\n") ;
        }
}
```

4.7.4 ฟังก์ชั่นรับข้อมูลสตริง

รูปแบบการใช้งาน

```
        gets (string) ;
        fgets (string, stream) ;
```

โดย

String คือ อะเรย์ของ Character

Stream คือ ค่าคงที่ของ stream ที่ประกาศด้วยออปชั่น stream ในไดเร็กตีฟ #use rs232()

<u>ตัวอย่าง</u>

```
#use delay (clock=20000000)     // use built-in function : delay_ms() & delay_us()
#use rs232 (baud=9600, xmit=PIN_C6, rcv=PIN_C7, stream=PC_COM)
                                // use stream io RS232 port
void main (void) {
char str[20] ;

while (TRUE)   {
        printf ("Enter String : ") ;
        gets (str) ;
        printf ("\n\rYour string : \"%s\"",str) ;
```

```
        printf ("n\rEnter strin : ") ;
        fgets (str, PC_COM) ;
        fputs (str, PC_COM) ;
        }
}
```

4.7.5 ฟังก์ชั่น kbhit() ตรวจสอบการกดคีย์

รูปแบบการใช้งาน

```
Value = kbhit() ;
```

ค่าที่ส่งกลับ

Value : "0" (false) จนกว่าจะมี character ใดเข้ามาจึงเป็น "1" (true)

ตัวอย่าง

```
#use delay (clock=20000000)     // use built-in function : delay_ms() & delay_us()
#use rs232 (baud=9600, xmit=PIN_C6, rcv=PIN_C7)      // use stream io RS232 port
void main (void)        {
                        printf ("Press any key to continue :") ;
                        while (!kbhit()) ;//  do nothing
                        printf ("\r\nA key was pressed...\r\n") ;
```

4.7.6 ฟังก์ชั่นที่เกี่ยวข้องกับการส่งข้อมูลผ่านทางพอร์ตอนุกรม RS232

ประกอบไปด้วยฟังก์ชั่นดังต่อไปนี้

1. ฟังก์ชั่นส่งข้อมูลอักขระขนาด 8 บิต

รูปแบบการใช้งาน

```
Putc (cdata) ;
Putchar (cdata) ;
Fputc (cdata, stream) ;
```

โดยที่

Cdata : ข้อมูล character ขนาด 8 บิต

Stream : ค่าคงที่ stream ที่ประกาศด้วยออปชั่น stream ในไดเร็กตีฟ #use rs232()

ตัวอย่าง #use delay (clock=20000000) // use built-in function : delay_ms() & delay_us()

```
#use rs232 (baud=9600, xmit=PIN_C6, rcv=PIN_C7)        // use stream io RS232 port
void main (void)          {
                int i=0 ;
                        putc ('#') ;
                        do      {
                                putc('*') ;
                        } while (i++<10) ;
putc ('#') ;
}
```

2. ฟังก์ชั่นส่งข้อมูลสตริง

รูปแบบการใช้งาน

puts (string) ;

fputs (string, stream) ;

โดย

String คือ อะเรย์ของ Character

Stream คือ ค่าคงที่ของ stream ที่ประกาศด้วยออปชั่น stream ในไดเร็กตีฟ #use rs232()

<u>ตัวอย่าง</u>

```
#use delay (clock=20000000)     // use built-in function : delay_ms() & delay_us()
#use rs232 (baud=9600, xmit=PIN_C6, rcv=PIN_C7)      // use stream io RS232 port
void main (void)          {
puts ("CCS C Compiler = PIC C Compiler") ;
}
```

3. ฟังก์ชั่นส่งข้อความสตริงหรืออาร์เรย์ของอักขระ

รูปแบบการใช้งาน

Prints (string) ;

Printf (cstring, values…) ;

Fprintf (stream, cstring, values…)

โดย

String คือ อาร์เรย์ของ Character

Value คือ ค่าของตัวแปร

Stream คือ ค่าคงที่ของ stream ที่ประกาศด้วยออปชั่น stream ในไดเร็กตีฟ #use rs232()

ตัวอย่าง

```
Void main (void)        {
        Int i ;
        Float f ;
        Printf ("CCS C Compiler\n\r") ;
        Printf ("PIC C Compiler is Easy\n\r") ;
        Printf ("int I = %d, float = %f" , I, f) ;
}
```

4. ฟังก์ชั่น set_uart_speed ()

ใช้การเปลี่ยนแปลงอัตราบอดของการสื่อสารข้อมูลผ่านพอร์ตอนุกรม ขณะทำงานอยู่ (run-time)

รูปแบบการใช้งาน

Set_uart_speed (baud) ;

โดยที่ baud คือ ค่าระหว่าง 100-115200 เพื่อกำหนดอัตราบอดในหน่อยบิตต่อวินาที (bps)

ตัวอย่าง

