การติดตามวัตถุ

OBJECT TRACKING

นายอภิชาติ สุทธิธรรมานนท์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมการวัดคุม
ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2550

OBJECT TRACKING

APICHAT SUTTHITHAMANON

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

DEPARTMENT OF INSTRUMENTATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUE OF TECHNOLOGY LARDKRABANG

2007

ภาควิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

.....

หัวข้อปริญญานิพนธ์ การติดตามวัตถุ

OBJECT TRACKING

นักศึกษาผู้จัดทำ นายอภิชาติ สุทธิธรรมานนท์ **รหัสนักศึกษา** 47012084

ปริญญา วิศวกรรมศาสตรบัณฑิต

สาขาวิชา วิศวกรรมการวัดคุม

ปีการศึกษา 2550

อาจารย์ผู้ควบคุมปริญญานิพนธ์	ลายมือชื่อ
รศ. เกษตร์ ศิริสันติสัมฤทธิ์	

หัวข้อปริญญานิพนธ์ การติดตามวัตถุ

OBJECT TRACKING

นักศึกษาผู้จัดทำ นายอภิชาติ สุทธิธรรมานนท์ รหัสนักศึกษา 47012084

อาจารย์ที่ปรึกษา รศ. เกษตร์ ศิริสันติสัมฤทธิ์

ปีการศึกษา 2550

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ นำเสนอการติดตามการเคลื่อนที่ของวัตถุด้วยเทคนิคการประมวลผลภาพ ขั้นตอนการติดตามวัตถุประกอบด้วยการเก็บภาพพื้นหลังที่ตำแหน่งต่าง ๆ ในขอบเขตพื้นที่งาน การลบภาพวัตถุด้วยภาพพื้นหลังและหาจุดศูนย์กลางของวัตถุ และสั่งให้เซอร์โวมอเตอร์ที่มีกล้องติดอยู่หมุนไปยังตำแหน่งต่าง ๆ เมื่อจุดศูนย์กลางของวัตถุอยู่ในเงื่อนไขที่กำหนด ในผลการทดลองวิธีที่นำเสนอสามารถตรวจหาวัตถุที่กำลังเคลื่อนที่ในภาพ และสามารถติดตามวัตถุได้เพียงวัตถุเดียว

Thesis Title Object Tracking

Authors Mr. Apichat Sutthithamanon

Thesis Advisor Asso. Prof Kaset Sirisantisamrid

Year 2007

ABSTRACT

This Thesis proposes object tracking using image processing technique. The procedure of object tracking consists of capture the several background image in working space, subtract the background image from the object image and compute the center of object, and send the command to dc servo motor attached with camera to rotate the camera to some position if the center of object is satisfied with the specified conditions. In experimental results, the proposed method can be detected the objects in images and tracked only one object.

กิตติกรรมประกาศ

ขอขอบพระคุณ รศ. เกษตร์ ศิริสันติสัมฤทธิ์ ที่ช่วยให้คำปรึกษา แนะนำและกระตุ้นเตือนในการทำปริญญานิพนธ์ฉบับนี้จนสำเร็จลุล่วงไปได้ด้วยดี

ขอขอบคุณ อ. ธีรวัฒน์ เทพมณี ที่ช่วยให้คำแนะนำต่าง ๆ ที่เกี่ยวกับการประมวลผลภาพ

ขอบอบคุณอาจารย์และเจ้าหน้าที่ภาควิชาวิศวกรรมการวัดคุมทุก ๆ ท่านคอยช่วยเหลือและให้คำปรึกษาที่ดีตลอดมา

ขอขอบคุณ คุณ สิทธิชัย วงศ์เสน ที่ช่วยเหลือทางด้านฮาร์ดแวร์ต่าง ๆ

ขอขอบคุณผู้แต่งหนังสือทุกท่านที่ได้นำมาอ้างอิงในการทำปริญญานิพนธ์ฉบับนี้

ผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

ภาพที่ **หน้า**

สารบัญภาพ(ต่อ)

ภาพที่ **หน้า**

สารบัญภาพ(ต่อ)

ภาพที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ระบบรักษาความปลอดภัย โดยทั่วไปแล้วมักจะติดตั้งกล้องโทรทัศน์วงจรปิดสำหรับการบันทึกภาพเหตุการณ์ในแต่ละช่วงเวลา เพื่อตรวจสอบความเคลื่อนไหวของบุคคลในสถานที่ต่าง ๆ แต่ข้อจำกัดของการใช้กล้องโทรทัศน์วงจรปิด (CCTV) เพียง 1 ตัว ย่อมทำให้สามารถมองเห็นภาพได้ในมุมมองที่จำกัดและไม่ทั่วถึง จึงมีความจำเป็นที่จะต้องใช้กล้องโทรทัศน์วงจรปิดหลายตัวเพื่อติดตั้งในหลาย ๆ จุด เพื่อให้สามารถเก็บรายละเอียดได้ครอบคลุมทั่วทั้งบริเวณที่ต้องการ

งานวิจัยนี้จึงได้มีแนวคิดที่จะพัฒนาขีดความสามารถของกล้องวงจรปิดเพื่อให้สามารถที่จะทำการตรวจจับภาพในมุมมองที่กว้างขึ้น และสามารถบันทึกรายละเอียดของเหตุการณ์ต่าง ๆ ได้ครอบคลุมยิ่งขึ้น โดยการติดตั้งเซอร์โวมอเตอร์ไว้กับกล้องเพื่อหมุนจับภาพในมุมที่กว้างขึ้น และใช้จำนวนของกล้องที่น้อยลง

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

แนวความคิดพื้นฐานที่ใช้สำหรับการศึกษาเกี่ยวกับการติดตามวัตถุที่กำลังเคลื่อนที่ คือการลบภาพ 2 ภาพ ที่มีพื้นหลัง (Background) เหมือนกันแต่วัตถุ (Object) อยู่ในตำแหน่งที่ต่างกันเมื่อทำการลบภาพทั้ง 2 ภาพ ดังภาพที่ 1.1 (ก) และ (ข) จะทำให้พื้นหลังในภาพผลลัพธ์กลายเป็นสีดำจนหมด เหลือเพียงแต่วัตถุในภาพเท่านั้น อย่างไรก็ตามยังคงปรากฏวัตถุ 2 วัตถุอยู่ในภาพ ดังแสดงในภาพที่ 1.1 (ค) ด้วยเหตุนี้จึงได้มีความคิดที่ว่า ทำอย่างไรเมื่อลบภาพที่มีพื้นหลังเหมือนกันแล้วภาพผลลัพธ์ที่ได้รับปรากฏวัตถุที่สนใจในภาพเพียง 1 วัตถุเท่านั้น

(ก)

(ข)

(ค)

ภาพที่ 1.1 ตัวอย่างของปัญหาที่เกิดขึ้น (ก) วัตถุเข้ามาในระยะมองเห็นกล้อง (ข) ภาพวัตถุขณะเคลื่อนที่ไปที่ตำแหน่งอื่น (ค) ผลลัพธ์จากการลบภาพปรากฏ 2 วัตถุในภาพ

1.3 ทฤษฎีหรือแนวความคิดที่ใช้ในการวิจัย

แนวความคิดสำหรับแก้ไขปัญหาที่เกิดขึ้นคือ การเก็บภาพที่มีแต่ภาพพื้นหลังไว้ก่อน จากนั้นก็จึงเก็บภาพวัตถุที่มีพื้นหลังเหมือนกัน เมื่อนำภาพทั้งสองมาลบกันจะได้ภาพผลลัพธ์ ดังภาพที่ 1.2 (ค) จะเห็นว่าภาพผลลัพธ์ที่ได้จะเหลือเพียง 1 วัตถุที่สนใจ ซึ่งจะทำให้สามารถลดขั้นตอนต่างๆ ลงได้มากกว่าการลบภาพแล้วเหลือ 2 วัตถุ ซึ่งจะต้องนำภาพมากระทำการแอนด์ (AND) กันทางโลจิก เพื่อทำให้วัตถุในภาพเหลือเพียง 1 วัตถุ อย่างไรก็ตาม การกระทำดังกล่าวข้างต้น ไม่สามารถรับประกันได้ว่าจะเหลือวัตถุในภาพเพียงวัตถุเดียว ทั้งนี้อาจจะเป็นเพราะความสว่างของภาพมีความแตกต่างกันในการถ่ายภาพแต่ละครั้ง

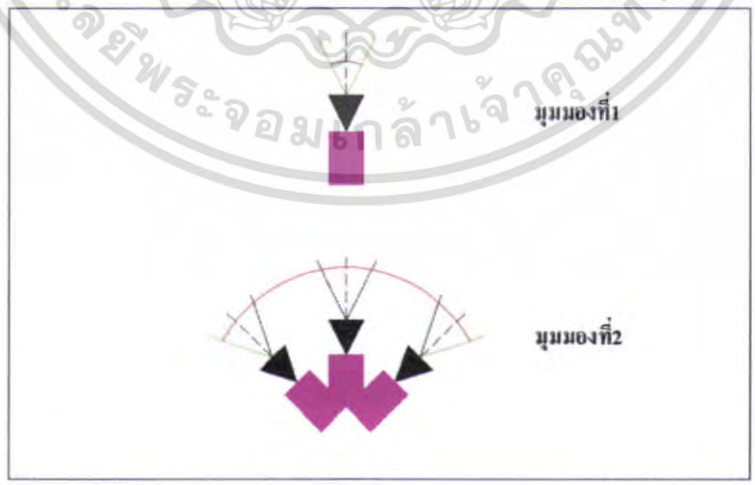
(ก)

(ข)

(ค)

ภาพที่ 1.2 ตัวอย่างของการแก้ปัญหา (ก) ภาพพื้นหลังขณะไม่มีวัตถุเข้ามาในมุมมองของกล้อง (ข) วัตถุเข้ามาในมุมมองของกล้อง (ค) ผลลัพธ์จากการลบภาพปรากฏวัตถุ 1 วัตถุในภาพ

การเพิ่มมุมมองเพื่อให้สามารถเก็บรายละเอียดของการติดตามวัตถุได้มากขึ้น ในส่วนนี้เองได้ติดตั้งเซอร์โวมอเตอร์(servo motor) เข้าไปในงานวิจัยเพื่อหมุนกล้องติดตามวัตถุไปในทิศทางต่างๆ ทำให้สามารถเก็บรายละเอียดได้มากขึ้น ดังภาพที่ 1.3 เห็นได้จากเส้นโค้งสีแดงซึ่งแสดงมุมมองของกล้อง ในมุมมองที่ 1 กล้องจะไม่มีการเคลื่อนที่ เนื่องจากไม่มีการติดตั้งเซอร์โวมอเตอร์เข้าไปทำให้มุมมองของกล้องจะมีค่าประมาณ 79 องศา (สเปคกล้องที่นำมาใช้งานในงานวิจัย) และในมุมมองที่ 2 เมื่อติดตั้งเซอร์โวมอเตอร์เข้าไป ทำให้กล้องสามารถเคลื่อนที่เพื่อเพิ่มมุมมองทำให้สามารถบันทึกรายละเอียดได้มากขึ้น เห็นได้จากเส้นโค้งสีแดงที่มีมุมมองกว้างขึ้น



ภาพที่ 1.3 การเพิ่มมุมมองของกล้องโดยการติดตั้งเซอร์โวมอเตอร์

1.4 ขอบเขตการวิจัย

ขอบเขตของการศึกษาการติดตามวัตถุ มีดังนี้

1. สามารถตรวจหาและติดตามวัตถุที่มีการเคลื่อนที่อยู่ในพื้นที่ทำงาน (Working space) ของกล้องได้อย่างถูกต้อง

2. สามารถตรวจจับวัตถุที่มีความเร็วไม่มากนัก โดยโปรแกรมจะทำการประมวลผลทุก ๆ 20 เฟรมต่อวินาที

3. สามารถตรวจจับวัตถุได้ไม่เกิน 2 วัตถุ

4. กล้องที่นำใช้ในงานวิจัยเป็นกล้องเวบแคม (webcam) แทนการใช้กล้องโทรทัศน์วงจรปิด (CCTV)

บทที่ 2
ทฤษฎีการประมวลผลภาพเบื้องต้น

2.1 การมองเห็นของมนุษย์และคอมพิวเตอร์

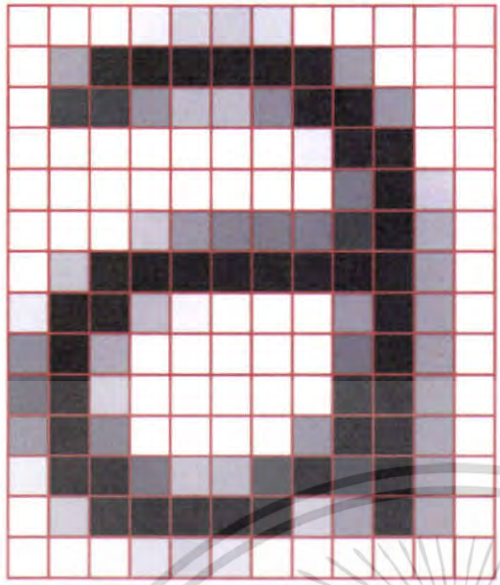
รูปภาพทั่วไปไม่ว่าจะเป็นภาพที่ถ่ายโดยใช้กล้องธรรมดาหรือกล้องแบบดิจิตอล ในมุมมองของมนุษย์และคอมพิวเตอร์ย่อมมีความแตกต่างกัน คือมนุษย์สามารถรับรู้ได้ว่าภาพที่ปรากฏอยู่นั้นต้องการจะสื่อถึงอะไร เพราะว่าภาพที่มองเห็นด้วยตาของมนุษย์จะมีทั้งความกว้าง ความยาวและความลึก ส่วนคอมพิวเตอร์จะมองเห็นภาพเพียง 2 มิติเท่านั้น คือภาพมีความกว้างและความยาวและแสดงความหมายต่าง ๆ โดยการแทนด้วยจุดสี ซึ่งเป็นค่าสีต่าง ๆ กันนำมาเรียงต่อ ๆ กันเพื่อที่จะสื่อถึงความหมายของภาพ

2.2 การแปลงภาพให้เป็นภาพเชิงดิจิตอล

ภาพเป็นกระบวนการทางแสง (Optical Process) ซึ่งเกิดจากพลังงานคลื่นแม่เหล็กไฟฟ้าหลายๆ ช่วงความถี่ เช่น แสงธรรมดา รังสีเอ็กเรย์ (X-Ray) รังสีอินฟาเรด (Infared) เป็นต้น และพลังงานเสียง เช่น อัลตราซาวด์ (Untrasound) ตกกระทบวัตถุแล้วสะท้อนกลับมาสู่ประสาทรับรู้ทางตาของมนุษย์หรืออุปกรณ์ตรวจจับ เช่น เซนเซอร์ (Sensor) เป็นต้น

ภาพดิจิตอลคือ ฟังก์ชัน 2 มิติ หรือ $f(x,y)$ ของค่าความเข้มของแสง โดยที่ x และ y คือค่าที่บอกถึงตำแหน่งในระบบพิกัดฉาก และค่าของฟังก์ชันที่ตำแหน่งใด ๆ จะเป็นสัดส่วนกับความสว่างของแสงที่ตำแหน่งนั้น ส่วนกระบวนการแปลงภาพให้เป็นภาพในเชิงดิจิตอลเราเรียกว่า Image Digitization มีกระบวนการ 3 ขั้นตอน คือ การนำภาพเข้ามา (Image Acquisition) การสุ่มเลือกจุดตำแหน่ง (Image Sampling) และการประมาณค่าความเข้มของแสง (image Quantization) ภาพที่ 2.1 แสดงการแทนภาพด้วยเมทริกซ์ โดยภาพทั้ง 2 ได้ถูกตีตารางเพื่อระบุตำแหน่งที่ตรงกันของจุดสีหรือพิกเซล โดยที่ภาพ 2.1 (ก) จะแสดงเฉพาะจุดสีที่ปรากฏ ส่วนภาพ 2.1 (ข) จะแสดงค่าประจำจุดสีที่ตำแหน่งเดียวกันของตัวอักษร a สังเกตเห็นว่าภาพตัวอักษร a ถ้าที่ตำแหน่งพิกเซลใดเป็นสีดำ ที่ตำแหน่งเดียวของภาพ 2.1 (ข) ก็จะแทนด้วยเป็น 0 และในทำนองเดียวกันถ้าตำแหน่งใดเป็นพิกเซลสีขาวก็จะถูกแทน 1 ส่วนสีอื่น ๆ ค่าตัวเลขก็จะลดลงหรือเพิ่มขึ้นตามความเข้มหรือความสว่างของสี

a



1.0	1.0	1.0	0.9	0.6	0.6	0.6	1.0	1.0	1.0	1.0	1.0
1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0	1.0	1.0
1.0	0.2	0.2	0.5	0.6	0.6	0.5	0.0	0.0	0.5	1.0	1.0
1.0	0.9	1.0	1.0	1.0	1.0	1.0	0.9	0.0	0.0	0.9	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
1.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5	0.4	0.0	0.5	1.0
1.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0
0.9	0.0	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.6	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.7	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.5	1.0
0.6	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.0	0.5	1.0
0.9	0.1	0.0	0.6	0.7	0.7	0.5	0.0	0.5	0.0	0.5	1.0
1.0	0.7	0.1	0.0	0.0	0.0	0.1	0.9	0.8	0.0	0.5	1.0
1.0	1.0	1.0	0.8	0.8	0.9	1.0	1.0	1.0	1.0	1.0	1.0

(ก) (ข)

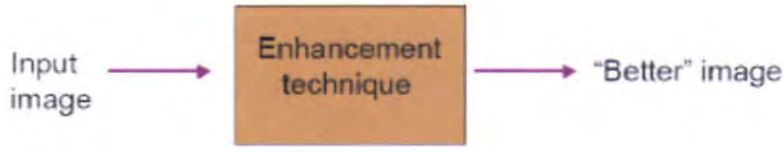
ภาพที่ 2.1 การแทนภาพด้วยเมทริกซ์

2.3 การปรับปรุงคุณภาพของภาพ (Image Enhancement)

Image Enhancement คือกระบวนการปรับปรุงภาพให้ดีขึ้นเพื่อประโยชน์การแปลภาพด้วยการมองด้วยตา (Visual Interpretation) โดยที่ไม่มีการเปลี่ยนแปลงเนื้อภาพ สามารถแบ่งได้เป็น 2 domains ได้แก่ Spatial Domain และ Frequency Domain สำหรับในหัวข้อนี้จะได้กล่าวถึง Image Enhancement แต่ใน Spatial Domain โดย Image Enhancement ซึ่งสามารถอธิบายด้วยสมการที่ (2.1)

$$f'(x,y) = S(f(x,y)) \qquad (2.1)$$

โดย f(x,y) คือ ภาพใด ๆ และ S คือ กระบวนการเปลี่ยนแปลงใด ๆ โดยที่ไม่มีการเปลี่ยนแปลงเนื้อภาพ (Image Content) ซึ่งภาพผลลัพธ์ที่ได้จะเก็บไว้ใน f'(x,y)



ภาพที่ 2.2 กระบวนการ Image Enhancement

Noise คือ สิ่งรบกวนในภาพ ทำให้ภาพที่ได้มาไม่ชัดเจนซึ่งเกิดจากความเข้มของแสง (Gray Scale) ที่มีค่าไม่เท่ากัน เช่นจุดเล็ก ๆ ที่เกิดขึ้นในภาพ ดังภาพที่ 2.3

(ก) (ข)

ภาพที่ 2.3 ภาพที่มี Noise (ก) รูปที่มี noise (ข) รูปที่มีการทำ Noise Reduction

ฮิสโตแกรม คือกราฟที่แสดงจำนวนพิกเซลของข้อมูลภาพ (แกน y) ตามค่าระดับความเข้มสีเทา (แกน x) ที่ปรากฏหรือแสดงอยู่บนภาพดิจิตอลใด ๆ ตัวอย่างของฮิสโตแกรมของภาพที่ 2.4 (ก) แสดงได้ดังภาพที่ 2.4 (ข)

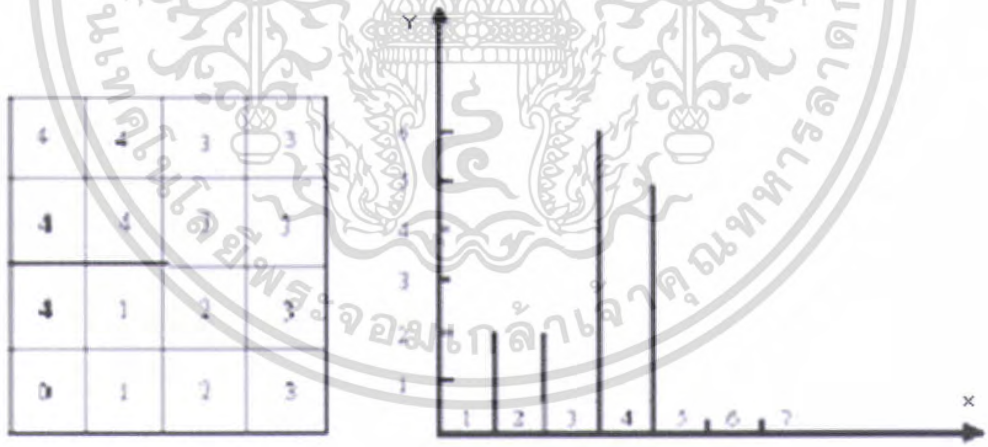
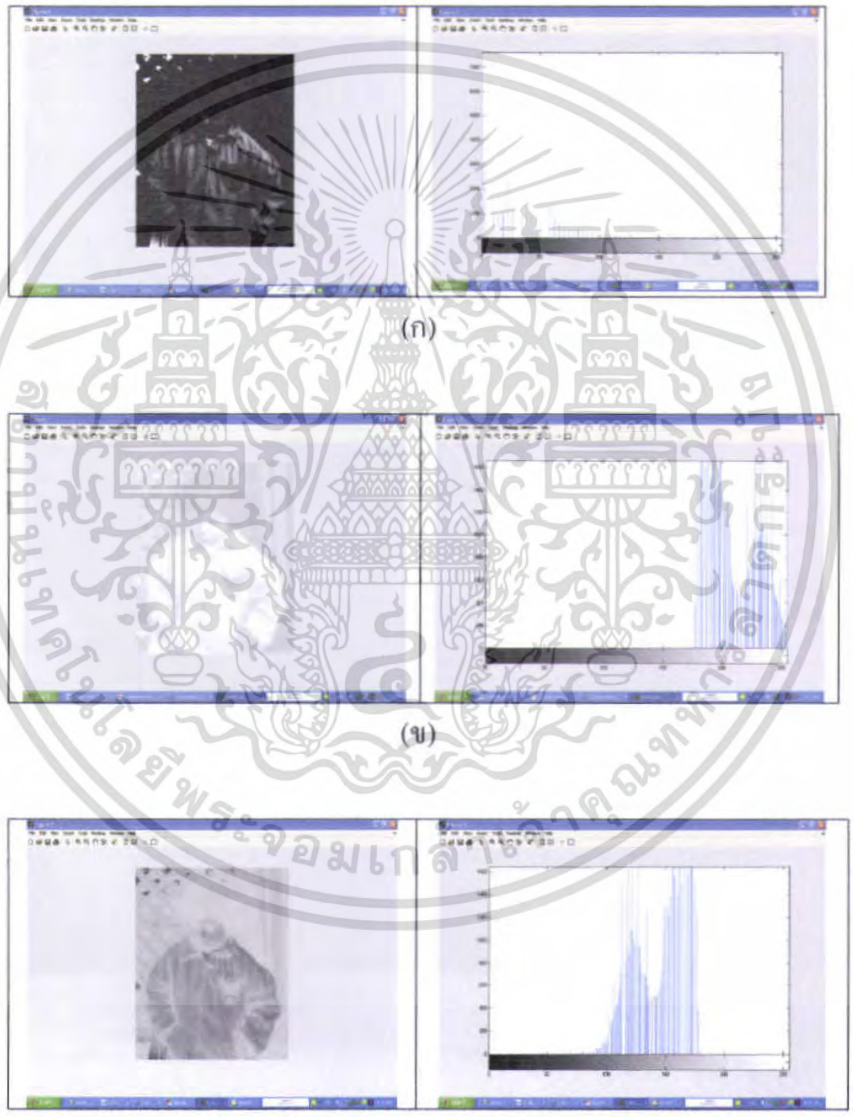


Image Pixel Intensity

(ก) (ข)

ภาพที่ 2.4 ฮิสโตแกรมของภาพ

เมื่อพิจารณาถึงฮิสโตแกรมของภาพ ก็จะสามารถที่จะแยกแยะประเภทของภาพจากกราฟของฮิสโตแกรมที่แสดงการกระจายของข้อมูลได้ดังตัวอย่างในภาพที่ 2.5 (ก) จากภาพจะเห็นได้ว่าภาพมีความมืด ดังนั้น ฮิสโตแกรมของภาพจะรวมกันเป็นกลุ่มอยู่ที่บริเวณที่ค่าระดับความเข้มสีเทาในช่วงบริเวณที่มีค่าต่ำ ส่วนในภาพที่ 2.5 (ข) จากภาพจะเห็นได้ว่าภาพมีความสว่างมาก ดังนั้น ฮิสโตแกรมของภาพจะรวมกันเป็นกลุ่มอยู่ที่บริเวณที่ค่าระดับความเข้มสีเทาในช่วงบริเวณที่มีค่าสูง ซึ่งภาพทั้งสองมีค่าความสว่างของภาพต่ำ และสูงจนเกินไปจึงทำให้ภาพดูไม่สมบูรณ์เท่าที่ควร และในภาพที่ 2.5 (ค) แสดงภาพและฮิสโตแกรมของภาพที่มองดูไม่ชัด



(ก)

(ข)

(ค)

ภาพที่ 2.5 ฮิสโตแกรมของภาพชนิดต่าง ๆ (ก) ภาพและฮิสโตแกรมของภาพที่มีความมืดจนเกินไป (ข) ภาพและฮิสโตแกรมของภาพที่มีความสว่างจนเกินไป (ค) ภาพและฮิสโตแกรมของภาพที่ไม่ชัดเจน

ในภาพที่ 2.6 จะแสดงตัวอย่างของภาพที่มีค่า Contrast ที่เหมาะสม หรือ ภาพที่ไม่มีความสว่างหรือมืด จนเกินไป ซึ่งเราจะเห็นได้ว่าฮิสโตแกรมของข้อมูลภาพจะมีการกระจายตลอดครอบคลุมทุกระดับค่าระดับสีเทา (Gray Scale)

ภาพที่ 2.6 ภาพที่ไม่มีความสว่างหรือมืดจนเกินไป

สำหรับในส่วนนี้จะอธิบายการปรับปรุงคุณภาพของภาพโดยอาศัยพื้นฐานของฮิสโตแกรม ซึ่งการกระจายข้อมูลของฮิสโตแกรมที่สมบูรณ์ควรเป็นรูประฆังคว่ำ ดังนั้นการทำการปรับปรุงคุณภาพของภาพ โดยอาศัยพื้นฐานฮิสโตแกรม โดยการแปลงฮิสโตแกรมเดิมของภาพให้มีความใกล้เคียงรูประฆังคว่ำที่มีความสมมาตร การปรับปรุงคุณภาพอธิบายได้ดังนี้

Image stretching คือ การขยายย่านฮิสโตแกรมเป็นย่านใหม่ที่ต้องการ โดยสมการดังนี้

$$j = \left[\frac{(\text{new max}) - (\text{new min})}{(\text{old max}) - (\text{old min})}\right] \times \left(i - (\text{old min})\right) + \text{new min} \qquad (2.2)$$

โดยที่ i คือ ค่าระดับความเข้มสีเทาเดิมของภาพ

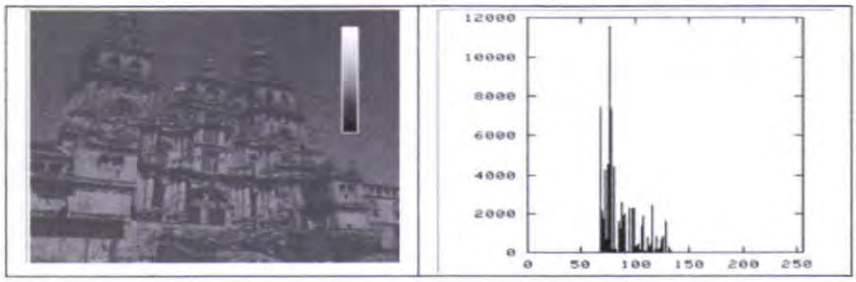
j คือ ค่าระดับความเข้มสีเทาใหม่ของข้อมูลภาพ หลังจาการทำ Histogram Stretching

oldmin คือ ค่าระดับความเข้มสีเทาต่ำสุดของข้อมูลภาพเดิม

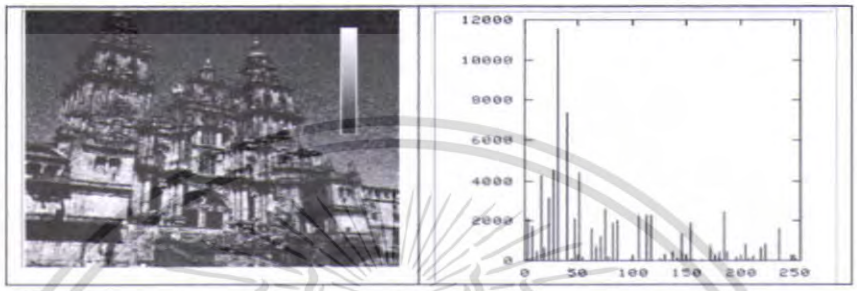
oldmax คือ ค่าระดับความเข้มสีเทาสูงสุดของข้อมูลภาพเดิม

newmin คือ ค่าระดับความเข้มสีเทาต่ำสุดที่เราต้องการของข้อมูลภาพใหม่

newmax คือ ค่าระดับความเข้มสีเทาสูงสุดที่เราต้องการของข้อมูลภาพใหม่



(ก)



(ข)

ภาพที่ 2.7 Histogram Stretching (a) ภาพต้นแบบและฮิสโตแกรมก่อนทำ Histogram Stretching (b) ภาพและฮิสโตแกรมที่ผ่านการทำ Histogram Stretching

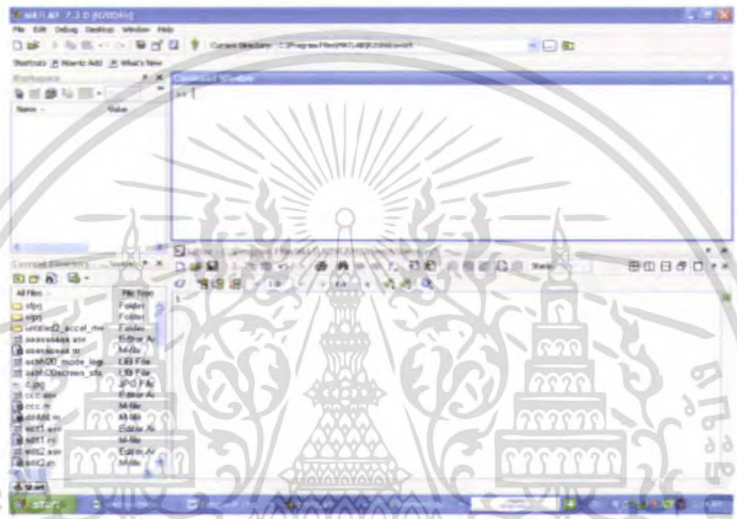
2.4 การประมวลผลภาพดิจิตอลด้วยโปรแกรม MATLAB

MATLAB เป็นโปรแกรมสำเร็จรูปที่นิยมใช้กันอย่างแพร่หลายในแวดวงของวิทยาศาสตร์และวิศวกรรมในปัจจุบัน โดยโปรแกรม MATLAB ได้เริ่มต้นขึ้นเพื่อต้องการให้ผู้ใช้สามารถแก้ปัญหา ตัวแปรที่มีลักษณะเป็น Matrix ได้ง่ายขึ้น ทั้งในด้านการคำนวณทั่วไป การสร้างแบบจำลองและการทดสอบแบบจำลอง การวิเคราะห์ข้อมูล การแสดงผลในรูปกราฟ และทางด้านวิทยาศาสตร์และวิศวกรรม สามารถสร้างโปรแกรมในลักษณะที่ติดต่อกับผู้ใช้ทางกราฟฟิก

การทำงานของโปรแกรม MATLAB จะสามารถทำงานได้ทั้งในลักษณะของการติดต่อโดยตรง (Interactive) คือการเขียนคำสั่งเข้าไปทีละคำสั่งเพื่อให้โปรแกรม MATLAB ประมวลผลไปเรื่อยๆหรือสามารถที่จะรวบรวมชุดคำสั่งเหล่านั้นเป็นโปรแกรมก็ได้ ข้อสำคัญอย่างหนึ่งของโปรแกรม MATLAB ก็คือข้อมูลทุกตัวจะถูกเก็บในลักษณะของ array คือในแต่ละตัวแปรจะได้รับการแบ่งเป็นส่วนย่อยเล็ก ๆ ขึ้น (หรือจะได้รับการแบ่งเป็น element นั่นเอง) ซึ่งการใช้ตัวแปรเป็น array ในโปรแกรม Matlab นี้ผู้ใช้ไม่จำเป็นจะต้องจอง dimension เหมือนกับการเขียนโปรแกรมในภาษาระดับต่ำทั่วไป ซึ่งทำให้สามารถที่จะแก้ปัญหาของตัวแปรที่อยู่ในลักษณะของ Matrix และVector ได้โดยง่ายซึ่งทำให้ลดเวลาการทำงานลงได้อย่างมากเมื่อเทียบกับการเขียนโปรแกรมโดยใช้ภาษา C หรือ ภาษา Fortran ภาพที่ 2.8 แสดงหน้าต่างการทำงานและสภาพแวดล้อมต่าง ๆ ของโปรแกรม MATLAB จะเห็นว่ามีทั้งส่วนของ Command Window ซึ่งมีไว้เขียนคำสั่งเพื่อให้

โปรแกรมทำงานทีละ 1 บรรทัด ส่วนของ Editor สำหรับการเขียนเป็นชุดคำสั่งหลาย ๆ บรรทัดเพื่อประมวลผล ส่วนของ Workspace สำหรับแสดงตัวแปรและค่าของตัวแปรที่ใช้งานอยู่ในปัจจุบัน ส่วนของ Current Directory สำหรับ แสดงไฟล์ต่าง ๆ ที่พร้อมเปิดใช้งาน

เหตุผลที่งานวิจัยนี้เลือกใช้โปรแกรม MATLAB เนื่องจากโปรแกรม MATLAB มีหนังสือให้ศึกษาค้นคว้ามากมายทั้งในด้านการประมวลผลภาพและด้านอื่น ๆ ซึ่งแตกต่างจากโปรแกรมภาษาอื่น ๆ เช่น ภาษา C ภาษา JAVA ภาษาเบสิก โปรแกรม Labview เป็นต้น ซึ่งมีหนังสือให้ค้นคว้าในหัวข้อการประมวลผลภาพไม่แพร่หลายมากนัก



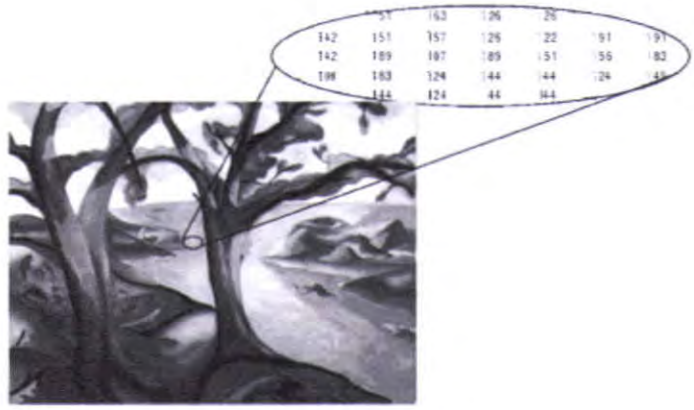
ภาพที่ 2.8 หน้าต่างการทำงานของโปรแกรม MATLAB

2.5 ชนิดของภาพ (Image Type)

ใน Toolbox "Digital Image Processing" ของโปรแกรม MATLAB สนับสนุนชนิดของภาพดังต่อไปนี้

2.5.1 ภาพความเข้มแสง (Intensity Image)

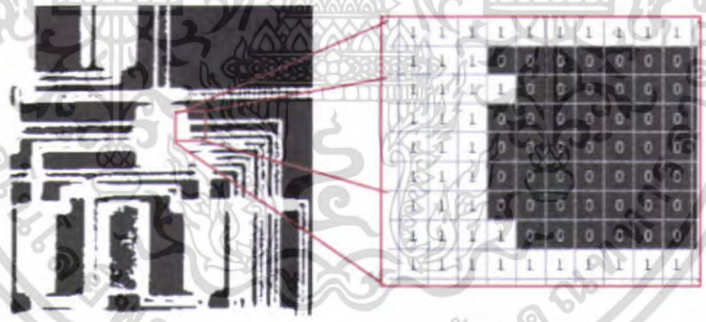
ภาพชนิดนี้ในแต่ละพิกเซล (Pixels) จะมีค่าของความเข้มของแสงในแต่ละระดับที่แตกต่างกันไปตั้งแต่สีขาวจนไปถึงสีดำ โดยสามารถกำหนดระดับความเข้มของแสงเหล่านั้นได้โดยใช้ค่าระดับสีเทา (Gray Scale) ซึ่งโดยปกติทั่วไปแล้วภาพระดับสีเทาจะมีความละเอียด (Resolution) เท่ากับ 8 บิท ซึ่งภาพจะมีค่าระดับความเข้มแสงสีดำเท่ากับ 0 ส่วนค่าระดับความเข้มแสงของสีขาวจะมีค่าเท่ากับ 255 ภาพความเข้มแสงแสดงได้ดังภาพที่ 2.9



ภาพที่ 2.9 ภาพความเข้มแสง

2.5.2 ภาพขาวดำ (Binary Image)

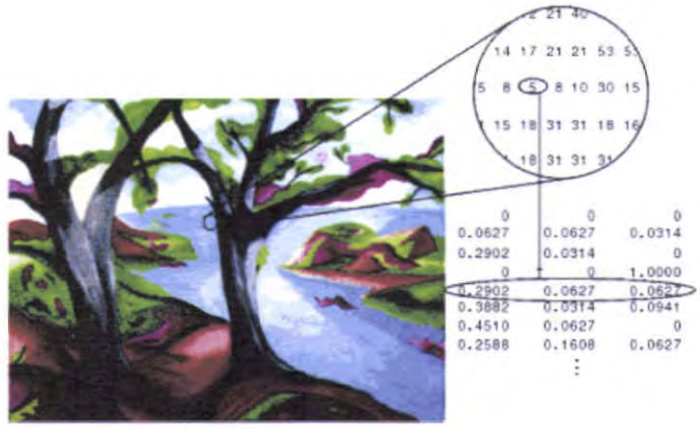
ภาพขาวดำ (Binary Image) คือ ในแต่ละพิกเซลจะแสดงด้วยค่าแบบไบนารี (Binary) คือมี 1 บิท ซึ่งจะประกอบไปด้วยค่า 1 หรือ 0 โดยที่ 1 หมายถึง จุดภาพสีขาว และ 0 หมายถึง จุดภาพสีดำโดยที่ภาพประเภทนี้เหมาะสำหรับภาพที่เกี่ยวข้องกับตัวอักษร (Text) ภาพลายนิ้วมือ (Fringer Print) เป็นต้น ภาพขาวดำแสดงได้ดังภาพที่ 2.10



ภาพที่ 2.10 ภาพขาวดำ

2.5.3 ภาพแบบดัชนี (Index Image)