```
Switch (input_b() & 3)   {
        Case 0 : set_uart_speed (2400) ;        break ;
        Case 1 : set_uart_speed (4800) ;        break ;
        Case 2 : set_uart_speed (9600) ;        break ;
        Case 3 : set_uart_speed (19200) ;       break ;
}
```

4.7.7 โมดูลฟังก์ชั่น INPUT.C

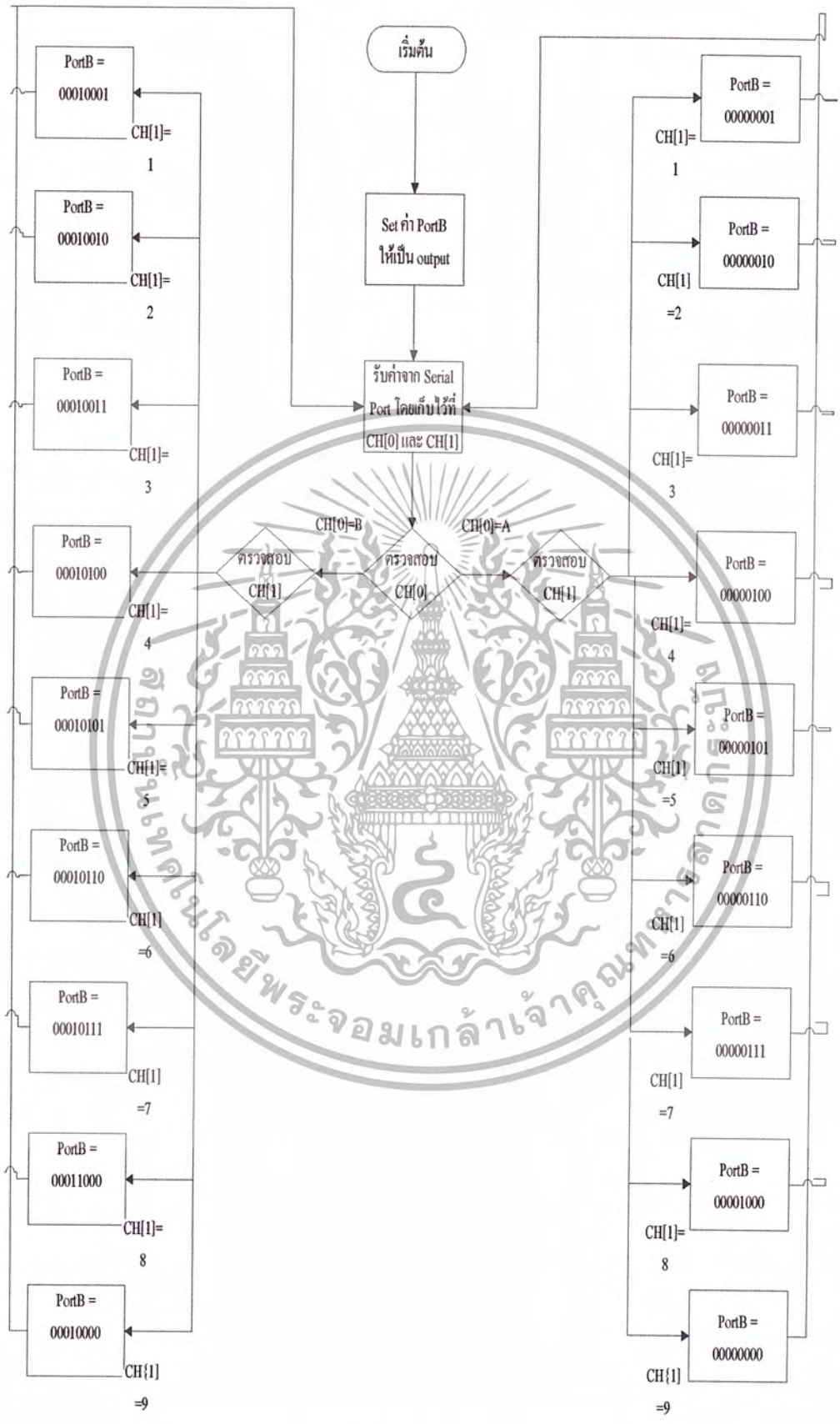
ประกอบด้วยฟังก์ชั่นที่เกี่ยวข้องกับการรับข้อมูลผ่านขาพอร์ตเชื่อมต่อพอร์ตอนุกรม (RS232 I/O) ใน CCS C จะไม่มีฟังก์ชั่น scanf()ในการรับข้อมูลแบบ ANSI C แต่จะใช้ฟังก์ชั่นต่อไปนี้แทน

1.gethex1 () , getthex ()	รับข้อมูลเลขฐานสิบหก
2.get_string ()	รับข้อมูลสตริง
3.get_int () , get_long ()	รับข้อมูลจำนวนเต็ม
4.get_float ()	รับข้อมูลจำนวนทศนิยม

4.8 การประยุกต์ใช้งานโปรแกรมภาษา C กับโครงงานนี้

จุดมุ่งหมายของการทำหุ่นยนต์ คือ

1. สามารถควบคุมผ่านคอมพิวเตอร์โดยใช้ Serial Port ได้
2. สามารถควบคุมด้วย RF ในกรณีที่ต้องการต่อแบบไร้สาย
3. สามารถใช้งาน PIC ในการควบคุมหุ่นยนต์ได้
4. สามารถควบคุมการทำงานของหุ่นยนต์ด้วยเดลไฟได้ถูกต้องแม่นยำ



รูปที่ 4.1 Flow Chart แสดงการทำงานของโปรแกรม PIC16F628 ในภาคส่ง

อธิบายรูปที่ 4.1

เริ่มต้น Set ค่า PortB ให้เป็น output จากนั้นรับค่าจาก serial port เก็บไว้ในอาร์เรย์ CH[0] และ CH[1] เสร็จแล้วจะทำการตรวจสอบค่า CH[0]

ถ้า CH[0] เป็น A ให้ตรวจสอบ CH[1]

ถ้า CH[1] เป็น 1 ให้ PortB เป็น 00000001

ถ้า CH[1] เป็น 2 ให้ PortB เป็น 00000010

ถ้า CH[1] เป็น 3 ให้ PortB เป็น 00000011

ถ้า CH[1] เป็น 4 ให้ PortB เป็น 00000100

ถ้า CH[1] เป็น 5 ให้ PortB เป็น 00000101

ถ้า CH[1] เป็น 6 ให้ PortB เป็น 00000110

ถ้า CH[1] เป็น 7 ให้ PortB เป็น 00000111

ถ้า CH[1] เป็น 8 ให้ PortB เป็น 00001000

ถ้า CH[1] เป็น 9 ให้ PortB เป็น 00000000

ถ้า CH[0] เป็น B ให้ตรวจสอบ CH[1]

ถ้า CH[1] เป็น 1 ให้ PortB เป็น 00010001

ถ้า CH[1] เป็น 2 ให้ PortB เป็น 00010010

ถ้า CH[1] เป็น 3 ให้ PortB เป็น 00010011

ถ้า CH[1] เป็น 4 ให้ PortB เป็น 00010100

ถ้า CH[1] เป็น 5 ให้ PortB เป็น 00010101

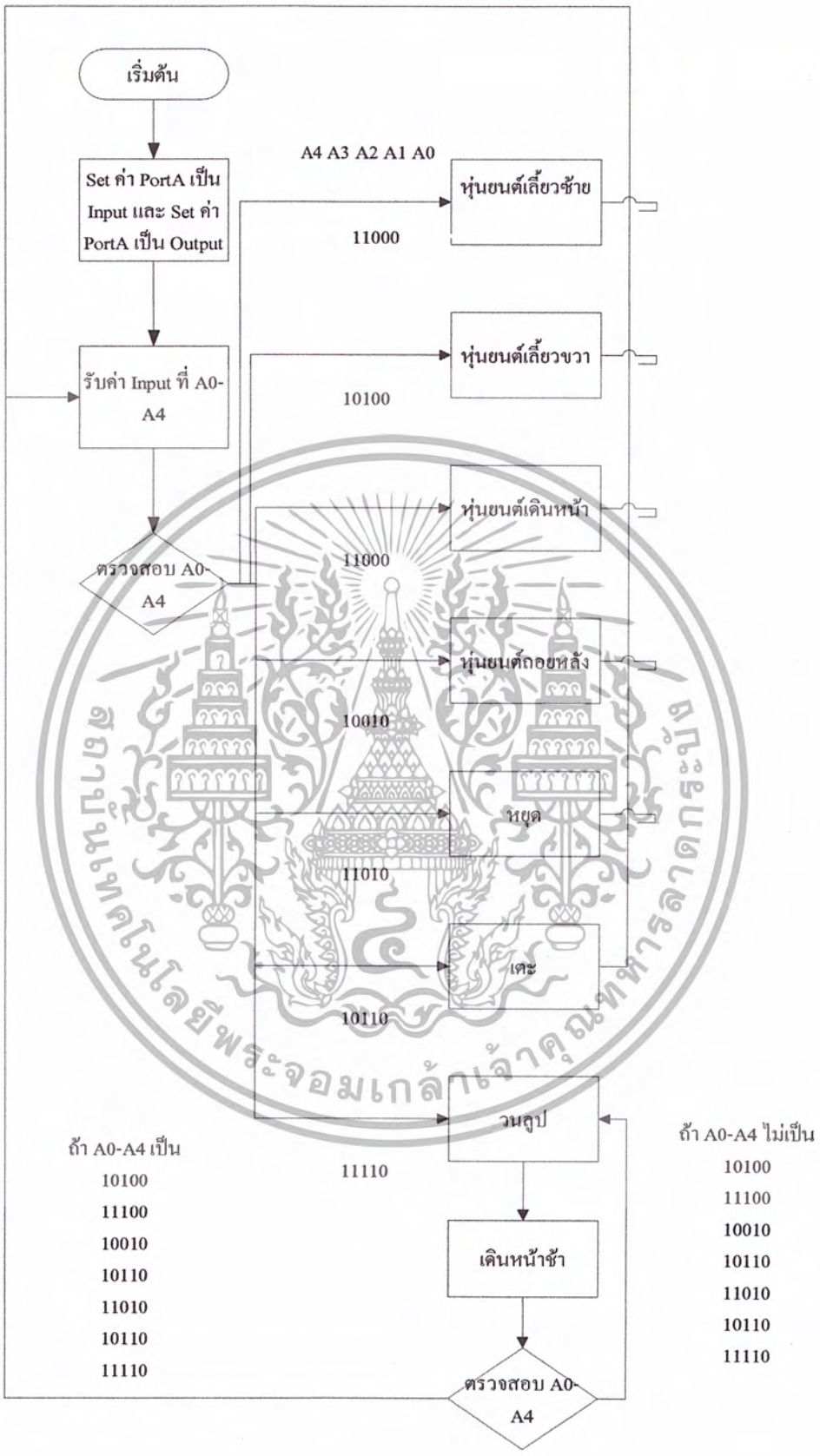
ถ้า CH[1] เป็น 6 ให้ PortB เป็น 00010110

ถ้า CH[1] เป็น 7 ให้ PortB เป็น 00010111

ถ้า CH[1] เป็น 8 ให้ PortB เป็น 00011000

ถ้า CH[1] เป็น 9 ให้ PortB เป็น 00010000

รูปที่ 4.1 Flow Chart แสดงการทำงานของโปรแกรม PIC16F628 ในภาคส่ง



รูปที่ 4.2 Flow Chart แสดงการทำงานของโปรแกรม PIC16F628 ในภาครับ

อธิบายรูปที่ 4.2

เริ่มต้น set ค่า PortA เป็น Input และ Set ค่า PortA เป็น Output จากนั้นรับค่าจาก Input ที่ขา A0-A4 แล้วทำการตรวจสอบค่า A0-A4 ถ้าค่า A4 A3 A2 A1 A0 ที่เรียงตามลำดับเป็นดังนี้

ถ้า เป็น 11000 ให้ หุ่นยนต์เลี้ยวซ้าย

ถ้า เป็น 10100 ให้ หุ่นยนต์เลี้ยวขวา

ถ้า เป็น 11100 ให้ หุ่นยนต์เดินหน้า

ถ้า เป็น 10010 ให้ หุ่นยนต์ถอยหลัง

ถ้า เป็น 11010 ให้ หุ่นยนต์หยุด

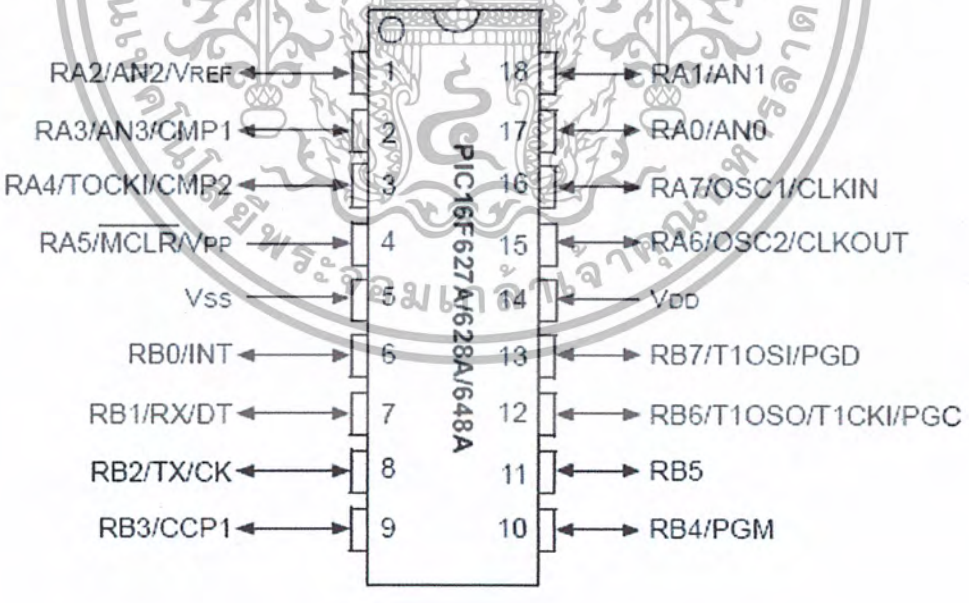
ถ้า เป็น 10110 ให้ หุ่นยนต์เตะ

ถ้า เป็น 11110 ให้ ทำการวนลูป แล้วเดินหน้าช้าๆ จากนั้นจะตรวจสอบ A0-A4

ถ้า A0-A4 เป็น 10100, 11100, 10010, 10110, 11010, 10110, 11110 ให้ทำการวนกลับมารับค่า Input ที่ A0-A4

ไม่เช่นนั้นให้ทำการวนลูปจนกว่าจะเจอกรณีดังกล่าว

4.9 ข้อมูลและรูปแบบการใช้งาน PIC16F628

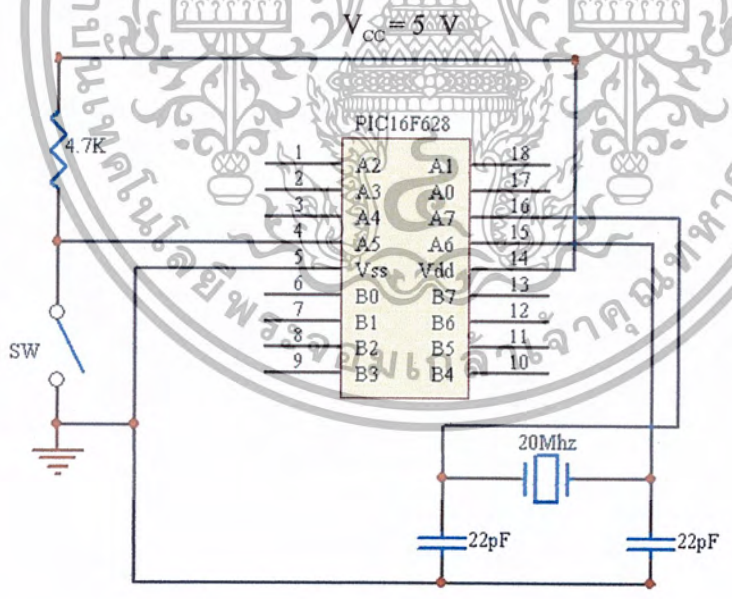


รูปที่ 4.3 แสดงรูป IC PIC16F628

*PIC16F628 จะใช้สัญญาณ Clock 20MHz เป็น Input

ข้อมูลที่จำเป็นสำหรับ PIC16F628

Ambient temperature under bias	-40 to +125 C
Storage temperature	-65 to +150 C
Voltage on V_{DD} with respect to V_{SS}	-0.3 to + 6.5V
Total power dissipation	800 mW
Maximum current out of V_{SS} pin	300 mA
Maximun current into V_{DD} pin	250 mA
Input clamp current,lik(Vi < 0 or Vi > V_{DD})	+/- 20 mA
Output clamp current,lok(Vo < 0 or Vo> V_{DD})	+/- 20 mA
Maximun output current sunk by any I/O pin	25 mA
Maximun output current sourced by any I/O pin	25 mA
Maximun output current sunk by PortA and PortB	200 mA
Maximun output current sourced PortA and PortB	200 mA



รูปที่ 4.4 แสดงวงจรการใช้งานPIC

อธิบายวงจรรูปที่ 4.4

สำหรับการนำPICไปต่อใช้งานนั้นเราต้องทำการต่อV_{cc} 5V เข้าที่ขา 14(V_{dd}) และต่อGndเข้าที่ขา6(Gnd) แล้วต้องทำการต่อวงจรเพื่อป้อนclockให้กับตัวPIC โดยใช้ crystal ความถี่20 MHzตามรูปที่ 4.2 และที่ขา 4(A5/$\overline{\text{MCLR}}$) ต้องทำการต่อสวิตท์ตามรูปเพื่อใช้ในการReset โปรแกรมของPIC ในกรณีที่โปรแกรมมีปัญหา

4.10 การประยุกต์ใช้งาน PIC ในโครงงานนี้คือ

ในภาคส่ง PIC16F628 ทำหน้าที่ เป็นตัวเช็คเงื่อนไขที่ได้รับมาจากคอมพิวเตอร์ ว่าตรงตามที่เราโปรแกรมลงไปหรือไม่ ถ้าตรงกับเงื่อนไขใด PIC ก็จะส่งคำสั่งที่แปรได้ไปให้กับ Encoder และไปที่ TLP315 ตามลำดับ โดยที่ PIC นั้นจะเชื่อมกับ PC โดยที่มี DS275 ทำหน้าที่เป็น IC Driver ที่ใช้เปลี่ยนแรงดัน โดยด้านที่ต่อกับคอมพิวเตอร์ จะเป็นแบบ สัญญาณมาตรฐาน RS232 โดยมีแรงดันสัญญาณที่มีค่า ตั้งแต่ +/- 3 V ถึง +/-15V ส่วนด้านที่ต่อกับ PIC จะเป็นสัญญาณ TTL ซึ่งมีค่าตั้งแต่ 0 – 5 V โดยที่ Input ของ PIC จะเป็น 20 บิต ซึ่งแบ่งเป็น ของ Address 10 บิต และของ Data 10 บิต โดยเข้าที่ขา A_0 และ A_1 Output ออกมาจะ ได้เป็น Address 8 บิต ออกทาง $B_4 - B_4$ และ Data 8 บิต ออกทาง $B_0 - B_3$ ส่งต่อไปยัง Encoder ต่อไป

ในภาครับ PIC16F628 จะทำการรับข้อมูลมาจาก Decoder ซึ่งข้อมูลจะเข้ามาทางขา D_{IN} ของ HT12D ข้อมูลที่ได้รับการ decode แล้วจะออกมาทางขา $D_8 - D_{11}$ และ VT รวมทั้งหมด 5 ขา ซึ่งจะต่อกับ PIC โดยจะเชื่อมดังต่อไปนี้

D_8 ของ HT12D เชื่อมต่อกับ A_0 ของ PIC

D_9 ของ HT12D เชื่อมต่อกับ A_1 ของ PIC

D_{10} ของ HT12D เชื่อมต่อกับ A_2 ของ PIC

D_{11} ของ HT12D เชื่อมต่อกับ A_3 ของ PIC

VT ของ HT12D เชื่อมต่อกับ A_4 ของ PIC

ซึ่ง PIC จะทำการประมวลผลและตรวจสอบเงื่อนไขและทำการส่งคำสั่งไปให้ มอเตอร์และโซลินอย โดยผ่านทาง 74LS245

บทที่ 5
การใช้งานพอร์ตอนุกรมของไมโครคอนโทรลเลอร์

สำหรับไมโครคอนโทรลเลอร์แล้วการมีพอร์ตที่เป็น Digital I/O อย่างเดียวนั้นยังไม่พอ หรือยังไม่ตรงกับความต้องการของผู้ออกแบบระบบ Embedded system สักเท่าไรนัก ในไมโครคอนโทรเลอร์ใหม่ๆนั้น มักจะมีพอร์ตอนุกรมเพิ่มเข้ามาอีกเพื่อทำให้การใช้งานในการติดต่อสื่อสารกับอุปกรณ์ภายนอกทำได้ง่ายขึ้น แต่การใช้งานก็มีข้อควรระวังอยู่บ้างเช่นกัน

5.1 การสื่อสารแบบอนุกรม

การสื่อสารแบบอนุกรมก็คือการที่อุปกรณ์หรือ Devices สองตัวทำการติดต่อกันโดยสัญญาณออกไปหรือรับเข้ามามีลักษณะของบวนสัญญาณที่เปลี่ยนแปลงสลับไปมาอย่างต่อเนื่อง ณ เวลาใดเวลาหนึ่งโดยมีรูปแบบการส่งที่แน่นอนเช่นการส่งรหัสมอร์ส การสื่อสารผ่าน RS232,RS485 ฯลฯ หลายคนอาจคิดว่าเมื่อพูดถึงพอร์ตอนุกรมของไมโครคอนโทรลเลอร์มักจะเหมารวมว่ามันคือพอร์ต RS232 แต่จริงๆแล้วไม่ใช่นะครับ RS232 คือข้อกำหนดของสัญญาณที่มีค่า+/- 3 V ถึง +/-15V เพราะฉะนั้นถ้าต้องการให้ไมโครคอนโทรลเลอร์สามารถสื่อสารแบบ RS232 ได้จะต้องมีตัวไอซีหรือวงจรที่จะทำให้สัญญาณอนุกรมที่เป็น TTL เปลี่ยนเป็นสัญญาณมาตรฐาน RS232 ก่อน เช่น MAX232 เป็นต้น แต่พอร์ตอนุกรมสามารถสื่อสารกับไมโครคอนโทรลเลอร์ได้โดยไม่ต้องมีไอซีหรือวงจรเข้ามาช่วย (ถ้าระยะทางไม่ไกล) หรือกับอุปกรณ์อื่นๆที่มีการสื่อสารแบบเดียวกัน

5.1.1 ประโยชน์ของพอร์ตอนุกรม

พอร์ตอนุกรมนั้นนอกจากจะใช้สื่อสารกับอุปกรณ์ภายนอกด้วยกันแล้วยังใช้เป็นอุปกรณ์สำหรับการดีบักโปรแกรม เช่น ใช้ดูค่าตัวแปรของของโปรแกรม ดูสถานะของพอร์ต ฯลฯ

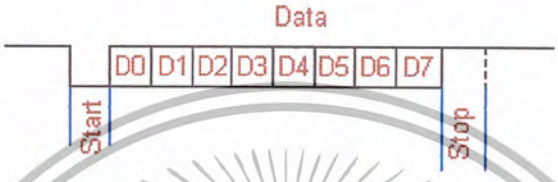
5.1.2 การเพิ่มระยะทางของพอร์ตอนุกรม

ปกติแล้วสัญญาณอนุกรมที่ออกจากตัวไมโครคอนโทรลเลอร์จะเป็นสัญญาณ TTL ที่มีขนาด 0-5V ถ้าการสื่อสารไม่ไกลกันเช่นอยู่บนบอร์ดเดียวกันก็ไม่ต้องใช้วงจรใดๆช่วยแต่ถ้าตัวบอร์ดหรือวงจรอยู่ไกลกันก็ใช้ไอซีช่วยเช่น DS232 ซึ่งก็จะกลายเป็นการสื่อสารแบบ RS232 ทันทีแต่ RS232 ก็มีข้อจำกัดด้านระยะทางเหมือนกันคือประมาณ 15-20 เมตรที่บอร์ดเร็ต 9600 และบอร์ดเร็ตจะแปรผกผันกับระยะทางด้วยและยังขึ้นกับไอซีไดร์เวอร์และชนิดของสายด้วยแต่ถ้าใช้สายยาวๆแล้ว

ใช้บอร์ดเร็ตสูงๆไม่แนะนำให้ใช้งานจริงถึงแม้จะทดลองว่าใช้ได้เพราะสถานที่ใช้งานจริงกับสถานที่ทดลองอาจมีสภาพแวดล้อมต่างกัน

5.1.3 ลักษณะสัญญาณของพอร์ตอนุกรม

สัญญาณของพอร์ตอนุกรมที่เป็นแบบที่นิยมใช้ทั่วไปคือ 1 บิต start, ข้อมูล 8 บิตและstop 1 บิตไม่มี parity บิต รวมทั้งหมดก็ 10 บิต จะมี สัญญาณตามรูปข้างล่าง

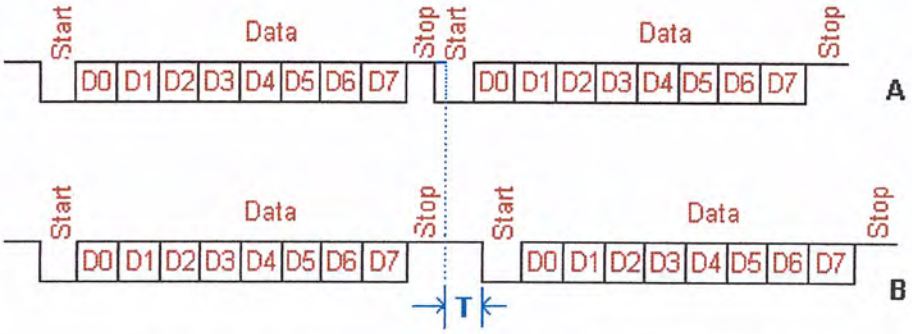


รูปที่ 5.1 แสดงรูปสัญญาณการส่งข้อมูลแบบ Serial Port

โดยบิตแรกจะเป็น start bit ซึ่งจะ active low ถัดไปอีก 8 บิตจะเป็นข้อมูลศูนย์หรือหนึ่งและบิตสุดท้ายจะเป็น stop บิตซึ่งจะ active high สำหรับความกว้างของแต่ละบิตก็คำนวนได้โดยอินเวิร์สค่าบอร์ดเร็ตหรือ 1/บอร์ดเร็ต เช่นบอร์ดเร็ต 9600 จะมีความกว้างของสัญญาณ 1 บิตเท่ากับ 104 uS เพราะฉะนั้น 10 บิตก็เท่ากับ 1.04 mS

5.1.4 ปัญหาด้านรับไม่สามารถรับข้อมูลไม่ได้

จากค่าของเวลาอันนี้ทำให้เรารู้ว่าถ้าเราส่งข้อมูลติดๆกันโดยไม่คำนึงถึงเวลาของแต่ละบิตจะทำให้ด้านรับซึ่งอาจเป็นไมโครคอนโทรลเลอร์หรือ PC แยกข้อมูลไม่ออกเช่นหลังจากส่งบิต stop ออกไป แล้วทำการส่งข้อมูลชุดใหม่ออกไปทันทีโดยที่ เวลาของ stop บิตยังไม่ครบทำให้ด้านรับเกิดอาการงงหรือรับข้อมูลได้มั่วๆ ดูรูปข้างล่างประกอบ



รูปที่ 5.2A ข้อมูลที่รับ-ส่งผิดพลาด รูปที่ 5.2B ข้อมูลที่รับ-ส่งถูกต้อง

จากรูปที่ 5.2 ถ้าการส่งข้อมูลเป็นแบบรูป A จะทำให้ด้านรับเกิดความผิดพลาดได้เนื่องจากบิต stop ของข้อมูลชุดแรกยังไม่ครบเวลาแล้วส่ง start บิตของข้อมูลชุดถัดไปทับ ซึ่งเหตุการณ์แบบนี้มักจะเกิดโดยไม่ได้ตั้งใจมากกว่าโดยเฉพาะไมโครคอนโทรลเลอร์ที่ทำงานที่ความเร็วสูงๆยิ่งใช้ภาษา assembly ด้วยแล้วมีโอกาสเกิดขึ้นได้มากเพราะส่วนใหญ่ใช้วิธีเช็ค Empty Tx bit หลังจาก Empty Tx bit ได้ Active แล้วเราก็เอาข้องมูลชุดใหม่ส่งทันทีโดยไม่หน่วงเวลา การส่งที่ใช้ได้และมีเสถียรภาพจะเป็นตามรูป B โดยหน่วงเวลาหลังจาก Empty Tx bit ได้ ative แล้วเป็นเวลามากว่าหรือเท่ากับเวลาของแต่ละบิตที่ได้กล่าวไปแล้ว ส่วนในการใช้ภาษาสูงเขียนอันนี้ไม่ค่อยเกิดให้เห็นเพราะว่าภาษาสูงนั้นกว่าจะส่งข้อมูลแต่ละไบต์หรือแต่ละชุดได้จะต้องมีการทำงานหลายอย่างเช่นกรณีของภาษาซีในคำสั่ง Printf กว่าจะส่งข้อมูลออกได้คอมไพล์เลอร์จะต้องแปลง, ตรวจสอบชนิดของตัวแปร,จับใส่ลง buffer ซึ่งตรงนี้ใช้เวลาไปมากพอสมควร แต่ก็ขึ้นกับเวอร์ชั่นหรือยี่ห้อของคอมไพล์เลอร์ด้วยบางตัวก็เก่งสามารถทำได้เร็วแต่ก็ยังไม่เคยเห็นสักที

รูปที่ 5.3 แสดงขาของพอร์ตอนุกรมแบบ 9 ขา

Signal Name	DB-9 Pin
FG (Frame Ground)	1
TD (Transmit Data)	3
RD (Receive Data)	2
RTS (Request To Send)	7
CTS (Clear To Send)	8
SG (Signal Ground)	5
DSR (Data Set Ready)	6
CD (Carrier Detect)	1
RI(Ring Indicator)	9
DTR (Data Terminal Ready)	4

ตารางที่ 5.1 แสดง หมายเลขประจำขาของ พอร์ตอนุกรมแบบ 9 ขา

FG (Frame Ground) = สัญญาณจากโมเดมให้ PC เตรียมพร้อม

TD (Transmit Data) = ส่งข้อมูลทีละบิตโดยเป็นลำดับไปที่โมเดม

RD (Receive Data) = รับข้อมูลทีละบิตโดยเป็นลำดับมาจากโมเดม

RTS (Request To Send) = สัญญาณจากโมเดมให้ PC พร้อมที่จะส่งข้อมูล

CTS (Clear To Send) = ตรวจสอบสัญญาณโมเดมว่าพร้อมจะรับข้อมูลจาก PC

SG (Signal Ground) = ขากราวน์

DSR (Data Set Ready) = สัญญาณบอกว่าโมเดมพร้อมทำงานแล้ว

CD (Carrier Detect) = เมื่อใดที่ตรวจสัญญาณเจอที่ปลายทางของสัญญาณจะทำให้สัญญาณ Active

RI(Ring Indicator) = ตรวจจับสัญญาณของโมเดม

ขาที่ 3 , 4 และ 7 เป็น Output , ขา 5 เป็น กราวน์ ส่วนขาที่เหลือเป็น Input

5.1.5 การ Test ว่าพอร์ตอนุกรมของ PC เสียหรือไม่

การ Test อันนี้ทำได้ง่ายมาก ทำได้โดยช็อตขา 2 กับขา 3 เข้าด้วยกันแล้วเปิดโปรแกรม Hyper terminal จากนี้ลองพิมพ์อะไรลงไปก็ได้ตัวที่พิมพ์จะถูกส่งออกไปที่ขา Td แล้วรับเข้ามาที่ขา Rd ปรากฏให้เราเห็นที่หน้าจอนั่นเอง แต่กรณีที่ต่อสายออกจาก DB-9 แล้วก็ช็อตที่ปลายสายก็ได้ ถ้าช็อตสายแล้วตัวอักษรที่พิมพ์ไม่ปรากฏก็แสดงว่าต่อสายผิดต้อง ถ้าพิมพ์แล้วมีข้อความออกมาก็ไม่ได้หมายความว่าต่อสายถูก อาจจะสลับสาย Rd กับ Td ก็ได้

* Rd กับ Td จะเป็นชื่อที่ใช้เรียกขารับและส่งข้อมูลของด้าน PC ส่วน Rx และ Tx หมายถึงขารับและส่งด้านของอุปกรณ์ภายนอกที่นำมาต่อกับ PC ซึ่ง x อาจจะหมายถึงอะไรก็ได้ ในที่นี้หมายถึงไมโครคอนโทรลเลอร์

5.1.6 การใช้งานไมโครคอนโทรลเลอร์ที่มีฮาร์ดแวร์ของพอร์ตอนุกรมในตัว(Hardware serial port)

ในการใช้งานชุดโมดุลของ Hardware serial port นั้นตัวไมโครคอนโทรลเลอร์จะต้องมีส่วนที่เรียกว่า UART หรือ USART ซึ่งก็แล้วแต่จะเรียกกันในต่อละยี่ห้อของเบอร์หรือตระกูลนั้นๆการทำงานจะคล้ายกันแต่ความสามารถจะต่างกัน

5.2 การเขียนโปรแกรมติดต่อและควบคุม Serial Port

5.2.1 พื้นฐานการสื่อสารแบบอนุกรม

ถึงแม้ว่าการสื่อสารแบบอนุกรมในคอมพิวเตอร์นั้นจะมีความเร็วในการสื่อสารช้ากว่าแบบขนาน ทั้งนี้เพราะว่าการเคลื่อนย้ายข้อมูลแบบอนุกรมนั้นเป็นการส่งข้อมูลทีละ 1 บิต แต่พอร์ตขนานนั้นสามารถส่งข้อมูลได้ทีละหลายๆบิตพร้อมกันส่งผลให้การสื่อสารข้อมูลแบบอนุกรมมีความเร็วต่ำกว่าแบบขนาน

แต่ว่าการส่งข้อมูลแบบอนุกรมนั้นมีข้อที่เหนือกว่าการส่งข้อมูลแบบขนานคือ การที่สามารถส่งข้อมูลได้ในระยะทางที่ไกลกว่าแบบขนาน อีกทั้งสายสัญญาณที่ใช้ยังมีน้อยกว่าการส่งข้อมูลแบบขนานอีกด้วย การสื่อสารแบบอนุกรมสามารถแบ่งออกได้เป็น 3 รูปแบบดังนี้

1. Simplex สามารถส่งข้อมูลได้อย่างเดียว เป็นการสื่อสารแบบทางเดียว
2. Half-Duplex สามารถส่งข้อมูลไปยังปลายทางและสามารถรับข้อมูลจากปลายทางได้ แต่ไม่สามารถทำการส่งและรับข้อมูลได้ในเวลาเดียวกัน
3. Full-Duplex สามารถรับและส่งข้อมูลได้ในเวลาเดียวกัน

นอกจากนี้แล้วยังสามารถแบ่งประเภทของการสื่อสารแบบอนุกรมตามลักษณะสัญญาณในการส่งได้อีก 2 แบบคือ

1. การสื่อสารแบบซิงโครนัส (Synchronous) สำหรับการสื่อสารแบบซิงโครนัสนี้จะใช้สัญญาณนาฬิกาควบคุมการส่งสัญญาณ เช่น สายคีย์บอร์ดคอมพิวเตอร์ โดยจะมีสายสัญญาณเส้นหนึ่งเป็นสายสัญญาณนาฬิกา ส่วนอีกเส้นหนึ่งเป็นสายของข้อมูล (และจะมีสายกราวน์ด้วย)

สำหรับการสื่อสารแบบซิงโครนัสนี้เหมาะสำหรับการทำงานในระยะใกล้ ข้อมูลที่จะส่งมีไม่มากนัก เพราะถ้าระยะทางไกลขึ้นจะทำให้สัญญาณนาฬิกามีปัญหา อีกทั้งต้องมีสายหลายเส้นทำให้สิ้นเปลืองมาก

2. การสื่อสารแบบอะซิงโครนัส (Asynchronous) สำหรับการสื่อสารแบบอะซิงโครนัสนั้นจะใช้สายข้อมูลเพียงตัวเดียว แต่จะใช้รูปแบบการส่งข้อมูล หรือ Bit Pattern เป็นตัวกำหนดว่าส่วนไหนเป็นส่วนเริ่มต้นข้อมูล, ส่วนไหนเป็นตัวข้อมูล, ส่วนไหนจะเป็นส่วนตรวจสอบความถูกต้องของข้อมูล และส่วนไหนเป็นส่วนปิดท้ายของข้อมูล โดยต้องกำหนดให้สัญญาณนาฬิกาเท่ากันทั้งภาคส่ง และภาครับ ซึ่งจะมีอุปกรณ์พิเศษที่ชื่อว่า UART หรือ Universal Asynchronous Receiver / Transmitter คอยควบคุมการรับ และส่งข้อมูล

UART ที่นิยมใช้งานในคอมพิวเตอร์คือ ชิป 8250 ซึ่งให้ความเร็วสูงสุด 57.6 Kbps (หรือ 56 K) ขณะที่ปัจจุบันเริ่มใช้งานกับชิป 16450 ซึ่งสนับสนุน 8250

5.2.2 รู้จักกับมาตรฐาน RS-232C

มาตรฐาน RS-232C เป็นมาตรฐานที่ได้รับการออกแบบมาเพื่อที่จะทำให้อุปกรณ์ต่อพ่วงจากผู้ผลิตต่างกันสามารถทำงานร่วมกันได้ มาตรฐานหลายชนิดได้รับการออกแบบขึ้นมา แต่มาตรฐานที่ได้รับความนิยมและใช้กันกว้างขวางมากที่สุดคือ มาตรฐาน RS-232C ซึ่งถูกประกาศใช้ในปี 1969 โดยสมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association : EIA) ในยุคแรกๆการอินเตอร์เฟสแบบ RS-232C ถูกออกแบบสำหรับเชื่อมต่อเทอร์มินอล (DTE : Data Terminal Equipment) กับโมเด็ม (DCE : Data Communication Equipment) ทั้งนี้ก็เพื่อป้องกันไม่ให้เกิดการส่งข้อมูลบนสายเดียวกัน