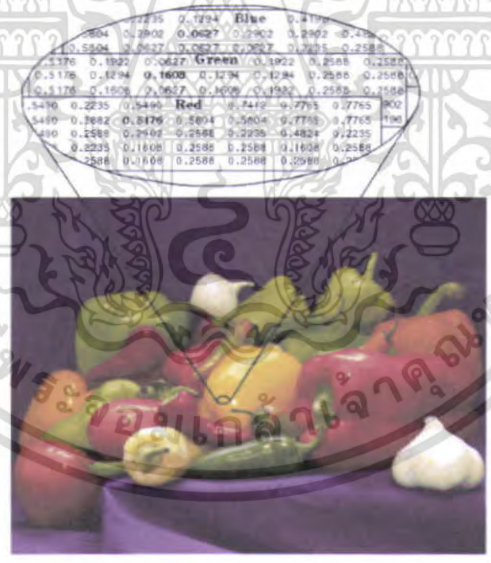
ภาพประชนิดนี้ในแต่ละพิกเซลของภาพจะเก็บค่าดัชนี (Idex Number) ซึ่งเป็นตัวเลขจำนวนเต็มซึ่งค่าดัชนีดังกล่าวจะถูกนำไปเทียบกับตารางสี (Color Table) ซึ่งเป็นตารางแสดงค่าแสงสี แดง เขียวและน้ำเงิน ซึ่งค่าดัชนีนี้จะเป็นตัวชี้ให้เห็นว่าภาพในตำแหน่งพิกเซลนั้น ๆ มีอัตราส่วนของแม่สี 3 สีในอัตราส่วนเท่าไร ภาพแบบดัชนีแสดงได้ดังภาพที่ 2.11



ภาพที่ 2.11 ภาพแบบดัชนี

2.5.4 ภาพสี (RGB Image)

ภาพชนิดนี้แต่ละพิกเซลของภาพจะเก็บค่าระดับความเข้มของแต่ละแถบแสงของแม่สีหลัก 3 สี ที่เรียงซ้อนกันคือ สีแดง (Red) สีเขียว (Green) สีน้ำเงิน (Blue) ซึ่งในแต่ละพิกเซลนั้น ๆ ก็จะแสดงผลของค่าสีแต่ละพิกเซลตามระดับความเข้มในแต่ละแถบสีนั้น แสดงได้ดังภาพที่ 2.12



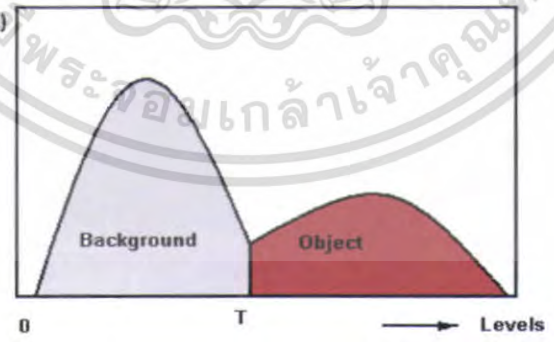
ภาพที่ 2.12 ภาพสี (RGB Image)

2.6 การแบ่งส่วนภาพ (Image Segmentation)

การแบ่งส่วนภาพเป็นขั้นตอนที่สำคัญก่อนการวิเคราะห์ภาพ เนื่องจากเป็นขั้นตอนที่เราจะแยกวัตถุที่เราสนใจออกจากพื้นหลัง การแบ่งส่วนภาพทำการแบ่งภาพออกตามองค์ประกอบหรือวัตถุที่อยู่ในภาพ ระดับของการแบ่งมักขึ้นกับปัญหาที่ศึกษา นั่นคือการแบ่งส่วนภาพอาจทำได้หลายวิธี แต่ในที่นี้จะขอกล่าวถึงการตั้งระดับเทรสโชว์เพียง 1 ระดับ ซึ่งจะเป็นการนำมาใช้ในงานวิจัยนี้

เทรสโชว์ (Threshold) คือการกำหนดค่าระดับความเข้มสีเทาคงที่ที่ค่าหนึ่งเพื่อทำการแยกสิ่งที่ต้องการ (Object) ออกจากพื้นหลังของภาพ (Background) และยังเป็นอีกขั้นตอนหนึ่งเพื่อใช้ในการสร้างภาพแบบไบนารี ซึ่งในการกำหนดค่าเทรสโชว์ ถ้ากำหนดค่าเทรสโชว์ไม่เหมาะสม เช่น ค่าเทรสโชว์ที่มีค่าน้อยหรือมากเกินไป อาจทำให้รายละเอียดบางส่วนของภาพวัตถุที่ต้องการขาดหายไปหรือภาพจะมีสิ่งไม่พึงประสงค์ปะปนมาด้วยเช่น สัญญาณรบกวน (Noise) ดังนั้นจะต้องมีการกำหนดค่าเทรสโชว์ที่เหมาะสม ซึ่งในปัจจุบันมีผู้เสนอวิธีในการหาค่าเทรสโชว์หลายวิธีซึ่งจะถูกนำไปใช้ในงานที่มีลักษณะแตกต่างกันไป

การทำเทรสโชว์โกบอลแบบง่าย เป็นวิธีการสำหรับการแยกส่วนของข้อมูลภาพที่ต้องการออกมาจากพื้นหลังของภาพ โดยการพิจารณาจากฮิสโตแกรมของภาพซึ่งกลุ่มของข้อมูลทั้งสองจะแยกออกเป็นสองกลุ่มตามการกระจายของข้อมูล โดยการใช้ระดับเทรสโชว์เดียว (Threshold) T ดังแสดงในภาพ 2.13 การแบ่งส่วนภาพ (Segmentation) เริ่มด้วยการสแกนภาพพิกเซลต่อพิกเซลแล้วพิจารณาว่าเป็นส่วนของวัตถุหรือพื้นหลัง ซึ่งขึ้นอยู่กับว่าค่าระดับสีเทาของพิกเซลว่ามีค่ามากกว่าหรือน้อยกว่าค่าระดับเทรสโชว์ T ความสำเร็จของวิธีนี้ขึ้นอยู่กับว่าเราสามารถแบ่งส่วนฮิสโตแกรมได้ดีเพียงใด



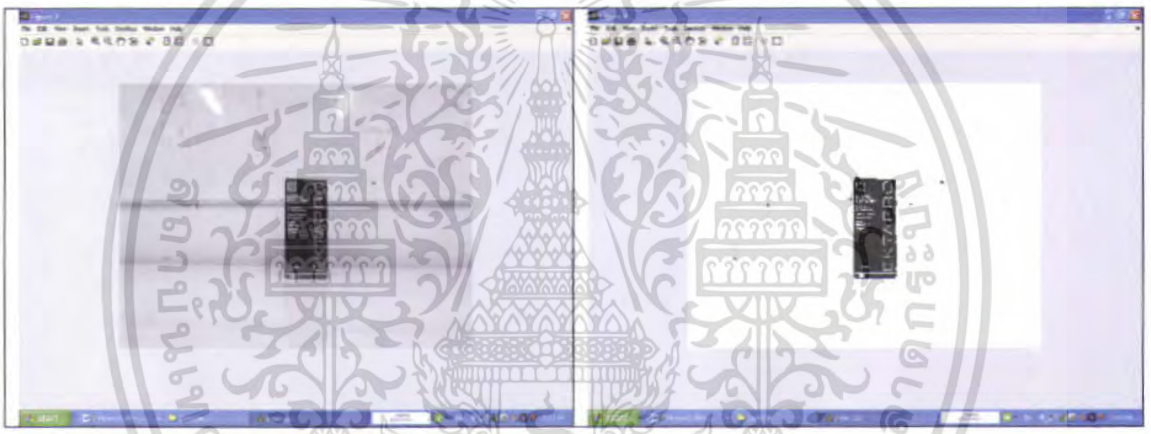
ภาพที่ 2.13 การตั้งค่าเทรสโชว์เพื่อแบ่งข้อมูลในภาพ

จากภาพที่ 2.13 จะเห็นว่าระหว่างส่วนที่เป็นวัตถุ (object) กับ พื้นหลัง (background) จะถูกกั้นด้วยค่า T ซึ่งก็คือค่าเทรสโชว์ที่ถูกกำหนดขึ้น เช่น ภาพที่ 2.14 จะเป็นการกำหนดค่าเทรสโชว์ที่ 100 สามารถแยกวัตถุออกมาจากพื้นหลังได้ ค่าเทรสโชว์ถูกนิยามตามสมการที่ (2.4)

$$g(x,y) = \begin{cases} 1 & \text{if} \quad f(x,y) \geq T \\ 0 & \text{,otherwise} \end{cases} \tag{2.4}$$

โดยที่ f(x,y) คือ ภาพอินพุต

g(x,y) คือ เป็นภาพผลลัพธ์ ซึ่งจะมีค่าเท่ากับหนึ่ง ถ้า f(x,y) มากกว่าหรือเท่ากับ T แต่ถ้าไม่เป็นไปตามเงื่อนไขจะเท่ากับศูนย์



ภาพที่ 2.14 แสดงภาพการทำเทรสโชว์ ที่ค่าเทรสโชว์เท่ากับ 100

2.7 การกำจัดสัญญาณรบกวนออกจากภาพ (Image Noise Reduction)

กระบวนการในการกำจัดสัญญาณรบกวนออกจากภาพสามารถแบ่งออกได้เป็น 2 ประเภท คือ การกำจัดสัญญาณรบกวนแบบเป็นเชิงเส้น (Linear Filtering) และการกำจัดสัญญาณรบกวนแบบไม่เป็นเชิงเส้น (Nonlinear Filtering)

2.7.1 การกำจัดสัญญาณรบกวนแบบเป็นเชิงเส้น (Linear Filtering)

การกำจัดสัญญาณรบกวนออกไปจากภาพ มีวัตถุประสงค์เพื่อกำจัดสัญญาณประเภทความถี่สูง โดยพื้นฐานอาศัยหลักการทำการเฉลี่ยค่าความเข้มแสงเฉพาะบริเวณ หรืออีกนัยหนึ่งก็คือการกรองสัญญาณความถี่ต่ำผ่าน (Low - pass Filter) ผลกระทบของการกรองด้วยสัญญาณความถี่ต่ำผ่าน จะทำให้สัญญาณรบกวนลดลง ในขณะที่ภาพผลลัพธ์จะมีความเรียบ (Smooth)

การทำภาพให้เรียบในบางกรณีภาพผลลัพธ์จะพร่ามัว (Blurring Effect) มีความคมชัดน้อยลง เนื่องจากขอบของวัตถุในรูปภาพจะเป็นบริเวณที่มีการเปลี่ยนแปลงระดับค่าความเข้มแสง ดังนั้น เทคนิคการทำภาพให้เรียบส่วนใหญ่จะเน้นที่การกำจัดสัญญาณรบกวนแต่จะไม่ทำลายขอบของวัตถุในภาพ และหน้าต่าง หรือ Mask ที่ใช้ในการทำภาพให้เรียบ สามารถที่จะเลือกใช้ได้หลายขนาด ตั้งแต่ 3x3 5x5 7x7 เป็นต้น ซึ่งจะมีลักษณะที่ค่าสัมประสิทธิ์ของตัวกรองทุกตำแหน่งจะมีค่าเป็นบวกทั้งหมด

2.7.1.1 ตัวกรองสัญญาณแบบค่าเฉลี่ย (Averaging Operator)

ตัวกรองแบบค่าเฉลี่ย คือตัวกรองแบบความถี่ต่ำผ่านชนิดหนึ่ง ซึ่งผลรวมของค่าสัมประสิทธิ์ของตัวกรองชนิดนี้จะมีค่าเป็นหนึ่ง ตัวอย่างของหน้าต่างของตัวกรองชนิดนี้ถูกแสดงไว้ดังสมการที่ (2.5) สัมประสิทธิ์ในแต่ละตำแหน่งจะมีค่าเป็นหนึ่งทั้งหมดคูณกับเลขจำนวนหนึ่งเพื่อทำให้ผลรวมของสัมประสิทธิ์มีค่าเป็นหนึ่ง ตัวอย่างเช่น เมื่อพิจารณาตัวกรองแบบค่าเฉลี่ยขนาด 3x3 สัมประสิทธิ์ในตำแหน่งจะมีค่าเท่ากับหนึ่งคูณด้วยเศษหนึ่งส่วนเก้า ซึ่งหน้าต่างของตัวกรองความถี่แบบค่าเฉลี่ยมีได้หลายขนาดเช่น 3x3 5x5 และ 7x7 เป็นต้น

$$\text{หน้าต่าง ค่าเฉลี่ยขนาด 3x3} = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \tag{2.5}$$

การเพิ่มขนาดของมาร์คจะช่วยลดสัญญาณรบกวนได้มากขึ้น หรืออาจกล่าวได้ว่าเมื่อขนาดของมาร์คยิ่งมีขนาดใหญ่เท่าไหร่ ปริมาณของสัญญาณรบกวนก็จะถูกลดลงไปได้มากเท่านั้น แต่ภาพจะมีลักษณะพร่ามัวยิ่งขึ้น

2.7.1.2 ตัวกรองแบบเกาส์เซียน (Gaussian Operator)

โดยทั่วไปตัวกรองแบบเกาส์เซียนเป็นที่นิยมใช้มากกว่าแบบค่าเฉลี่ย เนื่องจากมีผลกระทบต่อการพร่ามัว (Blur) ของภาพต้นฉบับน้อยกว่าการใช้ตัวกรองแบบค่าเฉลี่ย ตัวกรองประเภทนี้สามารถที่จะออกแบบสัมประสิทธิ์ของตัวกรองได้หลายรูปแบบ โดยที่สัมประสิทธิ์ของตัวกรองแบบเกาส์เซียนสองมิติ G(x,y) มีค่าตามสมการที่ (2.6)

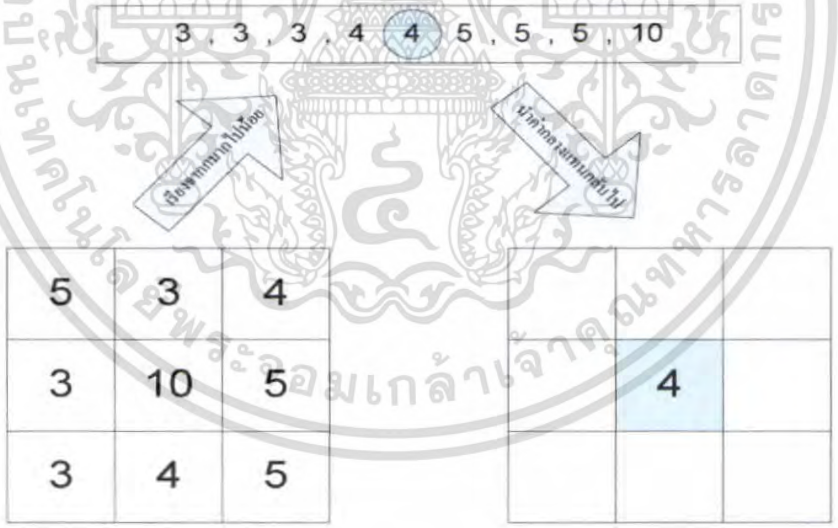
$$G(x,y) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\left(x^2+y^2\right/2\sigma^2)} \tag{2.6}$$

นอกจากนี้ หน้าต่างแบบเกาส์เซียนจะเหมาะสำหรับการกำจัดสัญญาณรบกวนที่มีการกระจายแบบเกาส์เซียน ผลลัพธ์จากการใช้การกรองสัญญาณรบกวนด้วยตัวกรองความถี่แบบเกาส์เซียนสามารถช่วยลดผลกระทบจากการสูญเสียของข้อมูลภาพได้มาก เมื่อเทียบกับการกรองสัญญาณด้วยตัวกรองแบบเฉลี่ย ซึ่งการเพิ่มขนาดของมาร์คจะไม่มีผลกับข้อมูลภาพมากนัก

2.7.2 การกำจัดสัญญาณรบกวนแบบไม่เป็นเชิงเส้น (Nonlinear Filtering)

ในบางครั้งสัญญาณรบกวนบางประเภทก็ไม่สามารถใช้ตัวกรองภาพแบบเชิงเส้นได้ ดังนั้นในส่วนนี้จะกล่าวถึงวิธีการขจัดสัญญาณรบกวนแบบจุดขาวดำ (Salt and Pepper Noise) ในภาพถ่ายดิจิตอลด้วยฟิลเตอร์แบบไม่เป็นเชิงเส้น (Nonlinear Filtering)

Median Filter เป็นตัวกรองความถี่ที่พิจารณาจากข้อมูลทางสถิติ โดยใช้ค่ามัธยฐาน (Median) การหาค่ามัธยฐานทำได้โดยการนำข้อมูลในมาร์ค (Mark) มาทำการเรียงค่าจากน้อยไปมากตามค่าระดับความเข้มสีเทาของข้อมูลกรณีที่ภาพเป็นระดับสีเทา หรืออาจเรียงตัวเลขศูนย์และหนึ่งในกรณีเป็นภาพไบนารี ซึ่งค่ามัธยฐานเป็นค่าตำแหน่งกึ่งกลางของกลุ่มข้อมูลที่พิจารณา จากนั้นนำค่ามัธยฐานที่ได้นั้นแทนค่ากลับลงไปในพิกเซลตำแหน่งตรงกลางของหน้าต่างจะได้ภาพผลลัพธ์ดังภาพที่ 2.15



ภาพที่ 2.15 การทำงานของตัวกรองแบบมีเดียน

ตัวกรองความถี่แบบ Median Filter สามารถเลือกใช้ได้หลายขนาดขึ้นอยู่กับความเหมาะสมและปริมาณของสัญญาณรบกวน เช่น หน้าต่างขนาด 3x3 5x5 7x7 เป็นต้น

2.8 มอร์โฟโลยี (Morphology)

มอร์โฟโลยี เป็นการประมวลผลภาพดิจิตอลโดยอาศัยโครงสร้างต้นแบบที่มีรูปร่างหรือรูปทรงต่าง ๆ มาดำเนินกับรูปต้นแบบ เพื่อนำไปสู่การวิเคราะห์คุณสมบัติทางเรขาคณิตของวัตถุในภาพโอเปอเรชั่นพื้นฐานที่นำมาใช้ในงานวิจัยนี้มี 4 โอเปอร์เรชั่น ได้แก่ Dilation Erosion Opening และ Closing

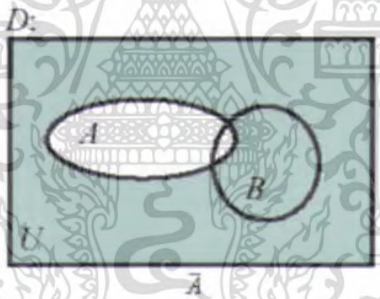
2.8.1 ความรู้เบื้องต้นเรื่องเซต

เนื่องจาก Morphological Operation ได้พัฒนาจากพื้นฐานของเซต ดังนั้นจะได้ขอแนะนำพื้นฐานของเซตที่สำคัญที่เกี่ยวข้องดังต่อไปนี้

กำหนดให้ A และ B เป็นเซตใน Z^2 ที่มีองค์ประกอบ (a_1, a_2) และ (b_1, b_2)

Complement คือ กลุ่มสมาชิกที่อยู่นอกเซตใด ๆ แต่อยู่ใน Universe เมื่อกำหนดให้ A เป็นเซตใด ๆ ดังนั้น Complement ของ A (ดูภาพที่ 2.16) สามารถอธิบายด้วยสมการที่ (2.7)

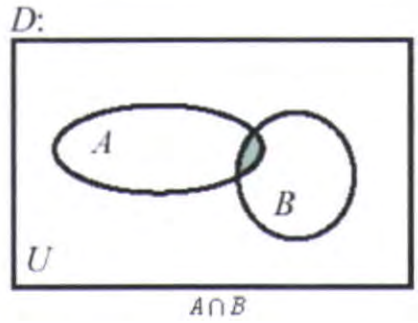
$$A^c = \{x \mid x \notin A\} \tag{2.7}$$



ภาพที่ 2.16 Complement

Intersection (AND) คือ สมาชิกที่อยู่ซ้อนทับระหว่างเซตใด ๆ ถ้ากำหนดให้ A และ B เป็นเซตใด ๆ ดังนั้น Intersection ของ A และ B (ดูภาพที่ 2.17) สามารถอธิบายด้วยสมการที่ (2.8)

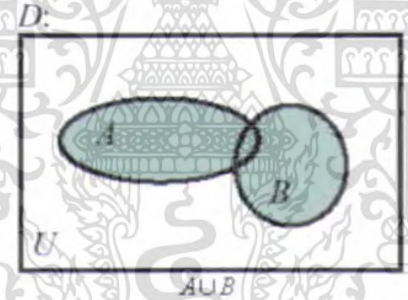
$$A \bigcap B = \{x \mid x \in A^x \in B\} \tag{2.8}$$



ภาพที่ 2.17 Intersection

Union (OR) คือ สมาชิกที่อยู่ในเซตใด ๆ เมื่อกำหนดให้ A และ B เป็นเซตใด ๆ ดังนั้น Union ของ A และ B (ดูภาพที่ 2.18) สามารถอธิบายด้วยสมการที่ (2.9)

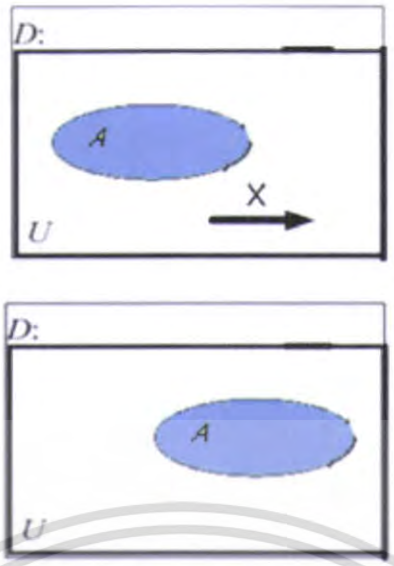
$$A \cup B = \{x \mid x \in A_x \in B\} \tag{2.9}$$



ภาพที่ 2.18 Union

Translation คือ การเลื่อนของเซตตามระยะและทิศทางที่กำหนด เมื่อกำหนดให้ A เป็นเซตใด ๆ และ Vector x คือ ระยะและทิศทางที่กำหนด ดังนั้น Translation ของ A และ B (ดูภาพที่ 2.19) สามารถอธิบายด้วยสมการที่ (2.10)

$$A_x = \{a + x \mid a \in A\} \tag{2.10}$$



ภาพที่ 2.19 Translation

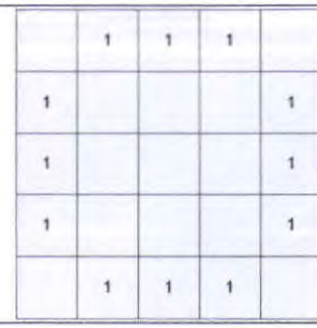
Reflection หรือการสะท้อนของ B เขียนแทนด้วย B_x มีคำจำกัดความดังสมการที่ (2.11)

$$B_x = \left\{ x \,\middle|\, x = -b, b \in B \right\} \tag{2.11}$$

2.8.2 Structure Element

Structure Element คือ เมตริกที่ถูกนิยามให้เป็นรูปร่าง สำหรับการทำ Morphological Operation โดยเมตริกจะประกอบด้วยค่าไบนารี 2 ค่าคือ 0 และ 1 ซึ่งสามารถมีรูปร่างต่าง ๆ ตามที่เรากำหนด ตัวอย่างของ Structure Element แสดงในภาพที่ 2.20

BOX	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
DISK	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

RING	

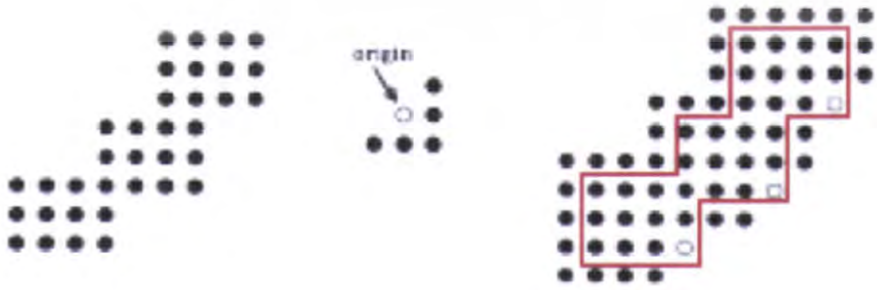
ภาพที่ 2.20 ตัวอย่างของ Structure Element ในลักษณะต่าง ๆ

2.8.3 การขยายภาพ (Dilation)

การขยายจะทำให้วัตถุกว้างขึ้น โดยทั่วไปมักใช้ในการเติมเต็มโดยที่ทั้งปริมาณและรูปแบบของการขยายจะขึ้นโดยตรงกับการเลือกรูปแบบของ Structure Element โดยการเลื่อน Structure Element ไปเรื่อย ๆ จากด้านซ้ายไปขวา และจากด้านบนลงล่าง จากภาพที่ 2.21 นำ Structure Element ขนาด 3x3 ไปวางครอบพื้นที่ใด ๆ ในภาพ แล้วพิจารณาพิกเซลรอบข้างของจุด origin ใน Structure Element กับรูปต้นแบบที่เป็นภาพไบนารีว่ามีค่าเหมือนกันอย่างน้อย 1 พิกเซลหรือไม่ ถ้าเหมือนกันอย่างน้อย 1 พิกเซลให้เปลี่ยนพิกเซลที่อยู่รอบข้างจุด Origin ในภาพไบนารีให้เป็นพิกเซลสีดำทั้งหมด แต่ถ้าพิกเซลรอบจุด Origin ในภาพไบนารีไม่มีพิกเซลไหนที่เหมือนกับพิกเซลรอบจุด Origin ใน Structure Element ให้เปลี่ยนพิกเซลรอบจุด Origin ในภาพไบนารีให้เป็นพิกเซลสีขาวทั้งหมด จากนั้นหน้าต่างที่เป็น Structure Element ก็จะเลื่อนไปที่พิกเซลอื่น ๆ จากซ้ายไปขวาและจากบนลงล่าง และทำแบบเดียวกันทั่วทั้งภาพ ซึ่งสามารถอธิบายด้วยสมการที่ (2.12)

$$A \oplus B = \{X \mid B_x \cap X \neq 0\} \tag{2.12}$$

โดย $A \oplus B$ หมายถึงการขยาย A ด้วย B



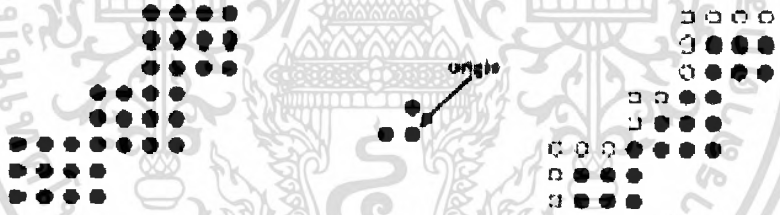
ภาพที่ 2.21 กระบวนการประมวลผลของ Dilation

2.8.4 การเซาะภาพ (Erosion)

การเซาะทำให้วัตถุแคบลง หรือลดขนาดวัตถุลงมักใช้ในการลบส่วนที่ยื่นออกมาจากวัตถุ โดยทั้งปริมาณและรูปแบบของการขยายจะขึ้นโดยตรงกับการเลือกรูปแบบของ Structure Element โดยการเลื่อน Structure Element ไปเรื่อย ๆ จากด้านซ้ายไปขวา และจากด้นบนลงล่าง จากภาพที่ 2.22 นำ Structure Element ขนาด 2x2 ไปวางครอบพื้นที่ใด ๆ ในภาพ แล้วพิจารณาพิกเซลรอบข้างของจุด origin ใน Structure Element กับรูปต้นแบบที่เป็นภาพไบนารีว่ามีค่าเหมือนกันอย่างน้อย 1 บิทหรือไม่ ถ้าเหมือนกันก็ให้แทนค่าบิทที่เหลือเป็นสีดำ แต่ถ้าพิกเซลที่อยู่รอบจุด origin ในภาพไบนารีไม่มีพิกเซลไหนที่เหมือนกับพิกเซลรอบข้างจุด origin ให้เปลี่ยนพิกเซลที่วางซ้อนทับกับจุด origin ในภาพไบนารีให้เป็นพิกเซลสีขาว ส่วนพิกเซลอื่น ๆ ก็ไม่มีการเปลี่ยนแปลงอีก จากนั้นหน้าต่างที่เป็น Structure Element ก็จะเลื่อนไปที่พิกเซลอื่น ๆ จากซ้ายไปขวา และจากบนลงล่าง และทำแบบเดียวกันทั่วทั้งภาพ ซึ่งสามารถอธิบายด้วยสมการที่ (2.13)

$$A \ominus B = \left\{ p \middle| B_p \subseteq A \right\} \tag{2.13}$$

โดย $A \ominus B$ หมายถึงการเซาะของ A ด้วย B



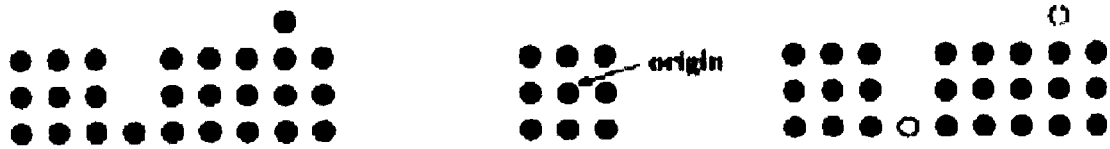
ภาพที่ 2.22 กระบวนการประมวลผลของ Erosion

2.8.5 การเปิด (Opening)

การเปิด คือการทำเซาะภาพก่อนแล้วตามด้วยการขยายภาพโดยทั่วไปแล้วเป็นการทำให้เส้นโครงร่าง (Contour) ของรูปมีความเรียบขึ้น (Smooth) ทำให้ช่องแคบที่ต่อภาพขาดจากกัน และกำจัดส่วนที่ยื่นออกมา ดังภาพที่ 2.23 ซึ่งสามารถอธิบายด้วยสมการที่ (2.14)

$$A \circ B = (A \ominus B) \oplus B \tag{2.14}$$

$A \circ B$ หมายถึงการเซาะของ A ด้วย B แล้วตามด้วยการขยายผลลัพธ์ด้วย B



ภาพที่ 2.23 กระบวนการประมวลผลของ Opening

2.8.6 การปิด (Closing)

การปิด คือการทำขยายแล้วตามด้วยการเซาะ โดยทั่วไปใช้เพื่อการต่อส่วนแคบ ๆ ของพื้นที่ ดังภาพที่ 2.24 ซึ่งสามารถอธิบายด้วยสมการที่ (2.15)

$$A \bullet B = (A \oplus B) \ominus B \tag{2.15}$$

$A \bullet B$ หมายถึงการขยายของ A ด้วย B แล้วตามด้วยการเซาะผลลัพธ์ด้วย B

ภาพที่ 2.24 กระบวนการประมวลผลของ Closing

บทที่ 3
ซอฟแวร์และฮาร์ดแวร์ที่ใช้ในการวิจัย

3.1 กล่าวนำ

การเขียนโปรแกรมด้วยภาษาระดับกลางหรือสูง เช่น ภาษา C หรือภาษาเบสิก สามารถช่วยลดเวลาในการเขียนโปรแกรม และเมื่อมีความจำเป็นต้องเปลี่ยนตระกูลของไมโครคอนโทรเลอร์ การเรียนรู้ใหม่ หรือการแก้ไขโปรแกรมเดิมสามารถทำได้ไม่ยากนักและใช้เวลาน้อยกว่าเมื่อเทียบกับการพัฒนาโปรแกรมด้วยภาษาแอสแซมบลี แต่ก็ไม่ได้หมายความว่าภาษาแอสแซมบลีไม่สำคัญหรือไม่ต้องเรียนรู้เลย เพียงแต่เรียนรู้เฉพาะเท่าที่ต้องนำมาใช้งาน และเฉพาะแก่นหลัก เนื่องจากโปรแกรมภาษาระดับกลาง หรือสูงเองก็มีข้อจำกัดเมื่อต้องลงไปกระทำงานในระดับบิท หรือในงานที่ต้องเกี่ยวข้องกับความเร็วและค่าเวลาที่สั้น ๆ

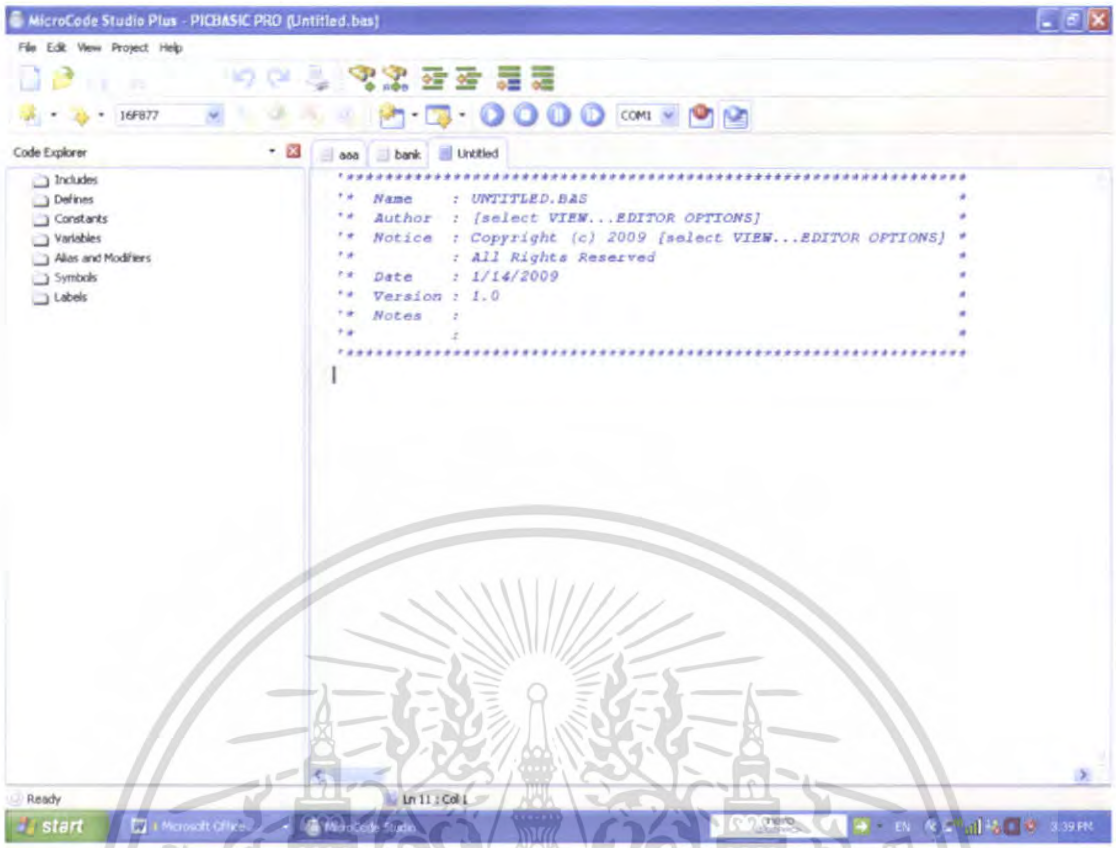
3.2 ภาษาเบสิก (BASIC Programming Language)

ภาษาเบสิก เป็นภาษาคอมพิวเตอร์ที่ได้รับความสนใจสูงมากจากผู้เริ่มต้นและผู้ที่ทำงานในระดับมืออาชีพ เพราะนอกจากจะง่ายต่อการเริ่มต้นเรียนรู้แล้วก็ยังง่ายต่อการพัฒนางานตามไปด้วย โดยเฉพาะอย่างยิ่งกับงานระบบควบคุมอัตโนมัติที่ใช้ไมโครคอนโทรเลอร์ตระกูล PIC

3.2.1 PicBasic Pro คอมไพเลอร์

PicBasic Pro คอมไพเลอร์ได้รับความนิยมอย่างกว้างขวางในหมู่นักพัฒนางานด้านไมโครคอนโทรเลอร์ทั่วโลก รวมถึงนักพัฒนาและผู้เริ่มต้นชาวไทยด้วยเนื่องจากเป็นการเขียนด้วยภาษาเบสิก ซึ่งง่ายต่อการเรียนรู้ อีกทั้งยังไม่จำเป็นต้องเข้าไปศึกษารายละเอียดฮาร์ดแวร์ของชิฟไมโครคอนโทรเลอร์ก็ยังสามารถใช้งานได้

ในการพัฒนาระบบงานของไมโครคอนโทรเลอร์ PIC ด้วยโปรแกรมภาษาเบสิก โดยใช้ PicBasic Pro คอมไพเลอร์ ผู้พัฒนางานสามารถสร้างฮาร์ดแวร์ได้เอง จากนั้นทำการเขียนโปรแกรมภาษาเบสิกแล้วคอมไพล์เป็นไฟล์ .hex เพื่อทำการโปรแกรมลงในไมโครคอนโทรเลอร์ PIC โดยใช้เครื่องโปรแกรมหรือบอร์ดที่สามารถโปรแกรมไมโครคอนโทรเลอร์ได้ในตัว



ภาพที่ 3.1 แสดงหน้าต่างโปรแกรม PicBasic Pro คอมไพล์เลอร์

3.2.2 ซอฟแวร์สำหรับการพัฒนาโปรแกรมภาษาเบสิก

มีด้วยกัน 3 ตัว หลัก ๆ คือ

1. โปรแกรมแปลหรือคอมไพล์ภาษาเบสิกในที่นี้คือ PicBasic Pro คอมไพเลอร์

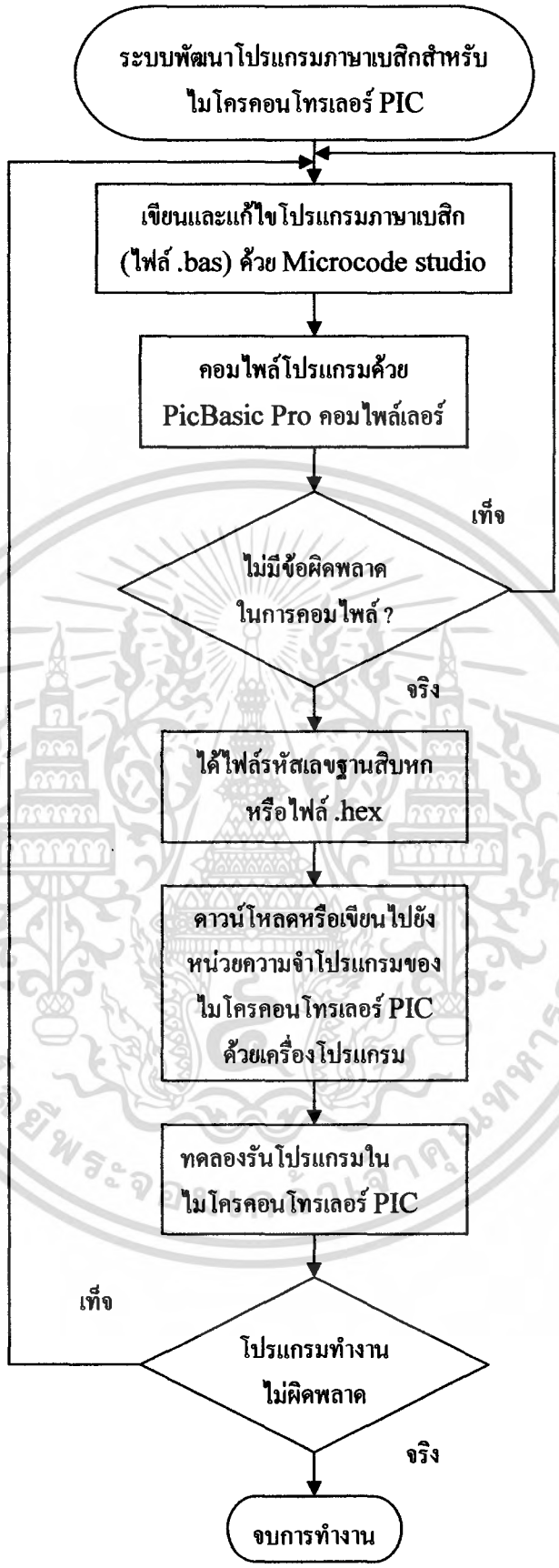
2. ซอฟแวร์ที่ช่วยในการเขียนโปรแกรม ตัวอย่าง เช่นเท็กซ์เอดิเตอร์จำพวก Notepad Wordpad หรือ Microsoft word หรืออาจจะเป็นซอฟแวร์ช่วยที่มีการจัดเตรียมสภาพแวดล้อมและเครื่องมือช่วยเหลืออย่างสมบูรณ์ในรูปแบบ IDE (Integrated Development Environment) ซึ่งในที่นี้จะเลือกแบบหลัง โดยใช้ซอรต์แวร์ชื่อ Microcode Studio