มาตรฐาน RS-232C ได้แบ่งอุปกรณ์ออกเป็น 2 ประเภท ซึ่งอุปกรณ์ทั้งสองประเภทนี้คือ

1. อุปกรณ์ DTE (Data Terminal Equipment) เป็นอุปกรณ์สำหรับส่งข้อมูล (เอาต์พุต)
2. อุปกรณ์ DCE (Data Communication Equipment) เป็นอุปกรณ์สำหรับรับข้อมูล (input)

ตามมาตรฐาน RS-232C แล้วคอนเน็กเตอร์ของ DTE จะเป็นตัวผู้ ส่วนคอนเน็กเตอรของ DCE จะเป็นตัวเมีย ซึ่งคอนเน็กเตอร์ที่นิยมใช้กันอยู่จะเป็นชนิด D-Type แบบ 9 ขา และแบบ 25 ขา โดยจะติดตั้งอยู่หลังเครื่องคอมพิวเตอร์ ระดับแรงดันจะมีค่าระหว่าง -3 V ถึง -15 V

สำหรับลอจิก High และลอจิก Low จะมีระดับแรงดันระหว่าง +3 V ถึง +15 V สามารถรับส่งข้อมูลได้ที่ความยาวของสัญญาณสูงสุด 50 ฟุต หรือ 150 เมตร แต่ถ้าเราต้องการสื่อสารกับอุปกรณ์อื่นที่อยู่ห่างกันมากๆ เราจำเป็นต้องใช้อุปกรณ์อื่นๆ เข้าช่วย เช่น การใช้โมเด็ม เป็นต้น

5.2.3 องค์ประกอบของการรับส่งข้อมูลแบบอนุกรม

การสื่อสารแบบอนุกรมที่นิยมใช้กับคอมพิวเตอร์นั้น เป็นการสื่อสารข้อมูลแบบอะซิงโครนัสนั่นคือ ต้องใช้สายสัญญาณเส้นเดียวทำหน้าที่ทั้งส่งส่วนที่เป็นข้อมูล ละส่วนที่ใช้ควบคุมการส่งข้อมูล ดังนั้นข้อมูลที่อ่านได้แต่ละบิตจากการส่งแบบอนุกรม จึงต้องถูกแยกแยะว่าใช้สำหรับวัตถุประสงค์ใด โดยเราสามารถแบ่งได้เป็น 4 ส่วนคือ

1. Start Bit	ขนาด 1 บิต
2. บิตข้อมูล (Data Character)	ขนาด 7 หรือ 8 บิต
3. Parity Bit	ขนาด 1 บิต
4. Stop Bit	ขนาด 1 หรือ 2 บิต

แต่ละตัวอักษรที่ถูกส่งออกไปเป็นกลุ่มจะประกอบไปด้วยบิตเริ่มต้น บิตข้อมูล บิตพาริตี้ (จะมีหรือไม่มีก็ได้) และบิตจบ โดยเราพอจะสรุปหน้าที่ของแต่ละส่วนได้ดังนี้

Start Bit หรือบิตเริ่มต้น จะใส่ที่จุดเริ่มต้นเสมอ เพื่อเตือนอุปกรณ์ฝ่ายรับว่าข้อมูลกำลังจะมาถึง

Data Character หรือบิตข้อมูล การส่งบิตข้อมูลจะส่งเป็นกลุ่มๆ โดยทั่วไปจะส่งเป็น 7 หรือ 8 บิต ซึ่งเพียงพอสำหรับการส่ง Ascii Word

Parity Bit หรือบิตพาริตี้ ใช้ในการตรวจสอบความถูกต้องของข้อมูลที่ส่ง เราจะใส่บิตพาริตี้เข้าไป แต่ทั้งตัวรับและตัวส่งจะต้องรู้กันว่าใช้พาริตี้แบบไหนในการส่งข้อมูลซึ่งหลักการในการกำหนดบิตพาริตี้มีหลายแบบดังนี้

พาริตี้คู่ (Even Parity) ค่าของบิตพาริตี้นี้เมื่อรวมกับทุกๆบิตของข้อมูลแล้ว จะต้องมีจำนวนบิตที่เป็นเลข 1 เป็นเลขคู่ ตัวอย่างเช่น ข้อมูล 1000101 มีเลข 1 ทั้งหมด 3 ตัว ดังนั้นบิตพาริตี้จะเป็น 0

พาริตี้คี่ (Odd Parity) ค่าของบิตพาริตี้นี้เมื่อรวมกับทุกๆบิตของข้อมูลแล้ว จะต้องมีจำนวนบิตที่เป็นเลข 1 เป็นเลขคี่ ตัวอย่างเช่น ข้อมูล 1000101 มีเลข 1 ทั้งหมด 3 ตัว ดังนั้นบิตพาริตี้จึงเป็น 1

ไม่มีพาริตี้ (None) ถ้าตั้งบิตพาริตี้เป็น None ทั้งภาครับและภาคส่งจะไม่มีการตรวจสอบบิตพาริตี้

Stop Bit หรือบิตจบ เป็นบิตที่ส่งมาปิดท้ายข้อมูล

5.2.4 อัตราเร็วในการรับส่งข้อมูลแบบอนุกรม

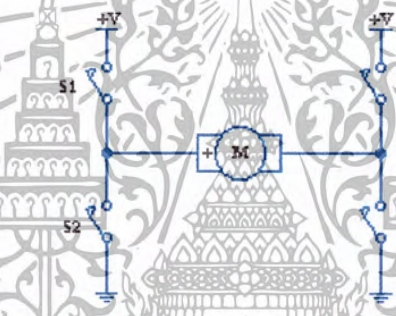
การที่อุปกรณ์ 2 อย่างจะติดต่อสื่อสารกันได้นั้น จะต้องทำงานด้วยอัตราเร็วเท่ากัน ซึ่งอัตราเร็วในการสื่อสารแบบอะซิงโครนัสคือ ค่าบอดเรต (Baud Rate) มีหน่วยเป็นบิตต่อวินาที ซึ่งค่าอัตราเร็วในการสื่อสารแบบอนุกรมสำหรับมาตรฐาน RS-232C นั้นมีใช้ดังนี้ 110, 150, 300, 600, 1200, 2400, 4800, 9600 และ 19200 บิตต่อวินาที

บทที่ 6
การขับมอเตอร์ (Drive Motor)

DC motor

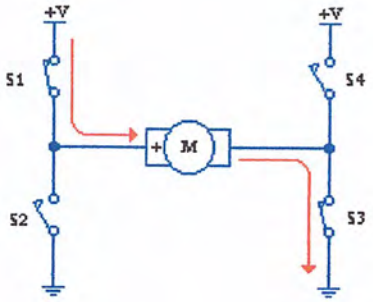
จะมีการทำงานเพียง 3 state เท่านั้น คือ หมุนตามเข็มนาฬิกา หมุนทวนเข็มนาฬิกา และหยุดนิ่งเท่านั้น ซึ่งอัตราความเร็วในการหมุนและแรงบิดจะขึ้นอยู่กับกระแสและแรงดันที่จ่ายให้มัน ถ้าหากแรงบิดของมอเตอร์ไม่เพียงพอต่อการทำงานเราก็สามารถที่จะทำการทดรอบของการหมุนลงก็จะได้แรงบิดเพิ่มมากขึ้น หรือ เราสามารถควบคุมความเร็วของมอเตอร์ได้โดยการจ่ายพลังงานในลักษณะ pulse width moduration (PWM)

การควบคุมทิศทางการหมุนของ DC Motor หลักการทำงานของวงจร H-Bridge Switching



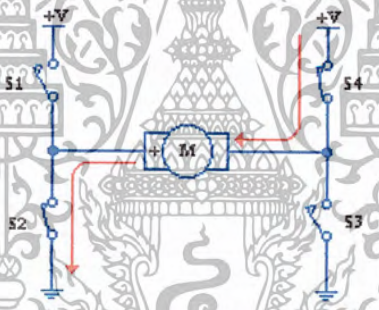
รูปที่ 6.1 แสดงสภาวะเริ่มต้น

เริ่มจากหลักการของวงจรนั้น จะประกอบไปด้วย สวิทช์ 4 ตัว นั่นก็คือ S1 ,S2 ,S3 และ S4 นั่นเอง ซึ่งในรูปตัวอย่าง จะใช้ DC-Motor เป็น Load ของวงจรนั่นเอง ในสภาวะเริ่มต้น สวิทช์ ทุกตัว Off อยู่ ก็จะไม่มีอะไรเกิดขึ้นทั้งสิ้น เพราะไม่มีกระแสไฟฟ้าไหลเข้าสู่ มอเตอร์ (รูปที่ 6.1)



รูปที่ 6.2 แสดงสภาวะเมื่อ On S1 กับ S3 พร้อมกัน

และเมื่อเราทำการ On สวิทช์ S1 และ S3 พร้อมกัน (รูปที่ 6.2) จะเป็นการเชื่อมวงจร ทำให้มีกระแสไฟฟ้า ไหลผ่านมอเตอร์ จากขั้วบวกของมอเตอร์ ไปยังขั้วลบของมอเตอร์ จึงทำให้มอเตอร์สามารหมุนได้ ในทิศทาง Forward (จะหมุนแบบตามเข็มนาฬิกา หรือทวนเข็มนาฬิกานั้น ขึ้นอยู่กับลักษณะของ การพันขดลวดภายในมอเตอร์)

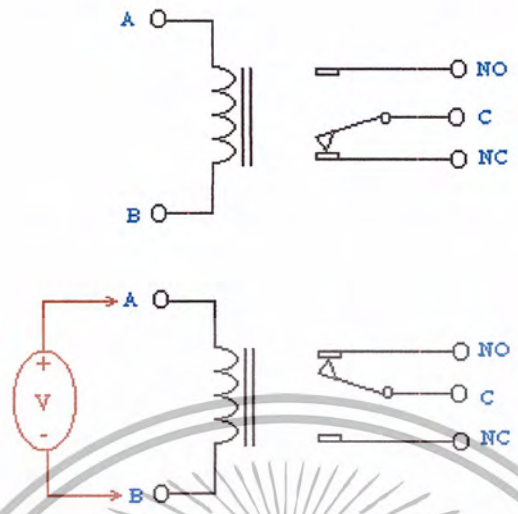


รูปที่ 6.3 แสดงสภาวะเมื่อ On S2 กับ S4 พร้อมกัน

และในทางกลับกัน ถ้าหากเราทำการ On สวิทช์ S2 และ S4 พร้อมกัน (รูปที่ 7.14) ก็จะเป็นการเชื่อมวงจร และทำให้เกิดกระแสไฟฟ้า ไหลผ่านมอเตอร์ จากขั้วลบของมอเตอร์ ไปยังขั้วบวกของมอเตอร์ จึงทำให้มอเตอร์สามารถหมุนได้ และเป็นการหมุนในทิศทาง Reverse (กลับทิศทางกับกรณีแรก)

สรุปได้ว่าวงจรนี้จะอาศัยสวิทช์ 4 ตัว เพื่อบังคับทิศทางการไหล ของกระแสไฟฟ้า ที่ไหลผ่านมอเตอร์ เพื่อควบคุมให้มอเตอร์หมุนตามทิศทางที่เราต้องการ โดยการผลัดกัน On และ Off สวิทช์พร้อมกัน 2 ตัว นั่นเอง

การควบคุมทิศทางการหมุนของ DC Motor สร้างวงจร H-Bridge Switching จาก Relay



รูปที่ 6.4 แสดงสภาวะเมื่อได้รับแรงดันตกคร่อม ขา A และ B

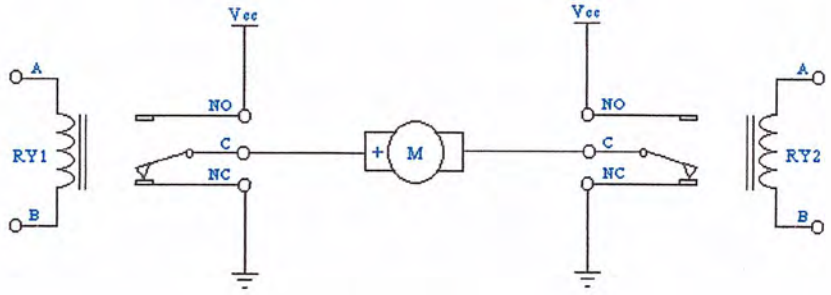
ภายในโครงสร้างของ รีเลย์ จะประกอบไปด้วยขดลวด (Coil) 1 ชุด และ หน้าสัมผัส (Contactor) ซึ่งในหน้าสัมผัส 1 ชุด จะประกอบไปด้วย

- หน้าสัมผัสแบบปกติปิด (Normally Close หรือ NC.) ซึ่งในสภาวะปกติ ขานี้จะต่ออยู่กับขาร่วม (Common)
- หน้าสัมผัสแบบปกติเปิด (Normally Open หรือ NO.) ขานี้จะต่อเข้ากับขาร่วม (Common) เมื่อขดลวดมีแรงดันตกคร่อม หรือกระแสไหลผ่าน (ในปริมาณที่เพียงพอ)

ใน รีเลย์ 1 ตัว อาจมีหน้าสัมผัสมากกว่า 1 ชุด เช่น 2 ชุด, 4 ชุด เป็นต้น.... (แล้วแต่ผู้ผลิต)

เมื่อขดลวดได้รับแรงดันตกคร่อม (ขา A และ B) จะทำให้มีกระแสไหลผ่านขดลวด ซึ่งจะทำให้เกิดอำนาจสนามแม่เหล็ก ดึงดูดให้หน้าสัมผัส NO และ C ติดกัน.... (ดูรูปที่ 6.4)

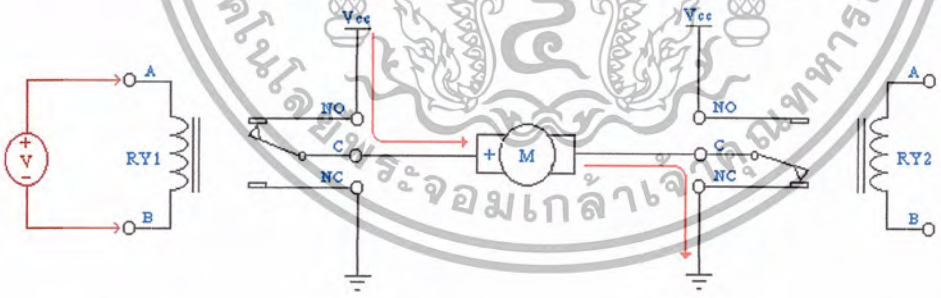
รูปวงจร



รูปที่ 6.5 แสดงรูปวงจรที่ในสภาวะเริ่มต้น

วงจรที่เราจะทดลองกันในคราวนี้ จะประกอบไปด้วย รีเลย์ 2 ตัว คือ RY1 และ RY2 ซึ่ง Load ก็คือ DC-Motor ซึ่งต่ออยู่กับขาร่วม (C.) ของ RY1. และ RY2. โดยขั้วบวก (+) ของมอเตอร์ ต่ออยู่ที่ขา C. ของ RY1 และขั้วลบ (-) ของมอเตอร์ ต่ออยู่ที่ขา C. ของ RY2 โดยที่ขา NO. ของ RY1 และ RY2 จะต่ออยู่กับขั้วบวก ของแหล่งจ่ายไฟ ที่จะจ่ายให้มอเตอร์ (Vcc) และขา NC. ของ RY1 และ RY2 จะต่อลงกราวด์

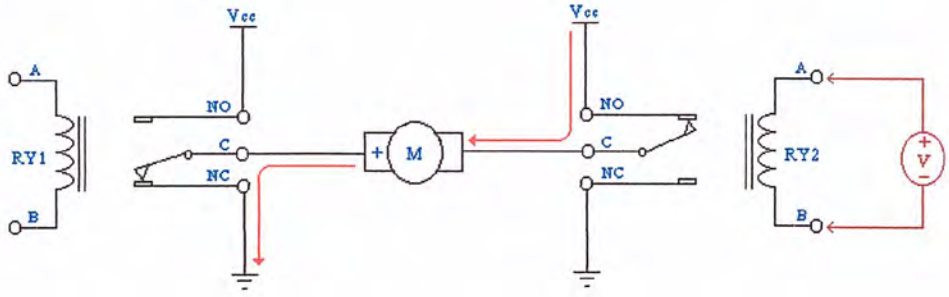
กรณีที่ RY1 ทำงาน



รูปที่ 6.6 แสดงสภาวะของวงจะขณะที่ RY1 ทำงาน

เมื่อ RY1 ทำงาน (มีกระแสไหลผ่านขดลวดในปริมาณที่เพียงพอ) จะทำให้เกิดอำนาจสนามแม่เหล็กไฟฟ้า ดึงดูดให้ขา NO และขา C ของ RY1 ติดกัน ส่งผลให้มีกระแสไหลจากแหล่งจ่าย (Vcc) ผ่านเข้าสู่ขั้วบวก (+) ของมอเตอร์ ผ่านไปยังขา C ของ RY2 ซึ่งต่ออยู่ที่ NC และลงกราวด์ ทำให้มีกระแสไหลผ่านมอเตอร์ในทิศทางบวก และครบวงจร จึงทำให้มอเตอร์สามารถหมุน ในทิศทาง Forward ได้

กรณีที่ RY2 ทำงาน



รูปที่ 6.7 แสดงสภาวะวงจรขณะที่ RY2 ทำงาน

เมื่อ RY2 ทำงาน (มีกระแสไหลผ่านขดลวดในปริมาณที่เพียงพอ) จะทำให้เกิดอำนาจสนามแม่เหล็กไฟฟ้า ดึงดูดให้ขา NO และขา C ของ RY2 ติดกัน ส่งผลให้มีกระแสไหลจากแหล่งจ่าย (Vcc) ผ่านเข้าสู่ขั้วลบ (-) ของมอเตอร์ ผ่านไปยังขา C ของ RY1 ซึ่งต่ออยู่ที่ NC และลงกราวด์ ทำให้มีกระแสไหลผ่านมอเตอร์ในทิศทางลบ และครบวงจร จึงทำให้มอเตอร์สามารถหมุน ในทิศทาง Reward ได้

ถึงแม้ว่า รีเลย์ จะเป็นอุปกรณ์ ที่นำมาประยุกต์สร้างเป็นวงจร H-Bridge switching ได้ง่าย ไม่ยุ่งยากก้อตาม แต่ รีเลย์ นั้นเป็นอุปกรณ์แม่เหล็ก ที่มีส่วนเคลื่อน ไหวทางกล เพราะฉะนั้นก้อย่อมจะมีข้อจำกัดทางกลอยู่บ้าง เช่น

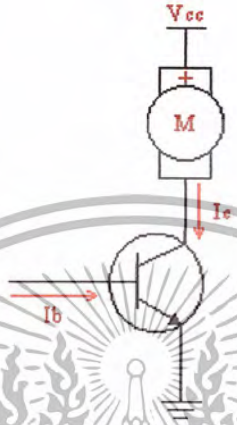
ผลกระทบของการตอบสนอง... รีเลย์ เป็นอุปกรณ์ที่มีความเร็วในการทำงานต่ำ เช่นรีเลย์ชนิดแรงดันต่ำ (กระตุ้นขดลวดไม่เกิน 24 V.) จะใช้เวลาในการทำงานประมาณ 10 - 50 ms. และรีเลยขนาดใหญ่ ที่ใช้ควบคุมมอเตอร์ในโรงงานอุตสาหกรรม นั้น อาจใช้เวลาในการทำงานมากกว่า 100 ms. เลยทีเดียว

ผลกระทบจากอำนาจแม่เหล็ก... รีเลย์ เป็นอุปกรณ์แม่เหล็ก ดังนั้นเราจึงไม่สามารถหนีปัญหานี้ได้ ดังนั้นจึงไม่ใช่เรื่องแปลก ที่หลายคนอาจเคยมีปัญหาของ รีเลย์ไปรบกวนการทำงานของวงจรไมโครฯ การแก้ไข อาจมีหลายวิธี เช่น

- แยกกราวด์ : คือ การแยกกราวด์ของแหล่งจ่ายไฟสำหรับวงจรไมโครฯ และแหล่งจ่ายไฟกระตุ้นรีเลย์ ออกจากกันโดยใช้อุปกรณ์ Opto Cuppler
- แยกบอร์ด : คือ การแยกการทำงานในส่วนของวงจรรีเลย์ ออกไปจากบอร์ดไมโครฯ แล้วทำการชีลด์ให้ดี

- ลดขนาดแรงดันกระตุ้น : คือ การเปลี่ยนตัวรีเลย์ เช่นเปลี่ยนจากรีเลย์ขนาด 24 v. มาเป็นขนาด 12 V .

การควบคุมทิศทางการหมุนของ DC Motor สร้างวงจร H-Bridge Switching จาก Transistor



รูปที่ 6.8 แสดงวงจรสาธิตแบบง่ายใช้ทรานซิสเตอร์ทำหน้าที่เป็นสวิตซ์ควบคุมมอเตอร์

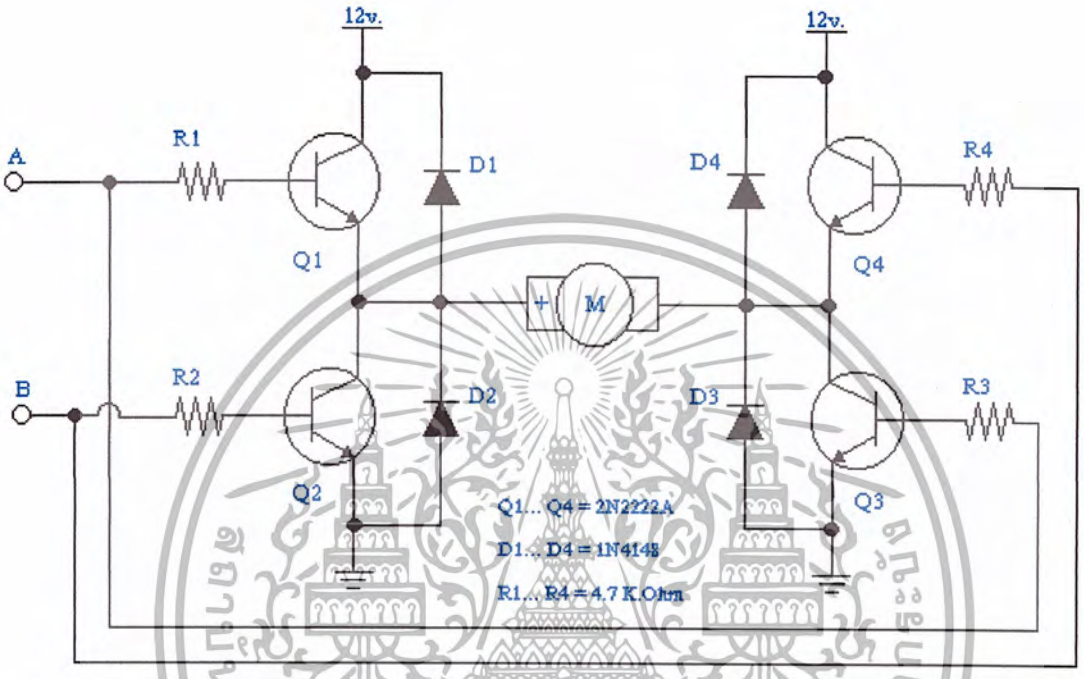
ทรานซิสเตอร์.... เป็นอุปกรณ์สารกึ่งตัวนำ (Semiconductor device) ที่เราสามารถนำคุณสมบัติของการ Cutoff และการ Saturation มาประยุกต์ใช้งานเป็นสวิทช์ได้ และที่สำคัญ มันเป็นสวิทช์อิเล็กทรอนิกส์ ที่เราสามารถควบคุมการปิด/เปิด ได้

จากรูปที่ 6.8 เป็นวงจรสาธิตแบบง่าย ๆ โดยการนำทรานซิสเตอร์ มาเป็นสวิทช์ควบคุมมอเตอร์ หลักการคิดง่าย ๆ ก็คือ เมื่อเราป้อนกระแส I_B ด้วยปริมาณที่มากพอ ก็จะทำให้ทรานซิสเตอร์ทำงาน (On) จะทำให้กระแส I_C ไหล แปลว่ามีกระแสไหลผ่านมอเตอร์ได้ (กระแส I_B มากเพียงพอที่จะทำให้ทรานซิสเตอร์อยู่ในสภาวะ "อิ่มตัว" ได้)

ในสภาวะ อิ่มตัว (Saturation mode) นี้ ทรานซิสเตอร์จะทำงานเหมือนกับสวิทช์ปิดวงจร ค่าความต้านทานระหว่างขา C และขา E จะมีค่าเข้าใกล้ศูนย์ กระแส I_C ที่ไหลจะมีค่าเข้าใกล้ $I_C(max)$

ในสภาวะคัทออฟ (Cutoff mode) นี้ จะเกิดขึ้นเมื่อเราหยุดจ่ายกระแส I_B ($I_B = 0$) ทรานซิสเตอร์จะทำงานเหมือนกับสวิทช์เปิดวงจร ค่าความต้านทานระหว่างขา C และขา E จะมีค่าเป็นอนันต์ กระแส I_C จะมีค่าเข้าใกล้ศูนย์

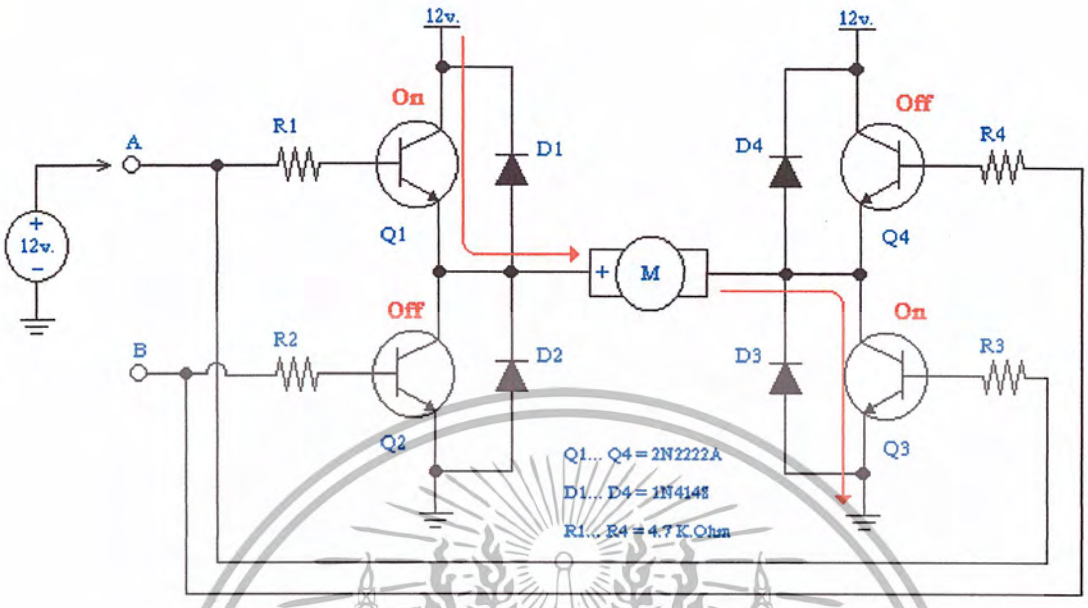
ข้อดีของการนำทรานซิสเตอร์ มาประยุกต์ใช้งานเป็นสวิทช์นั้น น่าจะเป็นส่วนของการควบคุมที่สามารถตอบสนองจังหวะของการเปิด/ปิด สวิทช์ได้นับล้านครั้งต่อวินาที (ความเร็วในการตอบสนองมีหน่วยเป็น ns) และที่สำคัญ คือไม่ทำให้เกิดปัญหารบกวนจากสนามแม่เหล็ก อย่างรีเลย์



รูปที่ 6.9 แสดงรูปวงจรในสภาวะเริ่มต้น

วงจรที่เราจะทดลองกันในคราวนี้ จะประกอบไปด้วย ทรานซิสเตอร์ Q1, Q2, Q3 และQ4 ทุกตัวใช้เบอร์ 2N2222A ส่วน R1, R2, R3 และR4 ใช้ค่าความต้านทาน 4.7k ทำหน้าที่จำกัดกระแส I_B ส่วนไดโอด D1, D2, D3 และD4 ใช้เบอร์ 1N4148 ทำหน้าที่ป้องกันกระแสไหลย้อนกลับจากมอเตอร์ในขณะทรานซิสเตอร์หยุดทำงาน แหล่งจ่ายของวงจรจะใช้เป็น แบตเตอร์รี่ขนาด 12 V. หรือจาก Power Supply ก็ได้ครับ ส่วนมอเตอร์นั้นให้ใช้เป็นมอเตอร์ตัวเล็ก ๆ ในของเด็กเล่น เนื่องจากจำกัดกระแสของวงจรเอาไว้ที่ 500 mA .

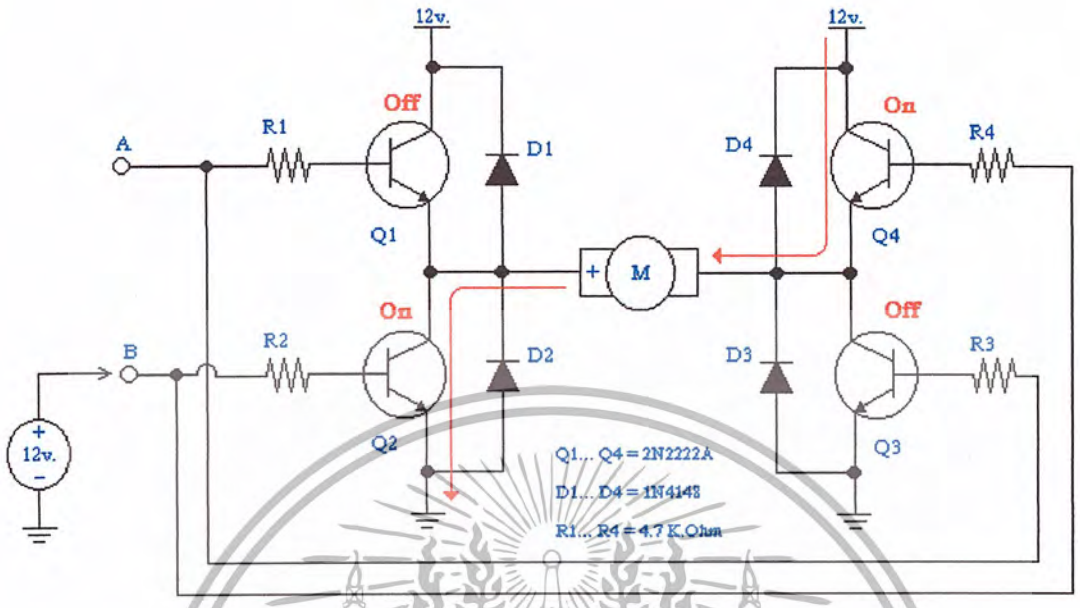
กรณีที่ Q1 และ Q3 ทำงาน



รูปที่ 6.10 แสดงสภาวะของวงจรขณะที่ Q1 และ Q3 ทำงาน

เมื่อมีการจ่ายแรงดัน 12 V. เข้าที่จุด A ทำให้มีกระแสไหลผ่าน R1 เข้าสู่ขา base ของ Q1 และมีกระแสไหลผ่าน R3 เข้าสู่ขา base ของ Q3 ทำให้ Q1 และ Q3 ทำงาน (On) เปรียบเสมือนสวิทช์ปิดวงจร ส่งผลให้มีกระแสไหลจากแหล่งจ่าย (12v.) ผ่านขา Collector และEmitter ของ Q1 ผ่านเข้าสู่ขั้วบวก (+) ของมอเตอร์ ผ่าน ไปยังขา Collector และEmitter ของ Q3 ทำให้มีกระแสไหลผ่านมอเตอร์ในทิศทางบวก และครบวงจร จึงทำให้มอเตอร์สามารถหมุน ในทิศทาง Forward ได้

กรณีที่ Q2 และ Q4 ทำงาน



รูปที่ 6.11 แสดงสภาวะของวงจรขณะที่ Q2 และ Q4 ทำงาน

เมื่อมีการจ่ายแรงดัน 12 V. เข้าที่จุด B ทำให้มีกระแสไหลผ่าน R2 เข้าสู่ขา base ของ Q2 และมีกระแสไหลผ่าน R4 เข้าสู่ขา base ของ Q4 ทำให้ Q2 และ Q4 ทำงาน (On) เปรียบเสมือนสวิทช์ปิดวงจร ส่งผลให้มีกระแสไหลจากแหล่งจ่าย (12v.) ผ่านขา Collector และEmitter ของ Q4 ผ่านเข้าสู่ขั้วลบ (-) ของมอเตอร์ ผ่านไปยังขา Collector และEmitter ของ Q2 ทำให้มีกระแสไหลผ่านมอเตอร์ในทิศทางลบ และครบวงจร จึงทำให้มอเตอร์สามารถหมุน ในทิศทาง Reward ได้

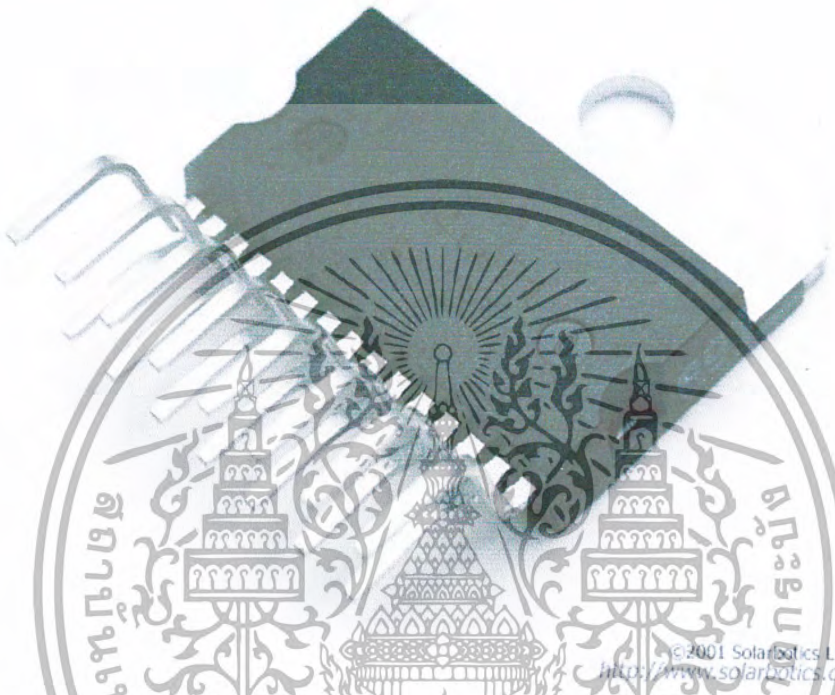
<u>ข้อควรระวัง</u>

อย่าทำให้ทรานซิสเตอร์ทั้ง 4 ตัว ทำงานพร้อมกันอย่างเด็ดขาด เพราะจะทำให้เกิดการลัดวงจร

ถึงแม้ว่าการสร้างวงจร H-Bridge switching จาก Transistor นั้นจะไม่มีปัญหา การรบกวนจากอำนาจสนามแม่เหล็ก และยังสามารถตอบสนองการทำงานได้เร็วมาก (เร็วกว่ารีเลย์แสนเท่า) แต่ก็ยังมีอุปกรณ์หลายตัว ทำให้วงจรมีขนาดใหญ่ และวุ่นวาย

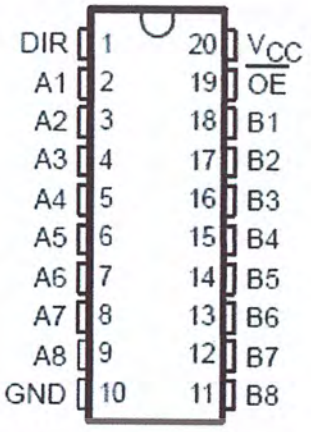
การนำ L298 มาใช้เป็น IC drive motor

การนำ L298 ไปใช้งานนั้น ไม่มีอะไรยุ่งยาก ไม่จำเป็นต้องต่ออุปกรณ์มากมาย เหมือนตอนใช้ทรานซิสเตอร์ ลำพังเพียงแค่ตัวมันเองก็สามารถทำงานได้



รูปที่ 6.12 แสดงรูป IC drive motor L298

L298 สามารถให้กระแสได้สูงสุดถึง 4A ควบคุมมอเตอร์ได้ 2 ตัว โดยที่ใช้กับแรงดันไฟฟ้าสูงสุดได้ 46 V ทนความร้อนได้ดี จะใช้มาตรฐาน TTL เป็น ลอจิกอินพุท แต่ต้องใช้อุปกรณ์ที่ป้องกัน back-EMF ในที่นี้เราใช้ ICเบอร์ 74LS245 เป็น บัฟเฟอร์ เพื่อป้องกัน back-EMF และใช้ drive อุปกรณ์ เช่น relays, solenoids และ DC motor



รูปที่ 6.13 แสดงภาพ IC 74LS245 ที่ใช้ป้องกันการเกิด back-EMF

IC 74LS245 เป็น IC ที่ใช้ป้องกันการเกิด back-EMF ในมอเตอร์ โดยที่ IC 74LS245 ใช้แรงดัน V_{CC} อยู่ในช่วง 4.75 to 5.25 V. มีดีเลย์จากขาต่อขา 8 ns ใช้สัญญาณอะซิงโครนัส และเป็นการเชื่อมต่อแบบ 2 ทิศทางระหว่าง Data Bus อุปกรณ์นี้ส่ง Data จาก A ไป B หรือจาก B ไป A ก็ได้ โดยอาศัยลอจิก ที่ขา DIR ส่วนขา OE สามารถ disable สัญญาณได้ขึ้นกะลอจิกดังตารางข้างล่างนี้

Inputs		Operation
OE	DIR	
L	L	B data to A bus
L	H	A data to B bus
H	X	Isolation

ตารางที่ 6.1 แสดงสถานการณ์ทำงานของ 74LS245 ขึ้นกับ ขา OE และ DIR

ส่วนมอเตอร์จะอยู่ที่ภาครับ โดยที่จะรับคำสั่งมาจาก PIC16F628 โดยที่มี 74LS245 เป็นตัวเชื่อมระหว่าง PIC และมอเตอร์ เพื่อป้องกัน back-EMF ซึ่งอาจทำให้ PIC เกิดความเสียหาย

DCมอเตอร์ที่ควบคุมล้อ มี 2 ตัว แบ่งควบคุมซ้ายและ ขวา โดยที่แต่ละตัวจะใช้ 2 บิตในการควบคุม และมีขดลวดโซลินอยทำหน้าที่เป็นตัวยิงลูกบอลโดยใช้ 2 บิต ควบคุมเช่นกัน

บทที่ 7
อิมเมจและการประยุกต์ใช้งาน

การเปลี่ยนแปลงรูปแบบทางเลขาคณิต

การเปลี่ยนแปลงรูปแบบทางเลขาคณิต คือ กระบวนการเปลี่ยนช่องว่างท่ามกลางวัตถุในรูป หรือเปลี่ยนรูปทรงทางเลขาคณิตของวัตถุในรูปภาพ

ขั้นตอนการคิด

เป็นการจำกัดความของรูปภาพที่เป็น f(x,y) ที่ (x,y) คือจุด (0,0) และ f(x,y) คือระดับสีเทาที่จุด (0,0) เพื่อที่จะแสดง การการเปลี่ยนแปลงรูปแบบทางเลขาคณิต 2 ข้อกำหนดที่จำเป็นคือ

1. การเปลี่ยนแปลงช่องว่าง
2. การสอดแทรกระดับสีเทา

การเปลี่ยนแปลงเชิงเส้นแบบมีช่องว่าง

คำจำกัดความ

การแมปปิ้ง F: E^2 -> E^2 คือ การแปลงเชิงเส้นของ E^2 ในตัวของมัน สำหรับทุกเวคเตอร์ X และ Y และสำหรับทุกสเกลล่า C และ D

$$F(cX+dY) = cF(x)+dF(Y)$$

ถ้า F และ G คือ เส้นตรง 2 เส้นที่ได้รับการเปลี่ยนแปลง

$$GF(x) = G(F(x))$$

ชั้นย่อยๆ

การหมุนในแบบตั้งฉากและการแปรความ

ความยาว มุม และ เส้นขนาน ที่ถูกกำหนดไว้

ความคล้ายคลึงของการหมุน การสเกล และการแปรความ

มุมและเส้นขนานที่ถูกกำหนดไว้ การเปลี่ยนแปลงแบบเลียนแบบ เช่น การเฉือน การหมุน การสเกล และ การแปรความ

เส้นขนานที่ถูกกำหนดไว้