3. ซอฟต์แวร์ใช้สำหรับการโปรแกรมข้อมูลลงในหน่วยความจำของไมโครคอนโทรเลอร์หรือโปรแกรมเมอร์ (Programmer) ซอฟต์แวร์ส่วนนี้จะสัมพันธ์กับเครื่องมือพัฒนาทางฮาร์แวร์ โดยซอฟต์แวร์โปรแกรมเมอร์นี้จะนำไฟล์รหัสเลขฐานสิบหกหรือไฟล์ .hex ที่ได้จากการคอมไพล์ไปเขียนลงในหน่วยความจำของโปรแกรม ทำได้โดยใช้เครื่องโปรแกรมไมโครคอนโทรเลอร์ตระกูล PIC ของบริษัท MICROCHIP ดังภาพที่ 3.2 โดยจะมีคุณสมบัติเทียบเท่ากับเครื่องโปรแกรม PICKIT 2 ของทาง MICROCHIP โดยสามารถโปรแกรมไมโครคอนโทรลเลอร์ PIC ที่มีหน่วยความจำแบบ FLASH MEMORY ได้หลายเบอร์ ET-PGM PIC USB เชื่อมต่อกับ

คอมพิวเตอร์พีซีในการใช้งานทาง PORT USB ใช้ไฟเลี้ยงจาก USB PORT มีความเร็วในการโปรแกรมสูง นอกจากนี้ยังสามารถอัพเกรดเฟิร์มแวร์เวอร์ชั่นใหม่ ๆ เบอร์ใหม่ ๆ ได้จาก WEB (www.microchip.com) ได้เองอีกด้วย

ภาพที่ 3.2 เครื่องโปรแกรมที่ใช้ในงานวิจัย

ภาพที่ 3.3 แสดงขั้นตอนการพัฒนาโปรแกรมภาษาเบสิกโดยใช้ซอฟต์แวร์ทั้งสามกลุ่มร่วมกันเพื่อสร้างโปรแกรมควบคุมการทำงานของไมโครคอนโทรเลอร์ PIC



ภาพที่ 3.3 ขั้นตอนการเขียนโปรแกรม

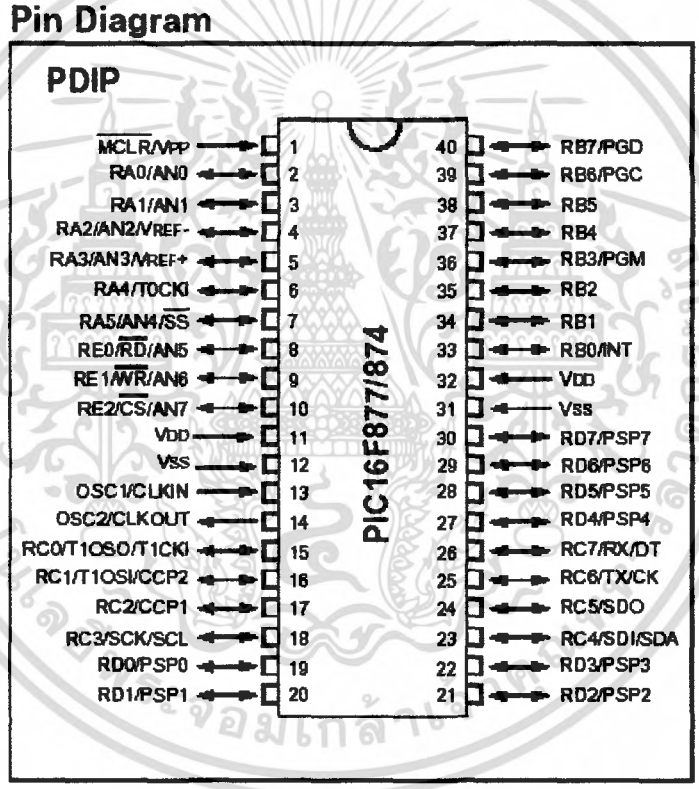
3.3 โครงสร้างและสถาปัตยกรรมของไมโครคอนโทรเลอร์ PIC 16F877

ไมโครคอนโทรเลอร์ได้มีการพัฒนามาอย่างต่อเนื่อง และในปัจจุบันมีไมโครคอนโทรเลอร์ตระกูลใหม่ ๆ เกิดขึ้นมามากมาย ล้วนแล้วแต่มีศักยภาพในการทำงานสูงด้วยกันทั้งสิ้น แต่ในที่นี้จะกล่าวถึงไมโครคอนโทรเลอร์ตระกูล PIC ของบริษัท Microchip ซึ่งเป็นไมโครคอนโทรเลอร์ที่มีความสามารถ และมีทรัพยากรหรือฟังก์ชันการใช้งานต่ าง ๆ มากมาย ตัวอย่างเช่น โมดูล Analog to Digital, Timer/Counter, USART, SPI, I^2C, PWM และ อื่น ๆ ซึ่งส่วนต่าง ๆเหล่านี้จะถูกสร้างรวมอยู่ภายใน CPU เพียงตัวเดียวทำให้ CPU เพียงตัวเดียวนี้สามารถทำงานได้หลาย ๆ อย่าง และสามารถลดในส่วนของฮาร์ดแวร์บางอย่างลง ส่วนในเรื่องของความเร็ว CPU ตระกูลนี้จะใช้เวลาในการกระทำคำสั่งต่าง ๆ เพียง 1 หรือ 2 ไซเคิล ต่อคำสั่งเท่านั้นโดยการทำงานนี้จะเป็นลักษณะไปร์ไลน์ (Pipe Line) ทำให้มีความเร็วในการทำงานมากกว่า CPU ทั่วไป

คุณสมบัติต่าง ๆ ของ PIC 16F877

- 35 คำสั่งหลัก
- ในการปฏิบัติงานคำสั่งต่าง ๆจะใช้ cycle เดียวและ 2 cycle ในคำสั่งที่เป็นการกระโดด
- ความถี่สูงสุดที่ทำงานได้คือ 20 MHz (16F877-20/P)
- การทำงานจะเป็นลักษณะ Pipeline ทำให้มีการทำงานที่เร็วขึ้น
- หน่วยความจำโปรแกรม FLASH Program Memory มีขนาด 8k (14-Bit Words)
- หน่วยความจำข้อมูล (RAM) 368 Byte
- หน่วยความจำข้อมูล (EEPROM) 256 Byte
- สามารถตอบสนองการอินเทอร์รัพได้ถึง 14 แหล่ง
- STACK 8 ระดับ (อยู่ในพื้นที่ของหน่วยความจำข้อมูลภายใน ทำหน้าที่จัดเก็บข้อมูล)
- เพาเวอร์ออนรีเซ็ต (POR) เพาเวอร์อัพไทม์เมอร์(PWRT) และ Oscillator Start-Up Timer
- Watchdog Timer
- สามารถเลือกการป้องกันข้อมูลได้ (Code Protection)
- โหมดประหยัดพลังงาน (Sleep Mode)
- เลือกโหมดสัญญาณนาฬิกาได้หลายโหมด
- สามารถโปรแกรมโดยใช้แรงดัน +5V ได้
- ฟังก์ชันการโปรแกรมแบบ ICSP (In-Circuit Serial Programing)
- ทำงานที่ไฟเลี้ยง 2.0V ถึง 5.5V
- กระแสทั้งซิงก์และซอสร์ของพอร์ตคือ 25mA
- Timer/Counter จำนวน 3 ตัว คือ Timer0, Timer1, Timer2
- โมดูล Capture/Compare/PWM จำนวน 2 ชุด
- Analog to Digital Converter ความละเอียด 10 บิท 8 แชนเนล ภายในตัว

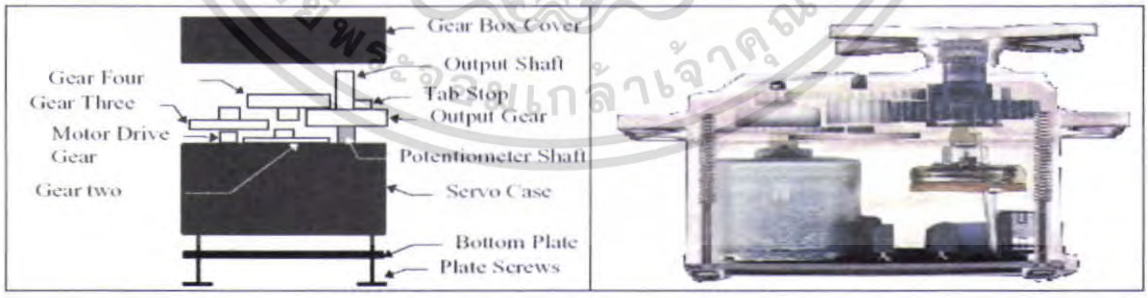
- มีโมดูลสื่อสาร USART
- มีโมดูลตรวจจับระดับแรงดันไฟเลี้ยง Brown-out reset (BOR)
- มีพอร์ต I/O 5 พอร์ต ประกอบด้วย A,B,C,D,E แต่ละพอร์ตจะมีจำนวนบิทไม่เท่ากันซึ่งรวมแล้วจะมี I/O จำนวน 33 บิท แบ่งออกเป็น
- PORTA = RA5-RA0 จำนวน 6 บิท (พอร์ต A บิทที่ 0-5)
- PORTB = RB7-RB0 จำนวน 8 บิท (พอร์ต B บิทที่ 0-7)
- PORTC = RC7-RC0 จำนวน 8 บิท (พอร์ต C บิทที่ 0-7)
- PORTD = RD7-RD0 จำนวน 8 บิท (พอร์ต D บิทที่ 0-7)
- PORTE = RE2-RE0 จำนวน 3 บิท (พอร์ต E บิทที่ 0-2)



ภาพที่ 3.4 แสดงตัวถังของ CPU PIC 16F877 และการจัดวางขาสัญญาณต่าง ๆ

3.4 เซอร์โวมอเตอร์ (Servo motor)

Servo motor คือ มอเตอร์ไฟฟ้ากระแสตรง (DC motor) ที่ถูกประกอบรวมกับชุดเกียร์และส่วนควบคุมต่าง ๆ ไว้ในโมดูลเดียวกันหรือภายในกล่องพาสติกเดียวกัน โดยมอเตอร์ชนิดนี้จะมีสายต่อใช้งานเพียง 3 เส้นเท่านั้น คือ VCC , GND และสายสัญญาณควบคุม (Control Line) ซึ่งสามารถควบคุมให้มอเตอร์หมุนซ้ายหรือขวาได้จากสายสัญญาณเพียงเส้นเดียว โดยสัญญาณที่ใช้ควบคุมนี้จะเป็สัญญาณพัลส์วิดมอด (PWM) แบบ TTL Level แรงดันที่จ่ายให้มอเตอร์นี้จะอยู่ในช่วงประมาณ 4 ถึง 6โวลท์ ขึ้นอยู่กับคุณสมบัติของมอเตอร์แต่ละตัว ข้อดีของมอเตอร์ชนิดนี้ก็คือ จะมีขนาดเล็กน้ำหนักเบา ให้แรงบิตสูง กินพลังงานน้อย และ สามารถควบคุมด้วยแรงดันลอจิกที่เป็น TTL ได้โดยตรงไม่จำเป็นต้องต่อวงจรขับ (Driver) อื่น ๆ เพราะมอเตอร์ชนิดนี้จะมีวงจรควบคุมบรรจุไว้ภายในอยู่แล้ว ซึ่งมอเตอร์ชนิดนี้สามารถควบคุมให้หมุนไปในตำแหน่งหรือทิศทางองศาที่ต้องการได้ โดยอาศัยสัญญาณความกว้างพัลส์ที่ป้อนให้มอเตอร์ แต่เซอร์โวมอเตอร์นี้จะหมุนได้แค่เพียงในช่วงประมาณ 180° หรือครึ่งรอบเท่านั้น หรือ บางรุ่นอาจหมุนได้ถึง 210° แต่จะไม่สามารถหมุนเป็นวงรอบได้ เนื่องจากโครงสร้างภายในจะประกอบด้วย ตัวต้านทานชนิดปรับค่าได้ (VR) ที่ทำหน้าที่ตรวจสอบตำแหน่งการหมุนของมอเตอร์ และตัวต้านทานนี้จะถูกยึดติดกับแกนหมุนของมอเตอร์ ซึ่งจากการที่ตัวต้านทานปรับค่านี้ไม่สามารถหมุนเป็นวงรอบได้ ดังนั้น เซอร์โวมอเตอร์จึงถูกออกแบบให้หมุนได้เพียงแค่ประมาณ 180 องศา หรือ ครึ่งรอบเท่านั้นเพื่อป้องกันความเสียหายที่จะเกิดกับตัวต้านทานปรับค่าได้ แต่ถ้าหากเราต้องการให้มอเตอร์หมุนเป็นวงรอบ (360°) ก็สามารถทำได้ โดยจะต้องทำการปรับแต่ง (Modify) ดัดแปลงชิ้นส่วนบางอย่างของมอเตอร์ ภาพที่ 3.5 และ 3.6 แสดงส่วนประกอบภายนอกและภายในของเซอร์โวมอเตอร์ตามลำดับ

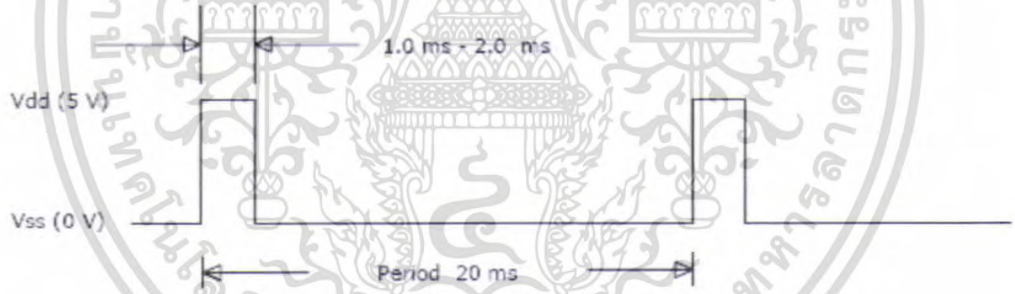


ภาพที่ 3.5 แสดงส่วนประกอบภายนอกของเซอร์โวมอเตอร์

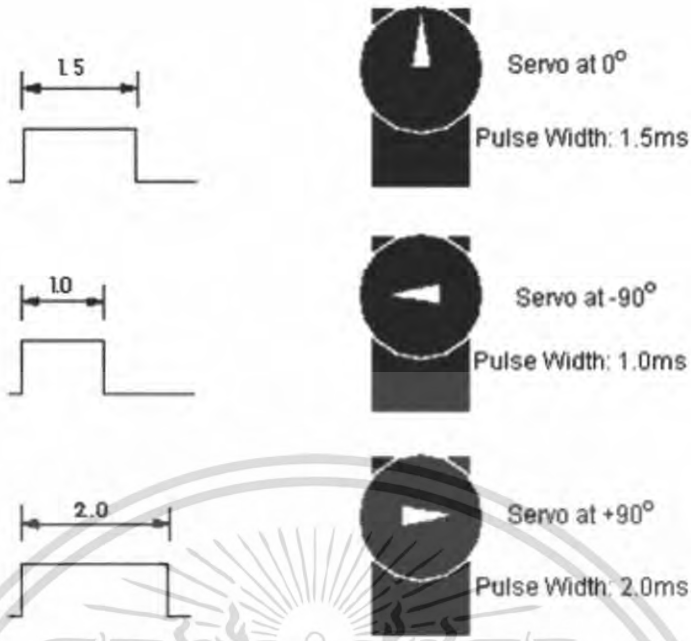
ภาพที่ 3.6 แสดงส่วนประกอบภายในของเซอร์โวมอเตอร์

หลักการทำงานของเซอโวมอเตอร์

การควบคุมการทำงานของเซอร์โวมอเตอร์ สามารถทำได้โดยการป้อนสัญญาณความกว้างพัลส์ ให้กับมอเตอร์ซึ่งตำแหน่งและทิศทางการหมุนของมอเตอร์นี้จะขึ้นอยู่กับขนาดของความกว้างของพัลส์นั้น ๆ โดยทั่วไปแล้วความกว้างของสัญญาณพัลส์ จะมีจุดให้อ้างอิง 3 จุด คือ 1.0 msec 1.5 msec และ 2.0 msec ดังภาพที่ 3.7 และ 3.8



ภาพที่ 3.7 แสดงพัลส์ที่ใช้ควบคุมเซอร์โวมอเตอร์



ภาพที่ 3.8 แสดงทิศทางการหมุนของเซอร์โวมอเตอร์เมื่อจ่ายพัลส์ขนาดต่างๆ

- สัญญาณความกว้างพัลส์ขนาด 1.5 msec จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม 0 องศา หรือ จุดกึ่งกลางของมอเตอร์
- สัญญาณความกว้างพัลส์ขนาด 1 msec จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม - 90 องศา หรือในทิศทางทวนเข็มนาฬิกา
- สัญญาณความกว้างพัลส์ขนาด 2 msec จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม + 90 องศา หรือในทิศทางตามเข็มนาฬิกา

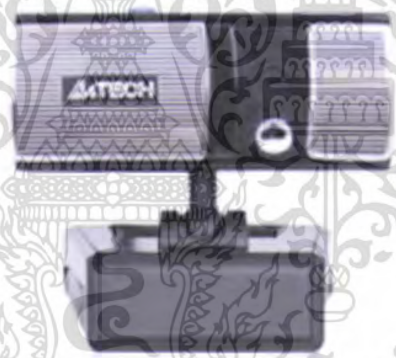
ส่วนการที่จะควบคุมให้เซอร์โวมอเตอร์หมุนเป็นมุมอื่น ๆ นั้นก็สามารถทำได้โดยการป้อนสัญญาณพัลส์เป็นระดับความกว้างต่าง ๆ โดยอ้างอิงจากจุด ทั้ง 3 จุดที่กล่าวมานี้ ตัวอย่างเช่น ถ้าต้องการให้มอเตอร์หมุนไปที่มุม -45 องศา เราก็จะต้องป้อนสัญญาณพัลส์ที่มีความกว้าง 1.25 msec เป็นต้น และ สัญญาณพัลส์นี้จะต้องจ่ายให้มอเตอร์ทุก ๆ 20 msec (Period) เพื่อรักษาสภาพตำแหน่งของมอเตอร์ไว้โดยหลักการก็ คือจะอาศัยการเปรียบเทียบช่วงเวลาของความกว้างพัลส์ที่จ่ายให้กับมอเตอร์ทางขาสัญญาณควบคุมกับค่าเวลาของวงจร RC ภายในบอร์ดควบคุมในตัวของมอเตอร์

(ดูภาพที่ 3.6) ซึ่งค่าเวลาของวงจร RC นี้จะมีการเปลี่ยนแปลงตามการหมุนของมอเตอร์ เนื่องจากตัวต้านทานปรับค่าจะถูกยึดติดอยู่กับแกนหมุนของมอเตอร์ ซึ่งการหมุนของมอเตอร์จะทำให้ค่าความต้านทานของตัวต้านทานปรับค่า (VR) เปลี่ยนแปลงไป เป็นผลทำให้ค่าเวลาของวงจร RC

เปลี่ยนแปลงตามไปด้วย โดยในขณะที่เราป้อนสัญญาณความกว้างพัลส์ให้กับเซอร์โวมอเตอร์ทางขาสัญญาณควบคุมสัญญาณนี้จะถูกนำไปเปรียบเทียบกับค่าเวลาของวงจร RC หากค่าทั้งสองไม่เท่ากันมอเตอร์ก็จะหมุนทำให้ค่าเวลาของวงจร RC เปลี่ยนแปลงจนกระทั่งค่าเวลาความกว้างพัลส์ของวงจร RC เปลี่ยนแปลงเท่ากับสัญญาณพัลส์ทางขาควบคุม (Control line) มอเตอร์จึงจะหยุดหมุน

3.5 กล้องเวบแคม (webcam)

อุปกรณ์ชิ้นสำคัญอีกชิ้นหนึ่งของการติดตามวัตถุก็คือ กล้อง webcam เพราะกล้องเป็นอุปกรณ์ที่จะนำข้อมูลภาพเข้ามาสู่การประมวลผลภาพ โดยในงานวิจัยนี้ได้เลือกใช้กล้อง webcam มาใช้งานเนื่องจากสามารถหาซื้อได้ง่าย ราคาถูก และง่ายต่อการติดต่อกับคอมพิวเตอร์ เพียงแค่ทำการติดตั้งไดร์เวอร์ (Driver) ที่ให้มากับตัวกล้อง จากนั้นก็ทำการติดต่อตัวกล้องกับคอมพิวเตอร์ผ่านทางพอร์ต USB ตัวอย่างของกล้อง webcam แสดงดังภาพที่ 3.9

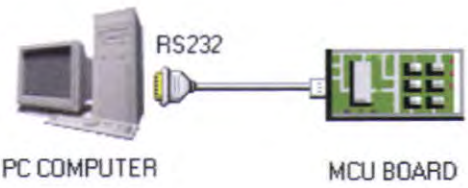


ภาพที่ 3.9 ตัวอย่างกล้องเวบแคม (webcam)

คุณสมบัติของกล้องที่นำมาใช้ในการวิจัย

- ขนาดของภาพที่ใช้ 320x240
- Frame Rate : 10 เฟรมต่อวินาที
- เลนส์ : F = 2.4, f = 3.8 mm
- มุมมองของกล้อง 79 องศา
- ระยะจับภาพ 10 cm ถึง infinity
- การเชื่อมต่อผ่านทาง USB

3.6 การสื่อสารผ่านพอร์ตอนุกรม

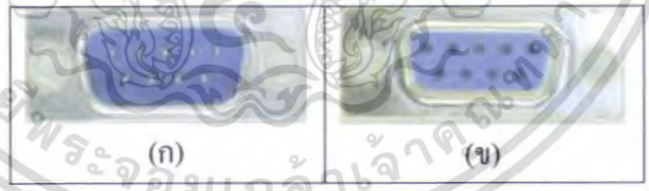


ภาพที่ 3.10 กราฟฟิกแสดงการติดต่อระหว่างบอร์ดไมโครคอนโทรเลอร์(MCU BOARD) กับ PC ผ่านพอร์ตอนุกรม

การสื่อสารแบบอนุกรมมีความสำคัญต่อการใช้งานไมโครคอนโทรลเลอร์มาก เพราะสามารถใช้แป้นพิมพ์และจอภาพของ PC เป็นอินพุต และเอาต์พุตในการติดต่อ หรือควบคุมไมโครคอนโทรลเลอร์ ด้วยสัญญาณอย่างน้อย เพียง 3 เส้นเท่านั้น คือ

- สายส่งสัญญาณ Tx
- สายรับสัญญาณ Rx
- สาย GND

โดยปกติพอร์ตอนุกรม RS-232C จะสามารถต่อสายได้ยาว 50 ฟุตโดยประมาณ ขึ้นอยู่กับชนิดของสายสัญญาณระยะทางและปริมาณสัญญาณรบกวน



ภาพที่ 3.11 พอร์อนุกรม (ก) พอร์ตอนุกรมของ PC DB9 ตัวผู้ (Male) (ข) แสดงพอร์ตอนุกรมของอุปกรณ์ภายนอก DB9 ตัวเมีย (Female)

- พอร์ตอนุกรมของ PC จะเป็นคอนเน็คเตอร์แบบ DB9 ตัวผู้ (Male)
- พอร์ตอนุกรมของอุปกรณ์ภายนอก จะเป็นคอนเน็คเตอร์แบบ DB9 ตัวเมีย (FeMale)

3.6.1 แสดงการจัดขาของคอนเน็กเตอร์อนุกรมแบบ DB9

หัวต่อแบบ D-Type ที่ใช้ในการสื่อสารแบบอนุกรมแบบ 9D ขา หรือบางครั้งจะเรียกว่า DB9

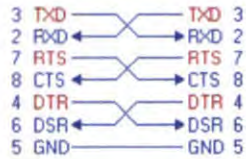
ภาพที่ 3.12 DB9 ตัวผู้

ตารางที่ 3.1 แสดงหน้าที่ของ DB9 แต่ละขา

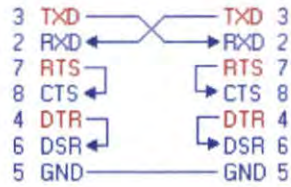
ขา	หน้าที่	แบบ
1	Data Carrier Detect (DCD)	Input
2	Received Data (RXD)	Input
3	Transmitted Data (TXD)	Output
4	Data Terminal Ready (DTR)	Output
5	Signal Ground (GND)	Input
6	Data Set Ready (DSR)	Input
7	Request To Send (RTS)	Output
8	Clear to Send (CTS)	Input
9	Ring Indicator (RI)	Input

3.6.2 การเชื่อมต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์ด้วยสาย DB9

การเชื่อมต่อใช้งานจะมี 2 แบบ คือ 1) การต่อใช้งานครบทุกสายสัญญาณ (Null modem) ซึ่งเป็นการใช้ออฟชั่น (Option) ทุกๆ อย่างที่มีและ 2) การต่อใช้งานเฉพาะสายสัญญาณที่สำคัญเพียง 3 เส้น ซึ่งก็คือ Tx , Rx และ GND ดังภาพที่ 3.13 และ 3.14



ภาพที่ 3.13 แสดงการเชื่อมต่ออุปกรณ์ภายนอกผ่าน DB9 แบบ Null modem



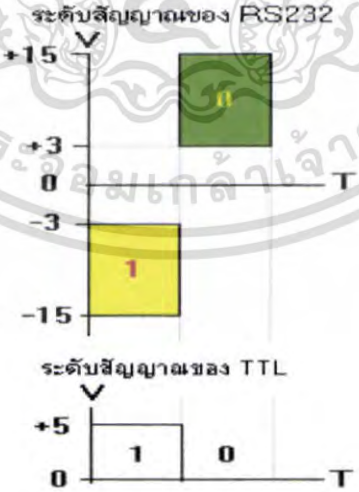
ภาพที่ 3.14 การต่ออุปกรณ์ภายนอกผ่าน DB9 แบบ 3 เส้น

การทำงานของขาสัญญาณ DB9

- TXD เป็นขาที่ใช้ส่งข้อมูล
- RXD เป็นขาที่ใช้รับข้อมูล
- DTR แสดงสภาวะพอร์ตว่าเปิดใช้งาน
- DSR ตรวจสอบว่าพอร์ตที่ติดต่อด้วย เปิดอยู่หรือไม่

เมื่อเปิดใช้งานพอร์ตอนุกรม ขา DTR จะ ON เพื่อให้อุปกรณ์ได้รับทราบว่าต้องการติดต่อด้วย ในขณะเดียวกันก็จะตรวจสอบขา DSR ว่าอุปกรณ์พร้อมหรือไม่ RTS แสดงสภาวะพอร์ตว่าต้องการส่งข้อมูล CTS ตรวจสอบว่าพอร์ตที่ติดต่ออยู่ ต้องการส่งข้อมูลหรือไม่ เมื่อต้องการส่งข้อมูลขา RTS จะ ON และจะส่งข้อมูลออกที่ขา TXD เมื่อส่งเสร็จก็จะ OFF ในขณะเดียวกันก็จะตรวจสอบขา CTS ว่าอุปกรณ์ต้องการที่จะส่งข้อมูลหรือไม่

3.6.3 ระดับสัญญาณของ RS232



ภาพที่ 3.15 ระดับสัญญาณของ RS232C และระดับสัญญาณของ TTL

สัญญาณรบกวนที่เกิดขึ้นในสายนำสัญญาณ มักจะมีแรงดันที่เป็นบวกเมื่อเทียบกับกราวน์ เพื่อป้องกันสัญญาณรบกวนนี้ จึงออกแบบแรงดันของโลจิก "1" เป็นลบ คืออยู่ในช่วง -3V ถึง -15V ส่วนแรงดันของโลจิก "0" อยู่ในช่วง +3V ถึง +15V และเหตุที่ระดับสัญญาณของ RS232 อยู่ในช่วง +15V ถึง -15V ก็เพื่อให้ต่อสายสัญญาณไปได้ไกลขึ้น ดังนั้นจึงจำเป็นจะต้องมีวงจรเปลี่ยนระดับแรงดันของ RS232 มาเป็นระดับแรงดันของ TTL

3.6.4 อัตราการส่งข้อมูล (Baud rate)

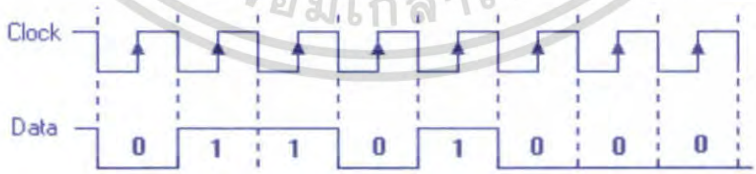
อัตราการส่งข้อมูล คือความเร็วของการรับ - ส่งข้อมูล เป็นจำนวนบิทต่อวินาทีเช่น 300 1,200 2,400 4,800 9,600 14,400 19,200 38,400 56,000 เป็นต้น การเลือกอัตราการส่งข้อมูลขึ้นอยู่กับชนิดของสายสัญญาณ ระยะทาง และปริมาณสัญญาณรบกวน

3.6.5 รูปแบบการสื่อสารแบบอนุกรม

รูปแบบการสื่อสารแบบอนุกรมมีด้วยกันอยู่ 2 แบบ คือแบบซิงโครนัส (Synchronous) และแบบอะซิงโครนัส(Asynchronous)

3.6.5.1 การสื่อสารแบบซิงโครนัส (Synchronous)

การสื่อสารแบบซิงโครนัสนี้จะใช้สัญญาณนาฬิกาควบคุมการรับ และส่งสัญญาณ เช่น สายคีย์บอร์ดคอมพิวเตอร์ โดยจะมีสายสัญญาณเส้นหนึ่งเป็นสายสัญญาณนาฬิกา ส่วนอีกเส้นหนึ่งเป็นสายของข้อมูล (และมักจะมีสายกราวน์ด้วย) การสื่อสารแบบนี้เหมาะสำหรับการสื่อสารในระยะใกล้ ข้อมูลที่ส่งมีไม่มากนัก เพราะถ้าระยะทางไกลขึ้นจะทำให้สัญญาณนาฬิกามีปัญหา อีกทั้งต้องมีสายหลายเส้นทำให้สิ้นเปลืองมาก

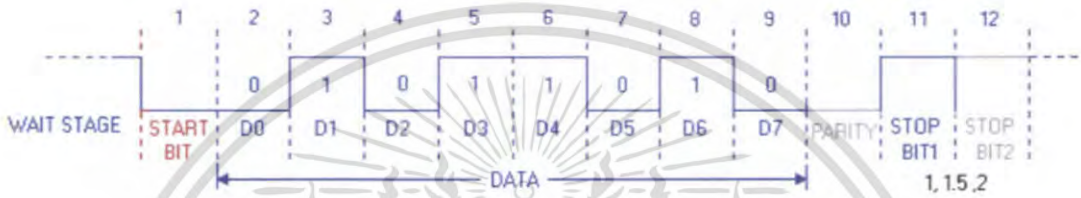


ภาพที่ 3.16 สัญญาณนาฬิกากำหนดการสื่อสารแบบซิงโครนัส

3.6.5.2 การสื่อสารแบบอะซิงโครนัส (Asynchronous)

การรับส่งข้อมูลโดยที่ไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วย แต่จะให้ตัวส่งและตัวรับมีอัตราส่งข้อมูลที่เท่ากัน รูปแบบข้อมูลแบบอะซิงโครนัส ประกอบด้วย 4 ส่วนคือ

1. บิทเริ่มต้น (Start bit) มีขนาด1 บิท
2. บิทข้อมูล (Data) มีขนาด 5 บิท 6 บิท 7 บิท หรือ 8 บิท
3. บิทตรวจสอบพาริตี้ (Parity bit) มีขนาด 1 บิตหรือไม่มี
4. บิทหยุด (Stop bit) มีขนาด 1 บิท 1.5 บิท หรือ 2 บิท



ภาพที่ 3.17 การสื่อสารแบบอะซิงโครนัส

จากรูปที่ 3.17 เมื่อไม่มีการส่งข้อมูลใด ๆ ขา data จะมีสถานะเป็นโลจิก "1" หรือสถานะหยุดรอ (Waiting stage) เมื่อเริ่มต้นส่งข้อมูลจะให้ขา data เป็นโลจิก "0" เป็นจำนวน 1 บิท เรียกว่าบิทเริ่มต้น (Start bit) จากนั้นก็จะเริ่มต้นส่งข้อมูล การส่งข้อมูลจะเริ่มต้นด้วยการส่งบิทต่ำ(LSB) ไปก่อน แล้วตามด้วยพาริตี้บิต ซึ่งอาจจะมีหรือไม่มีก็ได้ขึ้นอยู่กับการตั้งค่าของทั้งสองฝ่าย ตามด้วยการส่งบิทสูง (MSB) จากนั้นจึงส่ง stop bit เพื่อแสดงว่าสิ้นสุดข้อมูล การรับและส่งข้อมูลแบบอนุกรมยังแบ่งตามลักษณะการใช้งานได้ 3 แบบคือ

1. แบบซิมเพลกซ์ (Simplex) เป็นการส่ง หรือรับข้อมูลแบบทิศทางเดียวเท่านั้น
2. แบบฮาล์ฟดูเพลกซ์ (Half Duplex) เป็นการส่งและรับข้อมูลแบบสลับกันคือ เมื่อด้านหนึ่งส่งอีกด้านหนึ่งเป็นฝ่ายรับ สลับกันแต่ไม่สามารถรับ-ส่งในเวลาเดียวกันได้
3. แบบฟลูดูเพลกซ์ (Full Duplex) สามารถรับ-ส่งข้อมูลในเวลาเดียวกันได้

บทที่ 4
การออกแบบโปรแกรมสำหรับการติดตามวัตถุ

4.1 กล่าวนำ

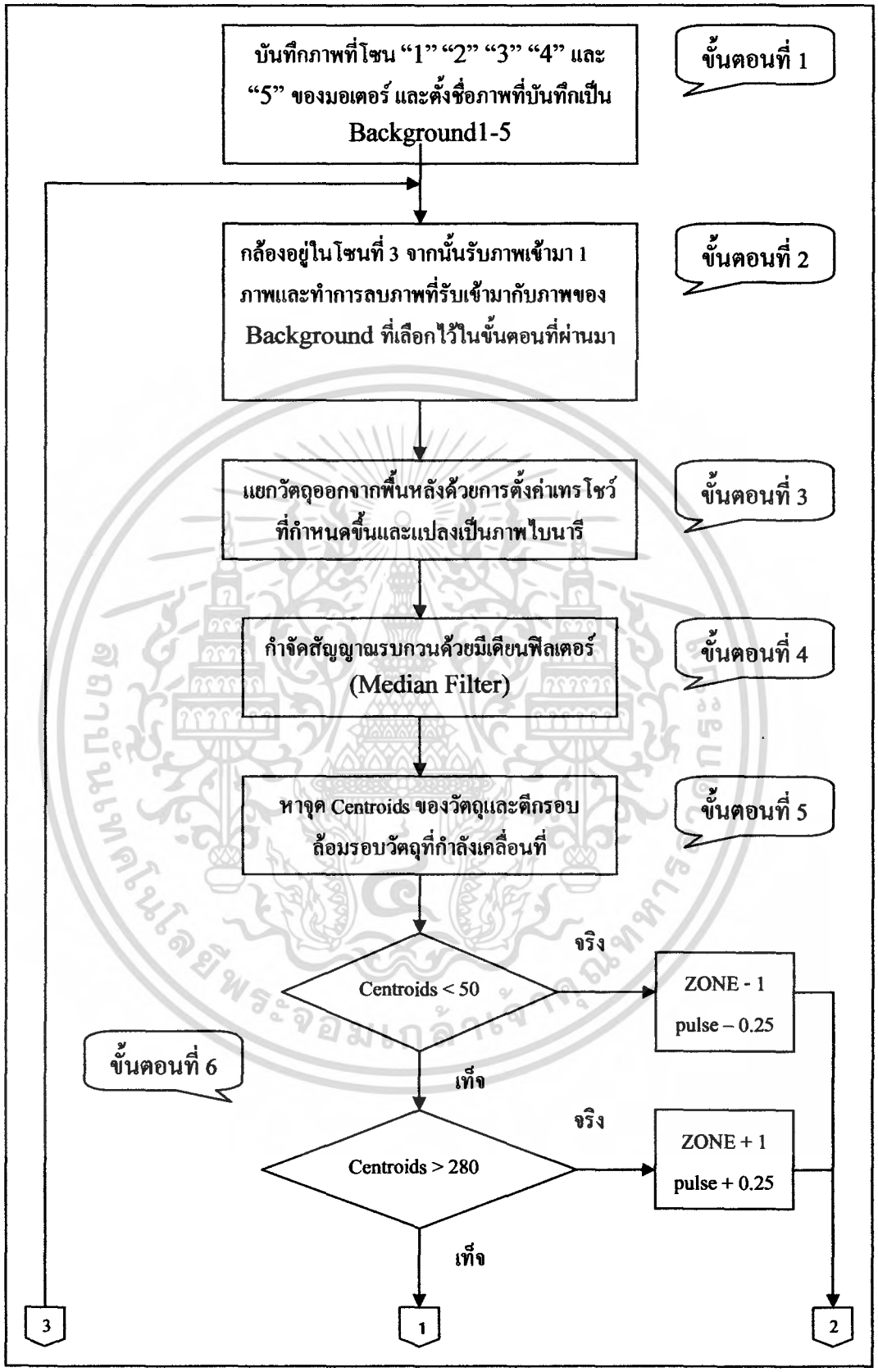
ในบทนี้เป็นการออกแบบโปรแกรมเพื่อการติดตามวัตถุ โดยแบ่งออกได้เป็น 2 หัวข้อ ได้แก่ ขั้นตอนของประมวลผลภาพเพื่อติดตามวัตถุ และขั้นตอนของการนำไมโครคอนโทรเลอร์มาใช้งานในการควบคุมเซอร์โวมอเตอร์ (Servo Motor)