การแปรความคือ ย้ายตำแหน่งของวัตถุจากตำแหน่งหนึ่งไปยังอีกตำแหน่งหนึ่งโดยมีลักษณะการเคลื่อนที่เป็นแนวเส้นตรง และไม่ทำให้รูปร่างของวัตถุมีการเปลี่ยนแปลงไปโดยจุด (x,y) เมื่อต้องการย้ายไปยัง (x',y') ทำได้โดยการบวกค่าระยะทางที่ต้องการย้ายไปตามสมการ

$$x' = x + T_x$$

$$y' = y + T_y$$

เมื่อนำมาเขียนให้อยู่ในรูปของเมตริกซ์จะได้ดังนี้

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} Tx \\ Ty \end{bmatrix}$$

การสเกลลิ่ง คือการเปลี่ยนขนาดของวัตถุ ทำได้โดยการคูณ scaling factor เข้าไปในสมการ

$$x' = s_x x$$

$$y' = s_y y$$

นำมาเขียนในรูปเมตริกซ์จะได้ดังนี้

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

โดยเมื่อ S_x หรือ S_y มีค่ามากกว่า 1 วัตถุจะมีขนาดใหญ่ขึ้น

S_x หรือ S_y มีค่าน้อยกว่า 1 แต่มากกว่า 0 วัตถุจะมีขนาดเล็กลง

S_x หรือ S_y มีค่าน้อยกว่า 0 วัตถุจะเกิดการสะท้อน

การหมุน

การหมุน คือ การย้ายตำแหน่งของจุด โดยมีเส้นทางของการย้ายเป็นแนววงกลมรอบจุดกำเนิดตามสมการ

$$x' = x\cos A - y\sin A$$

$$y' = x\sin A + y\cos A$$

นำมาเขียนในรูปเมตริกซ์จะได้ดังนี้

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos A & -\sin A \\ \sin A & \cos A \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

การเฉือน

การเฉือน คือ การทำให้วัตถุมีรูปร่างที่ผิดเพี้ยนไปจากเดิม ในการคำนวณหาตำแหน่งใหม่ของจุดทำได้ดังสมการ

1) เมื่อทำการ shear ในทิศทาง x

$$x' = x + Sh_x * y$$

$$y' = y$$

2) เมื่อทำการ shear ในทิศทาง y

$$y' = y + Sh_y * x$$

$$x' = x$$

เมื่อนำมาเขียนให้อยู่ในรูปของเมตริกซ์จะได้ดังนี้

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & Shx \\ Shy & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

เมื่อต้องการทำ transformation หลายๆรูปแบบ เราสามารถคำนวณหาตำแหน่งของจุดใหม่ได้ดังนี้

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos A & -\sin A \\ \sin A & \cos A \end{bmatrix} \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix} \begin{bmatrix} 1 & Shx \\ Shy & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} Tx \\ Ty \end{bmatrix}$$

พิจาณาจากสมการจะเห็นว่า translation matrix ไม่สามารถนำเข้าไปรวมกับเมตริกซ์อื่นได้ จึงได้มีการใช้ homogeneous coordinates โดย homogeneous coordinates จะต้องใช้ค่า 3 ค่า คือ (x_h, y_h, z) โดย $x_h = wh$ และ $y_h = wy$ โดยในระนาบ 2 มิตินั้นให้ w=1 และสามารถเขียน transformation matrix ต่างๆ ได้ดังนี้

Point Matrix

$$P_{x,y} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translation Matrix

$$T_{x,y} = \begin{bmatrix} 1 & 0 & Tx \\ 0 & 1 & Ty \\ 0 & 0 & 1 \end{bmatrix}$$

Scaling

$$S_{x,y} = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shear Matrix

$$Sh_{x,y} = \begin{bmatrix} 1 & Shx & 0 \\ Shy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation Matrix

$$R_0 = \begin{bmatrix} \cos A & -\sin A & 0 \\ \sin A & \cos A & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Forward Mapping

Forward Mapping คือการคำนวณหาพิกัดของจุดในภาพใหม่โดยคำนวณจากพิกัดของจุดในภาพเดิม

Backward Mapping

Backward Mapping คือการคำนวณหาพิกัดของจุดในภาพเก่าจากพิกัดของจุดในภาพใหม่ที่พิกัดที่ต้องการ โดยในการคำนวณจะใช้ Inverse ของ Transformation Matrix

Forward Mapping จะทำให้เกิดช่องว่างขึ้นในภาพใหม่เนื่องจากค่าพิกัดที่คำนวณได้ไม่ไปตกยังจุดนั้นๆ วิธีแก้คือใช้ Backward Mapping ซึ่งจะทำการกำหนดจุดในภาพใหม่ลงไปก่อนจากนั้นจึงทำการคำนวณหาจุดที่กำหนดจากภาพเก่า

2D Transformation Matrix

ภาพรวมของการ Transformation คือ การนำค่าแต่ละพิกัด (x,y) ไปคูณกับ Matrix ที่ใช้ในการ Transformation แล้วนำค่าพิกัด (x,y) ที่ได้จากการคูณไป mapping

1. Translation

เป็นการย้ายรูปภาพไปยังจุดต่างๆบนระนาบ x,y โดยใช้ Translation Matrix คูณกับ แต่ละพิกัด (x,y) ของรูปภาพ

Translation Matrix

$$T(d_x,d_y) = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \qquad T^{-1}(d_x,d_y) = \begin{bmatrix} 1 & 0 & -dx \\ 0 & 1 & -dy \\ 0 & 0 & 1 \end{bmatrix}$$

โดย d_x คือค่าที่ต้องการย้ายในแนวแกน x

d_y คือค่าที่ต้องการย้ายในแนวแกน y

2. Scaling

เป็นการขยายภาพ โดยการ Translate รูปภาพไปยัง Origin ก่อนแล้วทำการ Scaling โดยใช้ Scaling Matrix แล้ว Translate กลับมาที่เดิม

Scaling Matrix

$$S(s_x,s_y) = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad S^{-1}(s_x,s_y) = \begin{bmatrix} 1/Sx & 0 & 0 \\ 0 & 1/Sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

โดย S_x คือ ค่าที่ต้องการขยายในแนวแกน x

S_y คือ ค่าที่ต้องการขยายในแนวแกน y

3. Rotation

เป็นการหมุนภาพ โดยการ Translate รูปไปยัง Origin แล้วทำการ Rotate โดยใช้ Rotation Matrix แล้ว Translate กลับมาที่เดิม

Rotation Matrix

$$R(q) = \begin{bmatrix} \cos A & -\sin A & 0 \\ \sin A & \cos A & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad R^{-1}(q) = \begin{bmatrix} \cos A & \sin A & 0 \\ -\sin A & \cos A & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

โดยที่ A คือมุมที่ต้องการจะหมุน

4. Shear

เป็นการแปลงภาพแบบเฉียง คือ การทำภาพในแนวนอน หรือแนวตั้ง หรือทั้ง 2 แนว โดย Translate รูปภาพไปยัง Origin แล้วทำการ Shear โดยใช้ Shear Matrix แล้ว Translate กลับมาที่เดิม Shear Matrix

$$Sh(q) = \begin{bmatrix} 1 & Shx & 0 \\ Shy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad Sh^{-1}(q) = \frac{1}{1-(Shx * Shy)} \begin{bmatrix} 1 & -Shx & 0 \\ -Shy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

โดย Sh_x คือ ค่าที่ต้องการให้ภาพเฉียงในแนวแกน x

Sh_y คือ ค่าที่ต้องการให้ภาพเฉียงในแนวแกน y

ในแต่ละ Matrix ที่ใช้ในการ Transformation ที่กำหนดให้นี้จะใช้กระบวนการคูณทั้งหมดอย่างเช่น ถ้าอยากทำ Rotation จะได้ Matrix ที่ใช้ในการ Rotation ดังนี้

$$T = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos A & -\sin A & 0 \\ \sin A & \cos A & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -dx \\ 0 & 1 & -dy \\ 0 & 0 & 1 \end{bmatrix}$$

Matrix T เป็น Matrix ที่ใช้ในการ Rotation กระบวนการจะดูจากหลังมาหน้า โดย Translate รูปภาพไปยัง Origin ทำการหมุนภาพ แล้ว Translate กลับมาที่เดิม

การ Inverse Transformation จะได้ T^{-1} โดย T^{-1} เกิดจากการนำ Matrix Inverse ของ Matrix Transformation มาคูณกันนั่นเอง

5. การทำเทรสโชว์ (Image thresholding)

การทำเทรสโชว์จะได้เอาท์พุทของภาพออกมาเป็นไบนารี่ Image thresholding เป็นการเปลี่ยนภาพจาก Intensity ให้กลายเป็น Binary Image โดยกระเลือกค่า Threshold มาค่าหนึ่งหรือเป็นช่วงซึ่งเป็นตัวแบ่งว่า Pixel ใดมีค่าเป็น '1' หรือ '0' ดังนั้นหากเราใช้จุดสีสามสีคือ แดง เขียว น้ำเงิน เราจะสามารถทำการแบ่งภาพ RGB Image ออกเป็น Intensity Image ได้สามภาพ คือ Red, Green, Blue Channel Intensity Image แล้วนำมาทำการ Thresholding เพื่อแยกแยะวัตถุ

6. การหักลบภาพและตรวจการเปลี่ยนแปลง (Image Subtraction and Change Detection)

ในการประยุกต์ใช้งานในการประมวลผลภาพหลายอย่าง เราอาจต้องการที่จะเปรียบเทียบภาพที่มีความสลับซับซ้อน 2 ภาพ วิธีที่ง่ายแต่มีประสิทธิภาพคือ การเรียง(Align) ทั้งสองเข้าด้วยกัน แล้วทำการหารหาผลต่างจากนั้นก็ทำการปรับคุณภาพผลต่าง ตัวอย่างเช่น อุปกรณ์ที่หายไปจากบอร์ดวงจรสามารถตรวจหาได้โดยภาพของบอร์ดดังกล่าวกับภาพบอร์ดที่มีอุปกรณ์วางถูกต้อง การประยุกต์ใช้งานอีกอย่างพบใช้กรณีที่เราต้องการทำภาพของหลอดเลือดและเส้นเลือดในร่างกาย ซึ่งทำได้โดยการฉีดสารทึบแสงเข้าไปในหลอดเลือดแล้วทำการถ่ายภาพเอ็กเรย์ จากนั้นทำการลบภาพออกจากภาพเอ็กเรย์ของหลอดเลือดก่อนฉีดสารทึบแสงผลต่างของภาพทั้งสองจะแสดงทางเดินของเลือดอย่างชัดเจน การประยุกต์ใช้งานอื่นๆคือการตรวจจับความแตกต่างในระบบรักษาความปลอดภัย หรือการตรวจแผ่นปรินท์วงจรโดยอัตโนมัติ และอื่นๆ

รูปที่ 7.1 แสดงการหักลบภาพ

7. การแยก RGB

รูปที่ 7.2 แสดงสีภายใน 1 Pixel

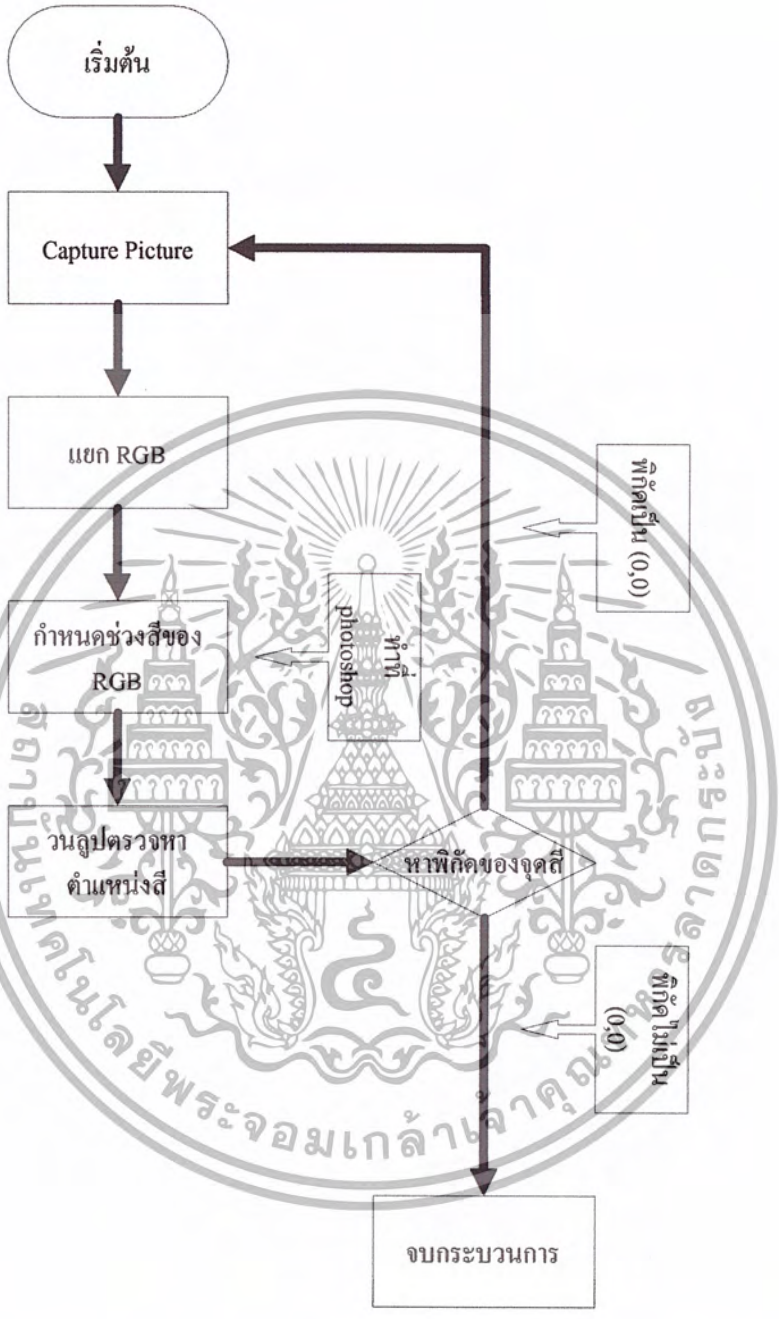
ใน 1 Pixel จะประกอบไปด้วยค่าสี 3 สี รวมกันเป็น 1 Pixel ซึ่งเราสามารถแยกค่าแต่ละสีออกมาจาก Pixel ได้ดังนี้

R = ค่าสี Mod 256

G = (ค่าสี div 256) Mod 256

B = 0 div 65536

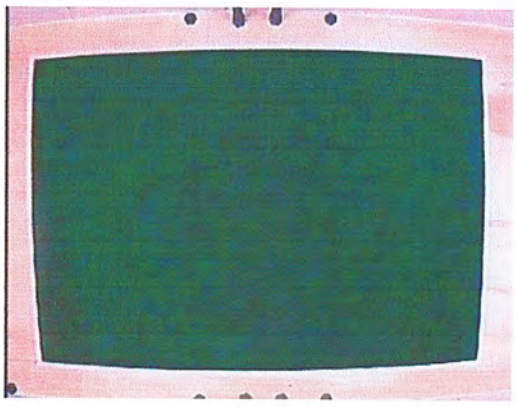
8. ขั้นตอนการหาตำแหน่งของจุดสีของบอลและหุ่นยนต์



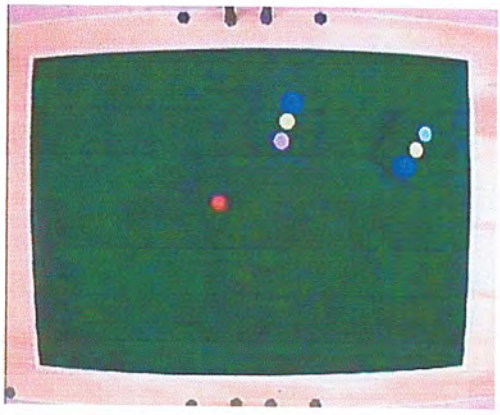
รูปที่ 7.3 แสดง flow chart การเขียนโปรแกรมหาตำแหน่งสี ด้วยวิธีที่ 1

วิธีที่ 1

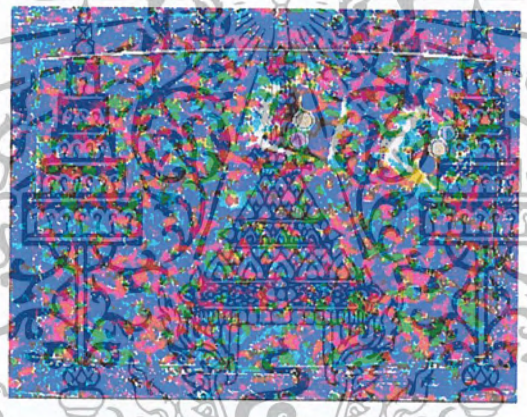
เป็นการแยกสีโดยนำจุดสีมแยกเป็น RGB แล้วเก็บไว้ใน Array 2 มิติ เมื่อทำการแยกสีเสร็จแล้ว ก็นำค่าต่ละสีที่ได้ไปเปรียบเทียบกับค่าที่กำหนดไว้ ถ้ามากกว่าค่าที่กำหนดให้สีนั้นเป็น 255 ถ้าน้อยกว่าค่าที่กำหนดให้สีนั้นเป็น 0



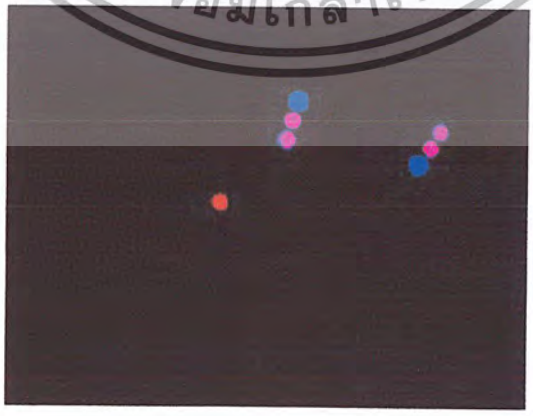
รูปที่ 7.4 แสดงภาพสนาม



รูปที่ 7.5 แสดงภาพสนาม หุ่นยนต์ และ ลูกบอล



รูปที่ 7.6 เกิดจากการนำรูป 7.5 ลบด้วยรูปที่ 7.4 โดยไม่แยก RGB



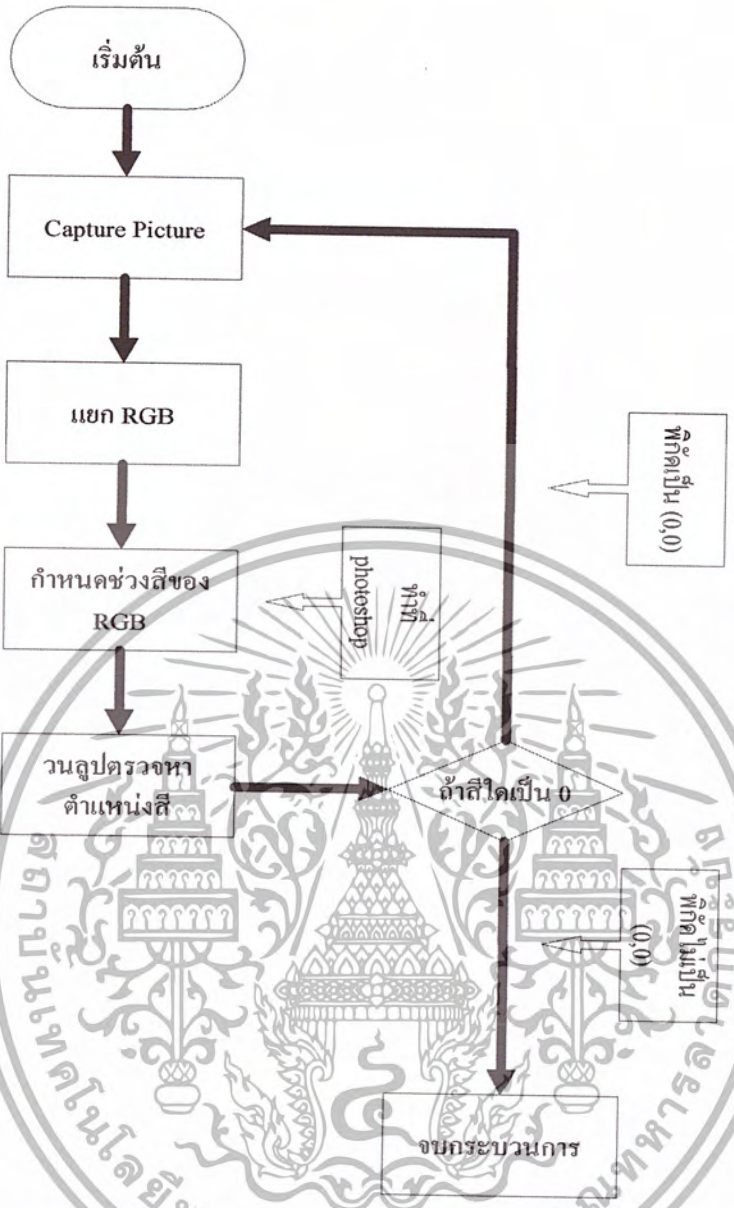
รูปที่ 7.7 แสดงภาพที่เกิดจากการทำเทรสโชว์แล้ว

เมื่อได้ภาพที่ทำการเปรียบเทียบมาแล้ว นำหน้ากากที่มีสีตามที่เรากำหนดขนาดเท่ากับหรือน้อยกว่าจุดสี วนทุกจุดของภาพเพื่อตรวจสอบสีที่กำหนด

	(i,j-1)	
(i-1,j)	(i,j)	(i+1,j)
	(i,j+1)	

รูปที่ 7.8 แสดงภาพหน้ากากขนาด 3*3 pixels

รูปที่ 7.9 ภาพแสดงการเคลื่อนที่ของหน้ากาก

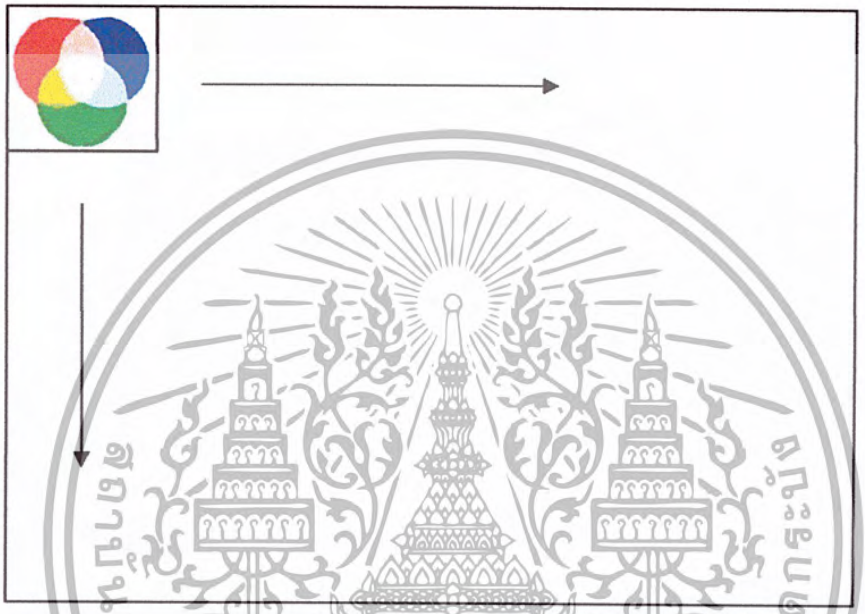


รูปที่ 7.10 แสดง flow chart การเขียนโปรแกรมหาตำแหน่งสี ด้วยวิธีที่ 2

วิธีที่ 2

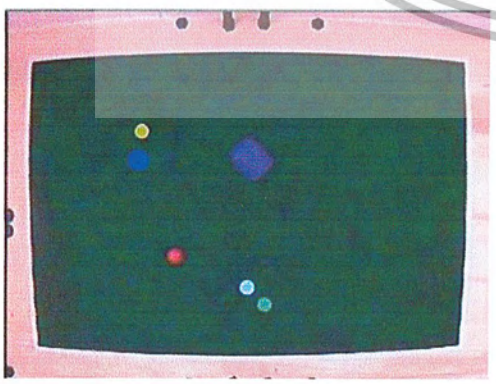
เป็นการแยกสีโดยนำจุดสีมาแยกเป็น RGB แล้วเก็บไว้ใน Array 2 มิติ เช่นเดียวกับวิธีที่ 1 แล้วเราก็จะกำหนดช่วงสีของ RGB ที่เราต้องการทราบตำแหน่งแล้วทำการวนลูปหาทั้งภาพดังภาพที่ 7.10

รูปที่ 7.11 แสดงส่วนประกอบของสีใน 1 pixel

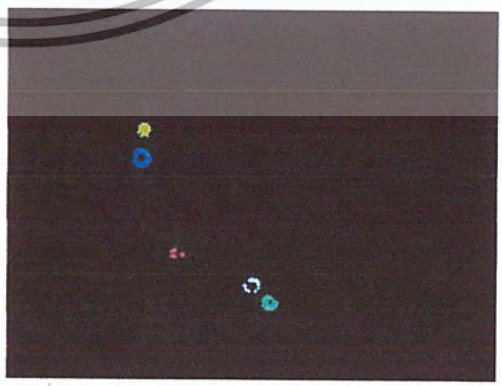


รูปที่ 7.12 แสดงการตรวจหาจุดสีบนภาพ

แล้วนำค่าที่ได้ทั้งหมดเก็บไว้ใน Array แล้วทำการสุ่มค่าที่อยู่ใน Array ขึ้นมา



รูปที่ 7.13 แสดงภาพตัวอย่างในการหาจุดสี

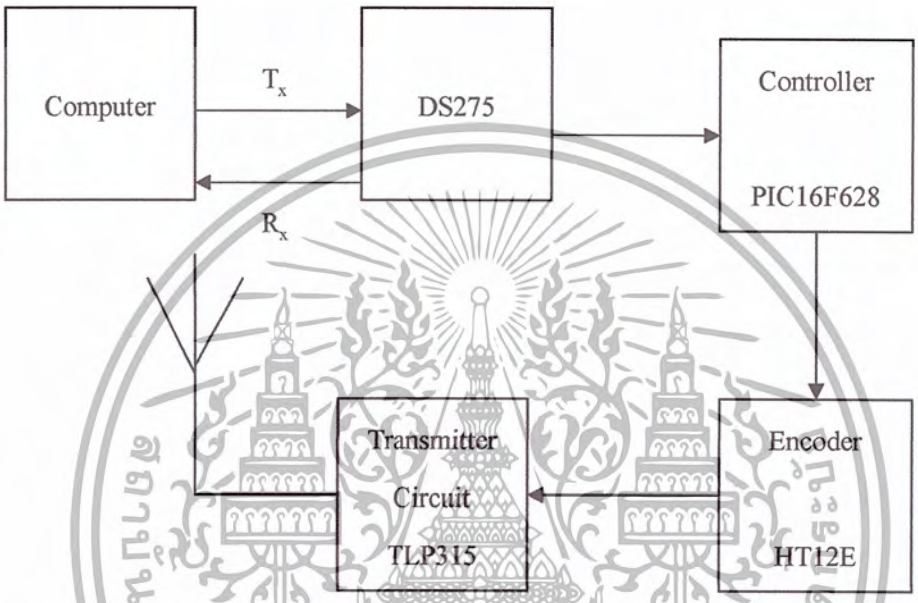


รูปที่ 7.14 แสดงผลของการหาจุดสี

บทที่ 8
การประยุกต์ใช้งานวงจร

การประยุกต์ใช้งาน

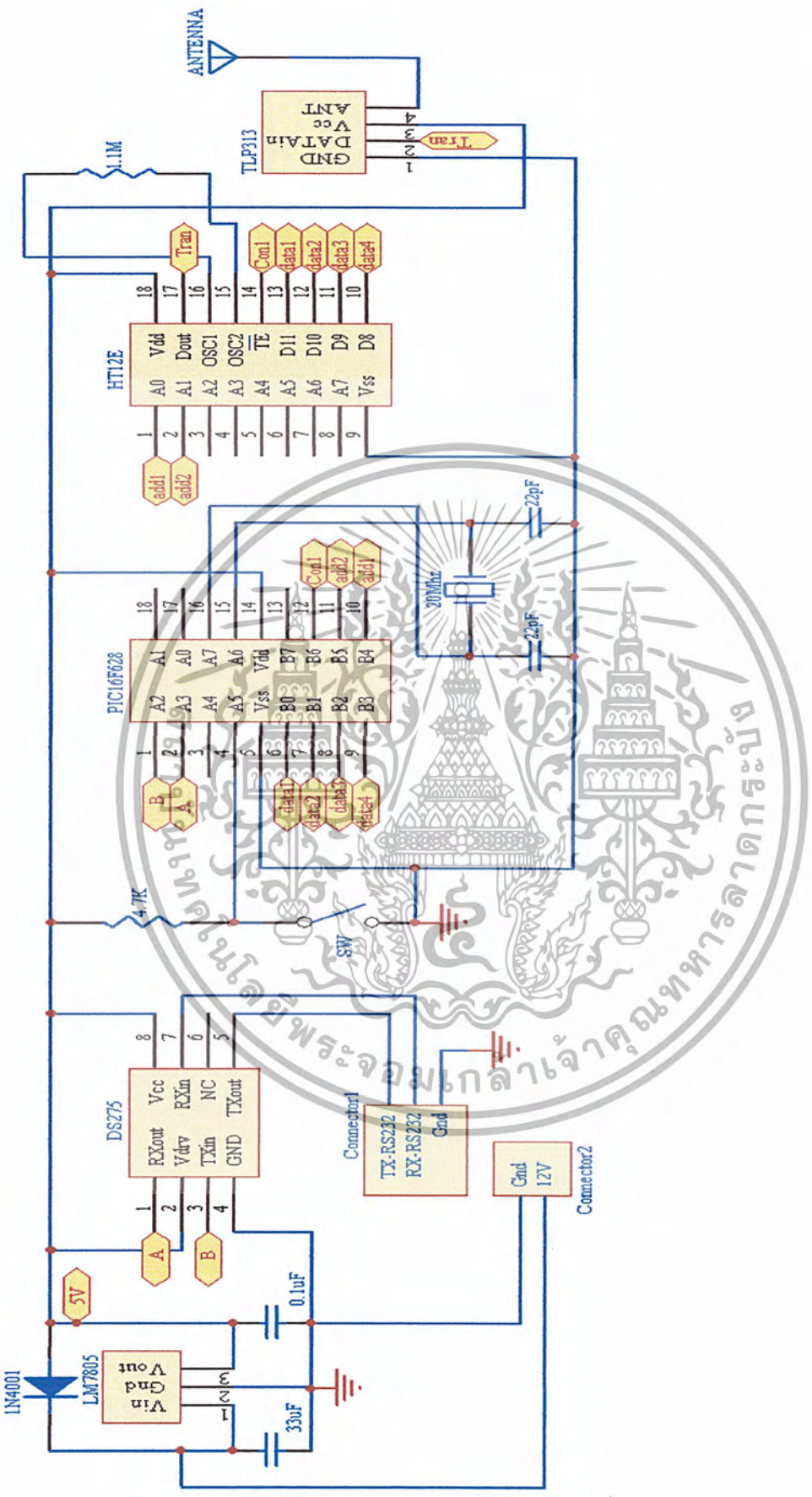
Block Diagram



รูปที่ 8.1 block diagram ของวงจรภาคส่ง

อธิบายวงจรในรูปที่ 8.1

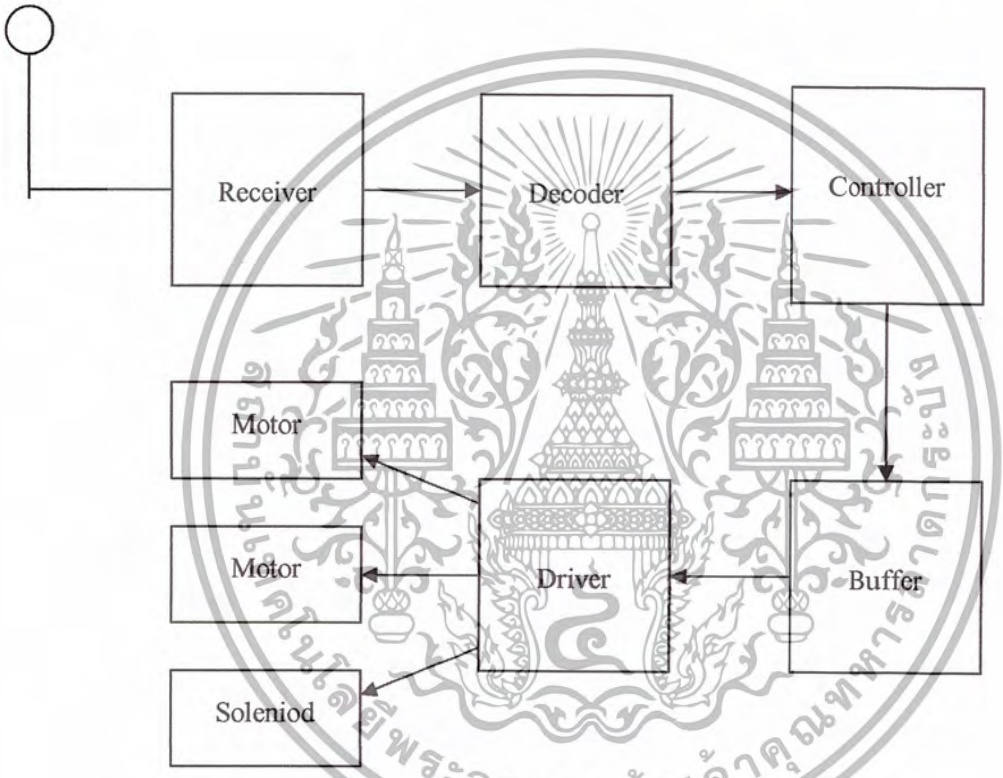
คอมพิวเตอร์จะส่งสัญญาณออกมาผ่านทางพอร์ตอนุกรมซึ่งใช้ DS275 เป็น IC ที่นำมาต่อเพื่อแปลงสัญญาณและแรงดัน โดยจะเปลี่ยนจากสัญญาณ RS232 ที่มีแรงดันอยู่ระหว่าง +/-3 ถึง +/-15 V มาเป็นสัญญาณ TTL ซึ่งมีแรงดันอยู่ระหว่าง 0-5 V จากนั้นสัญญาณที่เป็น TTL จะไปเข้า PIC16F628 ซึ่งจะทำการตรวจสอบเงื่อนไขจากสัญญาณที่รับมา ถ้าตรงกับเงื่อนไขใด PIC ก็จะสั่งให้ทำเงื่อนไขนั้นๆ โดยที่ส่งข้อมูลไปให้กับ Encoder HT12E เพื่อทำการเลือกส่งข้อมูลให้กับหุ่นยนต์ตัวที่ต้องการ โดยผ่านทาง TLP315



รูปที่ 8.2 แสดงภาพของวงจรภาคส่ง

อธิบายวงจรรูปที่ 8.2

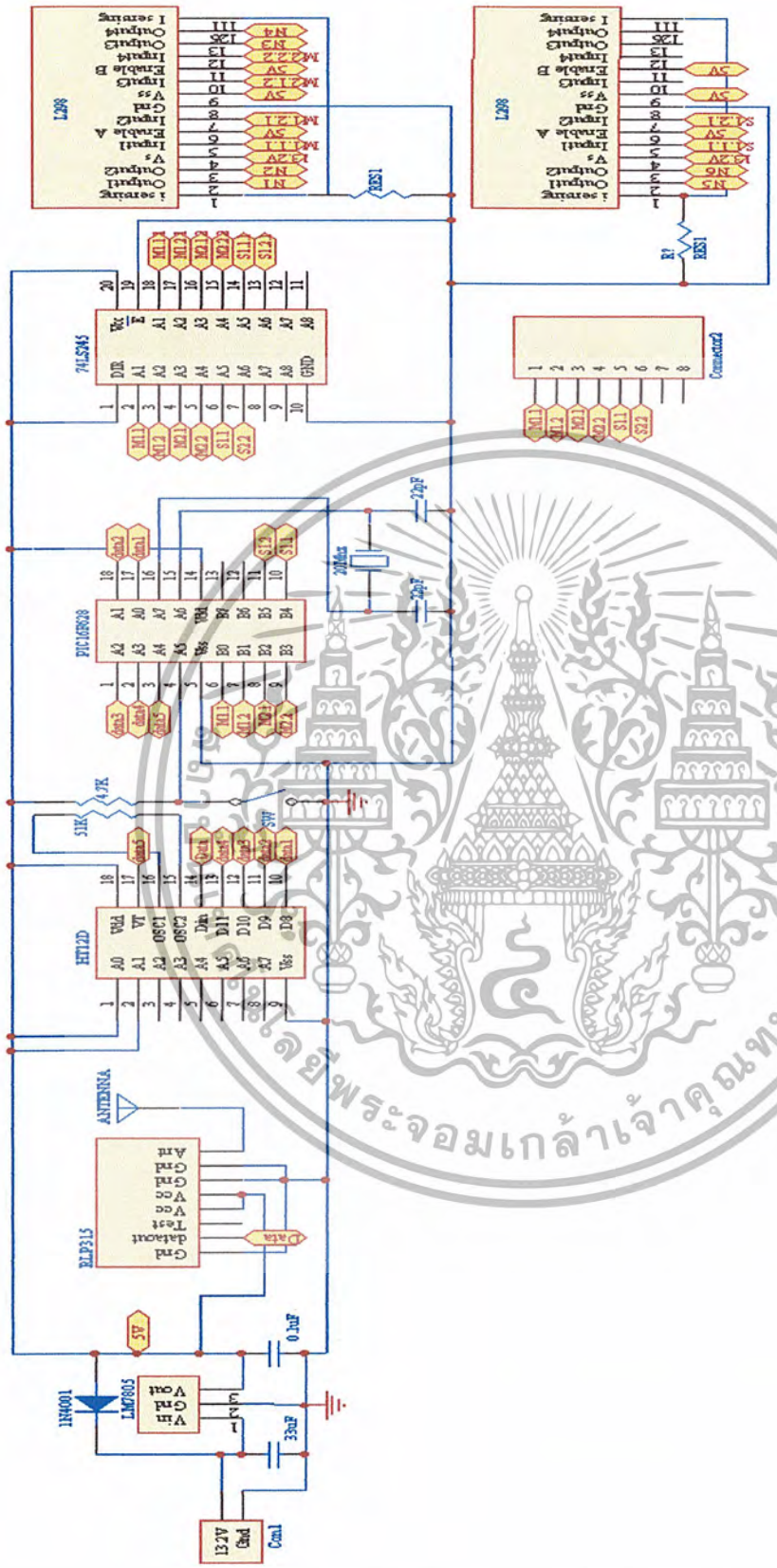
เมื่อคอมพิวเตอร์ส่งข้อมูลมาทางขา Tx ซึ่งส่งมาแบบ RS232 ซึ่งจะถูก DS275 เปลี่ยนแรงดัน Voltage ของสัญญาณให้เป็นสัญญาณ TTL แล้วจึงส่งไปที่ขา A2 ของ PIC16F628 ซึ่งกำหนดให้เป็นขา Rx ของ PIC จากนั้น PIC จะทำการประมวลผลข้อมูลที่ส่งมา แล้วก็จะทำการ Set ค่าของ B0-B6 ตามโปรแกรมโดยที่ D0-D3 เป็น Data ส่วน B4,B5 เป็น Address และขา B6 ใช้เป็นขา TE เมื่อ HT12E ได้รับ Data จาก PIC แล้วก็จะทำการ Encode ข้อมูลกับ Address แล้วส่งออกขา Dout ของ HT12E จากนั้นข้อมูลจะถูกส่งมาที่ TLP315 ซึ่งเป็นตัวส่งข้อมูลแบบ RF



รูปที่ 8.3 block diagram ของวงจรภาครับ

อธิบายวงจร รูปที่ 8.3

เมื่อมีข้อมูลมาถึง RLP315 แล้วก็จะถูกส่งไปให้ HT12D ทำการถอดรหัสออกมา ถ้ามี Address ตรงกับตัวส่งก็จะสามารถถอดรหัสได้ และเมื่อสามารถถอดรหัสจาก HT12D ได้แล้ว data จะถูกส่งไปที PIC PIC จะประมวลผลตามโปรแกรมที่เขียนไว้ในตัวมัน แล้วก็จะทำการส่ง data ที่สร้างขึ้นมาใหม่ไปที่ L298 เพื่อทำการขับมอเตอร์

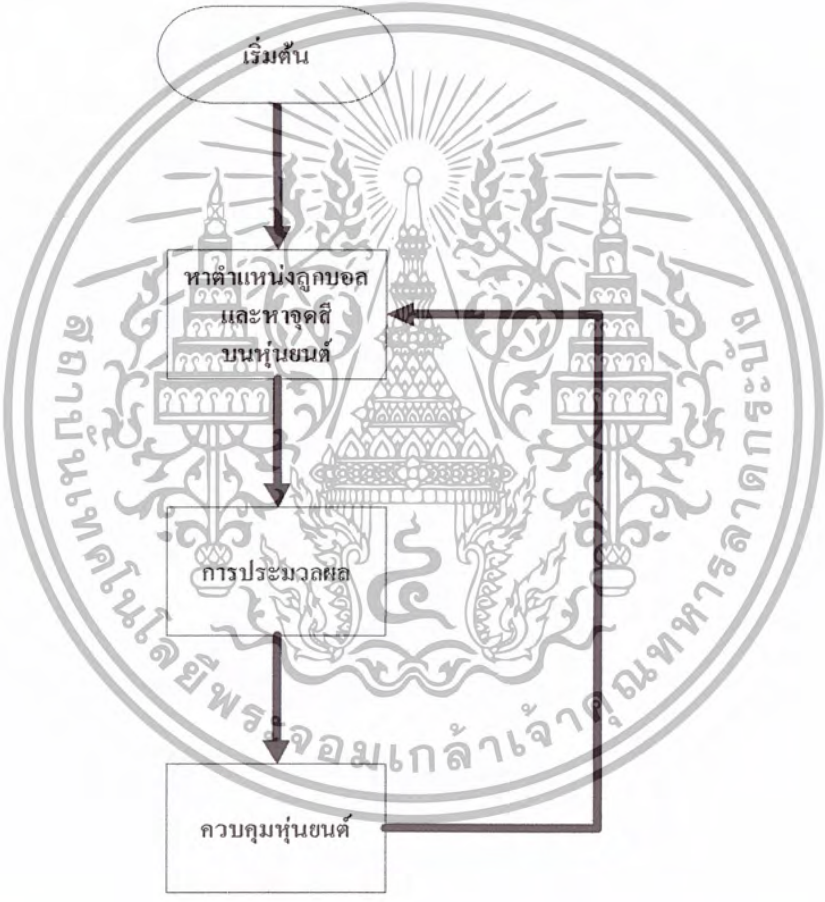


รูปที่ 8.4 วงจรภาครับ

อธิบายวงจรรูปที่ 8.4

เมื่อมีข้อมูลมาถึง RLP315 แล้วข้อมูลก็จะถูกส่งออกไปทางขา Digital Out ของ RLP315 เข้าสู่ขา DIN ของ HT12D เพื่อทำการ Decode ข้อมูลและนำข้อมูลที่ได้ส่งเข้าสู่ PIC16F628 แล้วทำการประมวลผลที่ได้แล้วส่งข้อมูลที่ได้ไปให้ L298 โดยใช้ B0-B4 ในการส่งข้อมูลเพื่อควบคุมมอเตอร์ โดยส่งข้อมูลเข้าทาง Input1 ถึง Input4 ของ L298 ส่วน B5,B6 ใช้ในการ ควบคุมโซลินอย

การประยุกต์ใช้งานเดลไฟในการประมวลผล Image ในโครงงาน

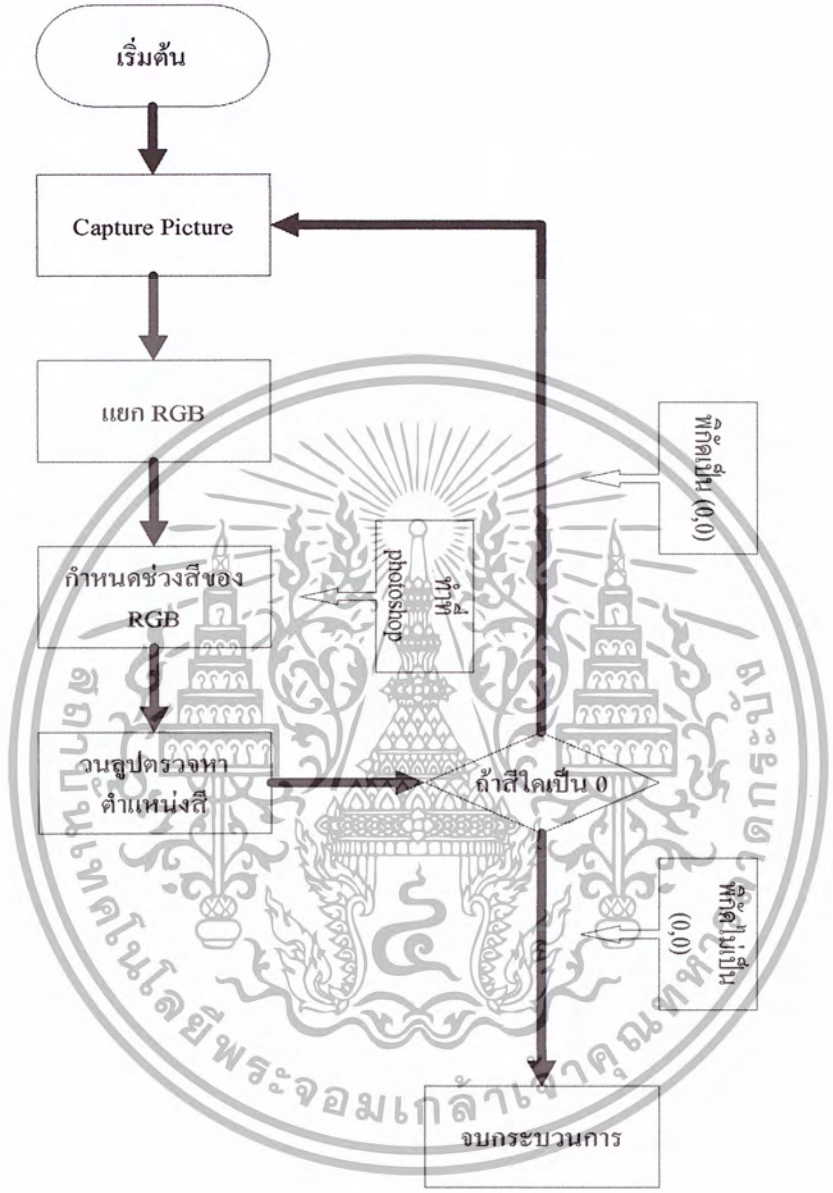


รูปที่ 8.5 แสดง Main Flowchart

คำอธิบายรูปที่ 8.5

เริ่มต้นโปรแกรมจะใช้ Image Processing ในการหาตำแหน่งของจุดสีบนหุ่นยนต์และลูกบอล แล้วนำค่าตำแหน่งที่ได้ไปประมวลผลและส่งผลที่ได้ไปควบคุมหุ่นยนต์โดยผ่าน RF และจะทำการหาตำแหน่งลูกบอลและหุ่นยนต์ซ้ำๆต่อไป ตาม Flow Chart ในรูปที่ 6.4

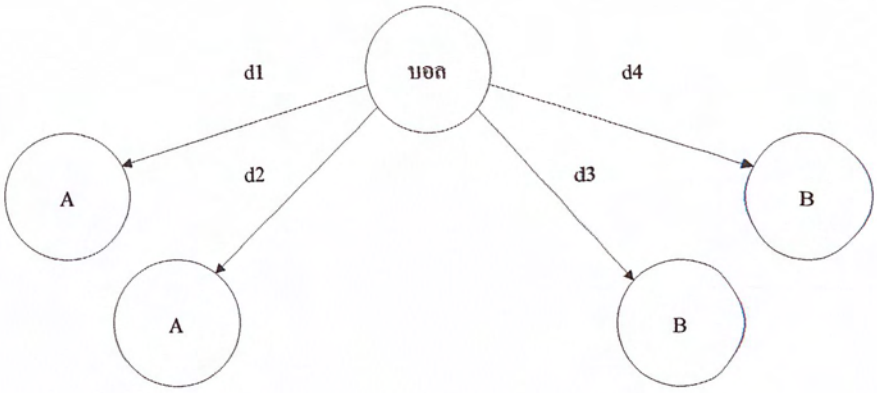
และสำหรับ 2 วิธีการกาค่าจุดเราใช้เป็นวิธีที่ 2 เนื่องจากสามารถประมวลผลได้เร็วกว่าแบบแรก เนื่องจากมีความซับซ้อนของโปรแกรมน้อยกว่า โดยมีหลักการทำงานดังต่อไปนี้



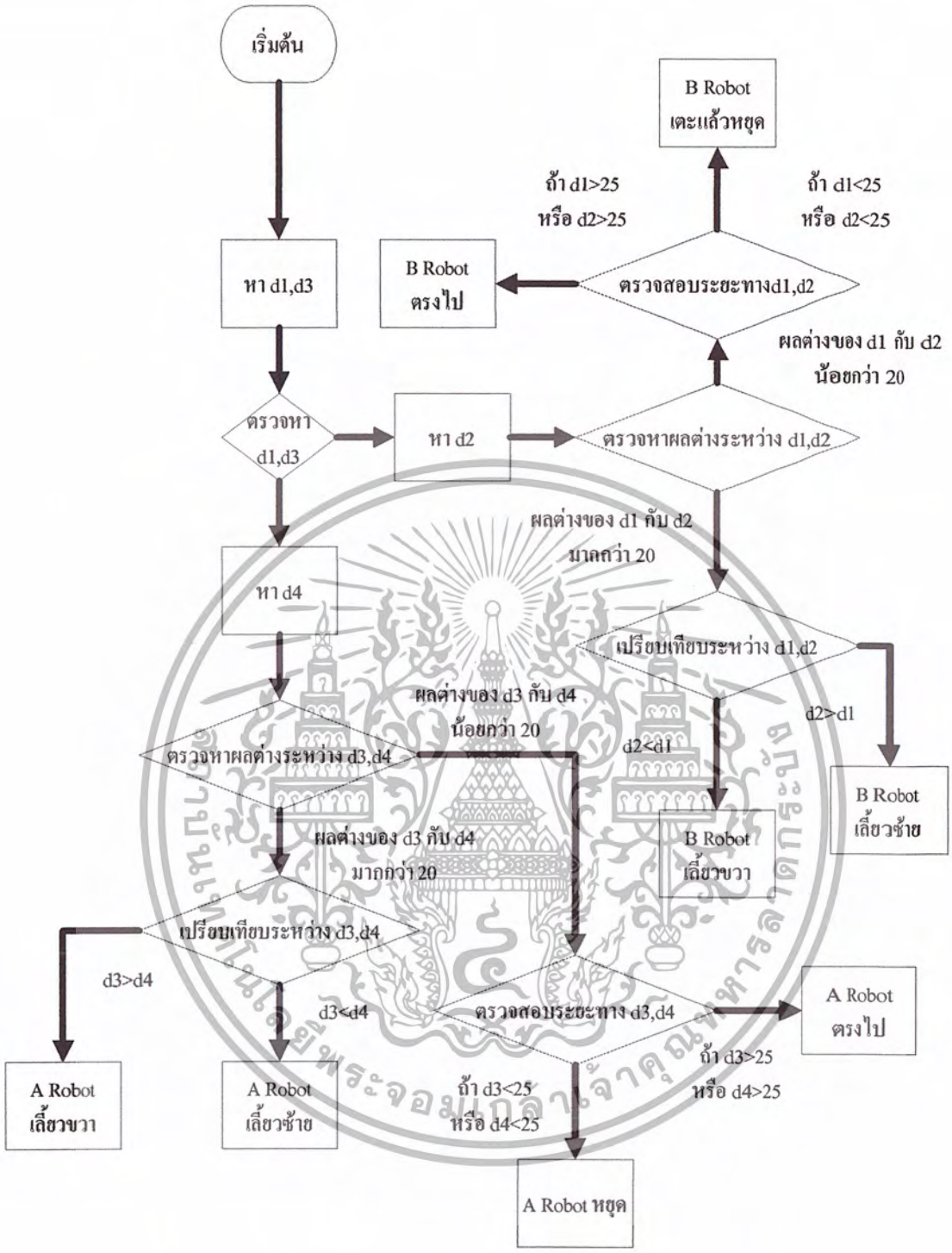
รูปที่ 8.6 แสดง flow chart การเขียนโปรแกรมหาตำแหน่งสี ด้วยวิธีที่ 2

วิธีที่ 2

เป็นการแยกสีโดยนำจุดสีมาแยกเป็น RGB แล้วเก็บไว้ใน Array 2 มิติ เช่นเดียวกับวิธีที่ 1 แล้วเราก็จะกำหนดช่วงสีของ RGB ที่เราต้องการทราบตำแหน่งแล้วทำการวนลูปหาทั้งภาพดังภาพที่ 8.6



รูปที่ 8.7 แสดงระยะทางจุดสีบนหุ่นยนต์ กับลูกบอล



รูปที่ 8.8 flow chart แสดงการประมวลผลและควบคุมหุ่นยนต์

อธิบายรูปที่ 8.8

เริ่มต้นหาระยะทาง d1 กับ d3 เพื่อหาว่าหุ่นยนต์ตัวไหนอยู่ใกล้ลูกบอลมากกว่ากัน ถ้า d1<d3 ก็จะทำการหาระยะ d2 และทำการตรวจสอบผลต่างระหว่าง d1 กับ d2

ถ้าผลต่างน้อยกว่า 20 ให้ ตรวจสอบd1กับd2ถ้าd1หรือd2มากกว่า25ให้B Robotตรงไปแต่ถ้าd1หรือd2น้อยกว่าB Robot ก็จะทำการเตะลูกแล้วหยุด

ถ้าผลต่างมากกว่า 20 ให้ ตรวจสอบ d1 กับ d2

ถ้า d2 > d1 ให้ B Robot เลี้ยวซ้าย

ถ้า d2 < d1 ให้ B Robot เลี้ยวขวา

ถ้า d1>d3 ก็จะทำการหาระยะ d4 และทำการตรวจสอบผลต่างระหว่าง d3 กับ d4

ถ้าผลต่างน้อยกว่า 20 ให้ ตรวจสอบd3กับd4ถ้าd3หรือd4มากกว่า25ให้B Robotตรงไปแต่ถ้าd3หรือd4น้อยกว่าB Robot ก็จะทำการเตะลูกแล้วหยุด

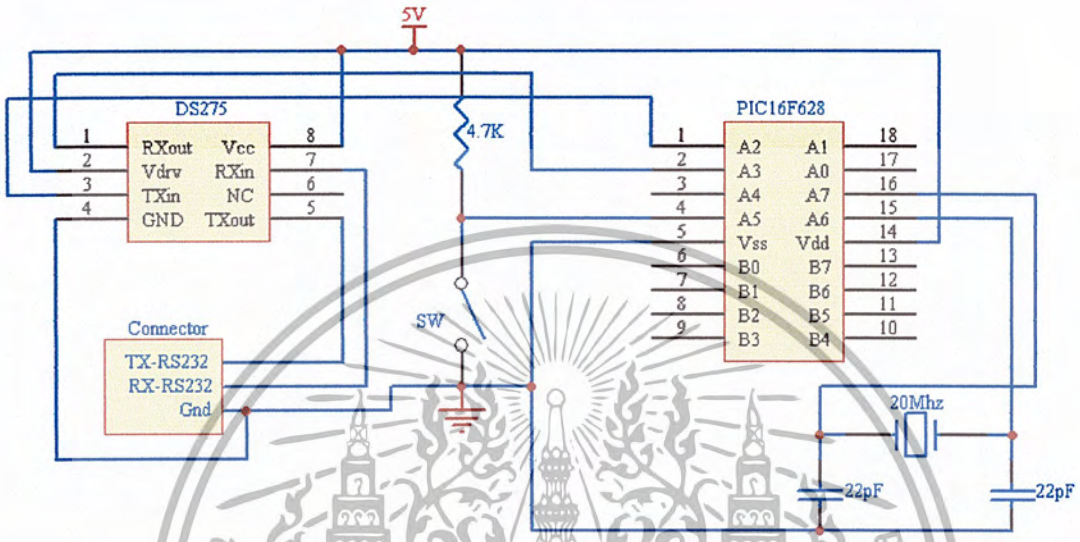
ถ้าผลต่างมากกว่า 20 ให้ ตรวจสอบ d3 กับ d4

ถ้า d3 > d4 ให้ B Robot เลี้ยวซ้าย

ถ้า d4 < d3 ให้ B Robot เลี้ยวขวา

บทที่ 9
การทดลอง

การทดลองที่ 1 Interface Serial Port with Delphi 7



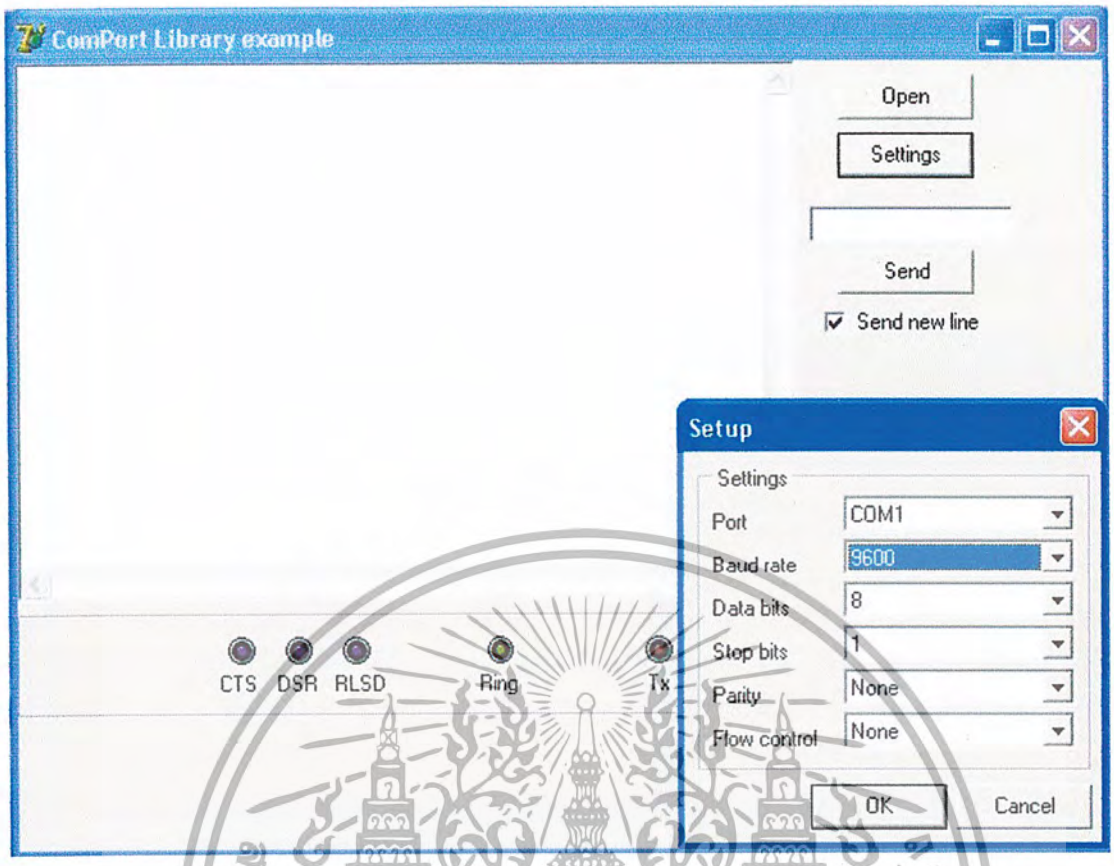
รูปที่ 9.1 ภาพการทดลองที่ 1

วัตถุประสงค์

เพื่อทดสอบการส่งผ่านข้อมูลระหว่าง Serial port กับ PIC 16F298

ขั้นตอนการทดลอง

1. ลงโปรแกรม Delphi 7 พร้อมติดตั้ง Component ติดต่อกับ Serial Port
2. ทำการเขียนโปรแกรม Delphi เพื่อติดต่อกับ Serial Port



รูปที่ 9.2 ภาพแสดงโปรแกรมแสดงการตั้งค่า baud rate

3. ต่อวงจรดังภาพที่ 9.1

4. ทำการเขียนโปรแกรมของ PIC ให้สามารถติดต่อกับ Serial Port โดยป้อน Source Code ดังนี้

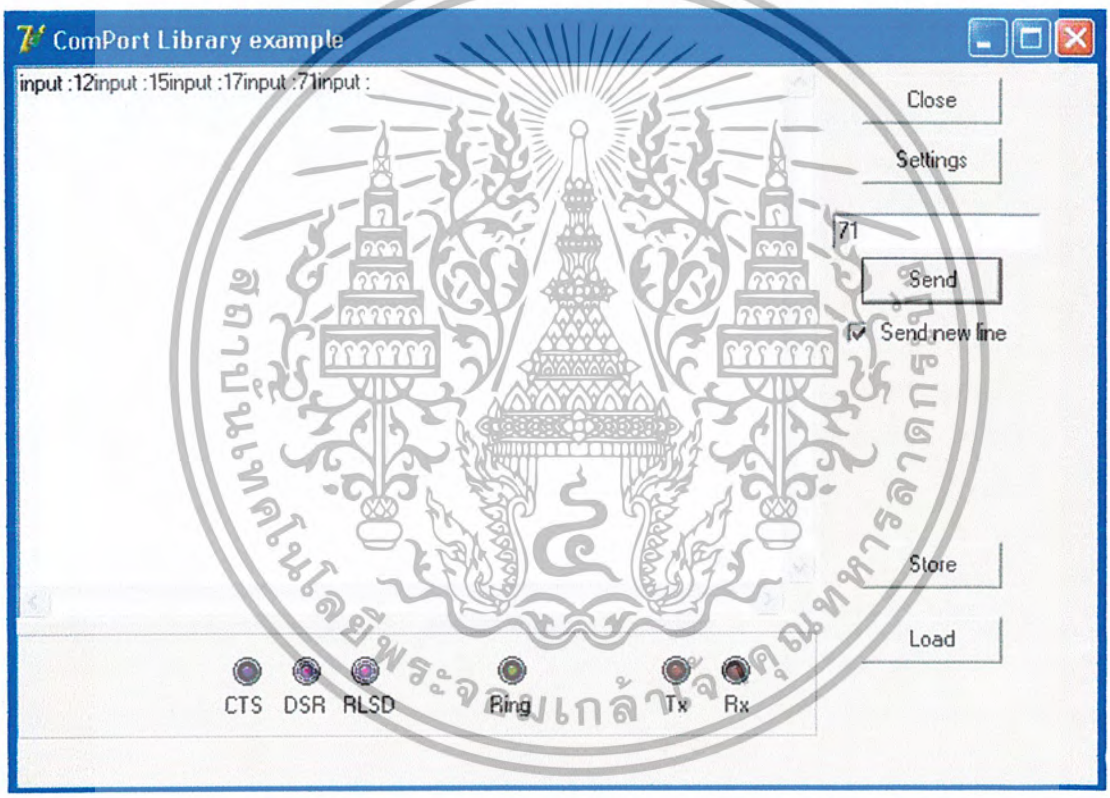
```
# include <16F628.h>
# define TxD     A1
# define RxD     A2
# define CLOCK_SP 20000000
# fuses  HS
# fuses  NOLVP, NOWDT
# use delay (clock = CLOCK_SP)
# use rs232 (baud = 9600, xmit = TxD, rcv = RxD)
# byte portb = 6
        void main (void)          {
        char ch[0];
```

```
int       i ;
while (TRUE)   {
          printf (" input :");
          gets(ch[2]);
          for (i = 0, i<2, i+t)          {
                    printf ("\c\,ch[i]); }}}
```

5. ทำการ Compile ให้เป็น file Hex แล้ว

6. แล้วนำ file hex Program ลงใน PIC

7. สังเกต ดูที่ Monitor

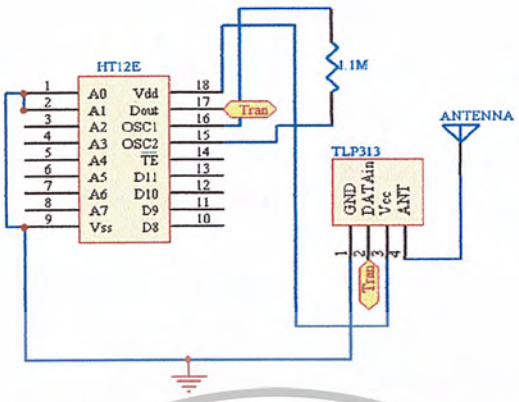


รูปที่ 9.3 ภาพโปรแกรมแสดงผลการทดลอง

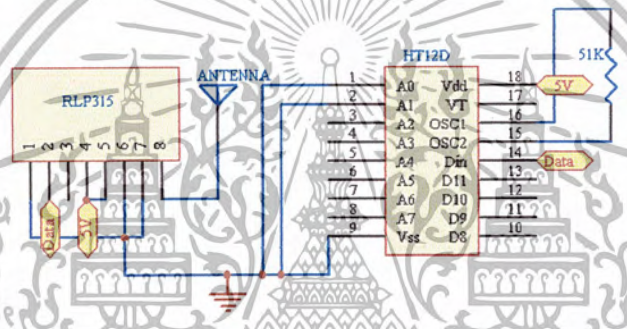
สรุปผลการทดลอง

สามารถส่งข้อมูลจาก Serial Port มาสู่ PIC ได้

การทดลองที่ 2 การส่งข้อมูล ผ่าน TLP315 กับ RLP315



รูปที่ 9.4 แสดงการเชื่อมต่อระหว่าง HT12E และ TLP315



รูปที่ 9.5 แสดงการเชื่อมต่อระหว่าง HT12D และ RLP315

วัตถุประสงค์

เพื่อทดสอบการส่งข้อมูลด้วย TLP315 กับ RLP315

ขั้นตอนการทดลอง

1. ต่อวงจรดังรูป
2. ทำการใส่ Data 4 Bit เข้าที่ขา D_0-D_3 ของ HT12E
3. สังเกตผลที่ D_0-D_3 ของ HT12D ว่าเหมือนกับของ HT12E หรือเปล่า

ผลการทดลอง

ส่ง				รับ			
D_3	D_2	D_1	D_0	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	1
0	1	1	0	0	1	1	0
0	1	1	1	0	1	1	1
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	1
1	0	1	0	1	0	1	0
1	0	1	1	1	0	1	1
1	1	0	0	1	1	0	0
1	1	0	1	1	1	0	1
1	1	1	0	1	1	1	0
1	1	1	1	1	1	1	1

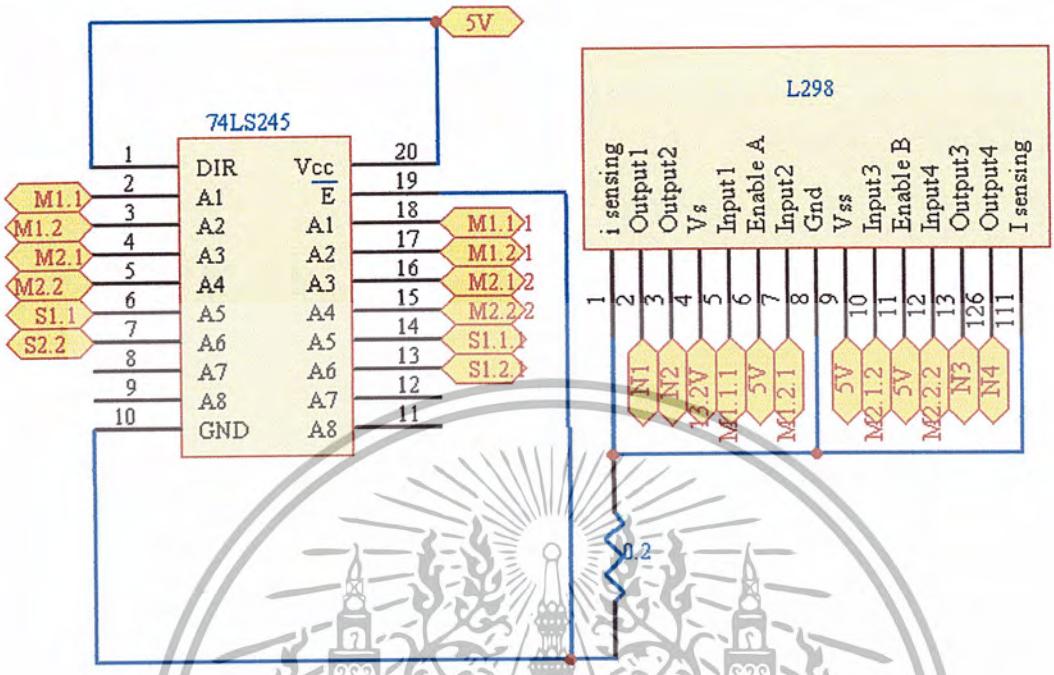
ตารางที่ 9.1 แสดงผลการรับ – ส่ง ข้อมูลด้วย TLP315 และ RLP315

สรุปผลการทดลอง

D_0-D_3 ที่ภาคส่งเท่ากับ D_0-D_3 ที่ภาครับแสดงว่าสามารถส่งรับข้อมูลทาง RF ได้

การทดลองที่ 3 การขับมอเตอร์ด้วย L298

วัตถุประสงค์ เพื่อใช้ทดสอบ IC drive motor



รูปที่ 9.6 แสดงการต่อวงจรเชื่อมระหว่าง 74LS245 กับ L298

วิธีการทดลอง

1.ต่อวงจรดังรูปที่ 9.6

2.ทำการป้อน I_1- I_4 ด้วยอินพุตข้อมูล 4 บิต

3.นำมอเตอร์มาต่อที่ out 1 out 2 1 ตัว และ out 3 out 4 1 ตัว

4.สังเกตมอเตอร์เมื่อมีการใส่ input

ผลการทดลอง

Input				Motor 1 (I_0,I_1)	Motor 2 (I_2,I_3)
I_3	I_2	I_1	I_0		
0	0	0	0	นิ่ง	นิ่ง
0	0	1	0	หมุนตามเข็ม	นิ่ง
0	0	0	1	หมุนทวนเข็ม	นิ่ง
1	0	1	1	นิ่ง	หมุนตามเข็ม
0	1	1	1	นิ่ง	หมุนทวนเข็ม
1	1	1	1	นิ่ง	นิ่ง

ตารางที่ 9.2 แสดงการทำงานของมอเตอร์เมื่อทำการป้อน I_1- I_4 ด้วยอินพุตข้อมูล 4 บิต

สรุปผลการทดลอง

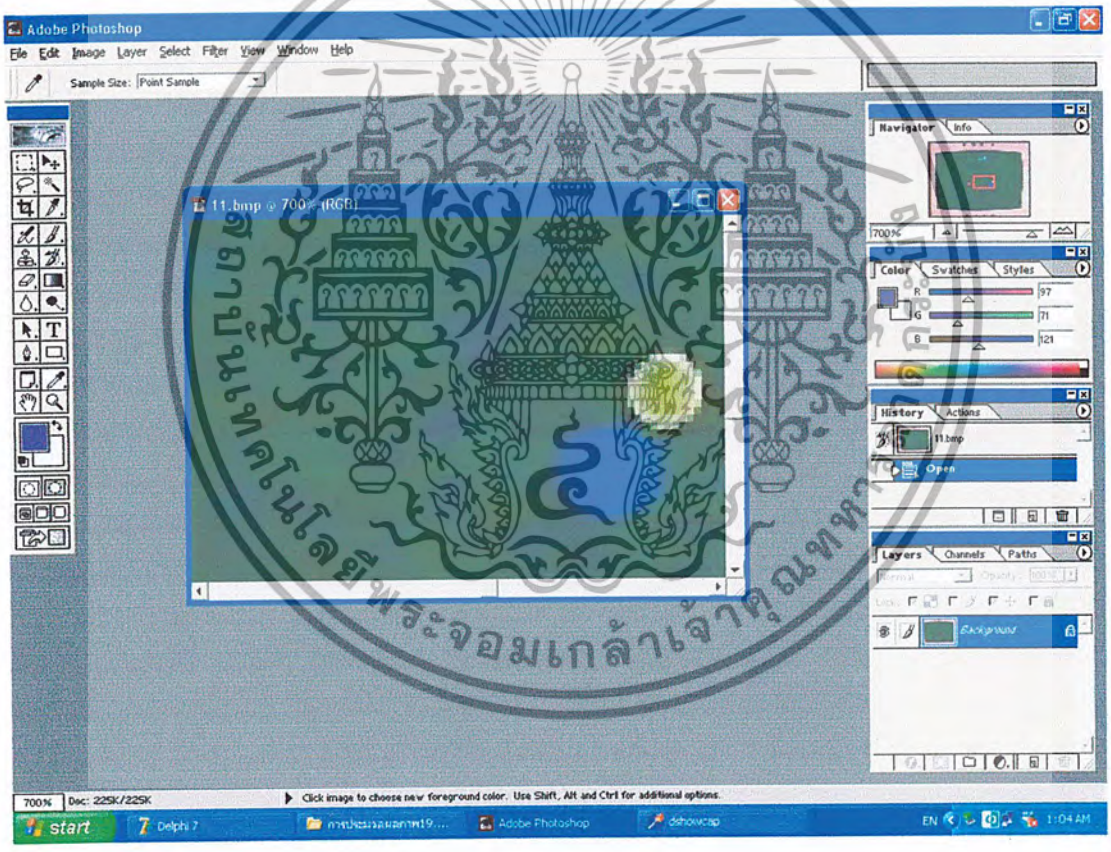
เราสามารถ Control Motor 1 และ Motor 2 ได้ด้วยการป้อนอินพุตที่ I_0-I_3

การทดลองที่ 4 การหาช่วงของ RGB โดยใช้ Photoshop

วัตถุประสงค์ เพื่อต้องการหาช่วงของ RGB ของสีที่ใช้เป็นสัญลักษณ์

วิธีการทดลอง

1. เปิดโปรแกรม Photoshop ขึ้นมา
2. เปิดภาพที่ใช้ในการทดลอบจุดสี
3. เลือก Tool ที่มีชื่อว่า EyeDroppper ดังภาพที่ 9.7
4. นำเครื่องมือนั้นไปกดที่ Pixel ที่ต้องการทราบค่าสี
5. สังเกตจากหน้าต่าง Color แล้วทำการบันทึกผลค่า RGB
6. ทำตารางช่วงของ RGB



รูปที่ 9.7 แสดงภาพการใช้งานโปรแกรม photoshop

ผลการทดลอง

สี	ส้ม		ฟ้า		น้ำเงิน		เหลือง	
	Min	Max	Min	Max	Min	Max	Min	Max
R	210	255	135	190	52	66	216	255
G	69	101	198	247	114	121	218	244
B	74	127	255	255	255	255	81	210
สี	ม่วง		ดำ		เขียวเข้ม		เขียวอ่อน	
	Min	Max	Min	Max	Min	Max	Min	Max
R	183	220	75	106	73	100	183	190
G	131	156	60	84	149	175	179	190
B	254	255	75	121	101	157	50	90

ตารางที่ 9.3.1 แสดงค่า RGB ที่มีค่า Min และ Max ของจุดสีต่างๆ

สี	ส้ม		ฟ้า		น้ำเงิน		เหลือง	
	Min	Max	Min	Max	Min	Max	Min	Max
R	203	255	134	192	54	64	223	255
G	68	101	195	250	115	119	214	249
B	71	128	255	255	255	253	80	209
สี	ม่วง		ดำ		เขียวเข้ม		เขียวอ่อน	
	Min	Max	Min	Max	Min	Max	Min	Max
R	183	219	73	102	73	100	184	190
G	129	157	61	84	152	172	177	192
B	255	254	74	123	100	159	52	89

ตารางที่ 9.3.2 แสดงค่า RGB ที่มีค่า Min และ Max ของจุดสีต่างๆ

สี	ส้ม		ฟ้า		น้ำเงิน		เหลือง	
	Min	Max	Min	Max	Min	Max	Min	Max
R	201	253	128	194	51	67	220	255
G	68	102	190	253	114	123	213	248
B	72	127	255	255	254	254	77	207
สี	ม่วง		ดำ		เขียวเข้ม		เขียวอ่อน	
	Min	Max	Min	Max	Min	Max	Min	Max
R	189	219	73	105	74	97	179	187
G	134	153	59	82	150	173	177	193
B	255	254	74	124	108	155	53	86

ตารางที่ 9.3.3 แสดงค่า RGB ที่มีค่า Min และ Max ของจุดสีต่างๆ

สี	ส้ม		ฟ้า		น้ำเงิน		เหลือง	
	Min	Max	Min	Max	Min	Max	Min	Max
R	220	255	125	194	54	65	217	255
G	69	100	193	251	110	120	210	242
B	72	126	255	255	254	255	78	210
สี	ม่วง		ดำ		เขียวเข้ม		เขียวอ่อน	
	Min	Max	Min	Max	Min	Max	Min	Max
R	186	218	75	106	75	99	180	186
G	130	154	59	83	149	173	176	192
B	255	255	73	124	105	157	51	90

ตารางที่ 9.3.4 แสดงค่า RGB ที่มีค่า Min และ Max ของจุดสีต่างๆ

สี	ส้ม		ฟ้า		น้ำเงิน		เหลือง	
	Min	Max	Min	Max	Min	Max	Min	Max
R	200	255	128	194	51	67	216	255
G	68	102	190	253	110	123	210	249
B	71	128	255	255	254	255	77	210
สี	ม่วง		ดำ		เขียวเข้ม		เขียวอ่อน	
	Min	Max	Min	Max	Min	Max	Min	Max
R	183	220	73	106	73	100	179	190
G	129	157	59	84	149	175	176	193
B	255	255	73	124	100	159	50	90

ตารางที่ 9.3.5 แสดงค่า RGB ที่มีค่า Min และ Max ของจุดสีต่างๆในแต่ละตารางรวมกัน

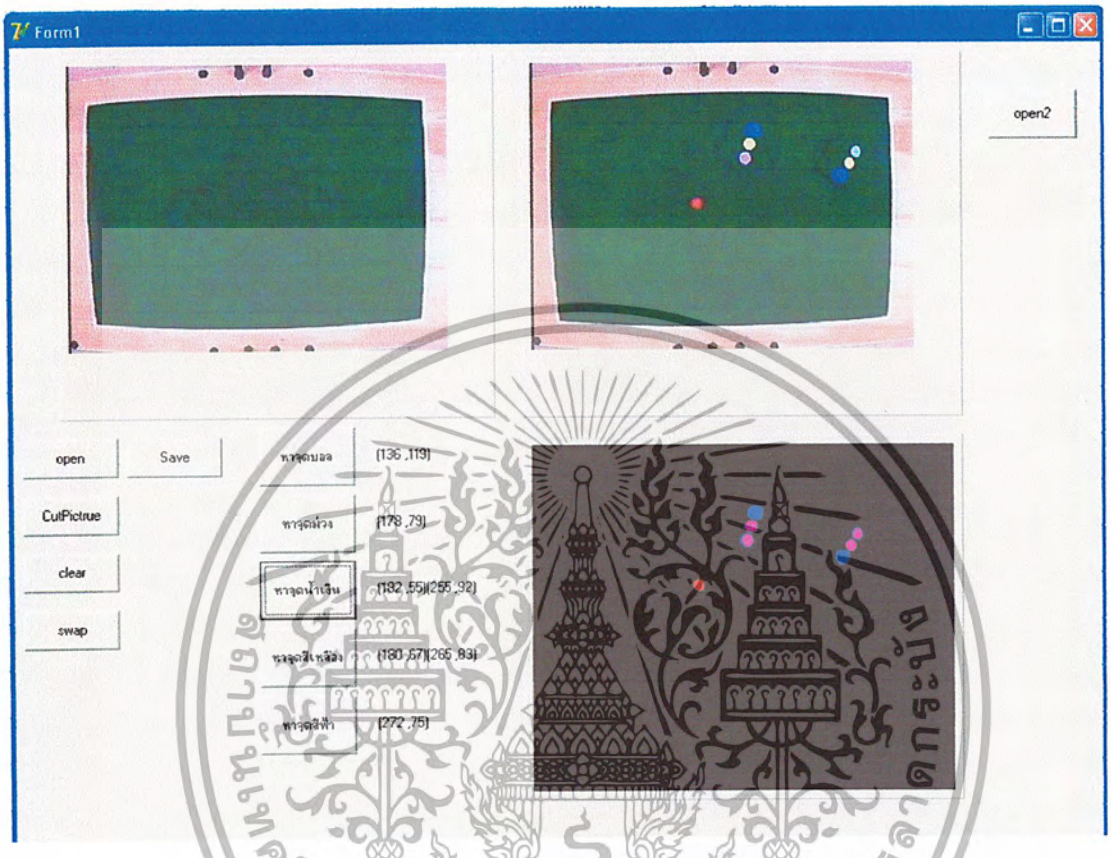
สรุปผลการทดลอง

การหาช่องของ RGB เราสามารถหาได้โดยเอาค่าสูงสุดและค่าต่ำสุดของ RGB ในสีนั้นๆ ช่วงที่ได้อยู่ในตารางที่ 9.3.5

การทดลองที่ 5 การหาตำแหน่งภาพโดยใช้วิธีที่ 1

วัตถุประสงค์ หาตำแหน่งของจุดสีต่างๆ

วิธีการทดลอง



รูปที่ 9.8 แสดงภาพโปรแกรมการทดลอง

1. เปิดโปรแกรม Image1.exe
2. คลิก Open เพื่อเปิดภาพพื้นสนามเปล่าๆคลิก open เพื่อเปิดภาพสนามที่มีหุ่นยนต์
3. ทำการคลิกเพื่อหาจุดสีต่างๆจากปุ่มบนสุดจนถึงปุ่มล่างสุดตามลำดับ
4. บันทึกผลการทดลอง
5. ทำตามวิธีข้างต้น 7 ภาพ

ผลการทดลอง

ภาพ	จุดสีส้ม	จุดสีเหลือง	จุดสีม่วง	จุดสีน้ำเงิน	จุดสีฟ้า
1	136,119	180,67 และ 265,83	-	182,55 และ 233,92	272,75
2	96,115	264,174	-	252,53 และ 261,190	265,166
3	40,185	44,189 และ 264,178	-	53,189 และ 261,190	265,166
4	-	155,88 และ 160,191	-	140,94 และ 160,179	167,95
5	127,103	151,93 และ 160,192	-	146,81 และ 160,179	153,107
6	126,92	151,93 และ 160,192	-	146,82 และ 160,180	157,107
7	125,94	152,130	-	151,78 และ 151,115	154,103

ตารางที่ 9.4 แสดงการหาจุดสีต่างๆจากปุ่มบนสุดจนถึงปุ่มล่างสุดตามลำดับ 7 ภาพ

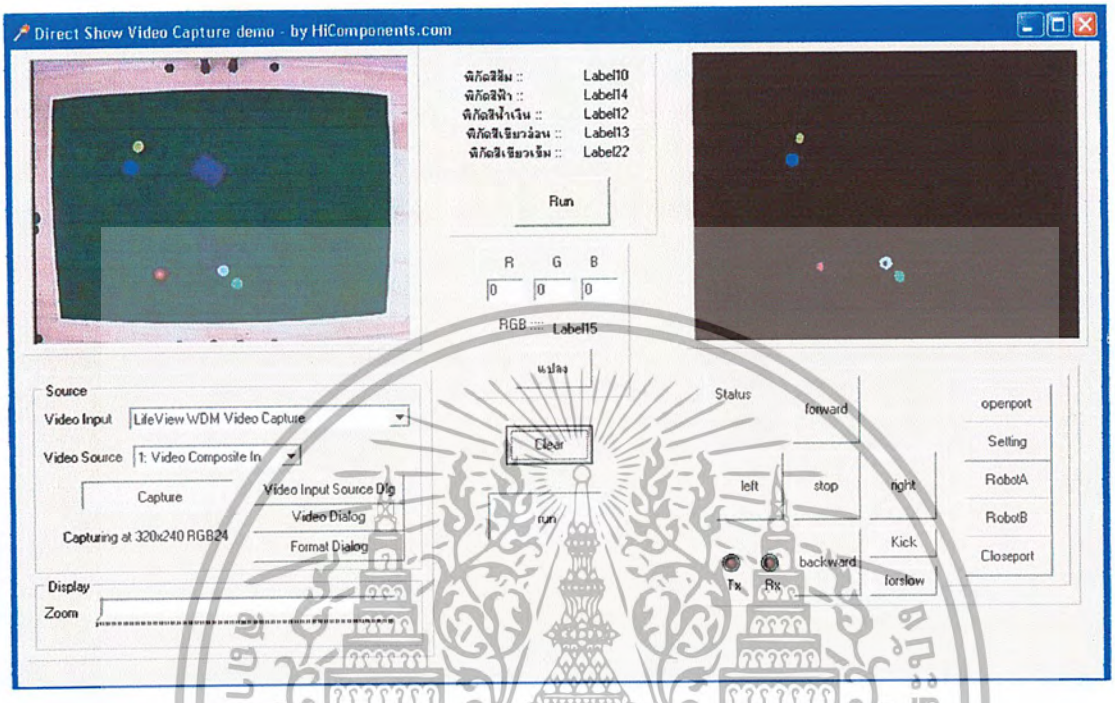
สรุปผลการทดลอง

วิธีนี้สามารถหาค่าสีได้แต่จะมีข้อผิดพลาดอยู่บ้างเพราะเกิดจากสภาพแสงที่ไม่เท่ากันของทั้งสนาม และวิธีนี้กำหนดช่วงการทำเทรสโชว์ได้ยาก และมีขั้นตอนการทำหลายขั้นตอน จึงทำให้โปรแกรมช้าลง

การทดลองที่ 6 การหาตำแหน่งภาพโดยใช้วิธีที่ 2

วัตถุประสงค์ หาตำแหน่งของจุดสีต่างๆ

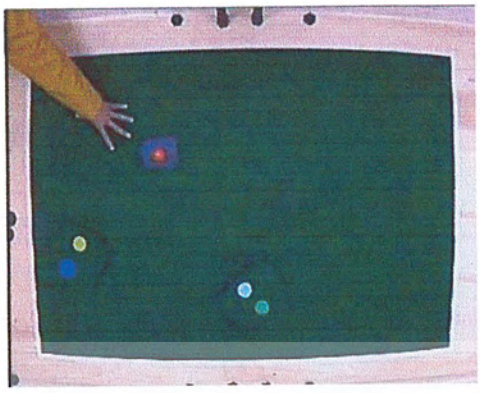
วิธีการทดลอง



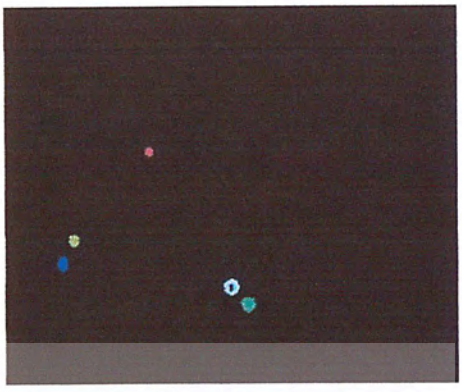
รูปที่ 9.9 แสดงภาพโปรแกรมการทดลอง

1. เปิดโปรแกรม Image2.exe
2. Set video dialog
3. กด Capture
4. กด Run ที่อยู่ในกรอบ
5. สังเกตจุดสีที่ขึ้นทางหน้าต่างขวา
6. บันทึกภาพแล้วนำมาเปรียบเทียบ

ผลการทดลอง



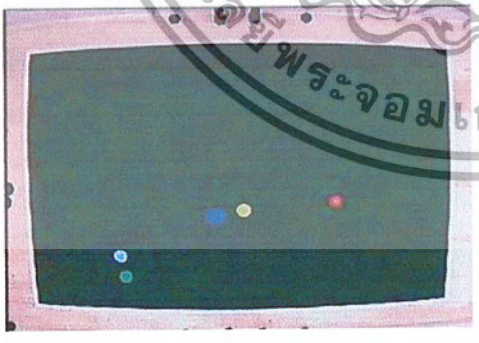
รูปที่ 9.10 แสดงภาพการทดลอง



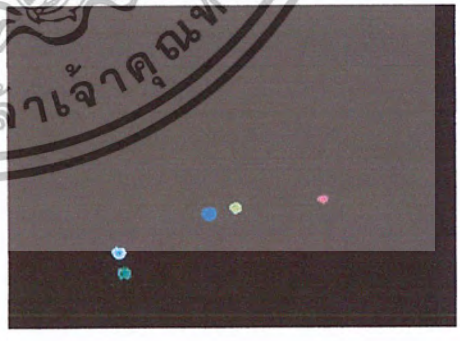
รูปที่ 9.11 แสดงภาพการทดลอง

รูปที่ 9.12 แสดงภาพการทดลอง

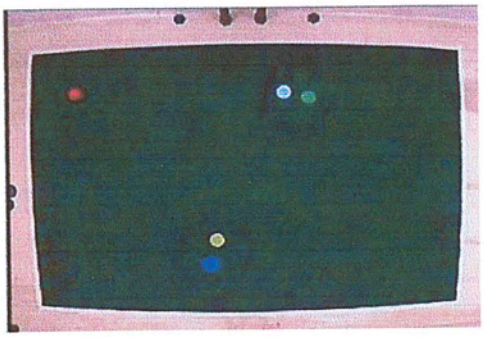
รูปที่ 9.13 แสดงภาพการทดลอง



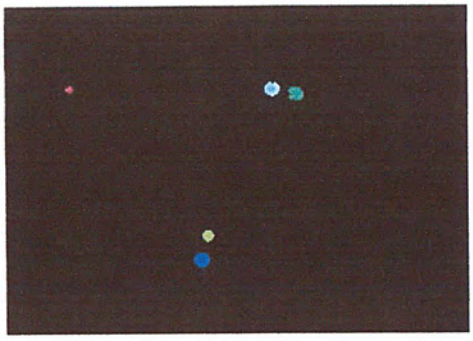
รูปที่ 9.14 แสดงภาพการทดลอง



รูปที่ 9.15 แสดงภาพการทดลอง



รูปที่ 9.16 แสดงภาพการทดลอง



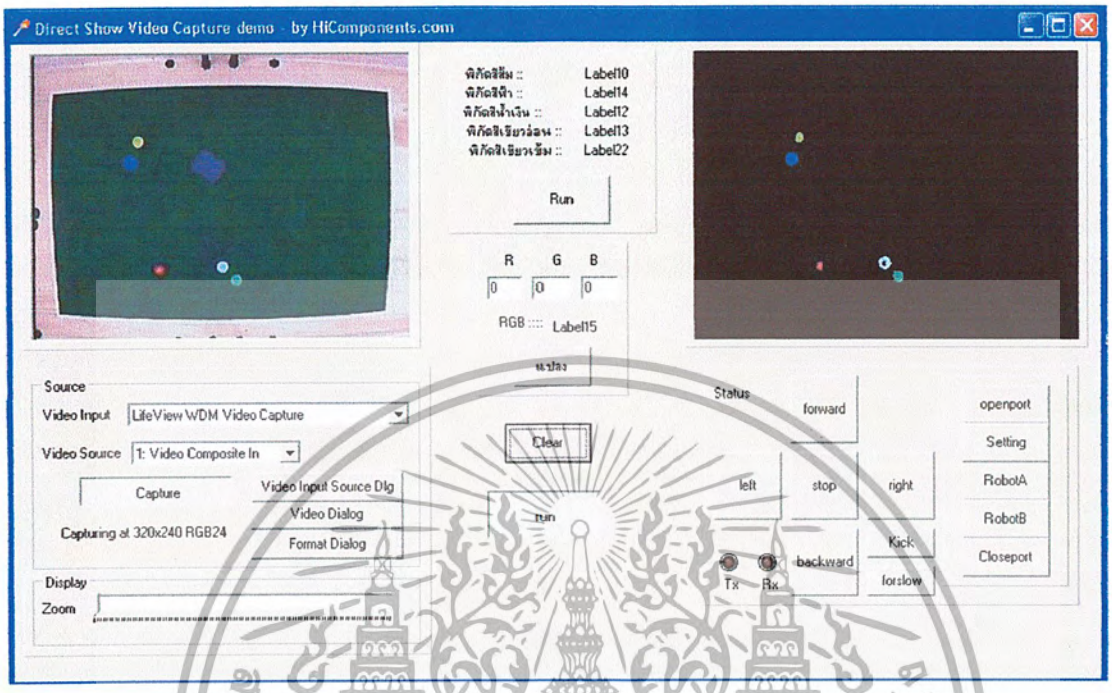
รูปที่ 9.17 แสดงภาพการทดลอง

สรุปผลการทดลอง

วิธีนี้สามารถหาค่าสีได้เกือบทุกจุดสีในสนาม เนื่องจากที่เราเก็บค่าช่วงของจุดมาจากการทดลองงที่แล้ว แล้วนำค่าช่วงที่ได้ไปวนลูปตรวจสอบกับภาพเลยทำให้สามารถตรวจสอบได้เร็วขึ้น เนื่องจากมีเพียงขั้นตอนเดียว และช่วงที่ใช้ในการ check จะเป็นช่วงแคบๆ ดังนั้นทำให้สีที่จะตรวจสอบเป็นสีที่เฉพาะเจาะจงค่อนข้างมาก

การทดลองที่ 7 การทดสอบการวิ่งเตะบอล

วัตถุประสงค์ เพื่อที่จะทดสอบการวิ่งเข้าหาบอลของหุ่นยนต์ A และ B

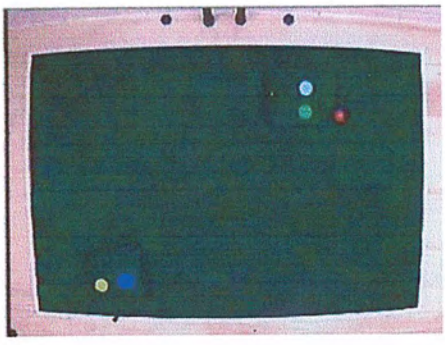


รูปที่ 9.18 แสดงภาพโปรแกรม

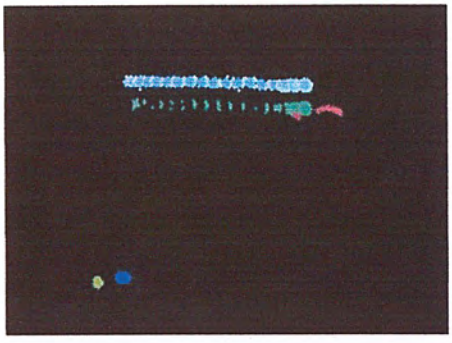
วิธีการทดลอง

1. เปิดโปรแกรม Image2.exe
2. Set video dialog
3. Set video source
4. Set format เป็น (320*280 pixels)
5. กดปุ่ม Run ที่อยู่นอกกรอบ
6. สังเกตภาพด้านซ้ายกับด้านขวา แล้วนำภาพด้านขวามาวิเคราะห์

ผลการทดลอง



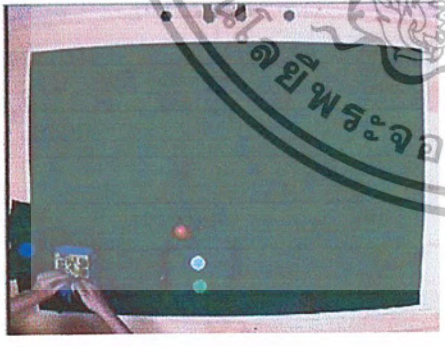
รูปที่ 9.19 แสดงภาพการทดลอง



รูปที่ 9.20 แสดงภาพการทดลอง

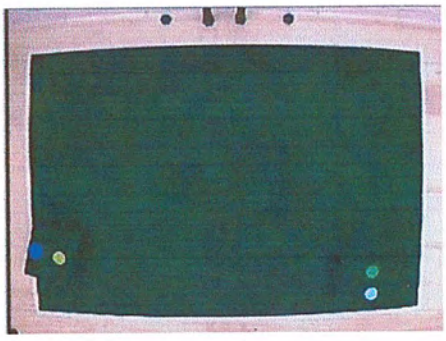
รูปที่ 9.21แสดงภาพการทดลอง

รูปที่ 9.22 แสดงภาพการทดลอง

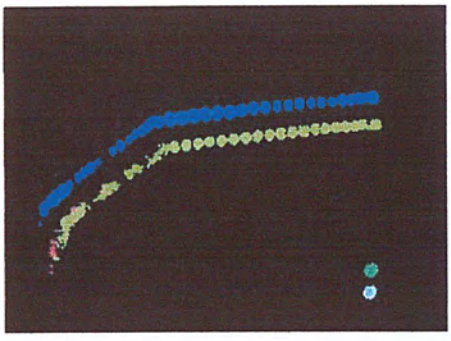


รูปที่ 9.23 แสดงภาพการทดลอง

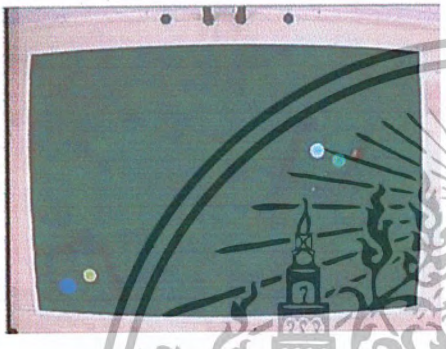
รูปที่ 9.24 แสดงภาพการทดลอง



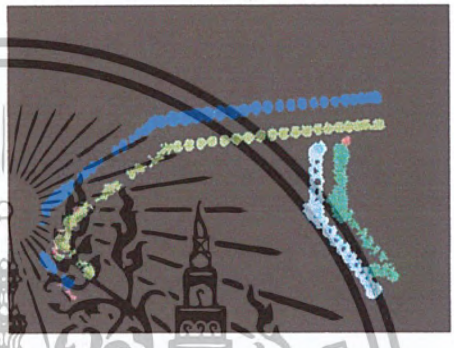
รูปที่ 9.25 แสดงภาพการทดลอง



รูปที่ 9.26 แสดงภาพการทดลอง

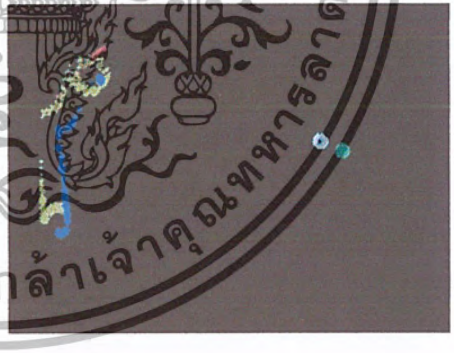


รูปที่ 9.27 แสดงภาพการทดลอง

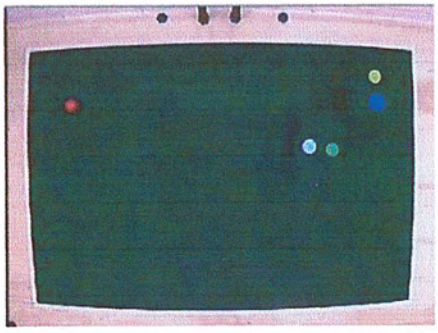


รูปที่ 9.28 แสดงภาพการทดลอง

รูปที่ 9.29 แสดงภาพการทดลอง



รูปที่ 9.30 แสดงภาพการทดลอง



รูปที่ 9.31 แสดงภาพการทดลอง

รูปที่ 9.32 แสดงภาพการทดลอง

สรุปผลการทดลอง

หุ่นยนต์สามารถเคลื่อนที่ได้ตามโปรแกรมที่กำหนดไว้ โดยมีความคิดพลาดคลาดเคลื่อนเล็กน้อยเนื่องจาก ความแตกต่างของแสงบนพื้นสนาม

บทที่ 10
สรุปและวิเคราะห์ผลการทดลอง

โครงงานนี้เป็นการทดลองเพื่อศึกษาถึงขบวนการ ทำงานของการสร้างรถบังคับวิทยุโดยใช้โปรแกรม Delphi ในการควบคุม ในโครงงานนี้จะมุ่งเน้นการควบคุมรถบังคับให้มีความสามารถในการเป็น หุ่นยนต์เตะฟุตบอล (Soccer Robot) โดยสามารถทำงานได้ถูกต้องตามโปรแกรมที่เขียนไว้ โดยใช้กล้องดิจิตอลในการจับภาพของวัตถุ

นอกจากนั้นในโครงงานนี้ยังมีการใช้เทคนิคการเขียนโปรแกรมการบังคับการขับเคลื่อนรถด้วย DC มอเตอร์ในการควบคุมองศาของหุ่นยนต์ให้เป็นไปตามทิศทางที่ต้องการและที่สำคัญยังต้องใช้ความรู้ทางทฤษฎีเกี่ยวกับการสื่อสารเบื้องต้นมาวิเคราะห์ปัญหาที่เกิดขึ้นเกี่ยวกับภาครับภาคส่งของรถบังคับวิทยุได้ ตลอดจนใช้ความรู้ในการเชื่อมต่อกับพอร์ตอนุกรม (Serial Port) กับฮาร์ดแวร์ภายนอกได้

ภาคผนวก

1. Source code โปรแกรมวิธีหาจุดสีวิธีที่ 1