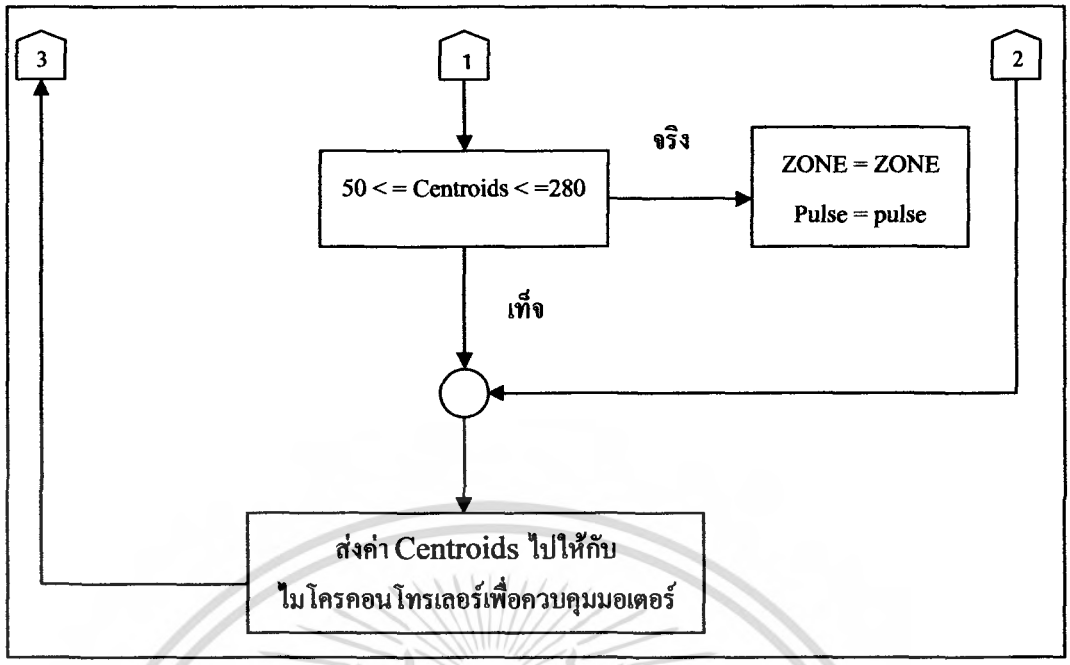
สำหรับแนวความคิดในการติดตามวัตถุนั้น จะออกแบบให้ระบบสามารถติดตามวัตถุที่เข้ามาในมุมมองของกล้อง ทั้งจากทางด้านซ้ายมือและขวามือของมุมมองกล้อง โดยจะกำหนดโซนสำหรับการหมุนของเซอร์โวมอเตอร์ออกเป็น 5 โซน ซึ่งสอดคล้องกับการหมุนของเซอร์โวมอเตอร์ โดยโซนที่ 1 จะตรงกับความกว้างพัลส์ 1.0 msec โซนที่ 2 จะตรงกับความกว้างพัลส์ 1.25 msec โซนที่ 3 จะตรงกับความกว้างพัลส์ 1.5 msec โซนที่ 4 จะตรงกับความกว้างพัลส์ 1.75 msec โซนที่ 5 จะตรงกับความกว้างพัลส์ 2.0 msec เพื่อให้ได้ภาพของพื้นหลังที่ต่อเนื่องกัน และสามารถติดตามวัตถุไปได้อย่างต่อเนื่อง และการทดลองทั้งหมดในงานวิจัยนี้ ได้ทำการทดลอง ณ ห้องทดลองภาควิชาวิศวกรรมการวัดคุม

4.2 การประมวลผลภาพเพื่อติดตามวัตถุ

ในหัวข้อนี้จะอธิบายการทำงานของโปรแกรมสำหรับประมวลผลภาพเพื่อให้เข้าใจถึงขั้นตอนการทำงานของโปรแกรม MATLAB ที่นำมาใช้ในการงานวิจัยนี้ โดยจะเขียนเป็นขั้นตอนเพื่อให้เข้าใจถึงการทำงานโดยรวมก่อน และในแต่ละขั้นตอนก็จะมีคำอธิบายเพิ่มเติมเพื่อให้เข้าใจมากยิ่งขึ้น และภาพประกอบทุกขั้นตอนของการทดลอง รวมทั้งโปรแกรมที่เขียนขึ้นเพื่อใช้ในการประมวลผลภาพในแต่ละขั้นตอนเพื่อให้เข้าใจมากยิ่งขึ้น

โปรแกรม MATLAB ที่นำมาใช้ในงานวิจัยนี้ จะเป็นโปรแกรม MATLAB เวอร์ชัน (Version) 7.3 ทำการเขียนโปรแกรมใน M-file โดยที่ M-file เป็นไฟล์สคริปท์ (Script File) ของโปรแกรม MATLAB ที่ประกอบด้วยซีเควันของคำสั่ง MATLAB เรียงบรรทัดกัน ที่ส่วนท้ายของบรรทัด จะใส่เครื่องหมาย ; เพื่อไม่ให้โปรแกรม MATLAB แสดงผลที่ได้ของคำสั่งในบรรทัดนั้น ส่วนคอมเมนท์ (Comment) จะใส่ไว้หลังเครื่องหมาย % M-file จะถูกสร้างด้วย Text Editor ใดก็ได้ หรือ Text Edit ของโปรแกรม MATLAB เอง ที่เมนู File-new M-file ที่สร้างขึ้นจะถูกบันทึกให้มีนามสกุลเป็น .m





ภาพที่ 4.1 ขั้นตอนการทำงานของการประมวลผลภาพ

อธิบายขั้นตอนการติดตามวัตถุ

- ขั้นตอนที่ 1 เมื่อโปรแกรมเริ่มต้นทำงานกล้องจะเคลื่อนที่มาอยู่ที่ตำแหน่งตรงกลาง จากนั้นก็เริ่มบันทึกภาพในโซนที่ 1 โซนที่ 2 โซนที่ 3 โซนที่ 4 และโซนที่ 5 เก็บไว้เพื่อใช้เปรียบเทียบกับภาพที่มีวัตถุ โดยจะตั้งชื่อของภาพทั้ง 5 ว่า BACKGROUND1 BACKGROUND2 BACKGROUND3 BACKGROUND4 และ BACKGROUND5 ตามลำดับ

ความกว้างพัลส์ที่ 1.00 ms คือ BACKGROUND1 และกำหนดให้ ZONE = 1

ความกว้างพัลส์ที่ 1.25 ms คือ BACKGROUND2 และกำหนดให้ ZONE = 2

ความกว้างพัลส์ที่ 1.50 ms คือ BACKGROUND3 และกำหนดให้ ZONE = 3

ความกว้างพัลส์ที่ 1.75 ms คือ BACKGROUND4 และกำหนดให้ ZONE = 4

ความกว้างพัลส์ที่ 2.00 ms คือ BACKGROUND5 และกำหนดให้ ZONE = 5

และกำหนดให้ตัวแปร pulse มีค่าเท่ากับ 1.5 msec

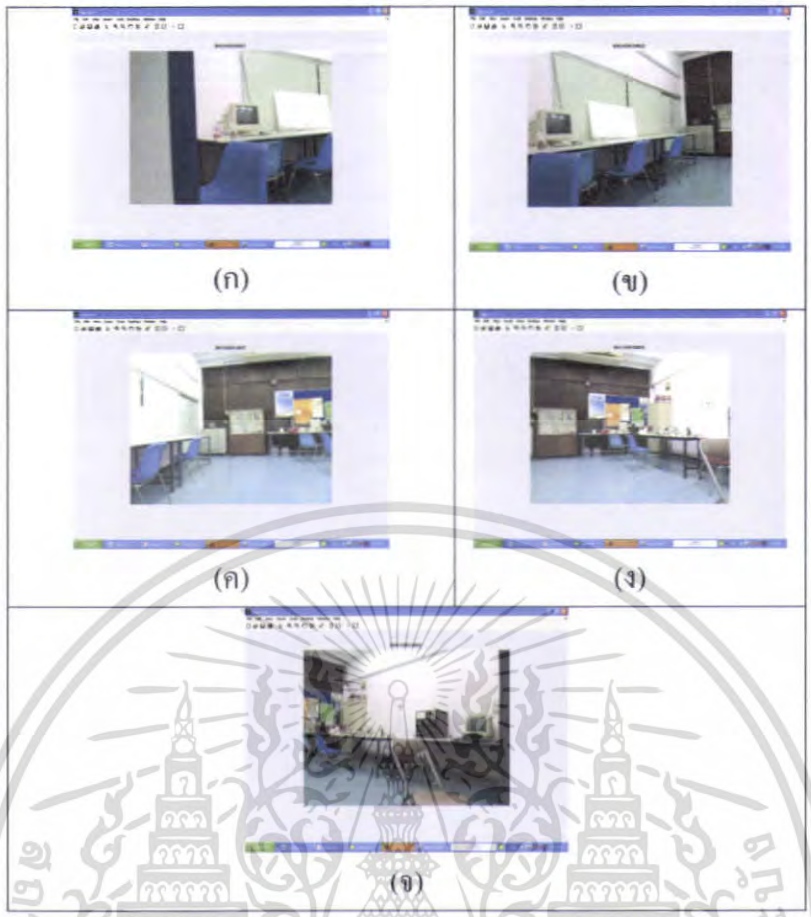
หลังจากบันทึกภาพที่ตำแหน่งต่างๆ ทั้ง 5 ภาพแล้ว กล้องจะกลับมาอยู่โซนที่ 3 ซึ่งเป็นตำแหน่งตรงกลาง เพื่อตรวจจับวัตถุที่จะมาจากทั้งทางซ้ายและขวาของกล้อง โปรแกรมการทำงานในขั้นตอนที่ 1. แสดงได้ดังนี้

```
%%%%%%%%%%%%%% ZONE 1 %%%%%%%%%%%%%%%
for q=1:50                  %วนรอบแบบ for-loop จำนวน 50 รอบ
   fprintf(s,'*');          %ส่ง * ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[250]); %ส่งตัวเลข 250 ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[stop]); %ส่งข้อมูล stop เท่ากับ 9  ออกไปทางพอร์ตอนุกรม
end                         %จบการทำงาน for-loop
pause(2)                    %หน่วงเวลา  2 วินาที
BACKGROUND1 = getsnapshot(vid);      %บันทึกภาพเข้ามา ตั้งชื่อเป็น BACKGROUND1
BACKGROUND1_GRAY = rgb2gray(BACKGROUND1); %แปลงภาพสี BACKGROUND1
                                                %เป็นภาพระดับเทา
%%%%%%%%%%%%%% ZONE 2 %%%%%%%%%%%%%%%
for q=1:50                          %วนรอบแบบ for-loop จำนวน 50 รอบ
   fprintf(s,'*');                  %ส่ง * ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[313]);         %ส่งตัวเลข 313 ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[stop]);        %ส่งข้อมูล stop เท่ากับ 9  ออกไปทางพอร์ตอนุกรม
end                                 %จบการทำงาน for-loop
pause(2)                            %หน่วงเวลา  2 วินาที
BACKGROUND2 = getsnapshot(vid);     %บันทึกภาพเข้ามา ตั้งชื่อเป็น BACKGROUND1
BACKGROUND2_GRAY = rgb2gray(BACKGROUND2);  %แปลงภาพสี BACKGROUND2
                                            % ปืนภาพระดับเทา
%%%%%%%%%%%%%% ZONE 3 %%%%%%%%%%%%%%%
for q=1:50                  %วนรอบแบบ for-loop จำนวน 50 รอบ
   fprintf(s,'*');          %ส่ง * ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[375]); %ส่งตัวเลข 375 ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[stop]); %ส่งข้อมูล stop เท่ากับ 9  ออกไปทางพอร์ตอนุกรม
end                         %จบการทำงาน for-loop
pause(2)                    %หน่วงเวลา  2 วินาที
BACKGROUND3 = getsnapshot(vid);      %บันทึกภาพเข้ามา ตั้งชื่อเป็น BACKGROUND3
BACKGROUND3_GRAY = rgb2gray(BACKGROUND3);  %แปลงภาพสี BACKGROUND3
                                            % เป็นภาพระดับเทา
%%%%%%%%%%%%%% ZONE 4 %%%%%%%%%%%%%%%
for q=1:50                  %วนรอบแบบ for-loop จำนวน 50 รอบ
```

```
   fprintf(s,'*');                    %ส่ง * ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[438]);           %ส่งตัวเลข 438 ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[stop]);          %ส่งข้อมูล stop เท่ากับ 9  ออกไปทางพอร์ตอนุกรม
end                %จบการทำงาน for-loop
pause(2)           %หน่วงเวลา  2 วินาที
BACKGROUND4 = getsnapshot(vid);          %บันทึกภาพเข้ามา ตั้งชื่อเป็น BACKGROUND4
BACKGROUND4_GRAY = rgb2gray(BACKGROUND4);      %แปลงภาพ BACKGROUND4
                                                  % เป็นภาพระดับเทา
%%%%%%%%%%%%%% ZONE 5 %%%%%%%%%%%%%%%%%%%%
for q=1:50                                 %วนรอบแบบ for-loop จำนวน 50 รอบ
   fprintf(s,'*');                         %ส่ง * ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[500]);                %ส่งตัวเลข 500 ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[stop]);               %ส่งข้อมูล stop เท่ากับ 9  ออกไปทางพอร์ตอนุกรม
end                                        %จบการทำงาน for-loop
pause(2)                                   %หน่วงเวลา  2 วินาที
BACKGROUND5 = getsnapshot(vid);      %บันทึกภาพเข้ามา ตั้งชื่อเป็น BACKGROUND5

BACKGROUND5_GRAY = rgb2gray(BACKGROUND5);  %แปลงภาพสี BACKGROUND5
                                                  % เป็นภาพระดับเทา
%%%%%%%%%%%%%% CENTER ZONE %%%%%%%%%%%%%%%%
for q=1:50                              %วนรอบแบบ for-loop จำนวน 50 รอบ
   fprintf(s,'*');                      %ส่ง * ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[375]);             %ส่งตัวเลข 375 ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[stop]);            %ส่งข้อมูล stop เท่ากับ 9  ออกไปทางพอร์ตอนุกรม
end                                     %จบการทำงาน for-loop
ZONE = 3;                               %กำหนดตัวแปร ZONE เท่ากับ 3
```

ภาพที่ 4.2 เป็นผลลัพธ์ที่ได้จากโปรแกรมข้างต้นในหน้าที่ 42 และ 43 ตามลำดับ ซึ่งภาพที่ 4.2 (ก)-(จ) เป็นรูปพื้นหลัง โซนที่ 1 โซนที่ 2 โซนที่ 3 โซนที่ 4 และโซนที่ 5 ตามลำดับ

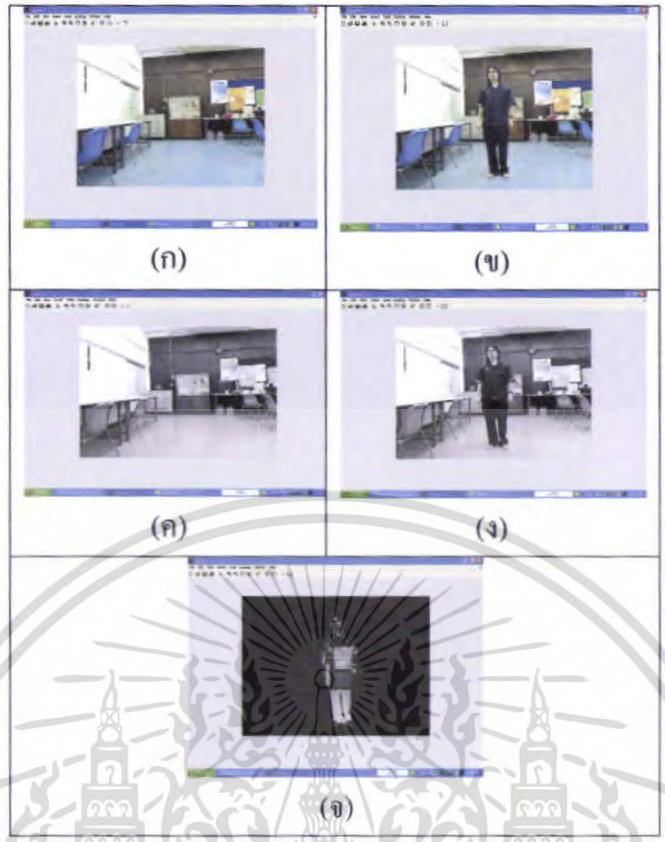


ภาพที่ 4.2 ภาพพื้นหลังทั้ง 5 ที่บันทึกเข้ามา (ก) BACKGROUND1 (ข) BACKGROUND2 (ค) BACKGROUND3 (ง) BACKGROUND4 (จ) BACKGROUND5

- ขั้นตอนที่ 2 กล้องรับภาพเข้ามา 1 ภาพ แล้วนำไปเก็บไว้ในตัวแปร FRAME จากนั้นโปรแกรมจะทำการตรวจสอบดูว่าภาพที่รับเข้ามามีพื้นหลังอยู่ในช่วงใดของภาพที่เก็บไว้ในตัวแปร BACKGROUND 1-5 เมื่อโปรแกรมสามารถตรวจสอบหาพื้นหลังของภาพได้แล้ว โปรแกรมก็จะนำภาพที่บันทึกเข้ามาเพื่อทำการลบกับภาพพื้นหลังที่โปรแกรมสามารถตรวจสอบได้ เพื่อกำจัดภาพพื้นหลังออกไป ซึ่งทำให้เหลือแต่เพียงวัตถุที่อยู่ในภาพที่บันทึกเข้ามาเท่านั้น การทำงานในขั้นตอนนี้เขียนเป็นคำสั่งได้ดังนี้

```
FRAME = getsnapshot(vid);          %บันทึกภาพเข้ามา ตั้งชื่อเป็น FRAME
FRAME_GRAY = rgb2gray(FRAME);              %แปลงภาพสี FRAME เป็นภาพระดับเทา
switch (ZONE)          %เงื่อนไขแบบ switch-case
    case 1             % case ที่ 1
      DIFF = imabsdiff(BACKGROUND1_GRAY,FRAME_GRAY);  %ลบภาพระหว่างภาพ
                                        %BACKGROUND1_GRAY กับภาพ FRAME_GRAY
    case 2    % case ที่ 2
      DIFF = imabsdiff(BACKGROUND2_GRAY,FRAME_GRAY);   %ลบภาพระหว่างภาพ
                                       %BACKGROUND2_GRAY กับภาพFRAME_GRAY
    case 3    % case ที่ 3
      DIFF = imabsdiff(BACKGROUND3_GRAY,FRAME_GRAY);    %ลบภาพระหว่างภาพ
                                       %BACKGROUND3_GRAY กับภาพ FRAME_GRAY
    case 4    % case ที่ 4
      DIFF = imabsdiff(BACKGROUND4_GRAY,FRAME_GRAY);   %ลบภาพระหว่างภาพ
                                       %BACKGROUND4_GRAY กับภาพ FRAME_GRAY
    case 5    % case ที่ 5
      DIFF = imabsdiff(BACKGROUND5_GRAY,FRAME_GRAY);   %ลบภาพระหว่างภาพ
                                       %BACKGROUND5_GRAY กับภาพ FRAME_GRAY
end
```

จากภาพที่ 4.3 เป็นการแสดงขั้นตอนการลบภาพโดยภาพที่ 4.3 (ก) คือภาพพื้นหลังและภาพที่ 4.3 (ข) คือภาพที่มีวัตถุ ซึ่งทั้งสองภาพนี้เป็นภาพแบบ RGB จากนั้นให้นำภาพทั้งสองไปแปลงเป็นภาพแบบระดับสีเทา จะทำให้ได้รูปผลลัพธ์ ดังแสดงในภาพที่ 4.3 (ค) และ (ง) ตามลำดับ เมื่อได้ภาพพื้นหลังและภาพวัตถุเป็นระดับสีเทาแล้ว ก็สามารถนำภาพทั้งสองไปลบกันได้ ซึ่งภาพผลลัพธ์ที่ได้จากการนำภาพระดับสีเทาทั้งสองไปลบกันแสดงดังภาพที่ 4.3 (จ)

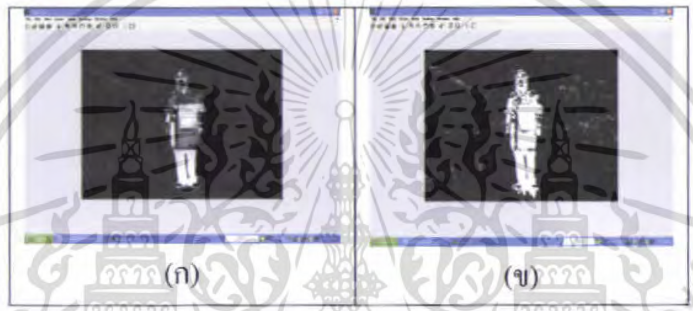


ภาพที่ 4.3 ผลลัพธ์จากการนำภาพระดับสีเทามาลบกัน

- ขั้นตอนที่ 3 แยกวัตถุออกจากพื้นหลังด้วยการทำเทรสโชว์ โดยการใช้ทูลบ๊อก (toolbox) graythresh ในโปรแกรม MATLAB หาค่าเทรสโชว์ที่เหมาะสม หลังจากนั้นก็จะนำผลลัพธ์ที่ได้ไปผ่านตัวกรองแบบมัธยฐาน (Median Filter) เพื่อกำจัด noise ที่อาจเกิดขึ้น จากนั้นก็นำผลลัพธ์ที่ได้มาทำมอร์โฟโลยีเพื่อกำหนดรูปร่างของวัตถุภาพ ตัวอย่างเช่น DISK , BOX เป็นต้น ผลลัพธ์แสดงดังภาพที่ 4.4 โดยภาพที่ 4.4 (ก) เป็นภาพที่ได้จากการลบกันของภาพระดับเทา ส่วนภาพที่ 4.4 (ข) เป็นภาพผลลัพธ์ที่ได้จากการทำเทรสโชว์ที่ค่า 50 และโปรแกรมแสดงการทำงานในขั้นตอนที่ 3 ได้ดังนี้

```
thresh = graythresh(DIFF);          %หาค่าเทรสโชว์แบบ auto
   BW    = im2bw(DIFF,thresh);      %แปลงภาพระดับเทาเป็นภาพขาวดำ
   med = medfilt2(BW,[29 29]);      %นำภาพขาวดำผ่านตัวกรองแบบมีเดียนขนาด 29x29
   se = strel('square',90);         %กำหนดรูปแบบการทำ morphology แบบสี่เหลี่ยม
                                    %จัสตุรัส
   closing = imclose(med,se);       %mophology โดยการ closing
```

```
for p=1:240                          %วนรอบแบบ for-loop จำนวน 240 รอบ
  for q=1:320                        %วนรอบแบบ for-loop จำนวน 320 รอบ
    if(closing(p,q)==1)              %เปรียบเทียบจุดภาพที่ผ่านการทำ morphology ว่าเท่
                                     %กับ 1หรือไม่
      move = move+1;                 %กำหนดให้ตัวแปร move มีการบวก 1
    end                              %จบการทำงาน if
  end                                %จบการทำงาน for
end                                  %จบการทำงาน for-loop
```



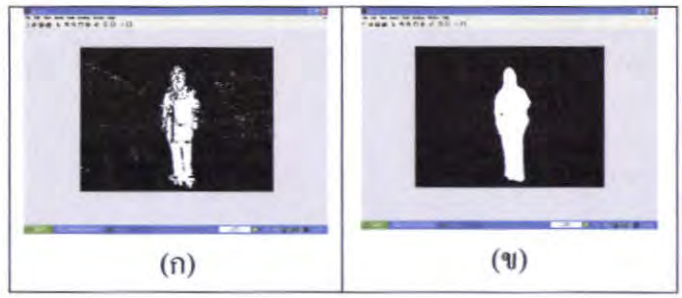
(ก) (ข)

ภาพที่ 4.4 ผลลัพธ์ที่ได้จากการทำเทรสโชว์ที่ ค่าเทรสโชว์ เท่ากับ 50

- ขั้นตอนที่ 4 กำจัดสัญญาณรบกวนด้วยมีเดียนฟิลเตอร์ (Median Filter) ผลลัพธ์ที่ได้มาจากขั้นตอนที่ 3 เมื่อแปลงภาพจากระดับสีเทาเป็นภาพไบนารีแล้วพบว่าภาพยังมีสัญญาณรบกวนปรากฏอยู่ในภาพ จึงต้องใช้ตัวกรองแบบมีเดียน (Median Filter) เพื่อกำจัดออกไป โปรแกรมในขั้นตอนนี้แสดงได้ดังนี้

```
med = medfilt2(BW,[29 29]);
```

จะเห็นว่ามีการใช้หน้าต่าง หรือ mark ขนาด 29x29 เนื่องจากจะต้องกำจัด noise ที่มีอยู่ในภาพออกไปให้หมด เพราะถ้าผ่านขั้นตอนนี้ไปแล้วแต่ยังมี noise อยู่ก็จะทำให้ขั้นตอนการตีกรอบเพื่อระบุตำแหน่งวัตถุมีความผิดพลาดได้ เพราะโปรแกรมจะประมวลผลให้ noise ที่เป็นจุดขาวเล็กๆ เหล่านี้เป็นวัตถุด้วย ภาพผลลัพธ์ของการนำภาพที่มี noise มาผ่านตัวกรองแบบมัธยฐานแสดงดังภาพที่ 4.5 โดยภาพที่ 4.5 (ก) เป็นภาพที่ได้จากการทำเทรสโชว์จากขั้นตอนที่ผ่านมา ส่วนภาพที่ 4.5 (ข) เป็นผลลัพธ์ที่ได้จากการนำภาพที่ 4.5 (ก) มาผ่านตัวกรองแบบมีเดียน



ภาพที่ 4.5 ผลลัพธ์ที่ได้จากการกรองภาพด้วยตัวกรองแบบมีเดียนขนาดหน้าต่าง 29x29

- ขั้นตอนที่ 5. หาจุด Centroids ของวัตถุจากภาพที่ 4.5 เพื่อติดตามและตีกรอบล้อมรอบวัตถุ ซึ่งทำให้สามารถระบุตำแหน่งวัตถุในรูปได้ โดยทำการ Label หรือการกำหนดหมายเลขให้กับกลุ่มพิกเซลสีขาวในรูปนี้ ซึ่งถ้ามีพิกเซลสีขาวอยู่เป็นกลุ่มก้อนเดียวกันโดยไม่ถูกแยกออกจากกันด้วยพิกเซลสีดำ กลุ่มพิกเซลสีขาวนั้นก็จะถูก Label ด้วยหมายเลขที่เหมือนกัน จากนั้นให้นำรูปผลลัพธ์ที่ได้จากการ Label มาหาจุด Centroids โดยจะหาได้จากคำสั่ง regionprops ซึ่งคำสั่งนี้จะมีออฟชั่น (Option) ให้เลือกใช้หลายออฟชั่น ตัวอย่างเช่น Area , BoundingBox และCentroid เป็นต้น สำหรับขั้นตอนนี้ได้เลือกใช้ออฟชั่น BoundingBox เพื่อใช้ตีกรอบล้อมรอบวัตถุ ซึ่งทำให้สามารถทราบค่าพิกัดมุมบนด้านซ้ายของวัตถุ 2 ค่า และอีก 2 ค่าจะเป็นความยาวตามแกน x และแกน y โดยเริ่มจากพิกัดที่มุมบนด้านซ้าย จากนั้นนำค่าความยาวดังกล่าวมาหาพิกัดของมุมทั้ง 4 ที่จะทำการตีกรอบล้อมรอบวัตถุ โปรแกรมข้างล่างค่าของพิกัดดังกล่าวจะเป็น (xb1, yb1) , ((xb1, yb2) , (xb2, yb1) , ((xb2, yb2) แต่เนื่องจากในงานวิจัยนี้มีการติดตามวัตถุเพียง 1 แกน ซึ่งก็คือแกน x ดังนั้นจึงสนใจเฉพาะการเปลี่ยนแปลงของแกน x เท่านั้น การหาจุด Centroids ก็จะเป็นการนำค่าของ xb1 มาบวก xb2 แล้วหาร 2 ค่าที่ได้มาจะเป็นค่าของจุด Centroids โปรแกรมในขั้นตอนดังที่กล่าวมาแล้วสามารถเขียนโปรแกรมได้ดังนี้ และแสดงผลลัพธ์ในภาพที่ 4.6

```
L = bwlabel(closing);                    %แบ่งส่วนพิกเซลที่เป็นพิกเซลสีขาวในภาพ
S = regionprops(L,'BoundingBox');        %หาพิกัดจุดเพื่อทำการตีกรอบล้อมรอบวัตถุโดยการ
                                         %ใช้ Toolbox regionprops
boxsize = cat(2,S.BoundingBox);          %กำหนดให้มีการเรียงค่าที่ได้เพียงแถวเดียวเท่านั้น
[row col] = size(boxsize);               %กำหนดตัวแปร row เท่ากับจำนวนแถว และ
                                         %col เท่ากับจำนวน คอลัมน์
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  if(move>8000)                          %กำหนดเงื่อนไขให้ move มีค่ามากกว่า 8000
    if(col==4)                           %กำหนดเงื่อนไขให้ col มีค่าเท่ากับ 4
```

```
xb1 = round(boxsize(1));      %ปัดค่าของ boxsize(1) ให้เป็น จำนวนเต็มในชื่อตัวแปร xb1
xb2 = round(boxsize(1)+boxsize(3));    %ปัดค่าของ boxsize(1)+ boxsize(2)
                                       %ให้เป็น จำนวนเต็มในชื่อตัวแปร xb2
yb1 = round(boxsize(2));      %ปัดค่าของ boxsize(2) ให้เป็น จำนวนเต็มในชื่อตัวแปร yb1
yb2 = round(boxsize(2)+boxsize(4));    %ปัดค่าของ boxsize(2)+ boxsize(4)
                                       %ให้เป็น จำนวนเต็มในชื่อตัวแปร yb2
centroids = round((xb1+xb2)/2);        %หาค่าของจุด centroids
if((centroids(1)>=50)&&(centroids(1)<=280))   %กำหนดเงื่อนไขขอบเขตการตรวจจับ
   ZONE = ZONE;                        %กำหนดให้ตัวแปร ZONE มีค่าเท่าเดิม
   pulse = pulse;                      %กำหนดให้ตัวแปร pulse มีค่าเท่าเดิม
elseif(centroids(1)<50)                %กำหนดเงื่อนไขขอบเขตการตรวจจับ
   ZONE = ZONE-1;                      %กำหนดให้ตัวแปร ZONE ลบ 1
   pulse = pulse-0.25;                 %กำหนดให้ตัวแปร pulse ลบ 1
   send = round(pulse*250);            %กำหนดค่าตำแหน่งวัตถุที่จะทำการส่ง
   for q=1:50                          %วนรอบแบบ for-loop จำนวน 50 รอบ
     fprintf(s,'*');                   %ส่ง * ออกไปทางพอร์ตอนุกรม
     fprintf(s,'%d\n',[send]);         %ส่งข้อมูล send ออกไปทางพอร์ตอนุกรม
     fprintf(s,'%d\n',[stop]);         %ส่งข้อมูล stop เท่ากับ 9  ออกไปทางพอร์ตอนุกรม
 end                                   %จบการทำงาน for-loop
elseif(centroids(1)>280)               %กำหนดเงื่อนไขขอบเขตการตรวจจับ
   ZONE = ZONE+1;                      %กำหนดให้ตัวแปร ZONE บวก 1
   pulse = pulse+0.25;                 %กำหนดให้ตัวแปร pulse บวก 1
   send = round(pulse*250)             %กำหนดค่าตำแหน่งวัตถุที่จะทำการส่ง
   for q=1:50                          %วนรอบแบบ for-loop จำนวน 50 รอบ
     fprintf(s,'*');                   %ส่ง * ออกไปทางพอร์ตอนุกรม
     fprintf(s,'%d\n',[send]);         %ส่งข้อมูล send ออกไปทางพอร์ตอนุกรม
     fprintf(s,'%d\n',[stop]);         %ส่งข้อมูล stop เท่ากับ 9  ออกไปทางพอร์ตอนุกรม
   end                                 %จบการทำงาน for-loop
end                                    %จบการทำงาน if
if(ZONE>5)                             %กำหนดเงื่อนไขของ ZONE มีค่ามากกว่า 5 หรือไม่
   ZONE = 5;                           %กำหนดค่าตัวแปร ZONE เท่ากับ 5
end                                    %จบการทำงาน if
```

```
if(ZONE<1)                    %กำหนดเงื่อนไขของ ZONE มีค่าน้อยกว่า 1 หรือไม่
  ZONE = 1;                   %กำหนดค่าตัวแปร ZONE เท่ากับ 1
End                           %จบการทำงาน if
if(pulse>2)                   %กำหนดเงื่อนไขของ pulse มีค่ามากกว่า 2 หรือไม่
  pulse = 2;                  %กำหนดค่าตัวแปร pulse เท่ากับ 2
end                           %จบการทำงาน if
if(pulse<1)                   %กำหนดเงื่อนไขของ pulse มีค่าน้อยกว่า 1 หรือไม่
  pulse = 1;                  %กำหนดค่าตัวแปร pulse เท่ากับ 1
end                           %จบการทำงาน if
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
    %create boundingbox
    figure,imshow(FRAME)   %แสดงภาพชื่อ FRAME ที่บันทึกไว้
    for x=xb1              %วนรอบแบบ for-loop ตั้งแต่ x ถึง xb1
      for y=yb1:yb2        %วนรอบแบบ for-loop ตั้งแต่ yb1 ถึง yb2
        hold on            %เก็บภาพที่แสดงไว้
        plot(x,y,'r.')     %พล๊อตค่า
        hold off           %คืนค่าปกติไม่เก็บภาพไว้อีก
      end                  %จบการทำงาน for-loop
    end                    %จบการทำงาน for-loop
    for x=xb1:xb2          %วนรอบแบบ for-loop ตั้งแต่ xb1 ถึง xb2
      for y=yb2            %วนรอบแบบ for-loop ตั้งแต่ y ถึง yb2
        hold on            %เก็บภาพที่แสดงไว้
        plot(x,y,'r.')     %พล๊อตค่า
        hold off           %คืนค่าปกติไม่เก็บภาพไว้อีก
      end                  %จบการทำงาน for-loop
    end                    %จบการทำงาน for-loop
    for x=xb1:xb2          %วนรอบแบบ for-loop ตั้งแต่ xb1 ถึง xb2
      for y=yb1            %วนรอบแบบ for-loop ตั้งแต่ y ถึง yb1
        hold on            %เก็บภาพที่แสดงไว้
        plot(x,y,'r.')     %พล๊อตค่า
        hold off           %คืนค่าปกติไม่เก็บภาพไว้อีก
```

```
    end             %จบการทำงาน for-loop
  end               %จบการทำงาน for-loop
  for x=xb2         %วนรอบแบบ for-loop ตั้งแต่ x ถึง xb2
    for y=yb1:yb2 %วนรอบแบบ for-loop ตั้งแต่ yb1 ถึง yb2
      hold on     %เก็บภาพที่แสดงไว้
      plot(x,y,'r.') %พล๊อตค่า
      hold off    %คืนค่าปกติไม่เก็บภาพไว้อีก
    end           %จบการทำงาน for-loop
  end             %จบการทำงาน for-loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
elseif(col==8)                    %กำหนดเงื่อนไขให้ col มีค่าเท่ากับ 4
  xb1 = round(boxsize(1));        %ปัดค่าของ boxsize(1) ให้เป็น จำนวนเต็มในชื่อตัวแปร xb1
  xb2 = round(boxsize(1)+boxsize(3));   %ปัดค่าของ boxsize(1)+ boxsize(2)
                                        %ให้เป็น จำนวนเต็มในชื่อตัวแปร xb2
  yb1 = round(boxsize(2));        %ปัดค่าของ boxsize(2) ให้เป็น จำนวนเต็มในชื่อตัวแปร yb1
  yb2 = round(boxsize(2)+boxsize(4));   %ปัดค่าของ boxsize(2)+ boxsize(4)
                                        %ให้เป็น จำนวนเต็มในชื่อตัวแปร yb2
   xb3 = round(boxsize(5));       %ปัดค่าของ boxsize(5) ให้เป็น จำนวนเต็มในชื่อตัวแปร yb3
  xb4 = round(boxsize(5)+boxsize(7));   %ปัดค่าของ boxsize(5)+ boxsize(7)
                                        %ให้เป็น จำนวนเต็มในชื่อตัวแปร xb4
  yb3 = round(boxsize(6));        %ปัดค่าของ boxsize(6) ให้เป็น จำนวนเต็มในชื่อตัวแปร yb3
  yb4 = round(boxsize(6)+boxsize(8));   %ปัดค่าของ boxsize(6)+ boxsize(8)
                                        %ให้เป็น จำนวนเต็มในชื่อตัวแปร yb4
  %create boundingbox
  figure,imshow(FRAME)      %แสดงภาพชื่อ FRAME ที่บันทึกไว้
  for x=xb1                 %วนรอบแบบ for-loop ตั้งแต่ x ถึง xb1
    for y=yb1:yb2           %วนรอบแบบ for-loop ตั้งแต่ yb1 ถึง xb2
      hold on               %เก็บภาพที่แสดงไว้
      plot(x,y,'g.')        %พล๊อตค่า
      hold off              %คืนค่าปกติไม่เก็บภาพไว้อีก
    end                     %จบการทำงาน for-loop
  end                       %จบการทำงาน for-loop
```

```
for x=xb1:xb2          %วนรอบแบบ for-loop ตั้งแต่ xb1 ถึง xb2
  for y=yb2            %วนรอบแบบ for-loop ตั้งแต่ y ถึง xb2
    hold on            %เก็บภาพที่แสดงไว้
    plot(x,y,'g.')     %พล๊อตค่า
    hold off           %คืนค่าปกติไม่เก็บภาพไว้อีก
  end                  %จบการทำงาน for-loop
end                    %จบการทำงาน for-loop
for x=xb1:xb2          %วนรอบแบบ for-loop ตั้งแต่ xb1 ถึง xb2
  for y=yb1            %วนรอบแบบ for-loop ตั้งแต่ y ถึง yb1
    hold on            %เก็บภาพที่แสดงไว้
    plot(x,y,'g.')     %พล๊อตค่า
    hold off           %คืนค่าปกติไม่เก็บภาพไว้อีก
  end                  %จบการทำงาน for-loop
end                    %จบการทำงาน for-loop
for x=xb2              %วนรอบแบบ for-loop ตั้งแต่ x ถึง yb2
  for y=yb1:yb2        %วนรอบแบบ for-loop ตั้งแต่ yb1 ถึง yb2
    hold on            %เก็บภาพที่แสดงไว้
    plot(x,y,'g.')     %พล๊อตค่า
    hold off           %คืนค่าปกติไม่เก็บภาพไว้อีก
  end                  %จบการทำงาน for-loop
end                    %จบการทำงาน for-loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for x=xb3              %วนรอบแบบ for-loop ตั้งแต่ x ถึง xb2
  for y=yb3:yb4        %วนรอบแบบ for-loop ตั้งแต่ yb3 ถึง yb4
    hold on            %เก็บภาพที่แสดงไว้
    plot(x,y,'r.')     %พล๊อตค่า
    hold off           %คืนค่าปกติไม่เก็บภาพไว้อีก
  end                  %จบการทำงาน for-loop
end                    %จบการทำงาน for-loop
for x=xb3:xb4          %วนรอบแบบ for-loop ตั้งแต่ xb3 ถึง xb4
  for y=yb4            %วนรอบแบบ for-loop ตั้งแต่ y ถึง yb4
    hold on            %เก็บภาพที่แสดงไว้
```

```
            plot(x,y,'r.')          %พล็อตค่า
            hold off                %คืนค่าปกติไม่เก็บภาพไว้อีก
          end                       %จบการทำงาน for-loop
        end                         %จบการทำงาน for-loop
        for x=xb3:xb4               %วนรอบแบบ for-loop ตั้งแต่ xb3 ถึง xb4
          for y=yb3                 %วนรอบแบบ for-loop ตั้งแต่ y ถึง yb4
            hold on                 %เก็บภาพที่แสดงไว้
            plot(x,y,'r.')          %พล็อตค่า
            hold off                %คืนค่าปกติไม่เก็บภาพไว้อีก
          end                       %จบการทำงาน for-loop
        end                         %จบการทำงาน for-loop
        for x=xb4                   %วนรอบแบบ for-loop ตั้งแต่ x ถึง xb4
          for y=yb3:yb4             %วนรอบแบบ for-loop ตั้งแต่ yb3 ถึง yb4
            hold on                 %เก็บภาพที่แสดงไว้
            plot(x,y,'r.')          %พล็อตค่า
            hold off                %คืนค่าปกติไม่เก็บภาพไว้อีก
          end                       %จบการทำงาน for-loop
        end                         %จบการทำงาน for-loop
      end                           %จบการทำงาน if
      clear col;                    %ล้างตัวแปรชื่อ col
    end                             %จบการทำงาน if
end                                 %จบการทำงาน for-loop
```