```
unit Unit1;
interface
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls, ImageEnView, VideoCap, ExtCtrls, Buttons, ImageEnProc, Menus,
ImageEnIO, ComCtrls, IEOpenSaveDlg, hyiedefs, IEVect, ieview, hyieutils,
ExtDlgs,Math, CPort;
type
TForm1 = class(TForm)
GroupBox1: TGroupBox;
SpeedButton2: TSpeedButton;
CheckBox1: TCheckBox;
ImageEnProc1: TImageEnProc;
ImageEnProc2: TImageEnProc;
ImageEnIO1: TImageEnIO;
ImageEnVideoView1: TImageEnVideoView;
Label5: TLabel;
Label6: TLabel;
SaveImageEnDialog1: TSaveImageEnDialog;
GroupBox4: TGroupBox;
butt_source: TButton;
butt_format: TButton;
GroupBox5: TGroupBox;
butt_freeze: TSpeedButton;
Button1: TButton;
butt_save: TButton;
Label8: TLabel;
OpenPictureDialog1: TOpenPictureDialog;
Panel1: TPanel;
Image1: TImage;
Panel2: TPanel;
```

```
Image2: TImage;
butt_run: TButton;
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
Label7: TLabel;
Label9: TLabel;
Label10: TLabel;
Label11: TLabel;
Label12: TLabel;
butt_open: TButton;
Label13: TLabel;
Button2: TButton;
Label14: TLabel;
ComPort: TComPort;
procedure SpeedButton2Click(Sender: TObject);
procedure CheckBox1Click(Sender: TObject);
procedure butt_freezeClick(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure butt_sourceClick(Sender: TObject);
procedure butt_formatClick(Sender: TObject);
procedure Button7Click(Sender: TObject);
procedure butt_saveClick(Sender: TObject);
procedure FormActivate(Sender: TObject);
procedure Button8Click(Sender: TObject);
procedure ImageEnVideoView1Job(Sender: TObject; job: TIEJob;
per: Integer);
procedure butt_open1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure butt_runClick(Sender: TObject);
procedure butt_openClick(Sender: TObject);
```

```
procedure Button2Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
procedure DisplayVideoSize;
end;
var
Form1: TForm1;
r,g,b,rr,gg,bb,rrr,ggg,bbb : array[0..400,0..300] of integer;
xtotal,ytotal,x,y,xorange,yorange,xyellow1,yyellow1,xblue1,yblue1,xlight,ylight,xpurple,ypurple,
xblue2,yblue2,xyellow2,yyellow2 : integer ;
dis1,dis2,dis3 :real;
str : string;
implementation
uses giflzw, tiflzw;
{$R *.DFM}
// Input ON
procedure TForm1.SpeedButton2Click(Sender: TObject);
begin
ImageEnVideoView1.ShowVideo:=SpeedButton2.Down;
Button1.enabled:=not SpeedButton2.Down;
DisplayVideoSize;
end;
// overlay
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
if checkbox1.checked then
ImageEnVideoView1.DisplayMode:=dmOverlay
else
ImageEnVideoView1.DisplayMode:=dmPreview;
end;
```

```
// freeze
procedure TForm1.butt_freezeClick(Sender: TObject);
begin
ImageEnVideoView1.Frozen:=butt_freeze.Down;
end;
// Color adjust
procedure TForm1.Button1Click(Sender: TObject);
begin
ImageEnProc1.DoPreviews(ppeColorAdjust);
end;
// Configure source
procedure TForm1.butt_sourceClick(Sender: TObject);
begin
if not ImageEnVideoView1.DoConfigureSource then
MessageDlg('Configure Source dialog not available',mtInformation,[mbOK],0)
else
DisplayVideoSize;
end;
// Configure Format
procedure TForm1.butt_formatClick(Sender: TObject);
begin
if not ImageEnVideoView1.DoConfigureFormat then
MessageDlg('Configure Format dialog not available',mtInformation,[mbOK],0)
else
DisplayVideoSize;
end;
// Configure display
procedure TForm1.Button7Click(Sender: TObject);
begin
if not ImageEnVideoView1.DoConfigureDisplay then
MessageDlg('Configure Display dialog not available',mtInformation,[mbOK],0)
else
```

```
DisplayVideoSize;
end;
// Save as...
procedure TForm1.butt_saveClick(Sender: TObject);
begin
ImageEnIO1.SaveToFile('C:\picture\1.bmp');
end;
//
procedure TForm1.DisplayVideoSize;
var
r:TRect;
begin
r:=ImageEnVideoView1.GetVideoSize;
Label6.caption:=inttostr(r.right+1)+'x'+inttostr(r.bottom+1);
end;
//
procedure TForm1.FormActivate(Sender: TObject);
var
q:integer;
begin
DefGIF_LZWDECOMPFUNC:=GIFLZWDecompress;
DefGIF_LZWCOMPFUNC:=GIFLZWCompress;
DefTIFF_LZWDECOMPFUNC:=TIFFLZWDecompress;
DefTIFF_LZWCOMPFUNC:=TIFFLZWCompress;
end;
// Compression
procedure TForm1.Button8Click(Sender: TObject);
begin
ImageEnVideoView1.DoConfigureCompression;
end;
procedure TForm1.ImageEnVideoView1Job(Sender: TObject; job: TIEJob;
per: Integer);
```

```
begin
case job of
iejNOTHING: Label8.Caption:='';
iejVIDEOCAP_CONNECTING: Label8.Caption:='Connecting...';
end;
Application.ProcessMessages;
end;
procedure TForm1.butt_open1Click(Sender: TObject);
begin
Image2.Picture.Bitmap.LoadFromFile('C:\picture\1.bmp')
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
Image1.Picture.Bitmap.LoadFromFile('C:\picture\0.bmp');
end;
procedure TForm1.butt_runClick(Sender: TObject);
var i,j,r1,r2,b1,b2,g1,g2,p1,p2,r3,g3,b3,p3,ii,jj : integer;
begin
x:=0;
y:=0;
{##############################}
{ PROCEES OF CAPTION PICTURE }
{##############################}
{  ImageEnVideoView1.Frozen := True ;
ImageEnIO1.SaveToFile('C:\picture\1.bmp');
ImageEnVideoView1.Frozen := False;
Image2.Picture.Bitmap.LoadFromFile('C:\picture\1.bmp');  }
{#########################}
{ PROCESS OF CHECK POINT  }
{#########################}
for i:=0 to 359 do        //Check Orage
begin
```

```
for j:=0 to 288 do
begin
p1:=Image1.Canvas.Pixels[i,j];
r1:= p1 Mod 256;
g1:= (p1 div 256)Mod 256;
b1:= p1 div 65536;
p2:=Image2.Canvas.Pixels[i,j];
r2:= p2 Mod 256;
g2:= (p2 div 256)Mod 256;
b2:= p2 div 65536;
r[i,j] :=  r1; g[i,j] := g1; b[i,j] := b1;
rr[i,j] := r2; gg[i,j] := g2; bb[i,j] := b2;
r3:=r1-r2;if r3<0 then r3:=r3*(-1);if r3<20 then r3:=0 else r3:=255;
g3:=g1-g2;if g3<0 then g3:=g3*(-1);if g3<50 then g3:=0 else g3:=255;
b3:=b1-b2;if b3<0 then b3:=b3*(-1);if b3<50 then b3:=0 else b3:=255;
rrr[i,j]:=r3; ggg[i,j]:=g3; bbb[i,j] :=  b3;
//Image3.Canvas.Pixels[i,j]:=RGB(r3,g3,b3);
end;
end;
Label9.Caption:=' ';ii:=0;jj:=0;
for i:=35 to 328 do
begin
for j:=40 to 248 do
begin
if rrr[i,j] <> 255then
begin
rrr[i,j]:=0;ggg[i,j]:=0;bbb[i,j]:=0;
end
else begin
if ((RGB(rrr[i-1,j],ggg[i-1,j],bbb[i-1,j])= 255) and (RGB(rrr[i+1,j],ggg[i+1,j],bbb[i+1,j])=255))
and((RGB(rrr[i-2,j],ggg[i-2,j],bbb[i-2,j]) = 255) and (RGB(rrr[i+2,j],ggg[i+2,j],bbb[i+2,j])=255))
```

```
//and((RGB(rrr[i,j-2],ggg[i,j-2],bbb[i,j-2]) = 255) and (RGB(rrr[i,j+2],ggg[i,j+2],bbb[i,j+2])
=255))
and ((RGB(rrr[i,j-1],ggg[i,j-1],bbb[i,j-1])= 255)and (RGB(rrr[i,j+1],ggg[i,j+1],bbb[i,j+1])= 255))
then
if (i+j)-(ii+jj)>60 then begin
xorange := i;  yorange := j;
Label9.Caption := Label9.Caption + '('+IntToStr(i)+' ,'+IntToStr(j)+')';
ii:=i;jj:=j;
end ;
end;
end;
end;
for i:=0 to 359 do      //Check Purple
begin
for j:=0 to 288 do
begin
r1:= r[i,j];
b1:= b[i,j];
r2:= rr[i,j];
b2:= bb[i,j];
r3:=r1-r2;if r3<0 then r3:=r3*(-1);if r3<120 then r3:=0 else r3:=255;
g3:=0;
b3:=b1-b2;if b3<0 then b3:=b3*(-1);if ((b3<170) and (b3>135)) then b3:=255 else b3:=0 ;
//Image3.Canvas.Pixels[i,j]:=RGB(r3,g3,b3);
rrr[i,j]:=r3; ggg[i,j]:=0;bbb[i,j]:=b3;
end;end;
Label12.Caption :=' ';ii:=0;jj:=0;
for i:=35 to 328 do
begin
for j:=40 to 248 do
begin
```

```
if ((RGB(rrr[i-1,j],ggg[i-1,j],bbb[i-1,j])  = RGB(255,0,255)) and
(RGB(rrr[i+1,j],ggg[i+1,j],bbb[i+1,j])=RGB(255,0,255)))
and((RGB(rrr[i-2,j],ggg[i-2,j],bbb[i-2,j]) = RGB(255,0,255)) and
(RGB(rrr[i+2,j],ggg[i+2,j],bbb[i+2,j])=RGB(255,0,255)))
//and((RGB(rrr[i-3,j],ggg[i-3,j],bbb[i-3,j]) = RGB(255,0,255)) and
(RGB(rrr[i+3,j],ggg[i+3,j],bbb[i+3,j])=RGB(255,0,255)))
and((RGB(rrr[i,j-3],ggg[i,j-3],bbb[i,j-3]) = RGB(255,0,255)) and
(RGB(rrr[i,j+3],ggg[i,j+3],bbb[i,j+3])=RGB(255,0,255)))
and((RGB(rrr[i,j-2],ggg[i,j-2],bbb[i,j-2]) = RGB(255,0,255)) and
(RGB(rrr[i,j+2],ggg[i,j+2],bbb[i,j+2])=RGB(255,0,255)))
and ((RGB(rrr[i,j-1],ggg[i,j-1],bbb[i,j-1])= RGB(255,0,255)) and
(RGB(rrr[i,j+1],ggg[i,j+1],bbb[i,j+1])= RGB(255,0,255)))  then
if (i+j)-(ii+jj)>40 then begin
xpurple := i ; ypurple := j ;
Label12.Caption := Label12.Caption + '('+IntToStr(i)+' ,'+IntToStr(j)+')';
ii:=i;jj:=j;
end ;
end;end;
for i:=0 to 359 do     //Check Blue
begin
for j:=0 to 288 do
begin
r1:= r[i,j];
b1:= b[i,j];
r2:= rr[i,j];
b2:= bb[i,j];
r3:=r1-r2;if r3<0 then r3:=r3*(-1);if r3<80 then r3:=0 else r3:=255;
g3:=0;
b3:=b1-b2;if b3<0 then b3:=b3*(-1);if b3<90 then b3:=0 else b3:=255 ;
rrr[i,j]:=r3;ggg[i,j]:=0;bbb[i,j]:=b3;
//Image3.Canvas.Pixels[i,j]:=RGB(r3,g3,b3);
end;end;
```

```
Label11.Caption :=' ';ii:=0;jj:=0;
for i:=35 to 328 do
begin
for j:=40 to 248 do
begin
if ((RGB(rrr[i-1,j],ggg[i-1,j],bbb[i-1,j])= RGB(0,0,255)) and
(RGB(rrr[i+1,j],ggg[i+1,j],bbb[i+1,j])=RGB(0,0,255)))
//and((RGB(rrr[i-2,j],ggg[i-2,j],bbb[i-2,j]) = RGB(0,0,255)) and
(RGB(rrr[i+2,j],ggg[i+2,j],bbb[i+2,j])=RGB(0,0,255)))
//and((RGB(rrr[i,j-2],ggg[i,j-2],bbb[i,j-2]) = RGB(0,0,255)) and
(RGB(rrr[i,j+2],ggg[i,j+2],bbb[i,j+2]) =RGB(0,0,255)))
and ((RGB(rrr[i,j-1],ggg[i,j-1],bbb[i,j-1])= RGB(0,0,255))and
(RGB(rrr[i,j+1],ggg[i,j+1],bbb[i,j+1])= RGB(0,0,255)))
then
if (i+j)-(ii+jj)>16 then
begin
Label11.Caption := Label11.Caption + '('+IntToStr(i)+' ,'+IntToStr(j)+')';
ii:=i;jj:=j;
if x=0 then
begin
xblue1:=i ;
yblue1:=j ;
end;
if x=1 then
begin
xblue2:=i ;
yblue2:= j;
end;
x:= x+1;
end ;
end;end;
for i:=0 to 359 do          //Check Yellow and Light Blue
```

```
begin
for j:=0 to 288 do
begin
r1:= r[i,j];
g1:= g[i,j];
r2:= rr[i,j];
g2:= gg[i,j];
r3:=r1-r2;if r3<0 then r3:=r3*(-1);if r3<150 then r3:=0 else r3:=255;
g3:=g1-g2;if g3<0 then g3:=g3*(-1);if g3<120 then g3:=0 else g3:=255;
b3:=0;
rrr[i,j]:=r3;ggg[i,j]:=g3;bbb[i,j]:=b3;
// Image3.Canvas.Pixels[i,j]:=RGB(r3,g3,b3);
end;end;
Label10.Caption:=' ';
ii:=0;jj:=0;
for i:=35 to 328 do
begin
for j:=40 to 248 do
begin
if ((RGB(rrr[i-1,j],ggg[i-1,j],bbb[i-1,j])= RGB(255,255,0)) and
(RGB(rrr[i+1,j],ggg[i+1,j],bbb[i+1,j])=RGB(255,255,0)))
and((RGB(rrr[i-2,j],ggg[i-2,j],bbb[i-2,j]) = RGB(255,255,0))and
(RGB(rrr[i+2,j],ggg[i+2,j],bbb[i+2,j])=RGB(255,255,0)))
and((RGB(rrr[i,j-2],ggg[i,j-2],bbb[i,j-2]) = RGB(255,255,0))and
(RGB(rrr[i,j+2],ggg[i,j+2],bbb[i,j+2]) =RGB(255,255,0)))
and ((RGB(rrr[i,j-1],ggg[i,j-1],bbb[i,j-1])= RGB(255,255,0))and
(RGB(rrr[i,j+1],ggg[i,j+1],bbb[i,j+1])= RGB(255,255,0)))  then
begin
if (i+j)-(ii+jj)>10 then begin
Label10.Caption := Label10.Caption + '('+IntToStr(i)+' ,'+IntToStr(j)+')';
ii:=i;jj:=j;
if y = 0 then
```

```
begin
xyellow1:= i;
yyellow1:= j ;
end;
if y = 1 then
begin
xyellow2:= i;
yyellow2:= j;
end;
y:= y+1;
end ;
end;
end;
end;
Label13.Caption :=' ';ii:=0;jj:=0;
for i:=35 to 328 do
begin
for j:=40 to 248 do
begin
if ((RGB(rrr[i-1,j],ggg[i-1,j],bbb[i-1,j])= RGB(0,255,0)) and
(RGB(rrr[i+1,j],ggg[i+1,j],bbb[i+1,j])=RGB(0,255,0)))
and((RGB(rrr[i-2,j],ggg[i-2,j],bbb[i-2,j]) = RGB(0,255,0))and
(RGB(rrr[i+2,j],ggg[i+2,j],bbb[i+2,j])=RGB(0,255,0)))
and((RGB(rrr[i,j-2],ggg[i,j-2],bbb[i,j-2]) = RGB(0,255,0))and
(RGB(rrr[i,j+2],ggg[i,j+2],bbb[i,j+2]) =RGB(0,255,0)))
and ((RGB(rrr[i,j-1],ggg[i,j-1],bbb[i,j-1])= RGB(0,255,0))and
(RGB(rrr[i,j+1],ggg[i,j+1],bbb[i,j+1])= RGB(0,255,0)))  then
if (i+j)-(ii+jj)>11 then begin
Label13.Caption := Label13.Caption + '('+IntToStr(i)+' ,'+IntToStr(j)+')';
ii:=i;jj:=j;
end ;
end;end;
```