ภาพที่ 4.6 ผลลัพธ์ที่ได้จากการตีกรอบเพื่อระบุตำแหน่งของวัตถุ

- ขั้นตอนที่ 6. นำค่าที่เก็บไว้ในตัวแปร Centroids ส่งไปให้กับไมโครคอนโทรเลอร์เพื่อควบคุมเซอร์โวมอเตอร์ ซึ่งมีเงื่อนไขดังต่อไปนี้

ถ้า Centroids < 50 ให้ค่าที่เก็บไว้ในตัวแปร ZONE ลดลง 1 ค่า และ pulse ลดลง 0.25 msec จากค่าเดิมตามลำดับ

ถ้า 50 $\leq$ Centroids $\leq$ 280 ให้ค่าที่เก็บไว้ในตัวแปร ZONE และ pulse จะมีค่าเท่าเดิม

ถ้า Centroids > 50 ให้ค่าที่เก็บไว้ในตัวแปร ZONE เพิ่มขึ้นไป 1 ค่า และ pulse เพิ่มขึ้นอีก 0.25 msec จากค่าเดิม ตามลำดับ

หลังจากนั้นโปรแกรมก็จะรับภาพใหม่เข้ามาเพื่อนำมาประมวลผลอีกรอบ

4.3 ขั้นตอนการเขียนโปรแกรมภาษาเบสิกและการโปรแกรมลงบนไมโครคอนโทรเลอร์

สำหรับรายละเอียดต่าง ๆ ของไมโครคอนโทรเลอร์ได้อธิบายไปแล้วบทที่ 3 บทนี้จะขอกล่าวถึงการเขียนโปรแกรมภาษาเบสิก เพื่อนำไปโปรแกรมลงบนตัวไมโครคอนโทรเลอร์ PIC 16F877 สำหรับใช้ควบคุมมอเตอร์ หน้าที่สำคัญของ PicBasic คอมไพเลอร์ คือ แปลโปรแกรมภาษาเบสิก (.bas) เป็นภาษาเครื่อง (.hex) หรือแมชีนโค๊ด กระบวนการทำงานของ PicBasic Pro คอมไพเลอร์ เพื่อให้ได้มาซึ่งแมชีนโค๊ด มีดังนี้

1. นำโปรแกรมภาษาเบสิกมาแปลงเป็นภาษาแอสแซมบลี ได้ผลลัพธ์เป็นไฟล์ .asm
2. ทำการแปลงภาษาแอสแซมบลีเป็นภาษาเครื่อง ได้ผลลัพธ์เป็นไฟล์ .hex

ดังนั้นส่วนประกอบที่สำคัญมากของ PicBasic Pro คอมไพเลอร์ คือส่วนแปลงภาษาเบสิกเป็นแอสแซมบลี โดยภายใน PicBasic Pro คอมไพลเลอร์จะมีตัวแปลภาษาแอสแซมบลี 2 ตัว คือ PM (PICmicro Macro assembler) ซึ่งเป็นของ Micro Engineering Labs เอง กับ MPASM ของ Microchip โดยปกติหรือค่าตั้งต้นที่กำหนดมากับตัวคอมไพเลอร์จะเลือกใช้ PM เป็นหลัก สำหรับ MPASM จะถูกเลือกใช้ก็ต่อเมื่อเลือกไมโครคอนโทรเลอร์ PIC ในอนุกรม 16 บิท (PIC18xxxxx) ทั้งนี้เนื่องจาก PM ยังไม่รองรับกับไมโครคอนโทรเลอร์ PIC ในอนุกรม 16 บิท

กระบวนการแปลหรือคอมไพล์ของ PIC Basic Pro คอมไพเลอร์จะดำเนินการไปในคราวเดียวกันตั้งแต่แปลภาษาเบสิกเป็นแอสแซมบลี และจากภาษาแอสแซมบลีเป็นแมชีนโค๊ด ได้เป็นไฟล์ .hex เพื่อนำไปโปรแกรมลงในไมโครคอนโทรเลอร์ต่อไป

ส่วนประกอบและคุณสมบัติที่น่าสนใจของ Microcode Studio

Microcode Studio เป็นโปรแกรมสร้างสภาวะแวดล้อมการเขียนโปรแกรมด้วยภาษาเบสิกสำหรับ PicBasic Pro คอมไพเลอร์ ทำให้เขียนโปรแกรมควบคุมไมโครคอนโทรเลอร์ PIC ด้วยภาษาเบสิกนั้นเป็นเรื่องง่าย โดยมีส่วนประกอบที่สำคัญดังนี้

1. พื้นที่เขียนโปรแกรม จะใช้สีและความหนาของตัวอักษรเป็นตัวแบ่งแยกคำสั่ง ลาเบล คอมเมนต์ และข้อมูลต่าง ๆ ออกจากกัน สามารถทำความเข้าใจโปรแกรมสามารถทำได้ง่าย

2. หน้าต่าง Code Explorer เพื่อกระโดดไปยังตำแหน่งลาเบล ตัวแปร ค่าคงที่ต่าง ๆ ได้อย่างรวดเร็ว ทั้งยังสามารถแจกแจงการกำหนดตำแหน่งลาเบลหรือตัวแปรต่าง ๆ ได้ง่ายอีกด้วย

3. หน้าต่างแสดงข้อผิดพลาด เมื่อคอมไพล์โปรแกรม อาจเกิดข้อผิดพลาดขึ้นได้ หน้าต่างนี้จะแสดงข้อผิดพลาดต่าง ๆ ขึ้นมา ผู้ใช้งานสามารถใช้เมาส์คลิกที่บรรทัดเหล่านั้นโปรแกรมจะแสดงแถบขึ้นที่บรรทัดที่ผิดพลาด ทำให้สามารถเข้าไปแก้ไขโปรแกรมได้ง่าย

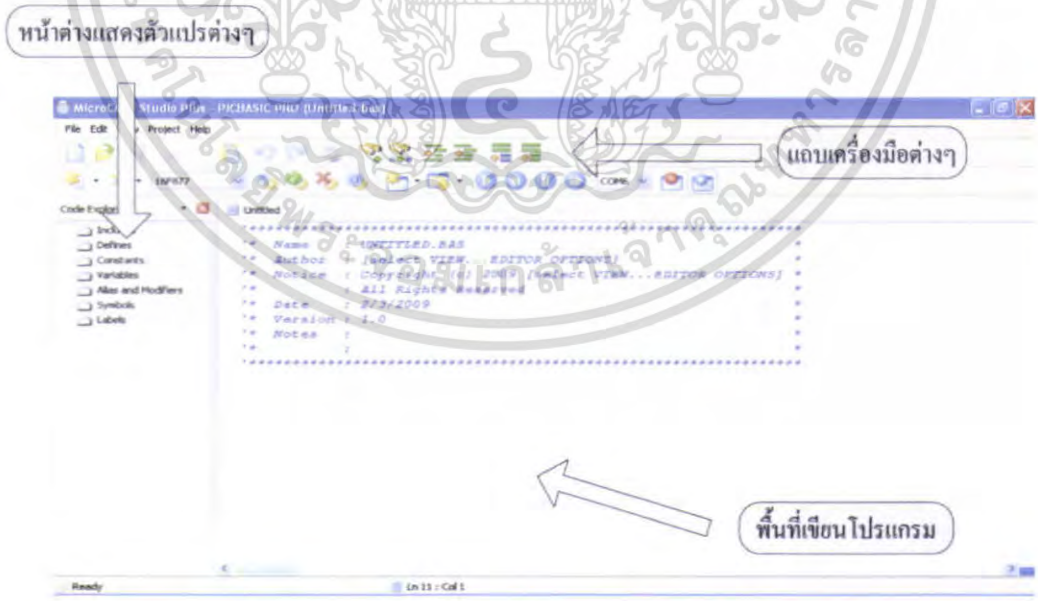
4. แถบแสดงสถานะ แสดงด้วยสัญลักษณ์รูป LED สีแดงและเขียว เพื่อให้สังเกตุได้ง่าย

5. แถบเครื่องมือ รวมคำสั่งแก้ไขและคอมไพล์โปรแกรมไว้ ทำให้ผู้เริ่มต้นสามารถใช้งานโปรแกรมได้อย่างรวดเร็ว

6. หน้าต่างสื่อสารข้อมูลอนุกรม (Serial Communications Window) ใช้แสดงผลข้อมูลอนุกรมที่ส่งมาจากไมโครคอนโทรเลอร์ผ่านคำสั่ง DEBUG หรือ SEROUT สามารถกำหนดค่าได้อย่างอิสระ

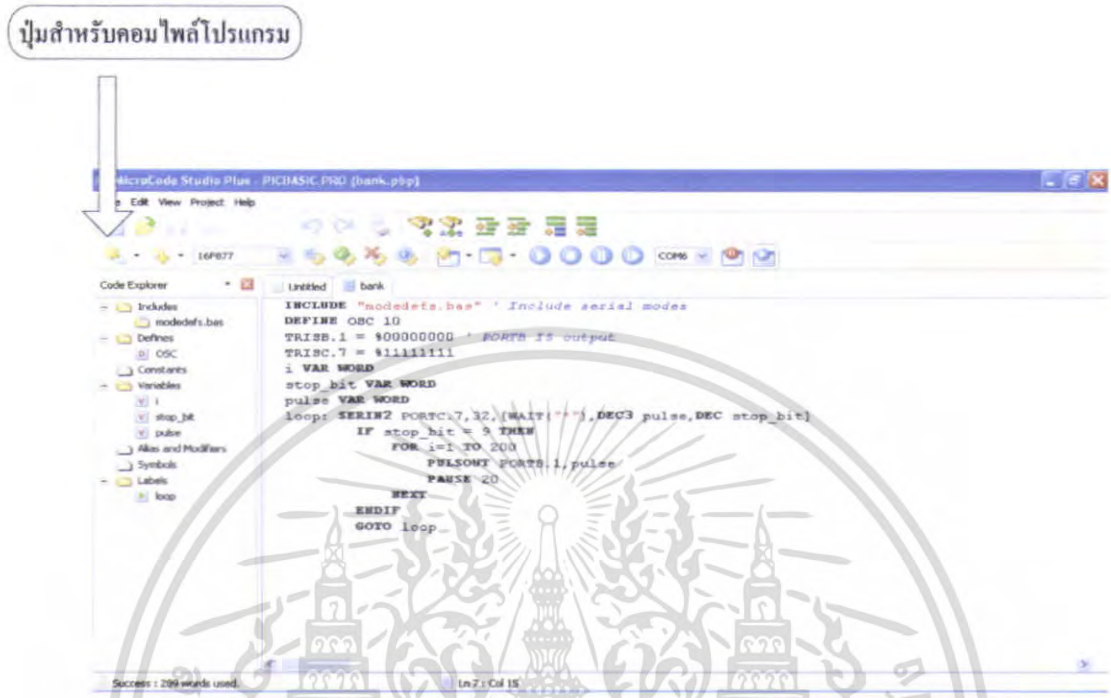
โดยการทำงานทั้งหมดดังที่กล่าวมาข้างต้นมีขั้นตอนการทำงานดังต่อไปนี้

1. เปิดโปรแกรม PicBasic Pro คอมไพเลอร์ ดังภาพที่ 4.7 แสดงหน้าต่างการทำงานของโปรแกรม PicBasic Pro คอมไพเลอร์ และส่วนประกอบต่างๆ ของโปรแกรม สำหรับการเขียนโปรแกรม



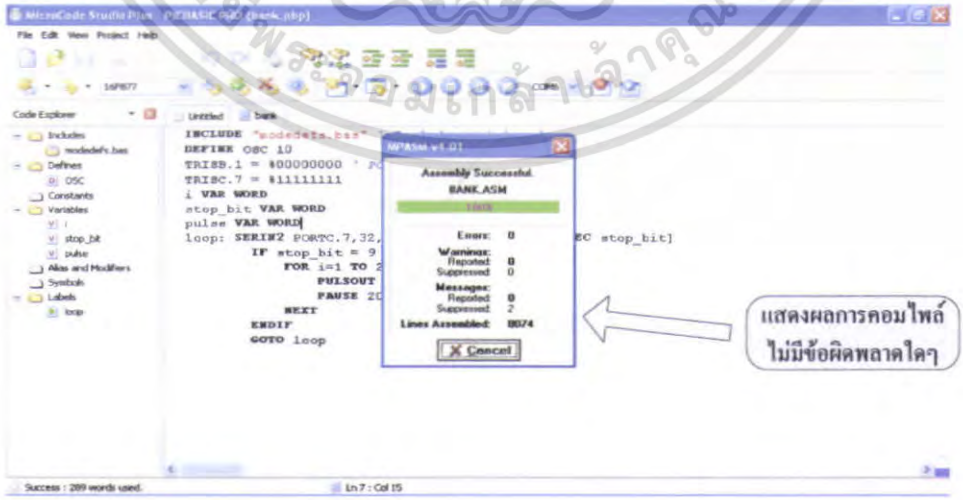
ภาพที่ 4.7 โปรแกรม PicBasic Pro

2. แสดงหน้าต่างโปรแกรมภาษาเบสิกที่เขียนเสร็จเรียบร้อยแล้ว และพร้อมที่จะทำการคอมไพล์ แสดงดังภาพที่ 4.8



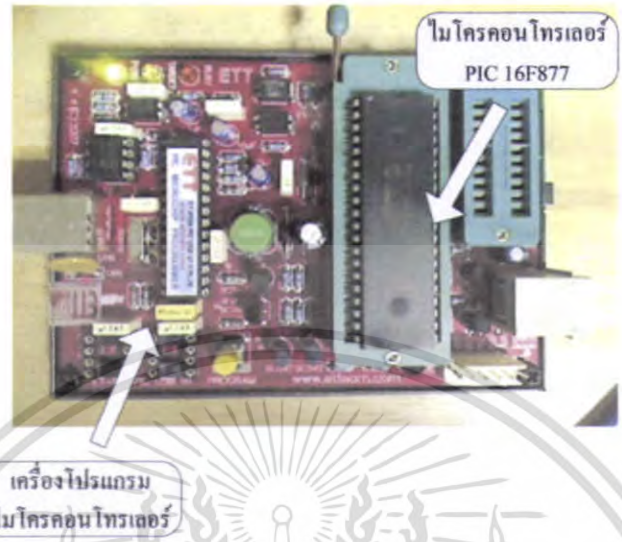
ภาพที่ 4.8 โปรแกรมภาษาเบสิกที่เขียนเสร็จแล้ว

3. แสดงการคอมไพล์โปรแกรมภาษาเบสิกเพื่อให้ได้ไฟล์ .hex เพื่อนำไปโปรแกรมให้กับไมโครคอนโทรเลอร์



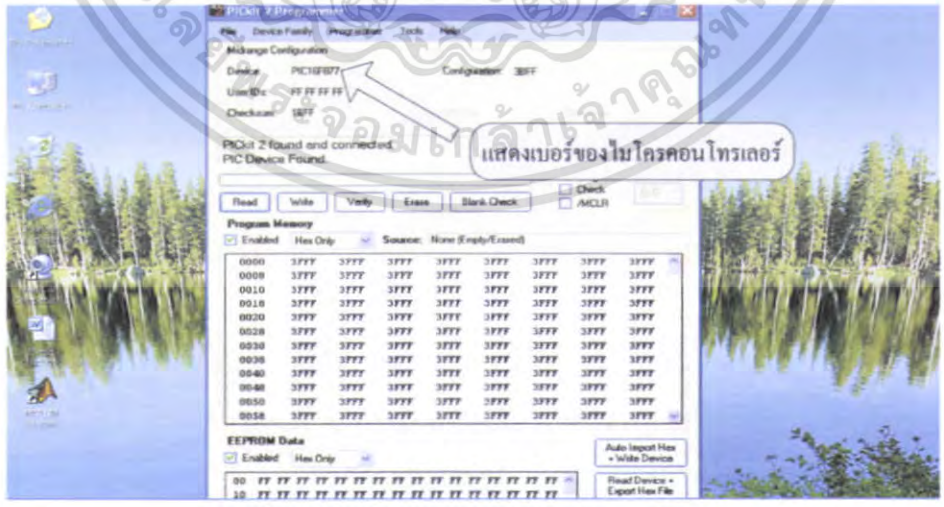
ภาพที่ 4.9 แสดงการคอมไพล์โปรแกรม PicBasic Pro

4. นำไมโครคอนโทรเลอร์วางลงใน TEXT Tool สำหรับใส่ไมโครคอนโทรเลอร์แล้วทำการล็อคขาไมโครคอนโทรเลอร์ให้เรียบร้อยดังภาพที่ 4.10



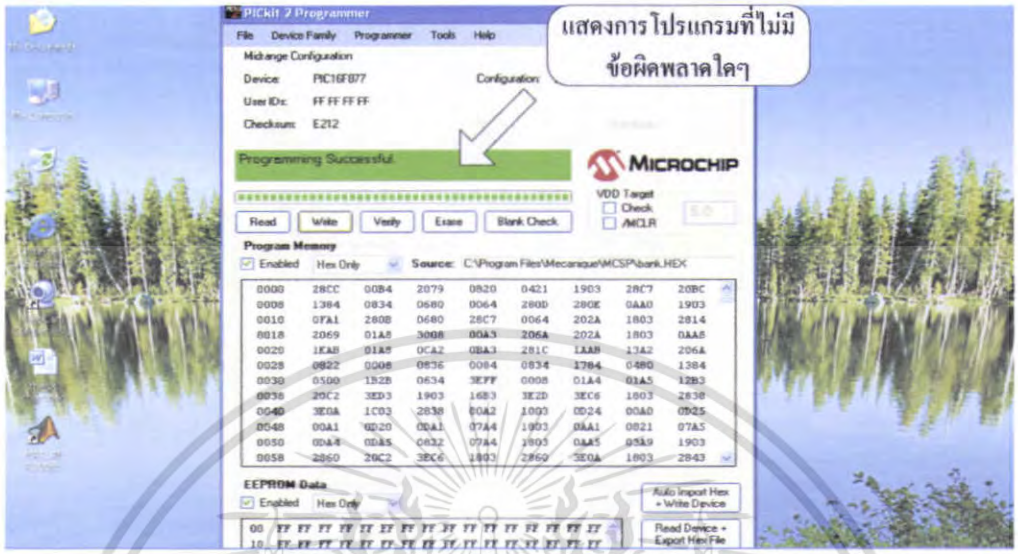
ภาพที่ 4.10 ไมโครคอนโทรเลอร์ที่พร้อมทำการโปรแกรมด้วยเครื่องโปรแกรม

5. เปิดโปรแกรม PICkit 2 Programmer เพื่อใช้สำหรับการโปรแกรมไฟล์ .hex ที่ได้ในขั้นตอนที่ 3 ลงบนตัวไมโครคอนโทรเลอร์ โดยโปรแกรม PICkit 2 จะทำการตรวจสอบไอซีบน TEXT Tool หากเป็นเบอร์ที่ PICkit 2 สนับสนุนการใช้งานอยู่ และการเชื่อมโยงสัญญาณต่าง ๆ ถูกต้อง ในช่อง Device จะแสดงเบอร์ของ PIC Micro ที่พบ ดังภาพที่ 4.11



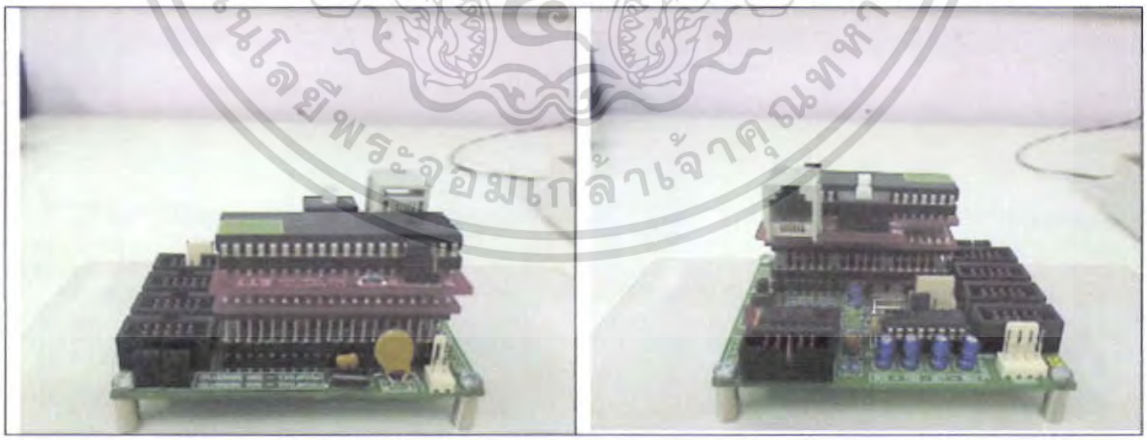
ภาพที่ 4.11 แสดงหน้าต่างโปรแกรม PICkit 2 Programmer

6. ทำการโปรแกรมไฟล์ .hex ที่ได้ ซึ่งจะถูกเก็บอยู่ในโฟเดอร์เดียวกับโปรแกรมที่ภาษาเบสิกที่เขียน ให้กับตัวไมโครคอนโทรเลอร์ ดังภาพที่ 4.12



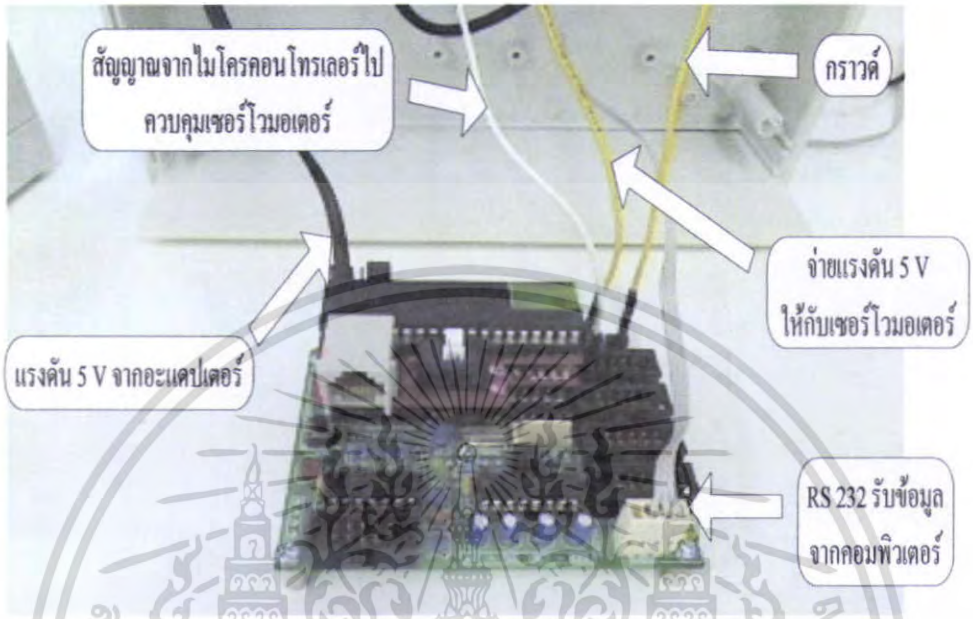
ภาพที่ 4.12 แสดงผลการโปรแกรมไฟล์ .hex ให้กับตัวไมโครคอนโทรเลอร์

เมื่อผ่าน 6 ขั้นตอนดังที่ได้กล่าวมาแล้ว ก็จะสามารถที่จะนำไมโครคอนโทรเลอร์ไปใช้งานได้ โดยจะต้องนำตัวไมโครคอนโทรเลอร์ไปเสียบในช่องสำหรับไมโครคอนโทรเลอร์ ในบอร์ดทดลองดังภาพที่ 4.13



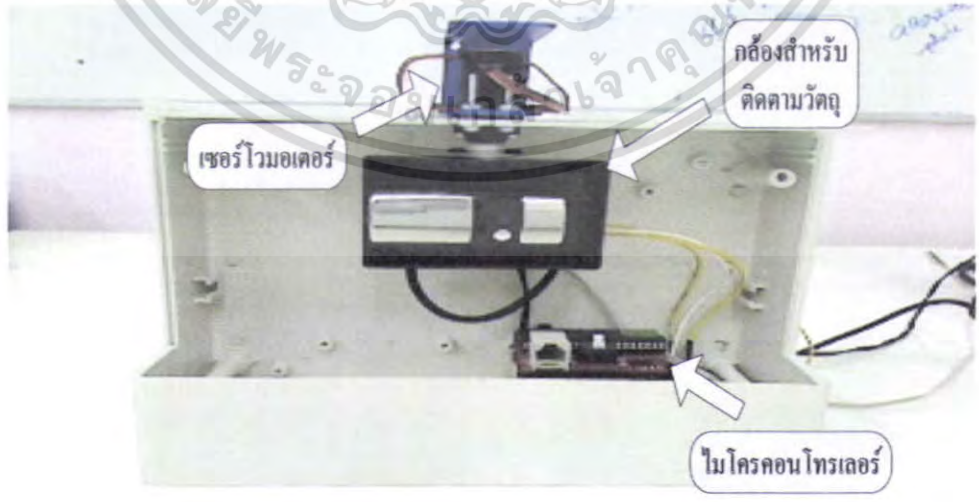
ภาพที่ 4.13 บอร์ดไมโครคอนโทรเลอร์ที่พร้อมใช้งาน (ด้านหน้าและด้านหลัง)

จากนั้นทำการต่อสายสัญญาณต่าง ๆ บนบอร์ดทดลองกับคอมพิวเตอร์และเซอร์โวมอเตอร์ เพื่อให้ไมโครคอนโทรเลอร์สามารถรับข้อมูลจากคอมพิวเตอร์ และนำข้อมูลดังกล่าวไปใช้ในการควบคุมเซอร์โวมอเตอร์ต่อไป ดังภาพที่ 4.14

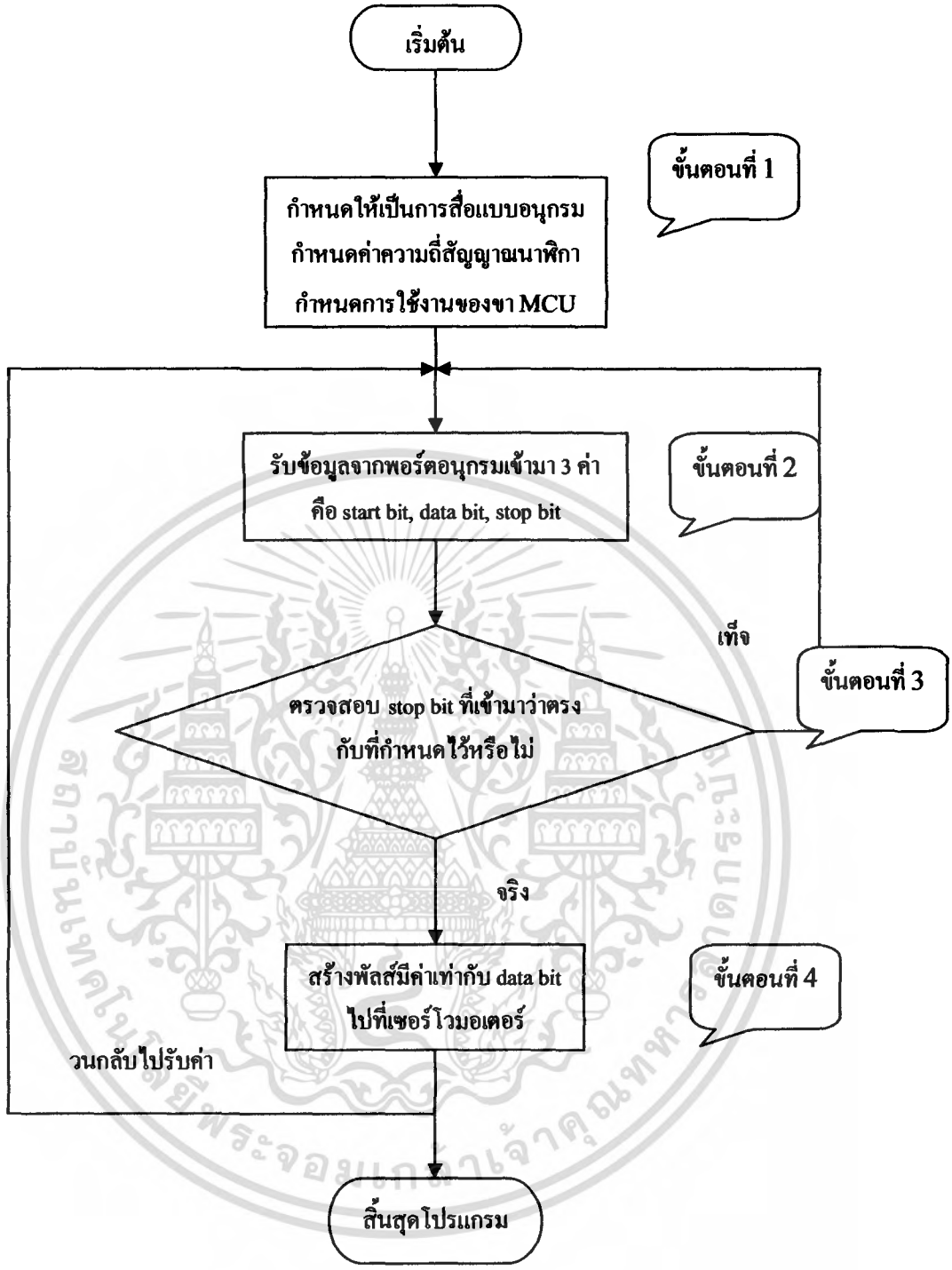


ภาพที่ 4.14 บอร์ดไมโครคอนโทรเลอร์เมื่อทำการต่อสายสัญญาณต่างๆแล้ว

สุดท้ายเป็นการนำส่วนประกอบต่าง ๆ ทั้งฮาร์ดแวร์และซอฟแวร์มาประกอบรวมกันเป็นชุดทดลองสำหรับการวิจัยในครั้งนี้ ดังภาพที่ 4.15



ภาพที่ 4.15 ชุดทดลองที่พร้อมนำมาใช้งาน



ภาพที่ 4.16 ขั้นตอนการทำงานของโปรแกรมภาษาเบสิก

จากภาพที่ 4.16 เป็นขั้นตอนการทำงานของโปรแกรมภาษาเบสิก ซึ่งจะอธิบายการทำงานได้ดังนี้

- ขั้นตอนที่ 1 กำหนดให้เป็นการสื่อแบบอนุกรม กำหนดค่าความถี่สัญญาณนาฬิกา และกำหนดการใช้งานของขา MCU โดยจะทำการผนวกไฟล์ modedefs.bas เข้าไปในโปรแกรมเพื่อใช้การสื่อสารแบบอนุกรม จากนั้นกำหนดค่าความถี่ของสัญญาณนาฬิกาเป็น 10 MHz และกำหนดการใช้งานของขา MCU ดังโปรแกรมด้านล่างนี้

```
INCLUDE "modedefs.bas"          'การผนวกไฟล์ modedefs.bas เข้ามาในโปรแกรมหลัก
define OSC 10                   'กำหนดค่าความถี่สัญญาณนาฬิกาหลักเท่ากับ 10 MHz
TRISB.1 = %00000000             'กำหนดให้ขา 1 ของพอร์ต B เป็นเอาต์พุต
TRISC.7 = %11111111             'กำหนดให้ขา 7 ของพอร์ต C เป็นอินพุต
```

- ขั้นตอนที่ 2 รับข้อมูลจากพอร์ตอนุกรมจำนวน 3 ค่า คือ start bit , data bit , stop bit จากโปรแกรมข้างล่าง โปรแกรมจะเปิดรับข้อมูลเข้ามาทางขา C.7 ของไมโครคอนโทรเลอร์ ตัวเลข 32 เป็นรหัสเพื่อบอกว่าการสื่อสารเชื่อมต่อด้วยอัตราบอดเรท (Baudrate) เท่ากับ 19200 บิทต่อวินาที ส่วนในวงเล็บใหญ่ คำสั่ง WAIT หมายถึงการรอข้อมูลเข้ามา 3 ข้อมูล โดยข้อมูลตัวแรกหมายถึง Start Bit จะต้องเป็นเครื่องหมาย * ตัวที่สองจะเป็นบิทข้อมูล และตัวสุดท้ายเป็น Stop Bit ซึ่งงานวิจัยนี้ได้กำหนดให้เป็นเลข 9 ตลอดงานวิจัย

```
loop: serin2 PORTC.7,32,[WAIT("*"),DEC3 pulse,DEC stop_bit]
```

- ขั้นตอนที่ 3 จะเป็นการตรวจสอบ stop bit ที่เข้ามาว่าตรงกับที่กำหนดไว้หรือไม่ ซึ่งในที่นี้คือ เลข 9 ดังโปรแกรมข้างล่าง ถ้าโปรแกรมตรวจสอบแล้วพบว่าไม่ใช่เลข 9 โปรแกรมก็จะวนรอบเพื่อรับข้อมูลเข้ามาใหม่อีกครั้ง

```
if stop_bit = 9 then                                    ' "เลขฐานสิบ" และเลข 9
      for i=1 to 200                                    'ไปเก็บในตัวแปร stop_bit
        PULSOUT PORTB.1,pulse     'สร้างพัลส์ออกไปที่พอร์ต B ขา 1 ขนาดเท่ากับ pulse
        PAUSE 20                  'หน่วงเวลา 20 millisec
      next
   endif
```

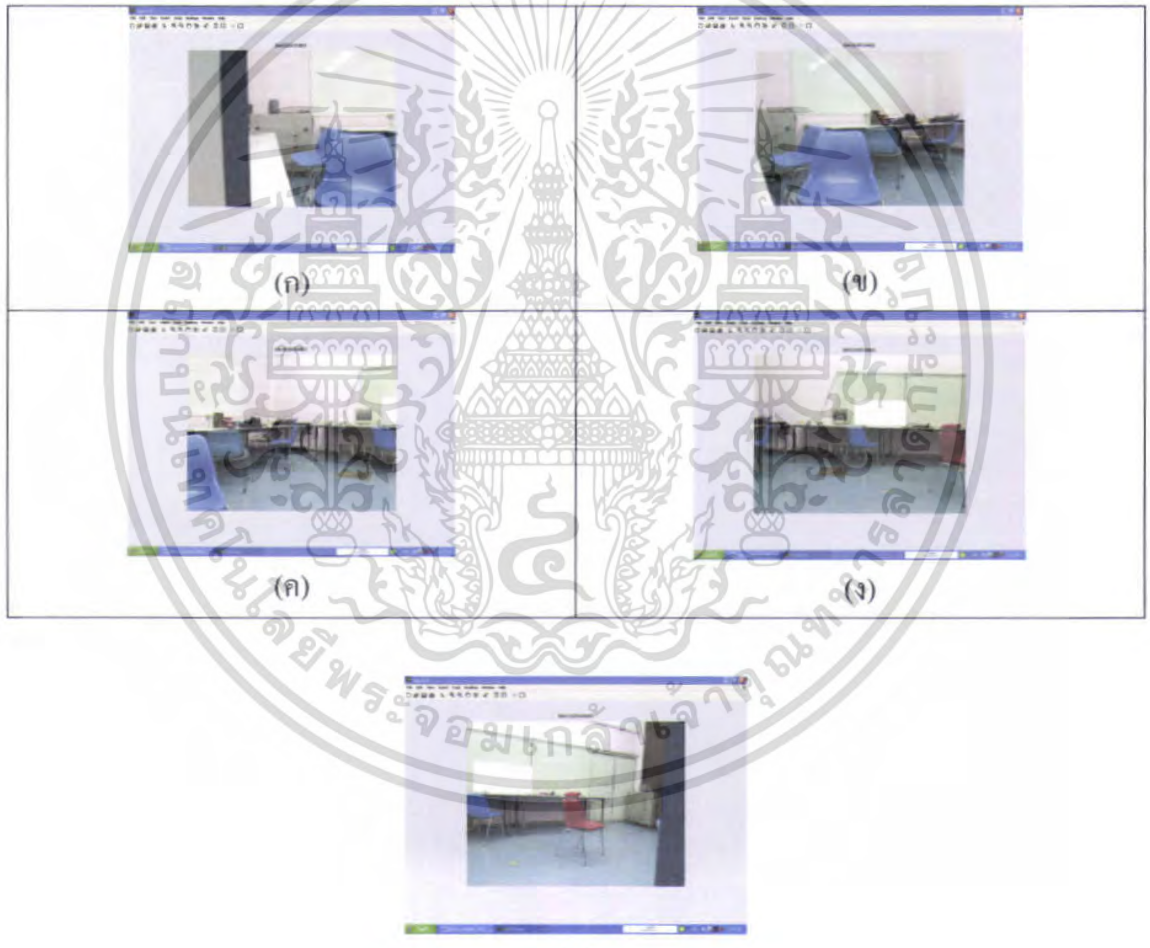
- ขั้นตอนที่ 4 ไมโครคอนโทรเลอร์จะสร้างพัลส์เท่ากับค่าของ data bit ที่รับเข้ามาใน ขั้นตอนที่ 2 โดยจะใช้คำสั่ง PULSEOUT ดังโปรแกรมข้างล่าง โดยพัลส์จะถูกส่งออกจากพอร์ต B บิทที่ 1 ของไมโครคอนโทรเลอร์ จากนั้นจะหน่วงเวลาไว้ 20 msec

```
PULSOUT PORTB.1,pulse     'สร้างพัลส์ออกไปที่พอร์ต B ขา 1 ขนาดเท่ากับ pulse
PAUSE 20                  'หน่วงเวลา 20 millisec
```

บทที่ 5

ผลการทดลอง

การทดลองจะแบ่งเป็น 2 ส่วนคือ การติดตามวัตถุ วัตถุเดียว และการติดตามวัตถุ 2 วัตถุ ซึ่งผลที่ได้จากการทดลองในแต่ละส่วน ดังแสดงในตารางที่ 5.1 และ 5.2 ตามลำดับ อย่างไรก็ตามในแต่ละตารางจะแสดงภาพพื้นหลังที่กล้องบันทึกเข้ามา ภาพของวัตถุที่กำลังเคลื่อนที่ และภาพผลลัพธ์ที่ได้จากการติดตามวัตถุ โดยโปรแกรมจะทำการตีกรอบล้อมรอบวัตถุเพื่อระบุถึงตำแหน่งของวัตถุในภาพนั้น ๆ



ภาพที่ 5.1 ภาพพื้นหลังทั้ง 5 ที่บันทึกเข้ามา

(ก) BACKGROUND1 (ข) BACKGROUND2 (ค) BACKGROUND3

(ง) BACKGROUND4 (จ) BACKGROUND5

จากภาพที่ 5.1 (ก) - (จ) แสดงภาพของพื้นหลังที่บันทึกเข้ามาเก็บไว้เพื่อนำไปใช้ในการเปรียบเทียบกับภาพที่ถ่ายเข้ามาใหม่เพื่อลบกัน โดยภาพที่ 5.1 (ก) เป็นภาพของพื้นหลังที่เก็บไว้ในตัวแปร ZONE 1 ภาพที่ 5.1 (ข) เป็นภาพของพื้นหลังที่เก็บไว้ในตัวแปร ZONE 2 ภาพที่ 5.1 (ค) เป็นภาพของพื้นหลังที่เก็บไว้ในตัวแปร ZONE 3 ภาพที่ 5.1 (ง) เป็นภาพของพื้นหลังที่เก็บไว้ในตัวแปร ZONE 4 และภาพที่ 5.1 (จ) เป็นภาพของพื้นหลังที่เก็บไว้ในตัวแปร ZONE 5

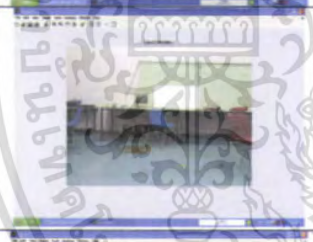
ตารางที่ 5.1 แสดงผลการทำงานของการติดตามวัตถุ

ลำดับที่	ภาพ BACKGROUND	ภาพ วัตถุกำลังเคลื่อนที่	ภาพผลลัพธ์ การติดตามวัตถุ
1			
2			
3			
4			
5			

ตารางที่ 5.1 แสดงผลการทำงานของการติดตามวัตถุ (ต่อ)

ลำดับที่	ภาพ BACKGROUND	ภาพ วัตถุกำลังเคลื่อนที่	ภาพผลลัพธ์ การติดตามวัตถุ
6			
7			
8			
9			
10			
11			
12			

ตารางที่ 5.1 แสดงผลการทำงานของการติดตามวัตถุ (ต่อ)

ลำดับที่	ภาพ BACKGROUND	ภาพ วัตถุกำลังเคลื่อนที่	ภาพผลลัพธ์ การติดตามวัตถุ
13			
14			
15			
16			
17			
18			
19			

ตารางที่ 5.1 แสดงผลการทำงานของการติดตามวัตถุ (ต่อ)

ลำดับที่	ภาพ BACKGROUND	ภาพ วัตถุกำลังเคลื่อนที่	ภาพผลลัพธ์ การติดตามวัตถุ
20			

การทดลองในส่วนนี้ เมื่อโปรแกรมเริ่มต้นทำงานกล้องจะเคลื่อนที่มาอยู่ที่ ZONE 3 แล้วเคลื่อนที่ไปที่ตำแหน่ง ZONE 1 , ZONE 2 , ZONE 3 , ZONE 4 และ ZONE 5 ตามลำดับ เพื่อบันทึกภาพ จากนั้นกล้องจะเคลื่อนที่มาอยู่ที่ ZONE 3 อีกครั้ง โดยในการทดลองวัตถุจะเคลื่อนที่จากซ้ายมือไปทางขวามือ และเคลื่อนที่กลับมาที่เดิม

- ภาพลำดับที่ 1 กล้องอยู่ที่ตำแหน่ง ZONE 3 เมื่อมีวัตถุเข้าในมุมมองของกล้องโปรแกรมก็ทำการบันทึกภาพและนำมาลบกับภาพพื้นหลังที่ตำแหน่ง ZONE 3 ที่ได้บันทึกไว้แล้ว ในภาพนี้จะเห็นได้ว่าโปรแกรมสามารถตีกรอบล้อมรอบวัตถุได้เพียงกรอบเล็ก ๆ เท่านั้น เนื่องจากมีวัตถุเข้ามาในมุมมองของกล้องไม่มากนัก ซึ่งจุด Centroids ของวัตถุมีค่าน้อยกว่า 50 พิกเซล ทำให้เซอร์โวมอเตอร์หมุนกล้องไปอยู่ใน ZONE ที่ 2

- ภาพลำดับที่ 2 กล้องอยู่ที่ตำแหน่ง ZONE 2 เมื่อมีวัตถุเข้าในมุมมองของกล้องโปรแกรมก็ทำการบันทึกภาพและนำมาลบกับภาพพื้นหลังที่ตำแหน่ง ZONE2 ที่ได้บันทึกไว้แล้ว ในภาพนี้จะเห็นได้ว่าโปรแกรมสามารถตีกรอบล้อมรอบวัตถุสามารถทำได้เพียง 0.5 เท่าของความสูงของวัตถุเท่านั้น เนื่องจากมีเก้าอี้สีฟ้าบังอยู่

- ภาพลำดับที่ 3 กล้องยังอยู่ที่ตำแหน่ง ZONE 2 เมื่อวัตถุเคลื่อนที่และอยู่ในมุมมองของกล้องโปรแกรมก็ทำการบันทึกภาพและนำมาลบกับภาพพื้นหลังที่ตำแหน่ง ZONE 2 ที่ได้บันทึกไว้แล้ว ในภาพนี้จะเห็นได้ว่าโปรแกรมสามารถตีกรอบล้อมรอบวัตถุได้รอบวัตถุ ซึ่งต่างจากภาพลำดับที่ 2 เพราะว่าเมื่อทำลบภาพกันแล้วจะได้ภาพวัตถุเต็มตัว เนื่องจากไม่โดนเก้าอี้บัง

- ภาพลำดับที่ 4 กล้องอยู่ที่ตำแหน่ง ZONE 2 วัตถุยังอยู่ในมุมมองของกล้องโปรแกรมก็ทำการบันทึกภาพและนำมาลบกับภาพพื้นหลังที่ตำแหน่ง ZONE 2 ที่ได้บันทึกไว้แล้วผลการติดตามวัตถุ โดยการตีกรอบล้อมรอบวัตถุทำสามารถตีกรอบได้รอบวัตถุ

- รูปลำดับที่ 5 กล้องอยู่ที่ตำแหน่ง ZONE 2 วัตถุเริ่มเคลื่อนที่ต่อไปและอยู่ในมุมมองของกล้องโปรแกรมก็ทำการบันทึกภาพและนำมาลบกับภาพพื้นหลังที่ตำแหน่ง ZONE 2 ที่ได้บันทึกไว้

ผลการติดตามวัตถุ โดยการตีกรอบล้อมรอบวัตถุทำสามารถตีกรอบได้รอบวัตถุ อย่างไรก็ตามจุด Centroids ของภาพมีค่ามากกว่า 280 พิกเซล ดังนั้นเซอร์โวมอเตอร์จึงหมุนกล้องกลับไปยัง ZONE ที่ 3

- ภาพลำดับที่ 6 กล้องอยู่ที่ตำแหน่ง ZONE 3 วัตถุอยู่ในมุมมองของกล้องโปรแกรมก็ทำการบันทึกภาพและนำมาลบกับภาพพื้นหลังที่ตำแหน่ง ZONE 3 ที่ได้บันทึกไว้แล้วตามลำดับ ซึ่งในภาพจะเห็นได้ว่าโปรแกรมสามารถตีกรอบได้รอบวัตถุ

- ภาพลำดับที่ 7 กล้องยังอยู่ที่ตำแหน่ง ZONE 3 วัตถุเคลื่อนที่ไปและอยู่ในมุมมองของกล้องโปรแกรมก็ทำการบันทึกภาพและนำมาลบกับภาพพื้นหลังที่ตำแหน่ง ZONE 3 ที่ได้บันทึกไว้ ซึ่งในภาพจะเห็นว่าโปรแกรมสามารถตีกรอบได้รอบวัตถุ อย่างไรก็ตามในขณะนี้เซอร์โวมอเตอร์ยังไม่มีการหมุน เนื่องจาก Centroids ของวัตถุยังอยู่ในช่วง 50 ~ 280 พิกเซล

- ภาพลำดับที่ 8 กล้องยังอยู่ที่ตำแหน่ง ZONE 3 วัตถุเข้าในมุมมองของกล้องโปรแกรมก็ทำการบันทึกภาพและนำมาลบกับภาพพื้นหลังที่ตำแหน่ง ZONE 3 ที่ได้บันทึกไว้แล้ว ผลการติดตามวัตถุ โดยการตีกรอบล้อมรอบวัตถุทำได้ไม่รอบวัตถุเนื่องจากวัตถุกำลังเคลื่อนออกนอกมุมมองของกล้อง ทำให้เซอร์โวมอเตอร์หมุนกล้องให้ไปอยู่ใน ZONE ที่ 4

- ภาพลำดับที่ 9 กล้องอยู่ที่ตำแหน่ง ZONE 4 วัตถุอยู่ในมุมมองของกล้องโปรแกรมก็ทำการบันทึกภาพและนำมาลบกับภาพพื้นหลังที่ตำแหน่ง ZONE 4 ที่ได้บันทึกไว้แล้ว ซึ่งในภาพจะเห็นได้ว่าโปรแกรมสามารถตีกรอบได้รอบวัตถุ

- ภาพลำดับที่ 10 กล้องอยู่ที่ตำแหน่ง ZONE 4 วัตถุอยู่หลังเก้าอี้และอยู่ในมุมมองของกล้องโปรแกรมก็ทำการบันทึกภาพและนำมาลบกับภาพพื้นหลังที่ตำแหน่ง ZONE 4 ที่ได้บันทึกไว้ ซึ่งในภาพจะเห็นได้ว่าโปรแกรมสามารถตีกรอบได้ไม่รอบวัตถุ เนื่องจากมีเก้าอี้บัง อย่างไรก็ตามเนื่องจากวัตถุกำลังเคลื่อนที่ออกนอกมุมมองของกล้อง เซอร์โวมอเตอร์จึงทำงานหมุนกล้องไปอยู่ใน ZONE ที่ 5

- ภาพลำดับที่ 11 กล้องอยู่ที่ตำแหน่ง ZONE 5 วัตถุอยู่ในมุมมองของกล้องโปรแกรมก็ทำการบันทึกภาพและนำมาลบกับภาพพื้นหลังที่ตำแหน่ง ZONE 5 ที่ได้บันทึกไว้แล้ว ซึ่งในภาพจะเห็นได้ว่าโปรแกรมสามารถตีกรอบได้รอบวัตถุ ขณะนี้วัตถุเคลื่อนที่ไปในทิศทางตรงกันข้าม

- ภาพลำดับที่ 12 กล้องอยู่ที่ ZONE 5 วัตถุอยู่ในมุมมองของกล้องโปรแกรมก็ทำการบันทึกภาพและนำมาลบกับภาพพื้นหลังที่ ZONE 5 ที่ได้บันทึกไว้แล้ว ซึ่งในภาพจะเห็นได้ว่าโปรแกรมสามารถตีกรอบได้ไม่รอบวัตถุ เพราะมีเก้าอี้บัง

- ภาพลำดับที่ 13 กล้องอยู่ที่ ZONE 5 วัตถุเคลื่อนที่ต่อไปและอยู่ในมุมมองของกล้องโปรแกรมก็ทำการบันทึกภาพและนำมาลบกับภาพพื้นหลังที่ ZONE 5 ตามที่ได้บันทึกไว้แล้ว ซึ่งในภาพจะเห็นได้ว่าโปรแกรมสามารถตีกรอบได้รอบวัตถุ

- ภาพลำดับที่ 14 กล้องอยู่ที่ ZONE 5 วัตถุอยู่ในมุมมองของกล้องโปรแกรมก็ทำการบันทึกภาพและนำมาลบกับภาพพื้นหลังที่ ZONE 5 ที่ได้บันทึกไว้แล้ว ซึ่งในภาพจะเห็นได้ว่าโปรแกรมสามารถตีกรอบได้รอบวัตถุ ขณะนี้จุด Centroids ของวัตถุยังอยู่ในเงื่อนไขที่กำหนดเซอร์โวมอเตอร์จะยังไม่ทำงาน

- ภาพลำดับที่ 15 กล้องอยู่ที่ ZONE 5 วัตถุอยู่ในมุมมองของกล้องโปรแกรมก็ทำการบันทึกภาพและนำมาลบกับภาพพื้นหลังที่ ZONE 5 ที่ได้บันทึกไว้แล้ว ในภาพนี้จะเห็นได้ว่าโปรแกรมสามารถตีกรอบล้อมรอบวัตถุได้เพียงกรอบเล็ก ๆ เท่านั้น เนื่องจากมีวัตถุเข้ามาในมุมมองของกล้องไม่มากนัก และจุด Centroids ของวัตถุมีค่าน้อยกว่า 50 พิกเซล ทำให้เซอร์โวมอเตอร์หมุนกล้องกลับไปใน ZONE ที่ 4

- ภาพลำดับที่ 16 กล้องอยู่ที่ ZONE 4 วัตถุอยู่ในมุมมองของกล้องโปรแกรมก็ทำการบันทึกภาพและนำมาลบกับภาพพื้นหลังที่ ZONE 4 ที่ได้บันทึกไว้แล้ว ซึ่งในภาพจะเห็นได้ว่าโปรแกรมสามารถตีกรอบได้รอบวัตถุ

- ภาพลำดับที่ 17 กล้องอยู่ที่ ZONE 4 วัตถุอยู่ในมุมมองของกล้องโปรแกรมก็ทำการบันทึกภาพและนำมาลบกับภาพพื้นหลังที่ ZONE 4 ที่ได้บันทึกไว้แล้ว ซึ่งในภาพจะเห็นได้ว่าโปรแกรมสามารถตีกรอบได้รอบวัตถุ เนื่องจากจุด Centroids ของวัตถุมีค่าน้อยกว่า 50 พิกเซล เซอร์โวมอเตอร์จึงหมุนกล้องไปอยู่ใน ZONE ที่ 3

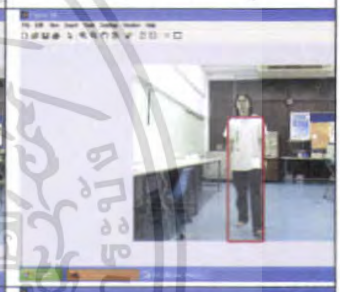
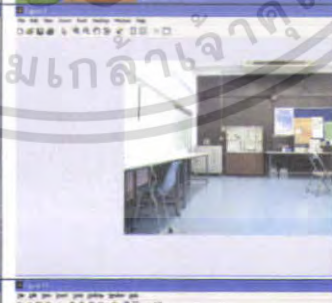
- ภาพลำดับที่ 18 กล้องอยู่ที่ ZONE 3 วัตถุอยู่ในมุมมองของกล้องโปรแกรมก็ทำการบันทึกภาพและนำมาลบกับภาพพื้นหลังที่ ZONE 3 ที่ได้บันทึกไว้แล้ว ซึ่งในภาพจะเห็นได้ว่าโปรแกรมสามารถตีกรอบได้รอบวัตถุ

- ภาพลำดับที่ 19 กล้องอยู่ที่ ZONE 3 วัตถุยังอยู่ในมุมมองของกล้องโปรแกรมก็ทำการบันทึกภาพและนำมาลบกับภาพพื้นหลังที่ ZONE 3 ที่ได้บันทึกไว้แล้ว ซึ่งในภาพจะเห็นได้ว่าโปรแกรมสามารถตีกรอบได้รอบวัตถุ

- ภาพลำดับที่ 20 กล้องอยู่ที่ ZONE 3 วัตถุอยู่ในมุมมองของกล้องโปรแกรมก็ทำการบันทึกภาพและนำมาลบกับภาพพื้นหลังที่ ZONE 3 ที่ได้บันทึกไว้แล้ว ในภาพนี้จะเห็นได้ว่าโปรแกรมสามารถตีกรอบล้อมรอบวัตถุสามารถทำได้เพียง 0.5 เท่าของความสูงของวัตถุเท่านั้น เนื่องจากมีเก้าอี้สีฟ้าบังอยู่ นอกจากนี้เซอร์โวมอเตอร์ยังหมุนกล้องไปอยู่ใน ZONE ที่ 2

สำหรับการติดตามวัตถุที่มี 2 วัตถุ ก็จะมีหลักการเดียวกับการติดตามวัตถุที่มี 1 วัตถุ แต่จะมีเงื่อนไขเพิ่มเติมเข้ามาเพื่อตัดสินใจในการติดตามวัตถุที่มี 2 วัตถุ ถ้าในการประมวลผลภาพพบว่ามีวัตถุ 2 วัตถุ โปรแกรมก็จะทำการตีกรอบเพื่อระบุตำแหน่งของวัตถุทั้ง 2 แต่โปรแกรมจะไม่ส่งค่าใดๆ ไปที่ไมโครคอนโทรเลอร์เพื่อควบคุมเซอร์โวมอเตอร์ จนกว่าวัตถุใดวัตถุหนึ่งจะออกไปจากมุมมองของกล้องเหลือเพียงวัตถุเดียว จากนั้นโปรแกรมจึงจะทำการส่งค่าตำแหน่งไปที่ไมโครคอนโทรเลอร์เพื่อควบคุมเซอร์โวมอเตอร์ต่อไป การติดตามวัตถุที่มี 2 วัตถุ แสดงดังตารางที่ 5.2

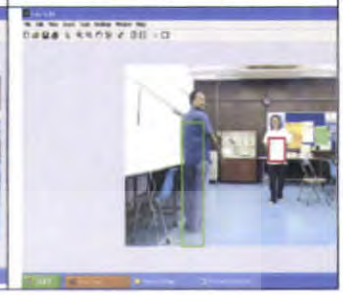
ตารางที่ 5.2 แสดงผลการทำงานของการติดตามวัตถุที่มี 2 วัตถุ

ลำดับที่	ภาพ BACKGROUND	ภาพ วัตถุกำลังเคลื่อนที่	ภาพผลลัพธ์ การติดตามวัตถุ
1			
2			
3			
4			

ตารางที่ 5.2 แสดงผลการทำงานของการติดตามวัตถุที่มี 2 วัตถุ (ต่อ)

ลำดับที่	ภาพ BACKGROUND	ภาพ วัตถุกำลังเคลื่อนที่	ภาพผลลัพธ์ การติดตามวัตถุ
5			
6			
7			
8			
9			

ตารางที่ 5.2 แสดงผลการทำงานของการติดตามวัตถุที่มี 2 วัตถุ (ต่อ)

ลำดับที่	ภาพ BACKGROUND	ภาพ วัตถุกำลังเคลื่อนที่	ภาพผลลัพธ์ การติดตามวัตถุ
10			

จากตารางที่ 5.2 แสดงผลลัพธ์ของการติดตามวัตถุที่เป็น 2 วัตถุ โดยจะแสดงภาพของพื้นหลัง ภาพของวัตถุที่กำลังเคลื่อนที่ และภาพผลลัพธ์การติดตามวัตถุ

- ภาพลำดับที่ 1 วัตถุเข้ามาในมุมมองของกล้อง 1 วัตถุ ทิศทางการเดินเข้าหากล้อง การระบุตำแหน่งด้วยการตีกรอบพบว่าสามารถระบุตำแหน่งของวัตถุได้ แต่ไม่ครอบคลุมทั้งตัวของวัตถุทั้งหมด เนื่องจากในการติดตามวัตถุแบบ 2 วัตถุ จะต้องทำให้รูปร่างของวัตถุในภาพไม่ขาดออกจากกันเพื่อที่จะสามารถตีกรอบให้เป็นวัตถุเดียวกันได้ โดยการใช้คำสั่ง imclose จึงทำให้รูปร่างของวัตถุถูกเซาะ เป็นผลให้รูปร่างของวัตถุหายไปบางส่วน

- ภาพลำดับที่ 2 วัตถุเข้ามาในมุมมองของกล้อง 1 วัตถุ ทิศทางการเดินเข้าหากล้อง เช่นเดียวกับภาพที่ 1 การระบุตำแหน่งด้วยการตีกรอบพบว่าสามารถระบุตำแหน่งของวัตถุได้ และครอบคลุมทั้งตัวของวัตถุทั้งหมด

- ภาพลำดับที่ 3 วัตถุยังอยู่ในมุมมองของกล้อง ทิศทางการเคลื่อนที่ของวัตถุจากซ้ายไปขวา การระบุตำแหน่งด้วยการตีกรอบพบว่าสามารถระบุตำแหน่งของวัตถุได้ แต่ไม่ครอบคลุมทั้งตัวของวัตถุทั้งหมด เนื่องจากในการติดตามวัตถุแบบ 2 วัตถุ จะต้องทำให้รูปร่างของวัตถุในภาพไม่ขาดออกจากกันเพื่อที่จะสามารถตีกรอบให้เป็นวัตถุเดียวกันได้ โดยการใช้คำสั่ง imclose จึงทำให้รูปร่างของวัตถุถูกเซาะ เป็นผลให้รูปร่างของวัตถุหายไปบางส่วน

- ภาพลำดับที่ 4 วัตถุเข้ามาในมุมมองของกล้อง 2 วัตถุ หยุดอยู่ด้านหน้ากล้อง การระบุตำแหน่งด้วยการตีกรอบพบว่าสามารถระบุตำแหน่งของวัตถุได้ แต่ไม่ครอบคลุมทั้งตัวของวัตถุ เนื่องจากในการติดตามวัตถุแบบ 2 วัตถุ จะต้องทำให้รูปร่างของวัตถุในภาพไม่ขาดออกจากกันเพื่อที่จะสามารถตีกรอบให้เป็นวัตถุเดียวกันได้ โดยการใช้คำสั่ง imclose จึงทำให้รูปร่างของวัตถุถูกเซาะ เป็นผลให้รูปร่างของวัตถุหายไปบางส่วน

- ภาพลำดับที่ 5 วัตถุเข้ามาในมุมมองของกล้อง 2 วัตถุ หยุดอยู่หน้ากล้องแต่มีขยับเล็กน้อย การระบุตำแหน่งด้วยการตีกรอบพบว่าสามารถระบุตำแหน่งของวัตถุได้ แต่ไม่ครอบคลุมทั้งตัวของวัตถุทั้งหมด เนื่องจากในการติดตามวัตถุแบบ 2 วัตถุ จะต้องทำให้รูปร่างของวัตถุในภาพไม่ขาดออกจากกันเพื่อที่จะสามารถตีกรอบให้เป็นวัตถุเดียวกันได้ โดยการใช้คำสั่ง imclose จึงทำให้รูปร่างของวัตถุถูกเซาะ เป็นผลให้รูปร่างของวัตถุหายไปบางส่วน

- ภาพลำดับที่ 6 วัตถุเข้ามาในมุมมองของกล้อง 2 วัตถุ โดยวัตถุหนึ่งเดินเข้ามาหน้ากล้องและอีกวัตถุเดินออกไปจากกล้อง การระบุตำแหน่งด้วยการตีกรอบพบว่าสามารถระบุตำแหน่งของวัตถุได้ แต่ไม่ครอบคลุมทั้งตัวของวัตถุทั้งหมด เนื่องจากเหตุผลที่กล่าวไปแล้วข้างต้น

- ภาพลำดับที่ 7 วัตถุเข้ามาในมุมมองของกล้อง 2 วัตถุ โดยวัตถุที่อยู่หน้ากล้องขยับออกไปและอีกวัตถุเดินเข้าหากล้อง การระบุตำแหน่งด้วยการตีกรอบพบว่าสามารถระบุตำแหน่งของวัตถุได้ แต่ไม่ครอบคลุมทั้งตัวของวัตถุทั้งหมด เนื่องจากในการติดตามวัตถุแบบ 2 วัตถุ จะต้องทำให้รูปร่างของวัตถุในภาพไม่ขาดออกจากกันเพื่อที่จะสามารถตีกรอบให้เป็นวัตถุเดียวกันได้ โดยการใช้คำสั่ง imclose จึงทำให้รูปร่างของวัตถุถูกเซาะ เป็นผลให้รูปร่างของวัตถุหายไปบางส่วน

- ภาพลำดับที่ 8 วัตถุเข้ามาในมุมมองของกล้อง 2 วัตถุ โดยวัตถุที่อยู่ใกล้กับกล้องกำลังขยับเพื่อขึ้นนั่งบนโต๊ะ และอีกวัตถุหยุดเดิน การระบุตำแหน่งด้วยการตีกรอบพบว่าสามารถระบุตำแหน่งของวัตถุได้ แต่ไม่ครอบคลุมทั้งตัวของวัตถุทั้งหมด

- ภาพลำดับที่ 9 วัตถุเข้ามาในมุมมองของกล้อง 2 วัตถุ โดยวัตถุที่อยู่ใกล้กับกล้องนั่งอยู่บนโต๊ะ และอีกวัตถุกำลังเดินเข้าหากล้อง การระบุตำแหน่งด้วยการตีกรอบพบว่าสามารถระบุตำแหน่งของวัตถุได้ แต่ไม่ครอบคลุมทั้งตัวของวัตถุทั้งหมด

- ภาพลำดับที่ 10 วัตถุเข้ามาในมุมมองของกล้อง 2 วัตถุ โดยวัตถุที่อยู่ใกล้กับกล้องกำลังเดินออกไปจากกล้อง และอีกวัตถุกำลังเดินเข้าหากล้อง การระบุตำแหน่งด้วยการตีกรอบพบว่าสามารถระบุตำแหน่งของวัตถุได้ แต่ไม่ครอบคลุมทั้งตัวของวัตถุทั้งหมด

บทที่ 6
สรุปผลการวิจัยและข้อเสนอแนะ

6.1 สรุปผลการวิจัย

ผลการทดลองเรื่องการติดตามวัตถุ จะแบ่งออกเป็น 3 หัวข้อ คือ 1) กล้องเคลื่อนที่ติดตามวัตถุ 2) การตีกรอบเพื่อระบุตำแหน่งของวัตถุในภาพ และ 3) ส่วนของไมโครคอนโทรเลอร์และเซอร์โวมอเตอร์ สำหรับหัวข้อที่ 1 กล้องเคลื่อนที่ติดตามวัตถุนั้น จากการทดลองพบว่ากล้องสามารถเคลื่อนที่ติดตามวัตถุไปได้อย่างไม่มีข้อผิดพลาดใดๆ ถ้าวัตถุยังเคลื่อนที่อยู่ในพื้นที่ทำงานหรือ ประมาณ 150 องศา ในส่วนนี้เองยังคงมีสิ่งที่ต้องปรับปรุงอีก เช่น ความเร็วในการติดตามวัตถุที่ยังช้าเกินไป เป็นต้น ในหัวข้อที่ 2 การตีกรอบเพื่อระบุตำแหน่งของวัตถุในภาพ จากการทดลองพบว่าผลลัพธ์ที่ได้ส่วนใหญ่สามารถตีกรอบล้อมรอบวัตถุได้ แต่ก็มีสิ่งที่ต้องปรับปรุงอีกอีก เช่น บางครั้งการตีกรอบไม่ครอบคลุมทั้งวัตถุ เนื่องจากการใช้คำสั่ง imclose เพื่อทำให้วัตถุในภาพไม่ขาดออกจากกัน เป็นต้น และหัวข้อที่ 3 ไมโครคอนโทรเลอร์และเซอร์โวมอเตอร์ สำหรับการทำงานในส่วนนี้ มีการทำงานสัมพันธ์กันเป็นอย่างดีโดยไมโครคอนโทรเลอร์รับค่าจากโปรแกรม MATLAB และสร้างพัลส์เพื่อควบคุมเซอร์โวมอเตอร์ได้อย่างต่อเนื่องไม่มีความผิดพลาดใด ๆ เกิดขึ้นเลย

สรุปการผลการทดลอง โดยรวมทั้งซอฟแวร์และฮาร์ดแวร์ต่างๆ ที่นำมาใช้ในการวิจัยครั้งนี้สามารถทำงานสัมพันธ์กันเป็นอย่างดี จะมีส่วนที่ต้องปรับปรุงก็คงจะเป็นเรื่องของความเร็ว ทั้งจากโปรแกรมที่เขียนขึ้นในขั้นตอนการประมวลผลภาพ และจากไมโครคอนโทรเลอร์ที่อาจจะมีการเปลี่ยนแปลงรุ่นหรือตระกูลเพื่อการประมวลผลที่เร็วมากขึ้น เป็นต้น

6.2 ปัญหาที่เกิดขึ้นในการทำวิจัย

1. โปรแกรมที่ออกแบบไว้ประมวลผลช้า ทำให้วัตถุที่เคลื่อนที่ไม่สามารถเคลื่อนที่ได้เร็วมากนัก

2. ขณะที่กล้องกำลังหมุนตามวัตถุที่กำลังเคลื่อนที่ ถ้าไมโครคอนโทรลเลอร์หยุดการสร้างพัลส์เพื่อควบคุมเซอร์โวมอเตอร์ เฟืองของเซอร์โวมอเตอร์จะขบกันไม่สนิท ทำให้เฟืองยังมีการขยับอยู่ เป็นผลทำให้ตำแหน่งของกล้องอาจจะผิดพลาดไปเล็กน้อย

6.3 ข้อเสนอแนะในการพัฒนางานวิจัย

ในงานวิจัยฉบับนี้มีข้อเสนอแนะเพื่อการพัฒนาครั้งต่อไปดังนี้

1. เรื่องของสัญญาณรบกวนที่เกิดจากกล้องเอง หรือเกิดจากสภาพแวดล้อมรอบตัว เช่น แสง

2. เรื่องของแนวคิดอัลกอริทึมในการเขียนโปรแกรม ที่มีขั้นตอนการคำนวณมากก็จะทำให้ใช้เวลาในการประมวลผลมากเช่นกัน

3. เรื่องของโปรแกรมจะตัดสินใจอย่างไร เมื่อมีวัตถุ 2 วัตถุ เข้ามาในมุมมองของกล้อง นอกจากการหยุดนิ่งของกล้องเพื่อรอให้วัตถุใดวัตถุหนึ่งออกไปจากมุมมองของกล้องเสียก่อน

บรรณานุกรม

[1] กิตติ ไพฑูรย์วัฒนกิจ. 2549. การประมวลผลภาพดิจิตอลเบื้องต้น. ภาควิชาวิศกรรมอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง พิมพ์ครั้งที่ 1

[2] ชูชาติ ปิณฑวิรุจน์. 2550. การประมวลผลภาพดิจิตอลด้วย Matlab. ภาควิชาวิศกรรมอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง พิมพ์ครั้งที่ 1

[3] สมเกียรติ อุดมหรรษากุล. 2550 . การประมวลผลภาพเบื้องต้น. ภาควิชาวิศกรรมสารสนเทศ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง : พิมพ์ครั้งที่ 1

[4] มนัส สังวรศิลป์. 2543. คู่มือโปรแกรม MATLAB ฉบับบสมบูรณ์. พิมพ์ครั้งที่ 1. กรุงเทพฯ สำนักพิมพ์ อินโฟเพรส

[5] ลัญฉกร วุฒิสิทธิกุลกิจ และคณะ. 2549. การใช้งานโปรแกรม MATLAB เบื้องต้น. พิมพ์ครั้งที่ 1. กรุงเทพฯ : สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย

[6] วัชรินทร์ เดารพ. 2546. "เรียนรู้และเข้าใจสถาปัตยกรรมไมโครคอนโทรเลอร์ PIC16F877" พิมพ์ครั้งที่ 1. กรุงเทพฯ : บริษัทอีทีที จำกัด

[7] วัชรินทร์ เดารพ. 2547. "เรียนรู้และเข้าใจไมโครคอนโทรเลอร์ PIC ด้วยภาษาเบสิก" พิมพ์ครั้งที่ 1. กรุงเทพฯ : บริษัทอีทีที จำกัด

[8] กฤษดา ใจเย็นม,ณัฏฐพล วงศ์สุนทรชัย,ชัยวัฒน์ ลิ้มพรจิตวิไล. 2546. "เรียนรู้และใช้งาน PICBASIC PRO คอมไพเลอร์ เขียนโปรแกรมภาษาเบสิกควบคุมไมโครคอนโทรเลอร์ PIC" กรุงเทพฯ : บริษัทอินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด

[9] สิทธิโชค ยอดระยับ. 2550. การเขียนโปรแกรม Digital Image Processing ด้วย Visual Basic พิมพ์ครั้งที่ 1. กรุงเทพ : สำนักพิมพ์ ส.ส.ท.

[10] นิรุธ อำนวยศิลป์. ประมวลผลภาพและวิดีโอด้วย C/C++. กรุงเทพฯ : ดวงกมลสมัย จำกัด

[11] นที เดยะดี,มณีรัตน์ คลื่นสนั่น,สิทธิพร ถึกสกุล. 2547. การติดตามวัตถุ ปริญญานิพนธ์ ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

[12] Rafael C. Gonzales,Richard E. Woods,Steven L. Eddins. 2004. Digital Image Processing USING MATLAB. USA : Pearson Education,Inc.

[13] ALASDAIR Mc ANDREW. 2004. INTRODUCTION TO DIGITAL IMAGE PROCESSING. USA : Thomson Learning, Inc.