```
{##############################}
{ PROCESS OF CALCULATE DISTANCE }
{##############################}
dis1 := sqrt(sqr(xorange-xyellow1)+sqr(yorange-yyellow1)); //Choose Robot
dis2 := sqrt(sqr(xorange-xyellow2)+sqr(yorange-yyellow2));
if dis1 > dis2 then
begin
xtotal:=xyellow2 ;
ytotal:=yyellow2 ;
end
else
begin
xtotal:=xyellow1 ;
ytotal:=yyellow1 ;
end;
if ((abs(xtotal-xpurple)<15)and(abs(ytotal-ypurple)<15)) then
begin
Str := 'a0' + #13#10;
ComPort.WriteStr(Str);
end;
if ((abs(xtotal-xlight)<15)and(abs(ytotal-ylight)<15)) then
begin
Str := 'b0' + #13#10;
ComPort.WriteStr(Str);
end;
end;
procedure TForm1.butt_openClick(Sender: TObject);
begin
if OpenPictureDialog1.Execute then
Image2.Picture.Bitmap.LoadFromFile(OpenPictureDialog1.FileName);
end;
procedure TForm1.Button2Click(Sender: TObject);
```

```
begin
label14.Caption:=
inttostr(xyellow1)+'/'+inttostr(xyellow2)+'/'+inttostr(xorange)+'/'+inttostr(yorange)+'/'+inttostr(xb
lue1)+'/'+inttostr(yblue2);
end;
end.
```

2. Source code โปรแกรมทดสอบการบังคับรถ

```
unit test;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, CPort;
type
TForm1 = class(TForm)
ComPort: TComPort;
butt_open: TButton;
butt_set: TButton;
GroupBox1: TGroupBox;
butt_forward: TButton;
butt_right: TButton;
butt_backward: TButton;
butt_left: TButton;
butt_stop: TButton;
GroupBox2: TGroupBox;
butt_robot1: TButton;
butt_robot2: TButton;
pong: TGroupBox;
Label1: TLabel;
butt_close: TButton;
procedure butt_openClick(Sender: TObject);
procedure butt_setClick(Sender: TObject);
```

```
procedure butt_forwardClick(Sender: TObject);
procedure butt_closeClick(Sender: TObject);
procedure butt_rightClick(Sender: TObject);
procedure butt_backwardClick(Sender: TObject);
procedure butt_leftClick(Sender: TObject);
procedure butt_stopClick(Sender: TObject);
procedure butt_robot1Click(Sender: TObject);
procedure butt_robot2Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
var
Form1: TForm1;
select : string;
implementation
{$R *.dfm}
procedure TForm1.butt_openClick(Sender: TObject);
begin
Comport.Open;
end;
procedure TForm1.butt_setClick(Sender: TObject);
begin
ComPort.ShowSetupDialog;
end;
procedure TForm1.butt_forwardClick(Sender: TObject);
var
str : string;
begin
if select = 'a'then
begin
```

```
str := 'a3'+#13#10;
ComPort.WriteStr(Str);
end;
if select = 'b'then
begin
str := 'b3'+#13#10;
ComPort.WriteStr(Str);
end;
end;
procedure TForm1.butt_closeClick(Sender: TObject);
begin
ComPort.Close;
end;
procedure TForm1.butt_rightClick(Sender: TObject);
var str : string;
begin
if select = 'a'then
begin
str := 'a2'+#13#10;
ComPort.WriteStr(Str);
end;
if select = 'b'then
begin
str := 'b2'+#13#10;
ComPort.WriteStr(Str);
end;
end;
procedure TForm1.butt_backwardClick(Sender: TObject);
var str : string;
begin
if select = 'a'then
begin
```

```
str := 'a4'+#13#10;
ComPort.WriteStr(Str);
end;
if select = 'b'then
begin
str := 'b4'+#13#10;
ComPort.WriteStr(Str);
end;
end;
procedure TForm1.butt_leftClick(Sender: TObject);
var
str : string;
begin
if select = 'a'then
begin
str := 'a1'+#13#10;
ComPort.WriteStr(Str);
end;
if select = 'b'then
begin
str := 'b1'+#13#10;
ComPort.WriteStr(Str);
end;
end;
procedure TForm1.butt_stopClick(Sender: TObject);
var
str : string;
begin
if select = 'a'then
begin
str := 'a5'+#13#10;
ComPort.WriteStr(Str);
```

```
end;
if select = 'b'then
begin
str := 'b5'+#13#10;
ComPort.WriteStr(Str);
end;
end;
procedure TForm1.butt_robot1Click(Sender: TObject);
begin
select := 'a';
Label1.Caption := 'Robot A is working';
end;
procedure TForm1.butt_robot2Click(Sender: TObject);
begin
select := 'b' ;
Label1.Caption := 'Robot B is working';
end;
end.
```

3. Source code โปรแกรมหาจุดด้วยวิธีที่2 และพร้อมใช้งานได้

```
unit umain;
interface
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls, ieview, imageenview, Buttons, ieds, ComCtrls, ExtCtrls, Menus,
ExtDlgs, CPort,Math, CPortCtl;
type
Tfmain = class(TForm)
Panel2: TPanel;
ImageEnView1: TImageEnView;
ComPort1: TComPort;
OpenPictureDialog1: TOpenPictureDialog;
```

Panel3: TPanel;

GroupBox1: TGroupBox;

Label1: TLabel;

Label4: TLabel;

Label8: TLabel;

SpeedButton1: TSpeedButton;

ComboBox1: TComboBox;

butt_videodi: TButton;

butt_formatdi: TButton;

ComboBox2: TComboBox;

butt_sourcedi: TButton;

GroupBox2: TGroupBox;

Label2: TLabel;

TrackBar1: TTrackBar;

Convertrgb: TPanel;

butt_convert: TButton;

Label16: TLabel;

Edit1: TEdit;

Edit2: TEdit;

Edit3: TEdit;

Label17: TLabel;

Label18: TLabel;

Label19: TLabel;

Label20: TLabel;

Label15: TLabel;

Panel5: TPanel;

Image2: TImage;

Panel6: TPanel;

butt_forward: TButton;

butt_stop: TButton;

butt_right: TButton;

butt_left: TButton;

```
butt_backward: TButton;
Panel7: TPanel;
butt_openport: TButton;
butt_robota: TButton;
butt_robotb: TButton;
butt_settingport: TButton;
butt_kick: TButton;
butt_forslow: TButton;
Label23: TLabel;
Panel8: TPanel;
Label3: TLabel;
Label6: TLabel;
Label7: TLabel;
Label9: TLabel;
Label10: TLabel;
Label12: TLabel;
Label13: TLabel;
Label14: TLabel;
Label21: TLabel;
Label22: TLabel;
Button6: TButton;
butt_clear: TButton;
ComLed1: TComLed;
Label5: TLabel;
ComLed2: TComLed;
Label11: TLabel;
SpeedButton2: TSpeedButton;
Button1: TButton;
procedure FormActivate(Sender: TObject);
procedure ImageEnView1DShowNewFrame(Sender: TObject);
procedure SpeedButton1Click(Sender: TObject);
procedure butt_videodiClick(Sender: TObject);
```

```
procedure butt_formatdiClick(Sender: TObject);
procedure TrackBar1Change(Sender: TObject);
procedure SaveAs1Click(Sender: TObject);
procedure ComboBox1Change(Sender: TObject);
procedure ComboBox2Change(Sender: TObject);
procedure Edit3Change(Sender: TObject);
procedure butt_sourcediClick(Sender: TObject);
procedure Button6Click(Sender: TObject);
procedure butt_convertClick(Sender: TObject);
procedure butt_openportClick(Sender: TObject);
procedure butt_settingportClick(Sender: TObject);
procedure butt_forwardClick(Sender: TObject);
procedure butt_leftClick(Sender: TObject);
procedure butt_rightClick(Sender: TObject);
procedure butt_backwardClick(Sender: TObject);
procedure butt_stopClick(Sender: TObject);
procedure butt_kickClick(Sender: TObject);
procedure butt_forslowClick(Sender: TObject);
procedure butt_robotaClick(Sender: TObject);
procedure butt_robotbClick(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure SpeedButton2Click(Sender: TObject);
procedure butt_clearClick(Sender: TObject);
procedure Button1Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
procedure Connect;
procedure Disconnect;
procedure ShowVideoFormats;
procedure Delay(msecs:integer);
```

```
end;
var
fmain: Tfmain;
i,j,ii,jj,x,y,k : integer;
z,r1,g1,b1 : array[0..400,0..400] of integer;
bluex,bluey,lightbx,lightby,orangex,orangey,greenx,greeny,lightgx,lightgy : integer;
select : string;
a,b,c,d,e,f :real;
str : string ;
implementation
{$R *.DFM}
procedure Tfmain.Delay(msecs:integer);
var
FirstTickCount:longint;
begin
FirstTickCount:=GetTickCount;
repeat
Application.ProcessMessages;
until ((GetTickCount-FirstTickCount) >= Longint(msecs));
end;
procedure Tfmain.FormActivate(Sender: TObject);
begin
// Fill video source combobox
ComboBox1.Items.Assign( ImageEnView1.IO.DShowParams.VideoInputs );
// Select first item
ComboBox1.ItemIndex:=0;
//
ShowVideoFormats;
end;
// We have got a new frame
procedure Tfmain.ImageEnView1DShowNewFrame(Sender: TObject);
begin
```

```
// copy current sample to ImageEnView bitmap
ImageEnView1.IO.DShowParams.GetSample(ImageEnView1.Layers[0].Bitmap);
// refresh ImageEnView1
ImageEnView1.Update;
end;
procedure Tfmain.Connect;
begin
if (not ImageEnView1.IO.DShowParams.Connected) then begin
// set video source as index of IO.DShowParams.VideoInputs
ImageEnView1.IO.DShowParams.SetVideoInput(ComboBox1.ItemIndex);
// enable frame grabbing
ImageEnView1.IO.DShowParams.EnableSampleGrabber:=true;
// connect to the video input
ImageEnView1.IO.DShowParams.Connect;
imageenview1.io.dshowparams.SaveGraph('c:\1.grf');
end;
end;
procedure Tfmain.Disconnect;
begin
// stop and disconnect
ImageEnView1.IO.DShowParams.Disconnect;
end;
// Capture button
procedure Tfmain.SpeedButton1Click(Sender: TObject);
var
w,h:integer;
f:string;
begin
if SpeedButton1.Down then begin
Connect;
// set video size
ImageEnView1.IO.DShowParams.SetCurrentVideoFormat(w,h,'');
```

```
// show info
ImageEnView1.IO.DShowParams.GetCurrentVideoFormat(w,h,f);
Label4.Caption:='Capturing at '+inttostr(w)+'x'+inttostr(h)+' '+f;
// start capture
ImageEnView1.IO.DShowParams.Run;
end else begin
Disconnect;
end;
end;
// video dialog
procedure Tfmain.butt_videodiClick(Sender: TObject);
begin
Connect;
ImageEnView1.IO.DShowParams.ShowPropertyPages(iepVideoInput,ietFilter);
end;
// Video source dialog
procedure Tfmain.butt_sourcediClick(Sender: TObject);
begin
Connect;
ImageEnView1.IO.DShowParams.ShowPropertyPages(iepVideoInputSource,ietFilter);
end;
// format dialog
procedure Tfmain.butt_formatdiClick(Sender: TObject);
begin
Connect;
ImageEnView1.IO.DShowParams.ShowPropertyPages(iepVideoInput,ietOutput);
end;
// Zoom
procedure Tfmain.TrackBar1Change(Sender: TObject);
begin
ImageEnView1.Zoom:=TrackBar1.Position;     //ZOOM
end;
```

```
// Save as
procedure Tfmain.SaveAs1Click(Sender: TObject);
begin
ImageEnView1.IO.SaveToFile( ImageEnView1.IO.ExecuteSaveDialog("",false,1,") );
end;
procedure Tfmain.ShowVideoFormats;
var
i:integer;
s:string;
begin
Connect;
// fills video formats list box (informative only box)
with ImageEnView1.IO.DShowParams do
for i:=0 to VideoFormatsCount-1 do begin
with VideoFormats[i] do
s:=Format + ' '+inttostr(MinWidth)+'x'+inttostr(MinHeight)+' to
'+inttostr(MaxWidth)+'x'+inttostr(MaxHeight);
end;
// fills video source inputs
ComboBox2.Items.Assign( ImageEnView1.IO.DShowParams.VideoInputSources );
// set current video source input
ComboBox2.ItemIndex:=ImageEnView1.IO.DShowParams.VideoInputSource;
Disconnect;
end;
// changes video source
procedure Tfmain.ComboBox1Change(Sender: TObject);
begin
ShowVideoFormats;
end;
// set video source input
procedure Tfmain.ComboBox2Change(Sender: TObject);
begin
```

```
ImageEnView1.IO.DShowParams.VideoInputSource:=ComboBox2.ItemIndex;
end;
// change tuner channel
procedure Tfmain.Edit3Change(Sender: TObject);
begin
if ImageEnView1.IO.DShowParams.TunerFindSignal then begin
end else begin
end;
end;
procedure Tfmain.Button6Click(Sender: TObject);
var
w,h:integer;
g:string;
begin                    //run Programe
{####################}
{Process openpicture}
{####################}
//for  k:=0 to 50 do
//begin
{####################}
{Process search point}
{####################}
for i:=25to 290 do          // separate RGB
begin
for j:=33 to 210 do
begin
z[i,j] := ImageEnview1.IEBitmap.Canvas.Pixels[i,j];
r1[i,j]:= z[i,j] Mod 256;
g1[i,j]:= (z[i,j] div 256)Mod 256;
b1[i,j]:= z[i,j] div 65536;
end;end;
for i:=25 to 290 do   //search orange
```

```
begin
for j:=33 to 210 do
begin
if ((200<r1[i,j])and(r1[i,j]<255)) then
if ((68<g1[i,j])and(g1[i,j]<102)) then
if ((71<b1[i,j])and(b1[i,j]<126)) then
begin
Image2.Canvas.Pixels[i,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i-1,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i+1,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i,j-1] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i,j+1] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i-2,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
orangex:=i;orangey:=j;
end;
// Label10.Caption := inttostr(orangex)+','+inttostr(orangey);
end;end;
for i:=25 to 290 do   //search light blue
begin
for j:=33 to 210 do
begin
if ((90<r1[i,j])and(r1[i,j]<255)) then
if ((150<g1[i,j])and(g1[i,j]<255)) then
if ((230<b1[i,j])and(b1[i,j]<255))then
begin
Image2.Canvas.Pixels[i,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i-1,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i+1,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i,j-1] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i,j+1] := rgb(r1[i,j],g1[i,j],b1[i,j]);
lightbx:=i;lightby:=j;
end;
```

```
//Label14.Caption := inttostr(lightbx)+','+inttostr(lightby);
end;end;
for i:=25 to 290 do   //search Blue
begin
for j:=33 to 210 do
begin
if ((20<r1[i,j])and(r1[i,j]<80)) then
if ((70<g1[i,j])and(g1[i,j]<130)) then
if ((210<b1[i,j])and(b1[i,j]<255))then
begin
Image2.Canvas.Pixels[i,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i-1,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i+1,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i,j-1] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i,j+1] := rgb(r1[i,j],g1[i,j],b1[i,j]);
bluex:=i;bluey:=j;
end;
//Label12.Caption := inttostr(bluex)+','+inttostr(bluey);
end;end;
for i:=25  to 290 do   //search light green
begin
for j:=33 to 210 do
begin
if ((160<r1[i,j])and(r1[i,j]<200)) then
if ((160<g1[i,j])and(g1[i,j]<200)) then
if ((50<b1[i,j])and(b1[i,j]<120))then
begin
Image2.Canvas.Pixels[i,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i-1,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i+1,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i,j-1] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i,j+1] := rgb(r1[i,j],g1[i,j],b1[i,j]);
```

```
lightgx:=i;lightgy:=j;
end;
// Label13.Caption := inttostr(lightgx)+','+inttostr(lightgy);
end;end;
for i:=25  to 290 do    //search black green
begin
for j:=33 to 210 do
begin
if ((60<r1[i,j])and(r1[i,j]<110)) then
if ((140<g1[i,j])and(g1[i,j]<160)) then
if ((100<b1[i,j])and(b1[i,j]<150))then
begin
Image2.Canvas.Pixels[i,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i-1,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i+1,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i,j-1] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i,j+1] := rgb(r1[i,j],g1[i,j],b1[i,j]);
greenx:=i;greeny:=j;
end;
// Label22.Caption := inttostr(greenx)+','+inttostr(greeny);
end;end;
{#####################}
{PROCESS SOLVE DISTANCE}
{#####################}
b:=sqrt(sqr(orangex-lightgx)+sqr(orangey-lightgy));
a:=sqrt(sqr(orangex-lightbx)+sqr(orangey-lightby));
if a<b then     //chosse Robot
begin
c := sqrt(sqr(orangex-greenx)+sqr(orangey-greeny));
d:= abs(a-c) ;
if d > 10 then        //Assume check degree
begin
```

```
if a>c then          //Control Robot
begin
str :=  'a2'+#13#10;          //RobotA turnright
ComPort1.WriteStr(Str);
end  else
if a < c then
begin
str := 'a1'+#13#10;          //RobotA turnleft
Comport1.WriteStr(str);
end;
end ;
if d < 20 then
begin
if ((a < 35)or( c < 35)) then
begin
str := 'a5'+#13#10;  // RobotA stop
Comport1.WriteStr(str);
end else
begin
str := 'a7'+#13#10; //  RobotA Ago
Comport1.WriteStr(str);
end;
end;
end;
if a > b then
begin
f := sqrt(sqr(orangex-bluex)+sqr(orangey-bluey));
e:= abs(b-f) ;
if e > 10 then
begin
if b > f then
begin
```

```
str := 'b2'+#13#10;     // RobotB turn right
Comport1.WriteStr(str);
end  else
if b < f then
begin
str := 'b1'+#13#10; //  RobotB turn left
Comport1.WriteStr(str);
end;
end ;
if e < 20 then
begin
if ((b < 25)or(f < 25)) then
begin
delay(100);
str := 'b8'+#13#10;
Comport1.WriteStr(Str); //  RobotB Kick
delay(100);
str := 'b5'+#13#10;
Comport1.WriteStr(Str); //  RobotB stop
end;
if ((b > 30)or(f > 30)) then
begin
str := 'b7'+#13#10;// RobotB Ago
Comport1.WriteStr(str);
end;
end;
end;
//end;
end;
procedure Tfmain.butt_convertClick(Sender: TObject); //convert RGB
begin
 Label15.Caption := inttostr(rgb(strtoint(Edit1.Text),strtoint(Edit2.Text),strtoint(Edit3.Text)));
```

```
end;
procedure Tfmain.butt_openportClick(Sender: TObject);
begin
Comport1.Open;        //open serail port
end;
procedure Tfmain.butt_settingportClick(Sender: TObject);
begin
ComPort1.ShowSetupDialog;    //setting serail port
end;
procedure Tfmain.butt_forwardClick(Sender: TObject);
var
str : string;
begin                  //forward
if select = 'a'then
begin
str := 'a3'+#13#10;
ComPort1.WriteStr(Str);
end;
if select = 'b'then
begin
str := 'b3'+#13#10;
ComPort1.WriteStr(Str);
end;
end;
procedure Tfmain.butt_leftClick(Sender: TObject);
var
str : string;
begin
if select = 'a'then    //turn left
begin
str := 'a1'+#13#10;
ComPort1.WriteStr(Str);
```

```
end;
if select = 'b'then
begin
str := 'b1'+#13#10;
ComPort1.WriteStr(Str);
end;
end;
procedure Tfmain.butt_rightClick(Sender: TObject);
var str : string;
begin
if select = 'a'then        //turn right
begin
str := 'a2'+#13#10;
ComPort1.WriteStr(Str);
end;
if select = 'b'then
begin
str := 'b2'+#13#10;
ComPort1.WriteStr(Str);
end;
end;
procedure Tfmain.butt_backwardClick(Sender: TObject);
var str : string;
begin                      //backward
if select = 'a'then
begin
str := 'a4'+#13#10;
ComPort1.WriteStr(Str);
end;
if select = 'b'then
begin
str := 'b4'+#13#10;
```

```
ComPort1.WriteStr(Str);
end;
end;
procedure Tfmain.butt_stopClick(Sender: TObject);
var
str : string;
begin
if select = 'a'then     //stop
begin
str := 'a5'+#13#10;
ComPort1.WriteStr(Str);
end;
if select = 'b'then
begin
str := 'b5'+#13#10;
ComPort1.WriteStr(Str);
end;
end;
procedure Tfmain.butt_kickClick(Sender: TObject);
var
str : string;
begin
if select = 'a'then     //kick
begin
str := 'a8'+#13#10;
ComPort1.WriteStr(Str);
end;
if select = 'b'then
begin
str := 'b8'+#13#10;
ComPort1.WriteStr(Str);
end;
```

```
end;
procedure Tfmain.butt_forslowClick(Sender: TObject);
var
str : string;
begin
if select = 'a'then      //forwad slow
begin
str := 'a7'+#13#10;
ComPort1.WriteStr(Str);
end;
if select = 'b'then
begin
str := 'b7'+#13#10;
ComPort1.WriteStr(Str);
end;
end;
procedure Tfmain.butt_robotaClick(Sender: TObject);
begin
select := 'a';              //select robotA
label23.Caption :='A work';
end;
procedure Tfmain.butt_robotbClick(Sender: TObject);
begin
select := 'b';              //select RobotB
label23.Caption :='B work';
end;
procedure Tfmain.Button4Click(Sender: TObject);
begin
for  i:=0 to 340 do
for  j:= 0 to 280 do
Image2.Canvas.Pixels[i,j] := 0;
end;
```

```
procedure Tfmain.SpeedButton2Click(Sender: TObject);
var
w,h:integer;
g:string;
begin
Comport1.Open;
{################}
{ Process start }
{################}
while  SpeedButton2.Down do
begin
{#########}
{ Capture }
{#########}
delay(100);
{###################}
{Process search point}
{###################}
for i:=25to 290 do          // separate RGB
begin
for j:=33 to 210 do
begin
z[i,j] := ImageEnview1.IEBitmap.Canvas.Pixels[i,j];
r1[i,j]:= z[i,j] Mod 256;
g1[i,j]:= (z[i,j] div 256)Mod 256;
b1[i,j]:= z[i,j] div 65536;
end;
end;
for i:=25 to 290 do    //search orange
begin
for j:=33 to 210 do
begin
```

```
if ((200<r1[i,j])and(r1[i,j]<255)) then
if ((68<g1[i,j])and(g1[i,j]<102)) then
if ((71<b1[i,j])and(b1[i,j]<126)) then
begin
Image2.Canvas.Pixels[i,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i-1,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i+1,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i,j-1] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i,j+1] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i-2,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
orangex:=i;orangey:=j;
end;
// Label10.Caption := inttostr(orangex)+','+inttostr(orangey);
end;
end;
for i:=25 to 290 do  //search light blue
begin
for j:=33 to 210 do
begin
if ((90<r1[i,j])and(r1[i,j]<255)) then
if ((150<g1[i,j])and(g1[i,j]<255)) then
if ((230<b1[i,j])and(b1[i,j]<255))then
begin
Image2.Canvas.Pixels[i,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i-1,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i+1,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i,j-1] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i,j+1] := rgb(r1[i,j],g1[i,j],b1[i,j]);
lightbx:=i;lightby:=j;
end;
//Label14.Caption := inttostr(lightbx)+','+inttostr(lightby);
end;end;
```

```
for i:=25 to 290 do   //search Blue
begin
for j:=33 to 210 do
begin
if ((20<r1[i,j])and(r1[i,j]<80)) then
if ((70<g1[i,j])and(g1[i,j]<130)) then
if ((210<b1[i,j])and(b1[i,j]<255))then
begin
Image2.Canvas.Pixels[i,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i-1,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i+1,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i,j-1] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i,j+1] := rgb(r1[i,j],g1[i,j],b1[i,j]);
bluex:=i;bluey:=j;
end;
//Label12.Caption := inttostr(bluex)+','+inttostr(bluey);
end;
end;
for i:=25  to 290 do   //search light green
begin
for j:=33 to 210 do
begin
if ((160<r1[i,j])and(r1[i,j]<200)) then
if ((160<g1[i,j])and(g1[i,j]<200)) then
if ((50<b1[i,j])and(b1[i,j]<120))then
begin
Image2.Canvas.Pixels[i,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i-1,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i+1,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i,j-1] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i,j+1] := rgb(r1[i,j],g1[i,j],b1[i,j]);
lightgx:=i;lightgy:=j;
```

```
end;
// Label13.Caption := inttostr(lightgx)+','+inttostr(lightgy);
end;
end;
for i:=25  to 290 do    //search black green
begin
for j:=33 to 210 do
begin
if ((60<r1[i,j])and(r1[i,j]<110)) then
if ((140<g1[i,j])and(g1[i,j]<160)) then
if ((100<b1[i,j])and(b1[i,j]<150))then
begin
Image2.Canvas.Pixels[i,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i-1,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i+1,j] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i,j-1] := rgb(r1[i,j],g1[i,j],b1[i,j]);
Image2.Canvas.Pixels[i,j+1] := rgb(r1[i,j],g1[i,j],b1[i,j]);
greenx:=i;greeny:=j;
end;
// Label22.Caption := inttostr(greenx)+','+inttostr(greeny);
end;
end;
{########################}
{PROCESS SOLVE DISTANCE}
{########################}
b:=sqrt(sqr(orangex-lightgx)+sqr(orangey-lightgy));
a:=sqrt(sqr(orangex-lightbx)+sqr(orangey-lightby));
if a<b then     //chosse Robot
begin
c := sqrt(sqr(orangex-greenx)+sqr(orangey-greeny));
d:= abs(a-c) ;
if d > 10 then        //Assume check degree
```

```
begin
if a>c then          //Control Robot
begin
str :=  'a2'+#13#10;          //RobotA turnright
ComPort1.WriteStr(Str);
end  else
if a < c then
begin
str := 'a1'+#13#10;          //RobotA turnleft
Comport1.WriteStr(str);
end;
end ;
if d < 20 then
begin
if ((a < 35)or( c < 35)) then
begin
str := 'a5'+#13#10;  // RobotA stop
Comport1.WriteStr(str);
end else
begin
str := 'a7'+#13#10; //  RobotA Ago
Comport1.WriteStr(str);
end;
end;
end;
if a > b then
begin
f := sqrt(sqr(orangex-bluex)+sqr(orangey-bluey));
e:= abs(b-f) ;
if e > 10 then
begin
if b > f then
```

```
begin
str := 'b2'+#13#10;     // RobotB turn right
Comport1.WriteStr(str);
end  else
if b < f then
begin
str := 'b1'+#13#10; //  RobotB turn left
Comport1.WriteStr(str);
end;
end ;
if e < 20 then
begin
if ((b < 25)or(f < 25)) then
begin
delay(100);
str := 'b8'+#13#10;
Comport1.WriteStr(Str); //  RobotB Kick
delay(100);
str := 'b5'+#13#10;
Comport1.WriteStr(Str); //  RobotB stop
end;
if ((b > 30)or(f > 30)) then
begin
str := 'b7'+#13#10;// RobotB Ago
Comport1.WriteStr(str);
end;
end;
end;
end;
end;
procedure Tfmain.butt_clearClick(Sender: TObject);
begin
```

```
for i:=0 to 340 do
for j:=0 to 280 do
Image2.Canvas.Pixels[i,j]:=0;
end;
procedure Tfmain.Button1Click(Sender: TObject);
begin
Comport1.Close;
end;
end.
```

กิตติกรรมประกาศ

รายงานฉบับนี้สำเร็จลุล่วงได้ด้วยความช่วยเหลือจากบุคคลหลายฝ่าย ทางคณะผู้จัดทำขอกราบพระคุณเป็นอย่างสูงไว้ ณ ที่นี้ โดยเฉพาะอย่างยิ่ง ผศ.ดร.ชูชาติ ปิณฑวิรุจน์ และ รศ.ดร.มนัส สังวรศิลป์ ที่ให้คำปรึกษาและคำแนะนำในโครงการชิ้นนี้ ขอขอบคุณรุ่นพี่นักศึกษาปริญญาโททุกท่านความช่วยเหลือ และกำลังใจจากเพื่อนๆที่ทำให้รายงานฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี

นายวรพงษ์ รุ่งวสันติสุข
นายวรวิทย์ มงคลนิตย์

เอกสารอ้างอิง

1. อภิชาติ ภู่พลับ, สัจจะ จรัสรุ่งรวีวง, "เริ่มต้นเขียนโปรแกรมติดต่อและควบคุมฮาดแวร์ด้วย Visual Basic"

2. สัจจะ จรัสรุ่งรวีวร, จักรพงษ์ สุขประเสริฐ, "เริ่มต้นอย่างมืออาชีพด้วย Delphi 7 "

3. ประกิจ ตังติสานนท์ , "วิศวกรรมการสื่อสาร (ไฟฟ้า-อิเล็กทรอนิกส์)" , คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, พิมพ์ครั้งที่ 4 กรุงเทพ 2539

4. ประจิน พลังสันติกุล , "เรียนรู้และใช้งาน CCS C Compiler เขียนโปรแกรมภาษา C ควบคุมไมโครคอนโทรลเลอร์"

5. ผศ.ดร. ชูชาติ ปิณฑวิรุจน์ , "เอกสารประกอบการสอน วิชา Digital Image Processing"