ภาคผนวก

ภาคผนวก ก.
โฟลว์ชาร์ตและโปรแกรมควบคุม

1. โปรแกรมการทำงานของการติดตามวัตถุ (MATLAB)

```
clear all;                %ล้างตัวแปรทั้งหมดในหน่วยความจำ
clc;                      %ล้างหน้าต่างการทำงาน
center_axis = 160;        %กำหนดค่าให้กับตัวแปร center_axis เท่ากับ 160
pulse = 1.5;              %กำหนดค่าให้กับตัวแปร pulse เท่ากับ 1.5
stop = 9;                 %กำหนดค่าให้กับตัวแปร stop เท่ากับ 9
s = serial('COM4')        %เปิดการสื่อสารแบบอนุกรมที่พอร์ต COM4 ให้เป็นตัวแปร s
set(s,'BaudRate',19200,'DataBits',8,'Parity','none','StopBits',1,'FlowControl','none')  %กำหนดค่าให้ s
fopen(s)                  %เปิดการทำงาน s
for q=1:50                %วนรอบแบบ for-loop จำนวน 50 รอบ
   fprintf(s,'*');        %ส่ง * ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[375]);      %ส่งตัวเลข 375 ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[stop]);     %ส่งข้อมูล stop เท่ากับ 9 ออกไปทางพอร์ตอนุกรม
end                       %จบการทำงาน for-loop
vid = videoinput('winvideo',1,'RGB24_320x240') %นำภาพวิดีโอเข้ามา กำหนดเป็น 320x240 pixels
scr = getselectedsource(vid);
set(scr,'FrameRate','10')       %กำหนดค่าของภาพวิดีโอเป็น 10 เฟรมต่อนาที
set(vid,'TriggerRepeat',Inf)    %กำหนดทริกเกอร์เป็น infinuty
set(vid,'framesPerTrigger',1)   %กำหนดให้ปืน 1 เฟรมต่อการทริกเกอร์
preview(vid)                    %แสดงภาพวิดีโอ
start(vid)                      %เริ่มต้นนำภาพวิดีโอเข้ามา
msgbox('READY')                 %แสดงข้อความ READY
%%%%%%%%%%%%%% ZONE 1 %%%%%%%%%%%%%%
for q=1:50                      %วนรอบแบบ for-loop จำนวน 50 รอบ
```

```
   fprintf(s,'*');                %ส่ง * ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[250]);       %ส่งตัวเลข 250 ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[stop]);      %ส่งข้อมูล stop เท่ากับ 9  ออกไปทางพอร์ตอนุกรม
end                               %จบการทำงาน for-loop
pause(2)          %หน่วงเวลา  2 วินาที
BACKGROUND1 = getsnapshot(vid);      %บันทึกภาพเข้ามา ตั้งชื่อเป็น BACKGROUND1
BACKGROUND1_GRAY = rgb2gray(BACKGROUND1);  %แปลงภาพสี BACKGROUND1เป็น
                                                    %ภาพระดับเทา
%%%%%%%%%%%%%% ZONE 2 %%%%%%%%%%%%%%
for q=1:50                        %วนรอบแบบ for-loop จำนวน 50 รอบ
   fprintf(s,'*');                %ส่ง * ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[313]);       %ส่งตัวเลข 313 ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[stop]);      %ส่งข้อมูล stop เท่ากับ 9  ออกไปทางพอร์ตอนุกรม
end                               %จบการทำงาน for-loop
pause(2)                          %หน่วงเวลา  2 วินาที
BACKGROUND2 = getsnapshot(vid);          %บันทึกภาพเข้ามา ตั้งชื่อเป็น BACKGROUND1
BACKGROUND2_GRAY = rgb2gray(BACKGROUND2);       %แปลงภาพสี BACKGROUND2
                                                    % เป็นภาพระดับเทา
%%%%%%%%%%%%%% ZONE 3 %%%%%%%%%%%%%%
for q=1:50                        %วนรอบแบบ for-loop จำนวน 50 รอบ
   fprintf(s,'*');                %ส่ง * ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[375]);       %ส่งตัวเลข 375 ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[stop]);      %ส่งข้อมูล stop เท่ากับ 9  ออกไปทางพอร์ตอนุกรม
end                               %จบการทำงาน for-loop
pause(2)                          %หน่วงเวลา  2 วินาที
BACKGROUND3 = getsnapshot(vid);          %บันทึกภาพเข้ามา ตั้งชื่อเป็น BACKGROUND3
BACKGROUND3_GRAY = rgb2gray(BACKGROUND3);     %แปลงภาพสี BACKGROUND3
                                                    % เป็นภาพระดับเทา
```

```
%%%%%%%%%%%%%% ZONE 4 %%%%%%%%%%%%%%
for q=1:50                          %วนรอบแบบ for-loop จำนวน 50 รอบ
   fprintf(s,'*');                  %ส่ง * ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[438]);         %ส่งตัวเลข 438 ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[stop]);        %ส่งข้อมูล stop เท่ากับ 9  ออกไปทางพอร์ตอนุกรม
end              %จบการทำงาน for-loop
pause(2)         %หน่วงเวลา  2 วินาที
BACKGROUND4 = getsnapshot(vid);          %บันทึกภาพเข้ามา ตั้งชื่อเป็น BACKGROUND4
BACKGROUND4_GRAY = rgb2gray(BACKGROUND4);       %แปลงภาพสี BACKGROUND4
                                                 % เป็นภาพระดับเทา
%%%%%%%%%%%%%% ZONE 5 %%%%%%%%%%%%%%
for q=1:50                          %วนรอบแบบ for-loop จำนวน 50 รอบ
   fprintf(s,'*');                  %ส่ง * ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[500]);         %ส่งตัวเลข 500 ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[stop]);        %ส่งข้อมูล stop เท่ากับ 9  ออกไปทางพอร์ตอนุกรม
end                                 %จบการทำงาน for-loop
pause(2)                            %หน่วงเวลา  2 วินาที
BACKGROUND5 = getsnapshot(vid);      %บันทึกภาพเข้ามา ตั้งชื่อเป็น BACKGROUND5
BACKGROUND5_GRAY = rgb2gray(BACKGROUND5);       %แปลงภาพสี BACKGROUND5
                                                 % เป็นภาพระดับเทา
%%%%%%%%%%%%%% CENTER ZONE %%%%%%%%%%%%%%
for q=1:50                          %วนรอบแบบ for-loop จำนวน 50 รอบ
   fprintf(s,'*');                  %ส่ง * ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[375]);         %ส่งตัวเลข 375 ออกไปทางพอร์ตอนุกรม
   fprintf(s,'%d\n',[stop]);        %ส่งข้อมูล stop เท่ากับ 9  ออกไปทางพอร์ตอนุกรม
end                                 %จบการทำงาน for-loop
ZONE = 3;                           %กำหนดตัวแปร ZONE เท่ากับ 3
move=0;                             %กำหนดตัวแปร move เท่ากับ 0
for u=1:100                         %วนรอบแบบ for-loop จำนวน 100 รอบ
```

```
FRAME = getsnapshot(vid);          %บันทึกภาพเข้ามา ตั้งชื่อเป็น FRAME
FRAME_GRAY = rgb2gray(FRAME);            %แปลงภาพสี FRAME เป็นภาพระดับเทา
switch (ZONE)           %เงื่อนไขแบบ switch-case
    case 1              % case ที่ 1
      DIFF = imabsdiff(BACKGROUND1_GRAY,FRAME_GRAY);  %ลบภาพระหว่างภาพ
                                                   %BACKGROUND1_GRAY กับภาพ FRAME_GRAY
    case 2     % case ที่ 2
      DIFF = imabsdiff(BACKGROUND2_GRAY,FRAME_GRAY);   %ลบภาพระหว่างภาพ
                                                   %BACKGROUND2_GRAY กับภาพ FRAME_GRAY
    case 3    % case ที่ 3
      DIFF = imabsdiff(BACKGROUND3_GRAY,FRAME_GRAY);    %ลบภาพระหว่างภาพ
                                                   %BACKGROUND3_GRAY กับภาพ FRAME_GRAY

    case 4    % case ที่ 4
      DIFF = imabsdiff(BACKGROUND4_GRAY,FRAME_GRAY);   %ลบภาพระหว่างภาพ
                                                   %BACKGROUND4_GRAY กับภาพ FRAME_GRAY

    case 5    % case ที่ 5
      DIFF = imabsdiff(BACKGROUND5_GRAY,FRAME_GRAY);   %ลบภาพระหว่างภาพ
                                                   %BACKGROUND5_GRAY กับภาพ FRAME_GRAY

  end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  thresh = graythresh(DIFF);            %หาค่าเทรสโชว์แบบ auto
  BW    = im2bw(DIFF,thresh);           %แปลงภาพระดับเทาเป็นภาพขาวดำ
  med = medfilt2(BW,[29 29]);           %นำภาพขาวดำผ่านตัวกรองแบบมีเดียน
  se = strel('square',90);              %กำหนดรูปแบบการทำ morphology แบบสี่เหลี่ยมจัสตุรัส
  closing = imclose(med,se);            %mophology โดยการ closing
```

```
for p=1:240                        %วนรอบแบบ for-loop จำนวน 240 รอบ
  for q=1:320                      %วนรอบแบบ for-loop จำนวน 320 รอบ
    if(closing(p,q)==1)            %เปรียบเทียบจุดภาพที่ผ่านการทำ morphology ว่าเท่ากับ 1
                                   %หรือไม่
      move = move+1;               %กำหนดให้ตัวแปร move มีการบวก 1
    end          %จบการทำงาน if
  end            %จบการทำงาน for
end              %จบการทำงาน for-loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
L = bwlabel(closing);   %แบ่งส่วนพิกเซลที่เป็นพิกเซลสีขาวในภาพ
S = regionprops(L,'BoundingBox');      %หาพิกัดจุดเพื่อทำการตีกรอบล้อมรอบวัตถุโดยการใช้
                                       %ทูลบ๊อก regionprops
boxsize = cat(2,S.BoundingBox);        %กำหนดให้มีการเรียงค่าที่ได้เพียงแถวเดียวเท่านั้น
[row col] = size(boxsize);             %กำหนดตัวแปร row เท่ากับจำนวนแถว และ
                                       %col เท่ากับจำนวน คอลัมม์
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if(move>8000)             %กำหนดเงื่อนไขให้ move มีค่ามากกว่า 8000
  if(col==4)              %กำหนดเงื่อนไขให้ col มีค่าเท่ากับ 4
    xb1 = round(boxsize(1));   %ปัดค่าของ boxsize(1) ให้เป็น จำนวนเต็มในชื่อตัวแปร xb1
    xb2 = round(boxsize(1)+boxsize(3));    %ปัดค่าของ boxsize(1)+ boxsize(2)
                                           %ให้เป็น จำนวนเต็มในชื่อตัวแปร xb2
    yb1 = round(boxsize(2));         %ปัดค่าของ boxsize(2) ให้เป็น จำนวนเต็มในชื่อตัวแปร yb1
    yb2 = round(boxsize(2)+boxsize(4));    %ปัดค่าของ boxsize(2)+ boxsize(4)
                                           %ให้เป็น จำนวนเต็มในชื่อตัวแปร yb2
    centroids = round((xb1+xb2)/2);        %หาค่าของจุด centroids
    if((centroids(1)>=50)&&(centroids(1)<=280))   %กำหนดเงื่อนไขขอบเขตการตรวจจับ
      ZONE = ZONE;                         %กำหนดให้ตัวแปร ZONE มีค่าเท่าเดิม
      pulse = pulse;                       %กำหนดให้ตัวแปร pulse มีค่าเท่าเดิม
```

```
elseif(centroids(1)<50)                %กำหนดเงื่อนไขขอบเขตการตรวจจับ
  ZONE = ZONE-1;                       %กำหนดให้ตัวแปร ZONE ลบ 1
  pulse = pulse-0.25;                  %กำหนดให้ตัวแปร pulse ลบ 1
  send = round(pulse*250);             %กำหนดค่าตำแหน่งวัตถุที่จะทำการส่ง
  for q=1:50                           %วนรอบแบบ for-loop จำนวน 50 รอบ
    fprintf(s,'*');                    %ส่ง * ออกไปทางพอร์ตอนุกรม
    fprintf(s,'%d\n',[send]);          %ส่งข้อมูล send ออกไปทางพอร์ตอนุกรม
    fprintf(s,'%d\n',[stop]);          %ส่งข้อมูล stop เท่ากับ 9 ออกไปทางพอร์ตอนุกรม
 end                                   %จบการทำงาน for-loop
elseif(centroids(1)>280)               %กำหนดเงื่อนไขขอบเขตการตรวจจับ
  ZONE = ZONE+1;                       %กำหนดให้ตัวแปร ZONE บวก 1
  pulse = pulse+0.25;                  %กำหนดให้ตัวแปร pulse บวก 1
  send = round(pulse*250)              %กำหนดค่าตำแหน่งวัตถุที่จะทำการส่ง
  for q=1:50                           %วนรอบแบบ for-loop จำนวน 50 รอบ
    fprintf(s,'*');                    %ส่ง * ออกไปทางพอร์ตอนุกรม
    fprintf(s,'%d\n',[send]);          %ส่งข้อมูล send ออกไปทางพอร์ตอนุกรม
    fprintf(s,'%d\n',[stop]);          %ส่งข้อมูล stop เท่ากับ 9 ออกไปทางพอร์ตอนุกรม
  end                                  %จบการทำงาน for-loop
end                                    %จบการทำงาน if
if(ZONE>5)                             %กำหนดเงื่อนไขของ ZONE มีค่ามากกว่า 5 หรือไม่
  ZONE = 5;                            %กำหนดค่าตัวแปร ZONE เท่ากับ 5
end                                    %จบการทำงาน if
if(ZONE<1)                             %กำหนดเงื่อนไขของ ZONE มีค่าน้อยกว่า 1 หรือไม่
  ZONE = 1;                            %กำหนดค่าตัวแปร ZONE เท่ากับ 1
End                                    %จบการทำงาน if
if(pulse>2)                            %กำหนดเงื่อนไขของ pulse มีค่ามากกว่า 2 หรือไม่
  pulse = 2;                           %กำหนดค่าตัวแปร pulse เท่ากับ 2
end                                    %จบการทำงาน if
if(pulse<1)                            %กำหนดเงื่อนไขของ pulse มีค่าน้อยกว่า 1 หรือไม่
```

```
    pulse = 1;                          %กำหนดค่าตัวแปร pulse เท่ากับ 1
  end                                   %จบการทำงาน if
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %create boundingbox
  figure,imshow(FRAME)    %แสดงภาพชื่อ FRAME ที่บันทึกไว้
  for x=xb1               %วนรอบแบบ for-loop ตั้งแต่ x ถึง xb1
    for y=yb1:yb2         %วนรอบแบบ for-loop ตั้งแต่ yb1 ถึง yb2
      hold on             %เก็บภาพที่แสดงไว้
      plot(x,y,'r.')      %พล็อตค่า
      hold off            %คืนค่าปกติไม่เก็บภาพไว้อีก
    end                   %จบการทำงาน for-loop
  end                     %จบการทำงาน for-loop
  for x=xb1:xb2           %วนรอบแบบ for-loop ตั้งแต่ xb1 ถึง xb2
    for y=yb2             %วนรอบแบบ for-loop ตั้งแต่ y ถึง yb2
      hold on             %เก็บภาพที่แสดงไว้
      plot(x,y,'r.')      %พล็อตค่า
      hold off            %คืนค่าปกติไม่เก็บภาพไว้อีก
    end                   %จบการทำงาน for-loop
  end                     %จบการทำงาน for-loop
  for x=xb1:xb2           %วนรอบแบบ for-loop ตั้งแต่ xb1 ถึง xb2
    for y=yb1             %วนรอบแบบ for-loop ตั้งแต่ y ถึง yb1
      hold on             %เก็บภาพที่แสดงไว้
      plot(x,y,'r.')      %พล็อตค่า
      hold off            %คืนค่าปกติไม่เก็บภาพไว้อีก
    end                   %จบการทำงาน for-loop
  end                     %จบการทำงาน for-loop
  for x=xb2               %วนรอบแบบ for-loop ตั้งแต่ x ถึง xb2
    for y=yb1:yb2         %วนรอบแบบ for-loop ตั้งแต่ yb1 ถึง yb2
      hold on             %เก็บภาพที่แสดงไว้
```

```
        plot(x,y,'r.') %พล๊อตค่า
        hold off    %คืนค่าปกติไม่เก็บภาพไว้อีก
      end           %จบการทำงาน for-loop
    end             %จบการทำงาน for-loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  elseif(col==8)  %กำหนดเงื่อนไขให้ col มีค่าเท่ากับ 4
    xb1 = round(boxsize(1));      %ปัดค่าของ boxsize(1) ให้เป็น จำนวนเต็มในชื่อตัวแปร xb1
    xb2 = round(boxsize(1)+boxsize(3));  %ปัดค่าของ boxsize(1)+ boxsize(2)
                                         %ให้เป็น จำนวนเต็มในชื่อตัวแปร xb2
    yb1 = round(boxsize(2));      %ปัดค่าของ boxsize(2) ให้เป็น จำนวนเต็มในชื่อตัวแปร yb1
    yb2 = round(boxsize(2)+boxsize(4));  %ปัดค่าของ boxsize(2)+ boxsize(4)
                                         %ให้เป็น จำนวนเต็มในชื่อตัวแปร yb2
     xb3 = round(boxsize(5));     %ปัดค่าของ boxsize(5) ให้เป็น จำนวนเต็มในชื่อตัวแปร yb3
    xb4 = round(boxsize(5)+boxsize(7));  %ปัดค่าของ boxsize(5)+ boxsize(7)
                                         %ให้เป็น จำนวนเต็มในชื่อตัวแปร xb4
    yb3 = round(boxsize(6));      %ปัดค่าของ boxsize(6) ให้เป็น จำนวนเต็มในชื่อตัวแปร yb3
    yb4 = round(boxsize(6)+boxsize(8));  %ปัดค่าของ boxsize(6)+ boxsize(8)
                                         %ให้เป็น จำนวนเต็มในชื่อตัวแปร yb4
    %create boundingbox
    figure,imshow(FRAME)     %แสดงภาพชื่อ FRAME ที่บันทึกไว้
    for x=xb1                %วนรอบแบบ for-loop ตั้งแต่ x ถึง xb1
      for y=yb1:yb2          %วนรอบแบบ for-loop ตั้งแต่ yb1 ถึง xb2
        hold on              %เก็บภาพที่แสดงไว้
        plot(x,y,'g.')       %พล๊อตค่า
        hold off             %คืนค่าปกติไม่เก็บภาพไว้อีก
      end                    %จบการทำงาน for-loop
    end                      %จบการทำงาน for-loop
    for x=xb1:xb2            %วนรอบแบบ for-loop ตั้งแต่ xb1 ถึง xb2
      for y=yb2              %วนรอบแบบ for-loop ตั้งแต่ y ถึง xb2
```

```
    hold on              %เก็บภาพที่แสดงไว้
    plot(x,y,'g.')       %พล๊อตค่า
    hold off             %คืนค่าปกติไม่เก็บภาพไว้อีก
  end                    %จบการทำงาน for-loop
end                      %จบการทำงาน for-loop
for x=xb1:xb2            %วนรอบแบบ for-loop ตั้งแต่ xb1 ถึง xb2
  for y=yb1              %วนรอบแบบ for-loop ตั้งแต่ y ถึง yb1
    hold on              %เก็บภาพที่แสดงไว้
    plot(x,y,'g.')       %พล๊อตค่า
    hold off             %คืนค่าปกติไม่เก็บภาพไว้อีก
  end                    %จบการทำงาน for-loop
end                      %จบการทำงาน for-loop
for x=xb2                %วนรอบแบบ for-loop ตั้งแต่ x ถึง yb2
  for y=yb1:yb2          %วนรอบแบบ for-loop ตั้งแต่ yb1 ถึง yb2
    hold on              %เก็บภาพที่แสดงไว้
    plot(x,y,'g.')       %พล๊อตค่า
    hold off             %คืนค่าปกติไม่เก็บภาพไว้อีก
  end                    %จบการทำงาน for-loop
end                      %จบการทำงาน for-loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for x=xb3                %วนรอบแบบ for-loop ตั้งแต่ x ถึง xb2
  for y=yb3:yb4          %วนรอบแบบ for-loop ตั้งแต่ yb3 ถึง yb4
    hold on              %เก็บภาพที่แสดงไว้
    plot(x,y,'r.')       %พล๊อตค่า
    hold off             %คืนค่าปกติไม่เก็บภาพไว้อีก
  end                    %จบการทำงาน for-loop
end                      %จบการทำงาน for-loop
for x=xb3:xb4            %วนรอบแบบ for-loop ตั้งแต่ xb3 ถึง xb4
  for y=yb4              %วนรอบแบบ for-loop ตั้งแต่ y ถึง yb4
```

```
            hold on              %เก็บภาพที่แสดงไว้
            plot(x,y,'r.')       %พล๊อตค่า
            hold off             %คืนค่าปกติไม่เก็บภาพไว้อีก
          end                    %จบการทำงาน for-loop
        end                      %จบการทำงาน for-loop
        for x=xb3:xb4            %วนรอบแบบ for-loop ตั้งแต่ xb3 ถึง xb4
          for y=yb3              %วนรอบแบบ for-loop ตั้งแต่ y ถึง yb4
            hold on              %เก็บภาพที่แสดงไว้
            plot(x,y,'r.')       %พล๊อตค่า
            hold off             %คืนค่าปกติไม่เก็บภาพไว้อีก
          end                    %จบการทำงาน for-loop
        end                      %จบการทำงาน for-loop
        for x=xb4                %วนรอบแบบ for-loop ตั้งแต่ x ถึง xb4
          for y=yb3:yb4          %วนรอบแบบ for-loop ตั้งแต่ yb3 ถึง yb4
            hold on              %เก็บภาพที่แสดงไว้
            plot(x,y,'r.')       %พล๊อตค่า
            hold off             %คืนค่าปกติไม่เก็บภาพไว้อีก
          end                    %จบการทำงาน for-loop
        end                      %จบการทำงาน for-loop
      end                        %จบการทำงาน if
      clear col;                 %ล้างตัวแปรชื่อ col
    end                          %จบการทำงาน if
  end                            %จบการทำงาน for-loop
```

2. ฟังก์ชันสำคัญต่างๆ ในโปรแกรม MATLAB ที่นำมาใช้

1. serial ใช้สำหรับสร้างการสื่อสารแบบอนุกรม

   รูปแบบการใช้งาน obj = serial('port')

   เมื่อ obj = ชื่อออปเจคของพอร์ตอนุกรมที่ใช้งาน

   'port' = พอร์ตอนุกรมที่ใช้งาน

2. fopen(serial) ใช้สำหรับเปิดการเชื่อมต่อทางพอร์ตอนุกรม

   รูปแบบการใช้งาน fopen(obj)

   เมื่อ obj = ชื่อออปเจคของพอร์ตอนุกรมที่ใช้งาน

3. fprintf ใช้สำหรับส่งข้อมูลออกไปทางพอร์ตอนุกรม

   รูปแบบการใช้งาน fprintf(obj,'format','cmd')

   เมื่อ obj = ชื่อออปเจคของพอร์ตอนุกรมที่ใช้งาน

   'cmd' = The string written to the device.

   'format' = ตัวอักษรกำหนดรูปแบบการแปลงในลักษณะต่างๆ เช่น %d คือ แปลงเป็นฐานสิบ เป็นต้น

4. videoinput ใช้สำหรับนำภาพจากสัญญาณวิดีโอจากกล้อง webcam เข้าโปรแกรม MATLAB

   รูปแบบการใช้งาน vid = videoinput(adaptorname,deviceID,format)

   เมื่อ adaptorname = อุปกรรับสัญญาณวิดีโอ มีหลายตัว เช่น winvideo , dcam เป็นต้น

   deviceID = แสดงจำนวนของกล้องที่ใช้

   format = ขนาดของสัญญาณวิดีโอที่จะนำเข้ามา

5. preview ใช้สำหรับสร้างหน้าต่างแสดงภาพของสัญญาณวิดีโอ

   รูปแบบการใช้งาน preview(vid)

   เมื่อ vid = ออปเจคที่นำสัญญาณวิดีโอเข้ามา

7. start ใช้สำหรับเริ่มการทำงานของออปเจค

   รูปแบบการใช้งาน start(vid)

   เมื่อ vid = ออปเจคที่นำสัญญาณวิดีโอเข้ามา

8. getsnapshot ใช้สำหรับนำเฟรมภาพจากสัญญาณวิดีโอ ณ เวลาปัจจุบันเข้ามาในโปรแกรม

   รูปแบบการใช้งาน frame = getsnapshot(vid)

   เมื่อ vid = ออปเจคที่นำสัญญาณวิดีโอเข้ามา

9. rgb2gray ใช้สำหรับแปลงภาพสี หรือRGB เป็นภาพระดับสีเทา

รูปแบบการใช้งาน I = rgb2gray(RGB)

เมื่อ RGB = ภาพสี หรือ RGB

10. pause ใช้สำหรับใช้สำหรับการหน่วงเวลา

รูปแบบการใช้งาน pause(n)

เมื่อ n = จำนวนของเวลา (วินาที) ที่ต้องการจะหน่วงเวลา

11. imabsdiff ใช้สำหรับลบภาพสองภาพแบบจุดต่อจุดโดยผลลัพธ์จากการลบจะเป็นค่าบวก

รูปแบบการใช้งาน Z = imabsdiff(X,Y)

เมื่อ X = ภาพที่ต้องการนำมาลบ ภาพที่ 1

Y = ภาพที่ต้องการนำมาลบ ภาพที่ 2

12. graythresh ใช้สำหรับหาค่าระดับเทรสโชว์

รูปแบบการใช้งาน level = graythresh(I)

เมื่อ I = ภาพระดับสีเทา

13. im2bw ใช้สำหรับแปลงภาพไปเป็นภาพไบนารีโดยใช้ร่วมกับฟังก์ชัน graythresh

รูปแบบการใช้งาน BW = im2bw(I,level)

เมื่อ I = ภาพระดับสีเทา

level = ค่าเทรสโชว์

14. medfilt2 ใช้สำหรับกำจัด noise ออกไปจากภาพ

รูปแแบบการใช้งาน B = medfilt2(BW,[m n])

เมื่อ A = ภาพที่มี noise

[m n] = ขนาดของหน้าต่างที่นำมาใช้ในการกำจัด noise

15. strel ใช้สำหรับสร้าง structuring element ในการทำมอร์โฟโลยี

รูปแแบบการใช้งาน SE = strel(shape,parameters)

เมื่อ shape = รูปร่างของ structuring element ที่นำมาทำมอร์โฟโลยี

ตัวอย่างเช่น 'disk' , 'rectangle' เป็นต้น

Parameters = ลักษณะต่างๆ ของรูปร่าง เช่น แบบ 'disk' ขนาดรัศมี = 3 เป็นต้น

16. imclose เป็นฟังก์ชันในการทำมอร์โฟโลยี

รูปแแบบการใช้งาน IM2 = imclose(IM,SE)

เมื่อ IM = ภาพไบนารี

SE = structuring element

17. bwlabel ใช้สำหรับกำหนดกลุ่มของพิกเซลสีขาวในภาพ

รูปแบบการใช้งาน L = bwlabel(BW,n)

เมื่อ BW = ภาพไบนารี

n = รหัสโซ่หรือ Chain Codes เป็นได้ 2 ค่า คือ 4 และ 8

18. regionprops ใช้สำหรับหาค่าสำคัญต่างๆ ของพื้นที่ในภาพขาวดำ ตัวอย่างเช่น การหาค่าพื้นที่หรือ 'Area' , การหาค่ากรอบสี่เหลี่ยมเล็กที่สุดที่ล้อมรอบวัตถุหรือ 'BoundingBox' เป็นต้น

รูปแบบการใช้งาน STATS = regionprops(L,properties)

เมื่อ L = label matrix หรือภาพที่ถูกกำหนดกลุ่มแล้ว

Properties = คุณสมบัติต่างที่ต้องการหาจากภาพที่เป็น label matrix

3. โปรแกรมการทำงานของการสร้างพัลส์เพื่อควบคุมเซอร์โวมอเตอร์ (PicBasic Pro)

```
INCLUDE "modedefs.bas"          'การผนวกไฟล์ modedefs.bas เข้ามาในโปรแกรมหลัก
define OSC 10                   'กำหนดค่าความถี่สัญญาณนาฬิกาหลักเท่ากับ 10 MHz
TRISB.1 = %00000000             'กำหนดให้ขา 1 ของพอร์ต B เป็นเอาต์พุต
TRISC.7 = %11111111             'กำหนดให้ขา 7 ของพอร์ต C เป็นอินพุต
i var word                      'กำหนดตัวแปรระดับเวิร์ดชื่อ i
stop_bit var word               'กำหนดตัวแปรระดับเวิร์ดชื่อ stop_bit
pulse var word                  'กำหนดตัวแปรระดับเวิร์ดชื่อ pulse
loop: serin2 PORTC.7,32,[WAIT("*"),DEC3 pulse,DEC stop_bit] 'รอรับ "*", "ข้อมูลตัวฐานสิบ"
    if stop_bit = 9 then                                    ' "เลขฐานสิบ" และเลข 9
      for i=1 to 200                                        'ไปเก็บในตัวแปร stop_bit
        PULSOUT PORTB.1,pulse   'สร้างพัลส์ออกไปที่พอร์ต B ขา 1 ขนาดเท่ากับ pulse
        PAUSE 20                'หน่วงเวลา 20 millisec
      next
    endif                       'จบเงื่อนไข if
    goto loop                   'กลับไปที่ลาเบล loop
```

ภาคผนวก ข.

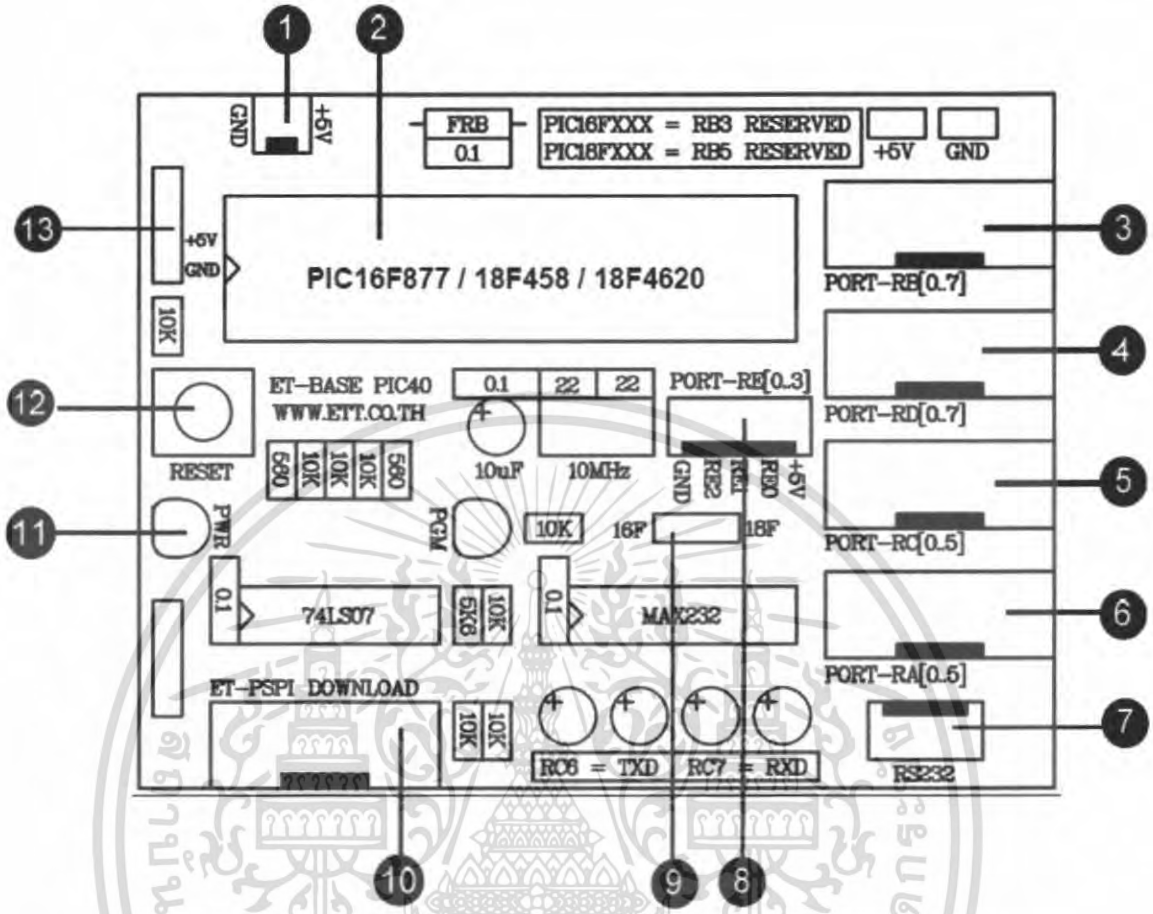
รายละเอียดวงจรของบอร์ดทดลอง ET-BASE PIC 40

ET-BASE PIC40 เป็นบอร์ดไมโครคอนโทรลเลอร์ในตระกูล PIC ขนาด 40 PIN ของบริษัท Microchip ซึ่งในเวอร์ชันนี้ได้นำเอา PIC MCU มาจัดวงจรใช้งานให้มีขนาดกะทัดรัดโดยเน้นการใช้งานทรัพยากรของ PIC MCU เป็นหลัก นอกจากนี้ยังออกแบบให้สนับสนุนการนำไปใช้งานร่วมกับบอร์ดทดลอง "ET-BASIC IO" อีกด้วย

คุณสมบัติของบอร์ด

- รองรับการใช้งานไมโครคอนโทรเลอร์ขนาด 40 PIN คือ PIC16F877 ,PIC18F458 และ PIC18F4620
- สัญญาณนาฬิกาคริสตอลออสซิลเลเตอร์ขนาด 10 MHz
- I/O Port ขนาด 10 PIN (จัดเรียงตามมาตรฐานของ อีทีที) จำนวน 4 พอร์ต
- I/O Port ขนาด 5 PIN จำนวน 1 พอร์ต
- ชุดวงจรไดร์ฟเวอร์ RS232 จำนวน 1 พอร์ต
- ชุดวงจรดาวน์โหลดแบบแรงดันต่ำ (Low Voltage Programming)
- ขั้วต่อแรงดันไฟ VCC และ GND

โครงสร้างบอร์ด ET-BASE PIC40



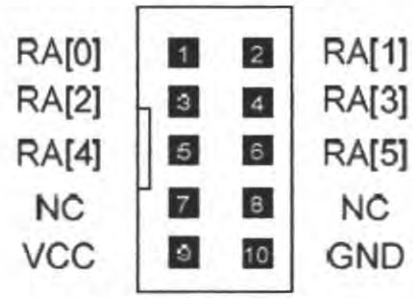
อธิบายรายละเอียดตามหมายเลขต่างๆ ดังนี้

• **หมายเลข 1** ขั้วต่อแหล่งจ่ายไฟ

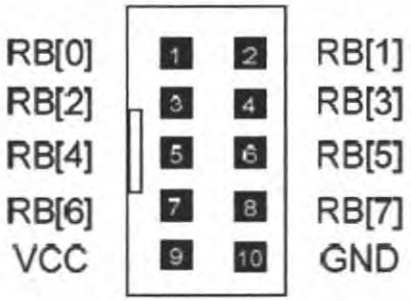
• **หมายเลข 2** ไมโครคอนโทรลเลอร์ PIC 40 PIN รองรับ PIC16F877, PIC18F458 และ PIC18F4620

• **หมายเลข 3 ,4 ,5 และ 6** เป็นพอร์ตของขาสัญญาณ I/O ของ PIC MCU คือ PORT-RA , PORT-RB, PORTRC และ PORT-RD ตามลำดับ โดยจะมีการจัดเรียงขาสัญญาณดังรูปต่อไปนี้

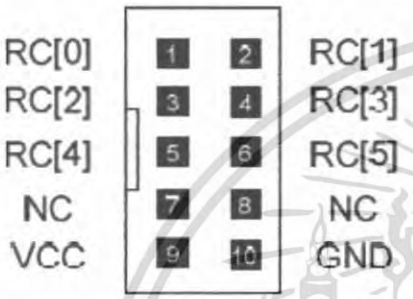
PORT-RA[0..5]



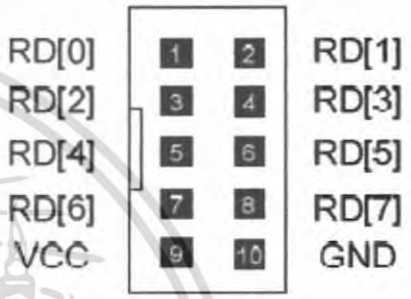
PORT-RB[0..7]



PORT-RC[0..5]



PORT-RD[0..7]



หมายเหตุ ขาสัญญาณ RB3 (กรณี PIC16F877) หรือ RB5 (กรณี PIC18F458 และ PIC18F4620) จะถูกสงวนไว้สำหรับฟังก์ชันการโปรแกรม ไม่สามารถใช้งานได้ ส่วน RB6 และ RB7 ให้ถอดสายสัญญาณดาวน์โหลดออกก่อนจึงจะ ใช้งานได้

• **หมายเลข 7** พอร์ต RS232 จัดเรียงสัญญาณดังนี้

• **หมายเลข 8** ขาสัญญาณ PORTE

• **หมายเลข 9** จัมพ์เปอร์เลือกการใช้งาน PIC MCU โดยสามารถเลือกได้ 2 แบบดังนี้

- 16F คือ ไมโครคอนโทรลเลอร์ 40 PIN เบอร์ PIC16F877
- 18F คือ ไมโครคอนโทรลเลอร์ 40 PIN เบอร์ PIC18F458 และ PIC18F4620

• **หมายเลข 10** พอร์ตสัญญาณสำหรับดาวน์โหลดโปรแกรม

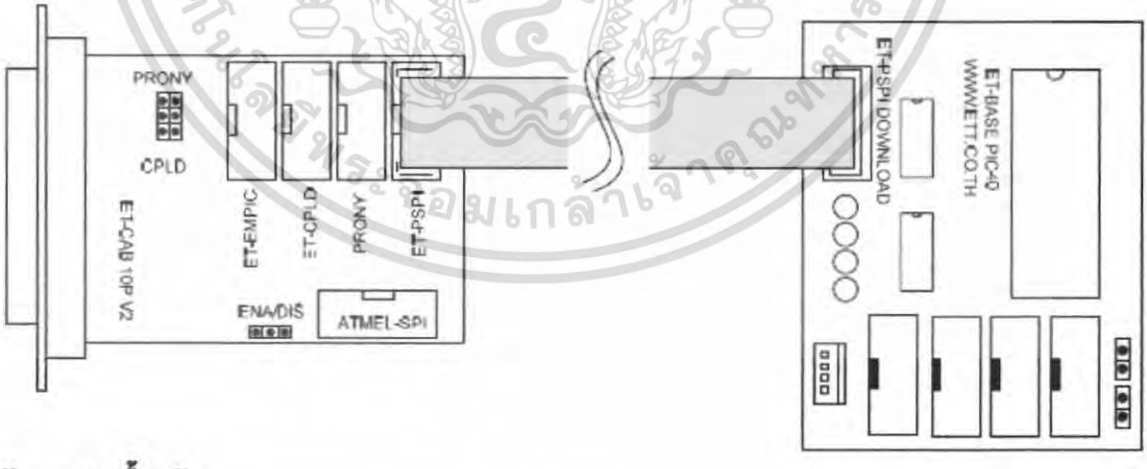
• **หมายเลข 11** LED แสดงสถานะของแหล่งจ่ายไฟภายในบอร์ด

• **หมายเลข 12** สวิตช์ RESET โปรแกรม

• **หมายเลข 13** ขั้วต่อแหล่งจ่ายไฟ สำหรับใช้ร่วมกับบอร์ด ET-BASIC I/O

การโปรแกรมซอร์สโค้ด (Hex File)

จะใช้ซอร์ฟแวร์ WinPic800 โดยจะต้องทำการเชื่อมต่อสายสัญญาณดาวน์โหลด ระหว่างบอร์ด กับคอมพิวเตอร์ โดยผ่าน ET-CAB 10P ดังนี้



ข้อแนะนำเบื้องต้น

- ตรวจสอบการเชื่อมต่อของสายสัญญาณต่างๆ
- ตรวจสอบการจ่ายพลังงานให้กับบอร์ด
- ตรวจสอบการเลือกจัมพ์เปอร์ 16F/18F ว่าถูกต้องตรงตามเบอร์ที่ใช้หรือไม่

- บอร์ดรับแรงดันไฟได้ 5 VDC ระวังห้ามจ่ายไฟเกิน
- ขาสัญญาณ RB6,RB7 หากไม่สามารถใช้งานได้ให้ถอดสายสัญญาณคาวน์โหลดออก จึงจะสามารถ

ใช้งานได้

ภาคผนวก ค.

คู่มือการใช้งาน ET-PGMPIC USB

ET-PGMPIC USB คือ เครื่องโปรแกรมไมโครคอนโทรลเลอร์ตระกูล PIC ซึ่งเป็นไมโครคอนโทรลเลอร์ ของ บริษัท ไมโครชิฟ (microchip) มีคุณสมบัติเทียบเท่ากับเครื่องโปรแกรม PicKit 2 ของ ไมโครชิฟ โดยสามารถโปรแกรมไมโครคอนโทรลเลอร์ PIC ที่มีหน่วยความจำแบบ Flash Memory ได้หลากหลายเบอร์ด้วยกัน (สามารถดูรายการเบอร์ไมโครคอนโทรเลอร์ PIC ที่ ET-PGMPIC USB ได้ในไฟล์ README ของซอร์ฟแวร์โปรแกรม PicKit 2)

จุดเด่นของเครื่องโปรแกรม ET-PGMPIC USB ก็คือ การเชื่อมต่อที่เป็นแบบ USB ทำให้ใช้งานได้สะดวกและ มีความเร็วในการโปรแกรมสูง ใช้เวลาในการโปรแกรมสั้นลง นอกจากนี้ยังสามารถอัฟเกรดเฟิร์มแวร์เวอร์ชันใหม่ๆ จากไมโครชิฟ (www.microchip.com) ได้อีกด้วย นอกจากคุณสมบัติในขั้นต้นแล้ว ทาง อีทีที ยังได้ออกแบบชุดโมดูลเสริม สำหรับทำการโปรแกรมแบบ Emulator ซึ่งสามารถโปรแกรมลง Target บอร์ดได้โดยตรงสะดวกต่อการพัฒนาโปรแกรมเป็นอย่างยิ่ง เพราะไม่ต้องคอยถอดไอซีเขา ลดความเสี่ยงต่อการหัก หรือ งอ ของขาไอซีได้

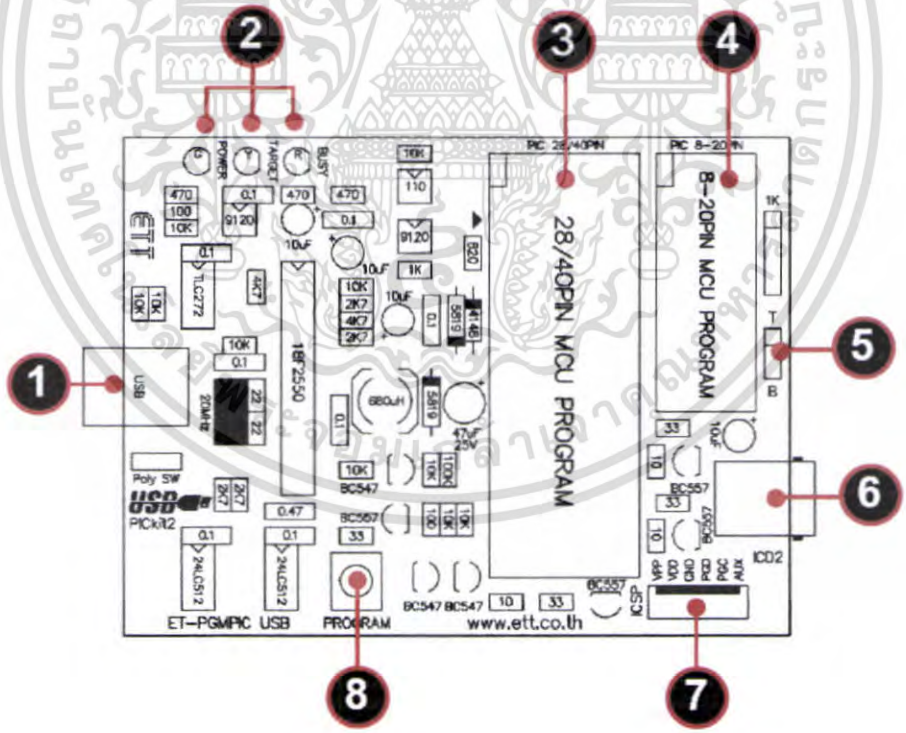
คุณสมบัติของ ET-PGMPIC USB

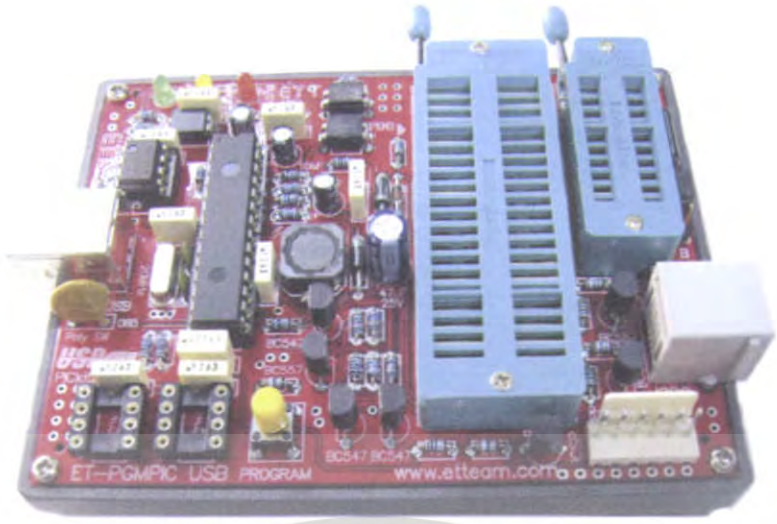
- รองรับการใช้งานกับไมโครคอนโทรเลอร์ตระกูล PIC
- เชื่อมต่อกับคอมพิวเตอร์ผ่าน USB Port
- ใช้ไฟเลี้ยงจาก USB Port (เฉพาะบอร์ด ET-PGMPIC USB เท่านั้น)
- สามารถโปรแกรมผ่าน Text Tool 40 PIN หรือ 20PIN ได้
- มีพอร์ต ICSP สำหรับนำไปต่อโปรแกรมแบบ In-Circuit Serial Programming
- มีไฟแสดงสภาวะต่างๆ
- สามารถโปรแกรมโดยการกดสวิตช์ PROGRAM บนเครื่องโปรแกรม
- สามารถโปรแกรมผ่านโมดูล Emulator ต่างๆ ได้

ความหมายของคำศัพท์ต่างๆ ที่ใช้ในคู่มือ

คำศัพท์	ความหมาย
Target Board	บอร์ดไมโครคอนโทรลเลอร์ที่เชื่อมต่อ กับ ET-PGMPIC USB ผ่านขั้วต่อ ICD2 หรือ ICSP
Emulator Module	โมดูลที่ใช้ใส่แทนที่ ไมโครคอนโทรลเลอร์บนบอร์ด Target เพื่อการโปรแกรม
PIC Micro	ตัวไอซี ไมโครคอนโทรลเลอร์ตระกูล PIC
ICD2	เครื่องโปแกรม และ ดีบัก ของ บริษัท ไมโครชิฟ
ICSP	การโปรแกรมโดยนำสัญญาณโปรแกรม คือ VPP , VDD ,GND, PGD และ PGC ไปต่อตรงกับขาสัญญาณของไมโครคอนโทรลเลอร์เพื่อทำการโปรแกรม
TEXT TOOL	ช่องสำหรับใส่ไอซีเพื่อทำการโปรแกรม

โครงสร้างบอร์ด ET-PGMPIC USB





อธิบายตามหมายเลขต่างๆ ดังนี้

1. ช่องต่อสัญญาณ USB (USB Port Connection) เป็นพอร์ตสำหรับเชื่อมต่อสัญญาณจากบอร์ด ***ET-PGMPIC USB*** เข้ากับคอมพิวเตอร์

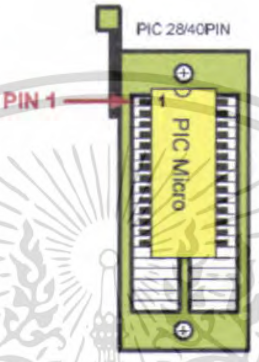
2. ไฟแสดงสถานะต่างๆ คือ POWER , TARGET และ BUSY

- **BUSY** หลอดไฟสีแดง แสดงสถานะการทำงานของเครื่องโปรแกรม หลอดไฟจะติดเมื่อเครื่องโปรแกรมกำลังทำงานอยู่ เช่นกำลัง อ่าน-เขียน Flash memory ของไมโครคอนโทรลเลอร์PIC

- **TARGET** หลอดไฟสีเหลือง แสดงสถานะของแหล่งจ่ายไฟเลี้ยงของบอร์ดปลายทาง (Target Board)

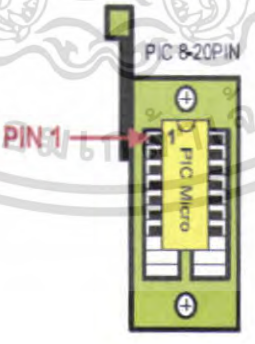
- **POWER** หลอดไฟสีเขียว แสดงสถานะพลังงานไฟเลี้ยงบอร์ด

3. TEXT TOOL ขนาด 40 PIN



- รองรับไอซีไมโครคอนโทรเลอร์ PIC ขนาด 28 PIN ขึ้นไปจนถึง 40 PIN ตัวถังแบบ DIP TYPE
- การใส่ให้ใส่ไอซีชิดด้านบนดังรูป
- จะต้องกดล็อคไอซีให้แน่นทุกครั้งที่ใช้งาน

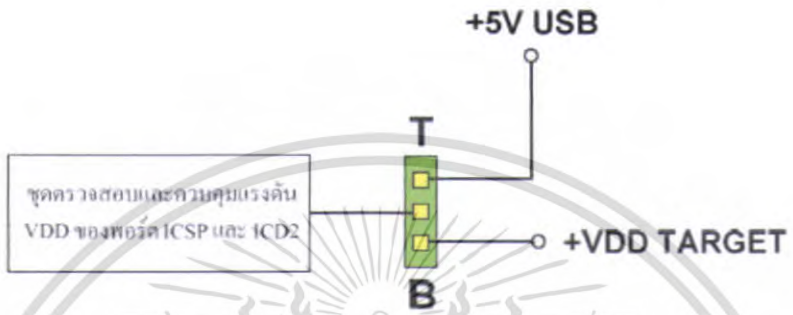
4. TEXT TOOL ขนาด 20 PIN



- รองรับไอซีไมโครคอนโทรเลอร์ PIC ขนาด 8 PIN ขึ้นไปจนถึง 20 PIN ตัวถังแบบ DIP TYPE
- การใส่ให้ใส่ไอซีชิดด้านบนดังรูป
- จะต้องกดล็อคไอซีให้แน่นทุกครั้งที่ใช้งาน

5. จัมพ์เปอร์ T/B สำหรับเลือกการจ่ายไฟเลี้ยงออกไปที่พอร์ต ICD2 และ ICSP

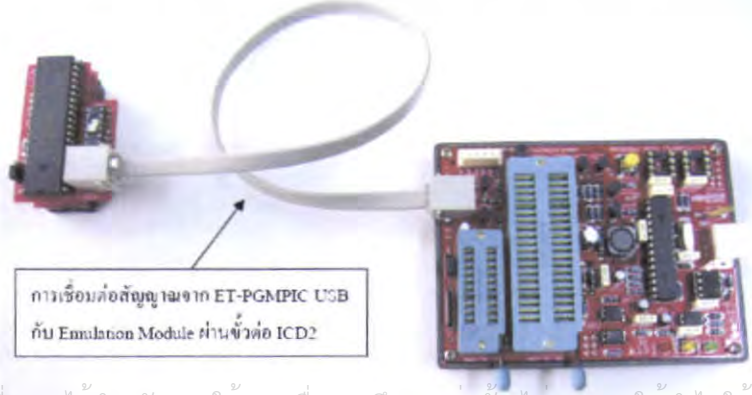
ET-PGMPIC USB นอกจากการโปรแกรมด้วยการใส่ไอซีลงบน TEXT TOOL แล้วยังมีพอร์ต ICSP และICD2 ที่สามารถต่อสัญญาณไปโปรแกรมอุปกรณ์ภายนอกบอร์ด หรือ บนบอร์ดไมโครคอนโทรลเลอร์(TARGET Board) ที่เราต้องการ ด้วยเหตุนี้ ET-PGMPIC USB จึงจำเป็นต้องมีวงจรตรวจสอบ และ ควบคุมแรงดันที่จ่ายไปยังพอร์ต ICSP และ ICD2 เพื่อป้องกันการชนกันระหว่างไฟเลี้ยงบอร์ด ET-PGMPIC USB กับ ไฟจากภายนอก (TARGET Board)



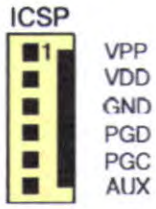
- กรณีการโปรแกรมไอซีบน TEXT TOOL ให้เลือกจัมพ์เปอร์ไปทาง T ชุดตรวจสอบแรงดันจะได้รับแรงดันไฟเลี้ยงที่มาจาก USB

- กรณีการโปรแกรมไอซีโดยการต่อสัญญาณจากพอร์ต ICSP หรือ ICD2 ให้เลือกจัมพ์เปอร์ไปทาง B ในกรณีนี้วงจรตรวจสอบแรงดันจะทำการตรวจสอบแรงดันของ TARGET Board ซึ่งถ้าพบว่ามีแรงดันที่ TARGET Board อยู่แล้ววงจรควบคุมแรงดัน จะควบคุมไม่ให้จ่ายไฟออกไปจากบอร์ด ET-PGMPIC USB แต่ถ้าพบว่าไม่มีแรงดันที่ TARGET Board วงจรควบคุมก็จะจ่ายไฟออกไปเลี้ยง TARGET Board

6. ช่องต่อสัญญาณ ICD2 เป็นพอร์ตของสัญญาณโปรแกรมที่จัดเรียงสัญญาณตามมาตรฐานของ ICD2 (เป็นชุดโปรแกรม และ ดีบักเกอร์ของบริษัท Microchip) สามารถนำไปใช้กับบอร์ดไมโครคอนโทรลเลอร์ต่างๆ ที่มีพอร์ตสัญญาณที่จัดเรียงตามมาตรฐานเดียวกับ ICD2 และสามารถนำไปเชื่อมต่อกับชุดโมดูล Emulator ต่างๆ ของ อีทีที ได้อีกด้วยดังรูปต่อไปนี้

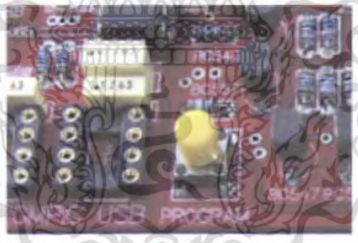


7. พอร์ต ICSP เป็นพอร์ตของสัญญาณโปรแกรมเช่นเดียวกับพอร์ต ICD2 แต่มีการจัดเรียงสัญญาณโดยใช้คอนเนคเตอร์ 6 PIN สำหรับเชื่อมต่อสัญญาณโปรแกรมซึ่งหากบอร์ดไมโครคอนโทรลเลอร์ไม่มีขั้วต่อ ICD2 แนะนำให้เชื่อมต่อสัญญาณโดยตรงจากพอร์ตนี้

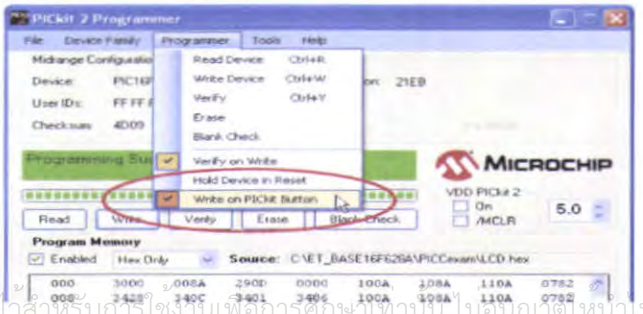


- **VPP (Programming Voltage)** สัญญาณแรงดันสำหรับโปรแกรม
- **VDD (Power Supply Positive Voltage)** แรงดันสำหรับเลี้ยงตัวไอซี
- **GND** ขาสัญญาณกราวด์
- **PGD (Programming Data)** ขาสัญญาณข้อมูลสำหรับการโปรแกรม
- **PGC (Programming Clock)** ขาสัญญาณนาฬิกาสำหรับการโปแกรม
- **AUX** สำรองไว้ไม่ได้ใช้งาน

8. สวิตช์สั่งโปรแกรม (PROGRAM)



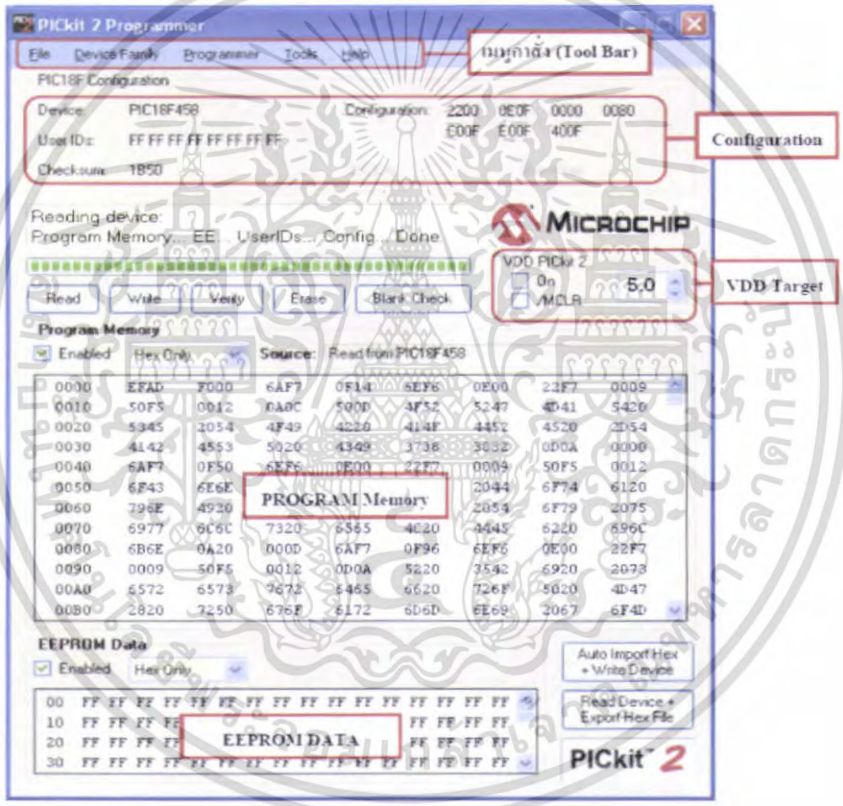
สำหรับสั่งโปรแกรมโดยการกดสวิตช์ ซึ่งมีคุณสมบัติเทียบเท่ากับการกดปุ่ม Write บนซอร์ฟแวร์ PICKit 2 ซึ่งฟังก์ชันการโปรแกรมด้วยสวิตช์นี้จะใช้ได้ก็ต่อเมื่อเราทำการกำหนดคุณสมบัติของโปรแกรม PICkit 2 Programmer ในเมนู Programmer -> Write on PICkit Button โดยการคลิกเครื่องหมายถูกดังรูปต่อไปนี้



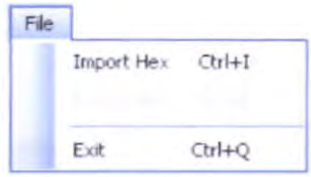
• ซอร์ฟแวร์ที่ใช้กับเครื่องโปรแกรม ET-PGMPIC USB

ในส่วนของซอร์ฟแวร์ บอร์ด ET-PGMPIC USB จะใช้ซอร์ฟแวร์ชื่อ PICkit 2 Programmer ซึ่งเป็นของทางบริษัทไมโครชิฟ โดยก่อนใช้งานโปรแกรมจะต้องทำการติดตั้งโปแกรมให้เรียบร้อยก่อน โดยจะต้องติดตั้งโปแกรม .NET Framework (dotnetfx) ก่อนตามด้วยโปรแกรม PICkit2Setup ดังต่อไปนี้

การใช้งานซอร์ฟแวร์โปรแกรม PICkit 2 Programmer



เมนูคำสั่งเกี่ยวกับการจัดการไฟล์(File)



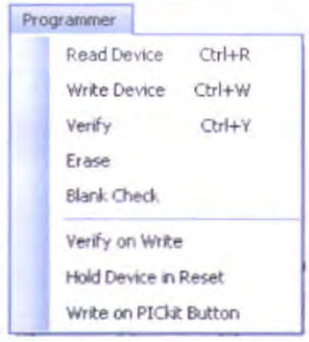
- **Import Hex** – โหลด hex file ที่ต้องการทำการโปรแกรมเข้ามาในโปรแกรม PICkit2
- **Export Hex** – Export hex file ที่อ่านได้จากตัวไมโครคอนโทรลเลอร์เพื่อบันทึกเป็นไฟล์

- **Exit** – ออกจากโปรแกรม

เมนูคำสั่งสำหรับเลือกตระกูลของไมโครคอนโทรลเลอร์ (DEVICE FAMILY)

- **Baseline (12-bit Core)** เลือกใช้งานโปรแกรมกับไมโครคอนโทรลเลอร์แบบ 12-bit Core Flash devices
- **Mid-range (14-bit Core)** เลือกใช้งานโปรแกรมกับไมโครคอนโทรลเลอร์แบบ 14-bit Core Flash devices
- **PIC18F** เลือกใช้งานโปรแกรมกับไมโครคอนโทรลเลอร์ตระกูล PIC18F Flash devices
- **PIC18F_J** เลือกใช้งานโปรแกรมกับไมโครคอนโทรลเลอร์ตระกูล PIC18FXXJXX Flash devices
- **PIC18F_K** เลือกใช้งานโปรแกรมกับไมโครคอนโทรลเลอร์ตระกูล PIC18FXXKXX Flash devices
- **PIC24** เลือกใช้งานโปรแกรมกับไมโครคอนโทรลเลอร์ตระกูล PIC24 Flash device
- **dsPIC33** เลือกใช้งานโปรแกรมกับไมโครคอนโทรลเลอร์ตระกูล dsPIC33 Flash Devices

เมนูคำสั่งสำหรับฟังก์ชันการโปรแกรม (PROGRAMMER)



- **Read Device** อ่านข้อมูลจากหน่วยความจำโปรแกรม (Program memory) , data EEPROM memory, ID locations, และ Configuration bits.

o มีคุณสมบัติเดียวกับปุ่ม Read

- **Write Device** เขียนข้อมูลลงหน่วยความจำ Program memory, data EEPROM, ID locations, และ Configuration bits.

o มีคุณสมบัติเดียวกับปุ่ม Write

- **Verify** ตรวจสอบข้อมูลใน Program memory, Data EEPROM , ID locations และ Configuration bits ของตัวไมโครคอนโทรลเลอร์ กับโค้ด (HEX File) ที่อยู่ในบัฟเฟอร์ของโปรแกรม PICkti2

o มีคุณสมบัติเดียวกับปุ่ม Verify

- **Erase** – ลบข้อมูลในหน่วยความจำของไมโครคอนโทรลเลอร์

o มีคุณสมบัติเดียวกับปุ่ม Erase

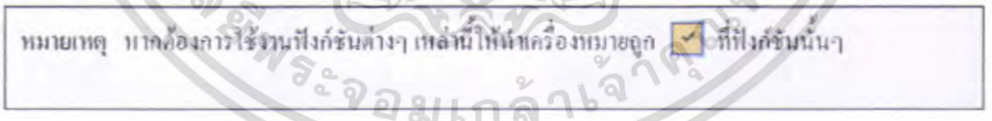
- **Blank Check** ตรวจสอบพื้นที่หน่วยความจำ Program memory, data EEPROM , ID locations และ Configuration bits ว่าอยู่ในสถานะ ว่างเปล่า (Blank) หรือไม่

o มีคุณสมบัติเดียวกับปุ่ม Blank Check

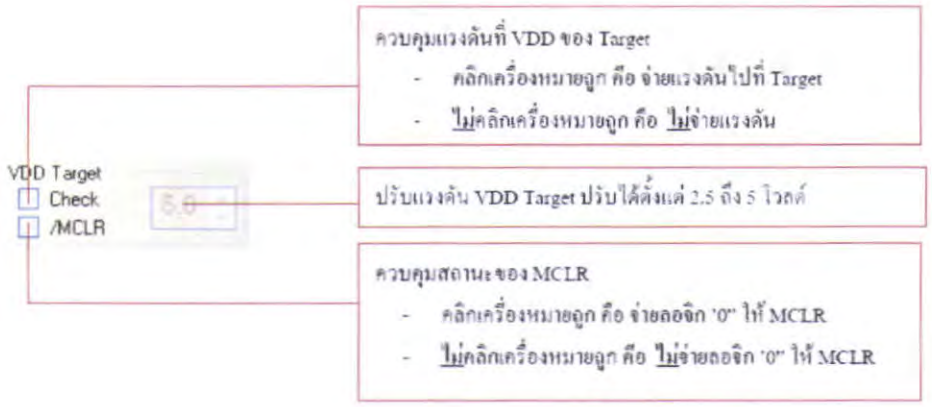
- **Verify on Write** ฟังก์ชันการตรวจสอบข้อมูลในหน่วยความจำ Program memory, data EEPROM, ID locations, และ Configuration bits ในขณะทำการ Write ข้อมูล

- **Hold Device in Reset** ค้างสถานะที่ขาสัญญาณรีเซตไว้เป็นลอจิก "0" (MCLR =0)

- **Write on PICkit Button** ฟังก์ชันการโปรแกรมจากการกดสวิตช์ (PROGRAM) บนบอร์ด ET-PGM USB

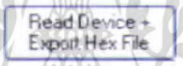


- **VDD Target** เป็นฟังก์ชันในการจ่ายไฟเลี้ยงให้อุปกรณ์ Target และ ควบคุมสัญญาณรีเซต (MCLR)

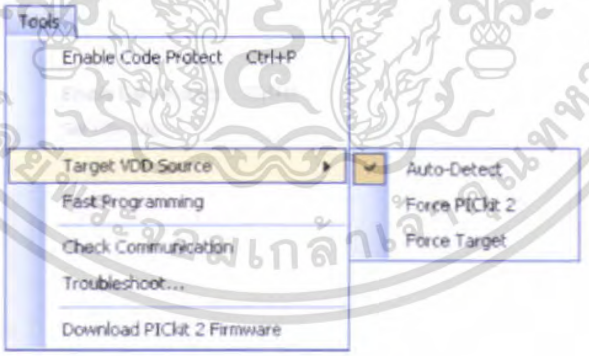


- **Auto Import Hex + Write Device** คือ ปุ่มคำสั่งที่ทำหน้าที่ทั้ง Import Hex File และ ทำการ Write ข้อมูล

- **Read Device + Export Hex File** คือปุ่มคำสั่งที่ทำหน้าที่ทั้งอ่านข้อมูลจากหน่วยความจำของไมโครคอนโทรลเลอร์และ ทำการ Export เป็น Hex File



เมนูคำสั่งเครื่องมือสำหรับการโปรแกรม (Tools)

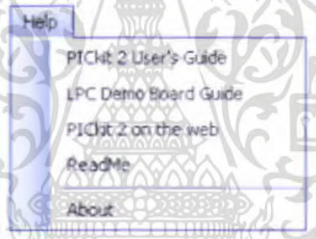


- **Enable Code Protect (Ctrl+P)** ฟังก์ชันปกป้องหน่วยความจำโค้ดโปรแกรม
- **Enable Data Protect (Ctrl+D)** ฟังก์ชันปกป้องหน่วยความจำข้อมูล EEPROM
- **Set OSCCAL** ใช้ค่าจากรีจิสเตอร์ OSCCAL เพื่อการปรับแต่งค่าความถี่ OSC ภายใน PIC

- **Target VDD Source** แนะนำให้เซตไว้ที่ตำแหน่ง Auto-Detect

- **Auto-Detect** ตรวจสอบแรงดันของอุปกรณ์ปลายทางโดยอัตโนมัติ
- **Force PICkit 2** กำหนดให้แรงดัน VDD ที่จ่ายให้กับ Target มาจากบอร์ด PICkit2
- **Force Target** กำหนดให้แรงดัน VDD ที่จ่ายให้กับ Target เป็นแรงดันของ Target เอง
- **Fast Programming** การโปรแกรมแบบรวดเร็ว
- **Check Communication** ตรวจสอบการเชื่อมต่อสื่อสารระหว่าง ET-PGMPIC กับ Computer
- **Troubleshoot...** เป็นฟังก์ชันของการให้ข้อมูลในการช่วยเหลือเมื่อเกิดปัญหาต่างๆ
- **Download PICkit 2 Firmware** คือ ฟังก์ชันที่ใช้สำหรับดาวโหลด Firmware ใหม่ๆ ของ PICkit2 ลงไปในบอร์ด ET-PGMPIC USB ใช้สำหรับการ Update Firmware

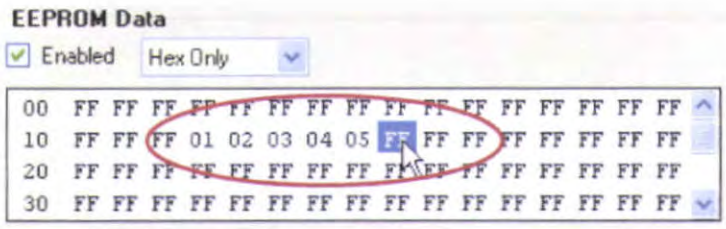
เมนูคำสั่งสำหรับการช่วยเหลือ (Help)



- **PICkit 2 User's Guide** คู่มือการใช้งาน PicKit2 เป็น PDF File
- **LPC Demo Board Guide** คู่มือบอร์ด Low Pin Count Demo Board ของ MICROCHIP
- **PICkit 2 on the web** ข้อมูลต่างๆของ PICkit2 บนเว็ปไซต์ของ MICROCHIP
- **ReadMe** ไฟล์ ReadMe ของโปรแกรม PICkit 2 แสดงรายละเอียด และ เบอร์ต่างๆ ของ PIC MCUที่
- PICkit 2 สนับสนุนการใช้งาน
- **About** ข้อมูลรายละเอียดของตัวซอร์ฟแวร์ PICkit 2

หน่วยความจำ EEPROM Data

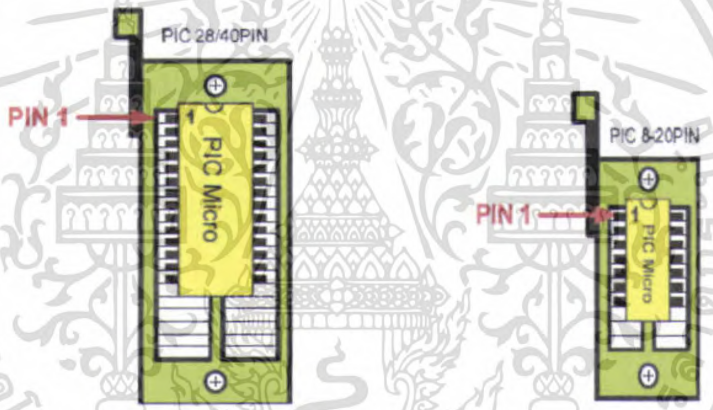
โปรแกรม PICkit 2 สามารถทำการแก้ไขข้อมูลในหน่วยความจำโปรแกรม EEPROM ของ PIC Micro ได้โดยจะมีหน้าต่างในการแก้ไขข้อมูล วิธีการแก้ไขก็เพียงแค่นำเมาส์ ไปคลิกแก้ไขในตำแหน่งข้อมูลที่เราต้องการซึ่งเมื่อการ Write ข้อมูลลงไปหน่วยความจำ EEPROM ของ PIC Micro ก็จะเปลี่ยนแปลงตามข้อมูลดังที่เรากำหนดดังรูปต่อไปนี้



ขั้นตอนการโปรแกรม

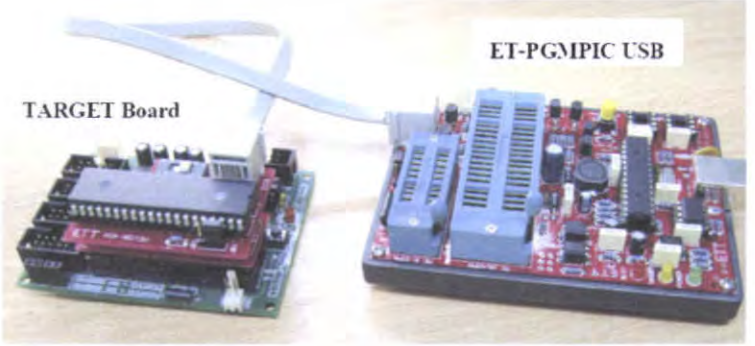
1. เชื่อมต่อสายสัญญาณ USB ระหว่างบอร์ด ET-PGMPIC USB กับ คอมพิวเตอร์
2. ใส่ไอซี PIC MCU ที่ต้องการโปรแกรมลงใน Text Tool หรือ ชุด Emulator ต่างๆ

- กรณีการโปรแกรมบน Text Tools



*หมายเหตุ การโปแกรมบน Text Tools ให้เลือกจัมพ์เปอร์ T B มาที่ตำแหน่ง T

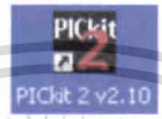
- กรณีการโปรแกรมบน Target Board ด้วยชุด Emulator Module



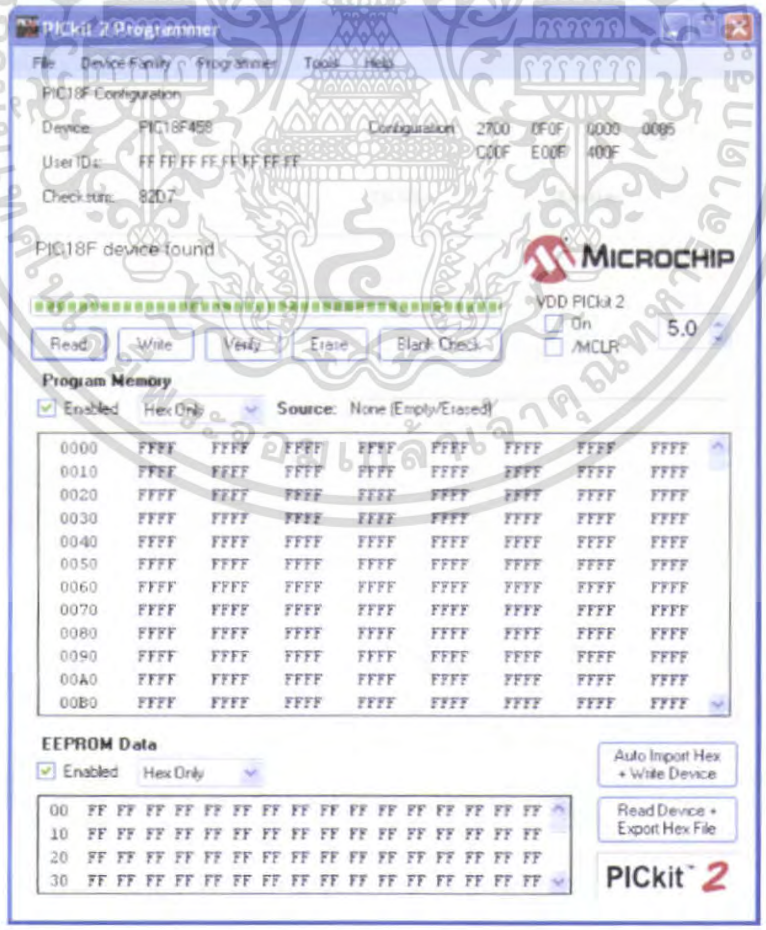
กรณีการโปรแกรมโดยผ่านโมดูล Emulator ควรต่อไฟเลี้ยงให้บอร์ด TARGET ด้วย เพื่อป้องกันปัญหาไฟเลี้ยงจาก USB ไม่เพียงพอ และ จะต้องเลื่อนตำแหน่งของสวิตช์ บนโมดูลมาที่ตำแหน่ง PRG ด้วยเพื่อทำการเชื่อมต่อสัญญาณโปรแกรม

*หมายเหตุ การโปแกรมบน Emulator Module ให้เลือกจัมพ์เปอร์ T/B มาที่ตำแหน่ง B

3. เปิดโปรแกรม PICkit 2 โดยการดับเบิ้ลคลิกที่ไอคอน PICkit2



4. โปรแกรม PICkit 2 จะทำการตรวจสอบ ไอซี บน TEXT Tool หากเป็นเบอร์ที่ PICkit 2 สนับสนุนการใช้งานอยู่ และ การเชื่อมโยงสัญญาณต่างๆ ถูกต้อง ในช่อง Device จะแสดงเบอร์ของ PIC Micro ที่พบ ดังรูป



5. ทำการลบข้อมูลเก่าใน PIC Micro ออกก่อนโดยคลิกที่ปุ่มคำสั่ง Erase ซึ่งจะเห็นว่าข้อมูลในช่อง Program Memory และ EEPROM Data จะมีค่าเป็น FF

6. ทำการ Import Hex File ที่เราต้องการ โดยคลิกที่ เมนูคำสั่ง File -> Import Hex

7. จะเห็นว่า ข้อมูลในช่อง Program Memory และ EEPROM Data จะมีค่าเปลี่ยนเป็นค่าต่างๆ ตามข้อมูลของ Hex File ที่โหลดเข้ามา

8. คลิกปุ่มคำสั่ง Write เพื่อทำการเขียนโปรแกรม Hex File ลงไปในหน่วยความจำของ PIC Micro

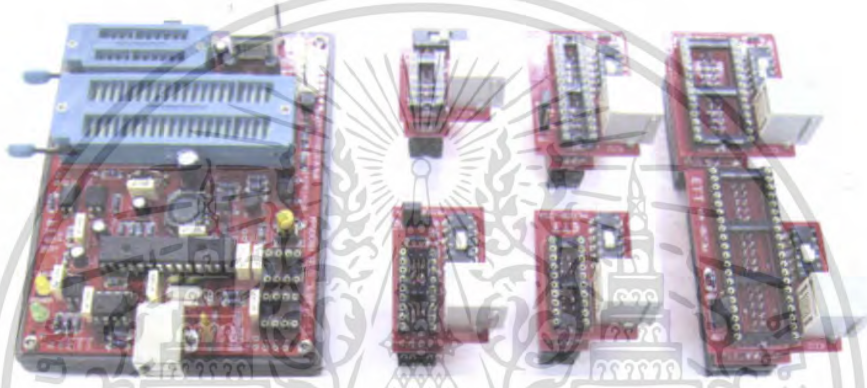
9. หากต้องการตรวจสอบว่าข้อมูลที่เขียนเข้าไปใน PIC Micro มีความถูกต้องหรือไม่ ให้ใช้การ Verify โดย คลิกที่ปุ่ม Verify

หากมีการ Enable Code Protect ไว้กระบวนการ Verify จะล้มเหลว (failed) เพราะโค้ดโปรแกรมถูกป้องกันการอ่านไว้ทำให้ไม่สามารถทำการ Verify ได้

ชุดอุปกรณ์ Emulation Module

คือ ชุดอุปกรณ์เสริมของเครื่องโปรแกรม ET-PGMPIC USB จุดประสงค์เพื่อรองรับการโปรแกรมบนบอร์ดไมโครคอนโทรลเลอร์ (TARGET Board) โดยไม่ต้องถอดไอซีเข้าออก เพิ่มความสะดวกในการพัฒนาโปรแกรม และ ช่วยป้องกันการหักงอของขาไอซี ที่มักเกิดขึ้นจากการถอดไอซีเข้า-ออก เครื่องโปรแกรม เป็นต้น

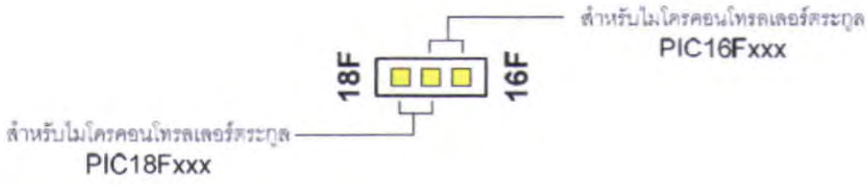
ชุดโมดูล Emulator มีทั้งหมด 6 โมดูลด้วยกัน คือ ขนาด 14-PIN, 18-PIN, 20-PIN, 28PIN (ขาแคบ), 28PIN (ขากว้าง) และ 40-PIN ทั้งนี้ก็เพื่อให้รองรับการใช้ไมโครคอนโทรลเลอร์ PIC ขนาดต่างๆ ของ MICROCHIP ให้ได้มากที่สุด ดังรูปต่อไปนี้



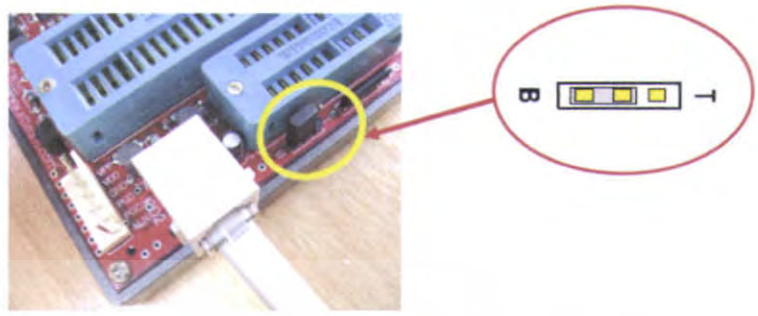
ในแต่ละโมดูลจะมีสวิตช์เลือกโหมด การโปรแกรม (PRG) และ โหมดการรัน (RUN) โดยเมื่อต้องการทำการโปรแกรมก็ให้เลือกสวิตช์มาที่ตำแหน่ง PRG และ เมื่อต้องการรัน ให้เลื่อนสวิตช์ไปที่ตำแหน่ง RUN ดังรูปต่อไปนี้

Programming Mode *Running Mode*

ในบางโมดูลจะมีจัมพ์เปอร์ 18F/16F สำหรับเลือกเบอร์ของไมโครคอนโทรเลอร์ PIC จะต้องทำการเซตจัมพ์เปอร์ให้ตรงกับเบอร์ที่เราใช้งานดังรูปต่อไปนี้

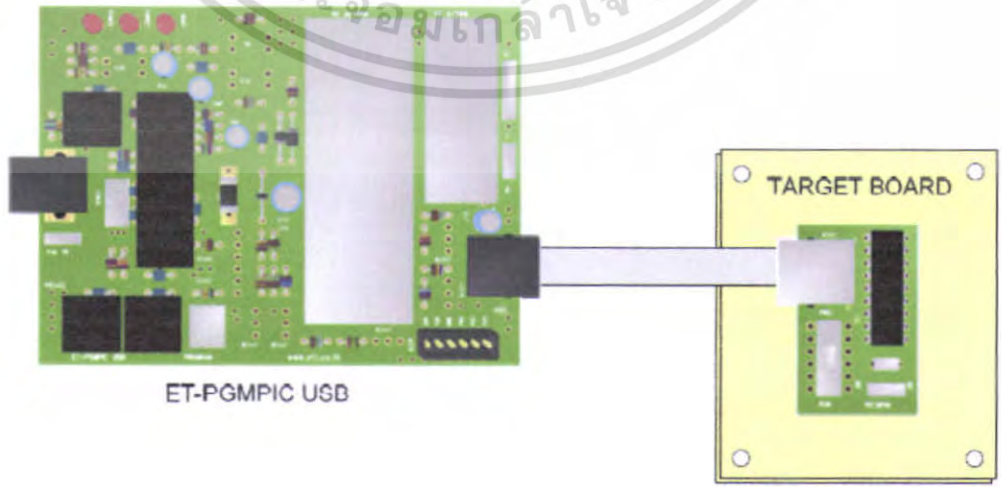


กรณีที่เราต้องการโปรแกรมผ่านโมดูลต่างๆ เหล่านี้จะต้องทำการเซตจัมพ์เปอร์ T/B มาที่ตำแหน่ง B ดังรูปต่อไปนี้

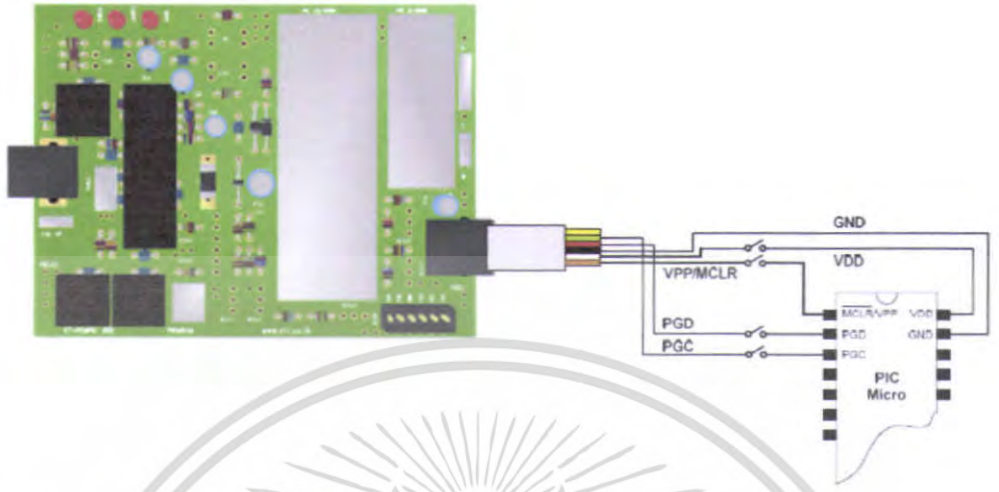


การจัดเรียงขาสัญญาณของพอร์ต ICD2

ลักษณะการโปรแกรมผ่านโมดูล Emulator



ลักษณะการโปรแกรมโดยการเชื่อมต่อสัญญาณโปรแกรมเข้ากับขาสัญญาณของไมโครคอนโทรลเลอร์โดยตรง



ข้อผิดพลาดและแนวทางการแก้ไข

ปัญหา

การผิดพลาดจากการเชื่อมต่อระหว่างคอมพิวเตอร์กับ บอร์ด PICkit2 จะฟ้องข้อความดังรูปต่อไปนี้

แนวทางการแก้ไข

- ตรวจสอบการเชื่อมต่อของสาย USB ระหว่างคอมพิวเตอร์ กับ บอร์ด ET-PGMPIC USB
- คลิก Tools -> Check Communication เพื่อทำการตรวจสอบอีกครั้ง

ปัญหา

ความผิดพลาดจากการตรวจสอบแรงดันที่ Target Board โดยจะมี Error Massage ดังรูปต่อไปนี้

แนวทางการแก้ไข

- กรณีการใช้งานเครื่องโปแกรมโดยผ่าน Text Tool ให้ตรวจสอบจัมพ์เปอร์ T/B ว่าอยู่ในตำแหน่ง T หรือไม่

- กรณีการใช้งานผ่านโมดูล Emulator ให้ตรวจสอบจัมพ์เปอร์ T/B ว่าอยู่ในตำแหน่ง B หรือ ไม่และตรวจสอบไฟเลี้ยงของ Target Board ว่ามีไฟเลี้ยงหรือไม่ ถ้าไม่มีให้ทำการจ่ายไฟเลี้ยงที่บอร์ดปลายทาง (Target Board) ให้เรียบร้อย

ปัญหา

ปัญหาจากการตรวจไม่พบ ไมโครคอนโทรลเลอร์

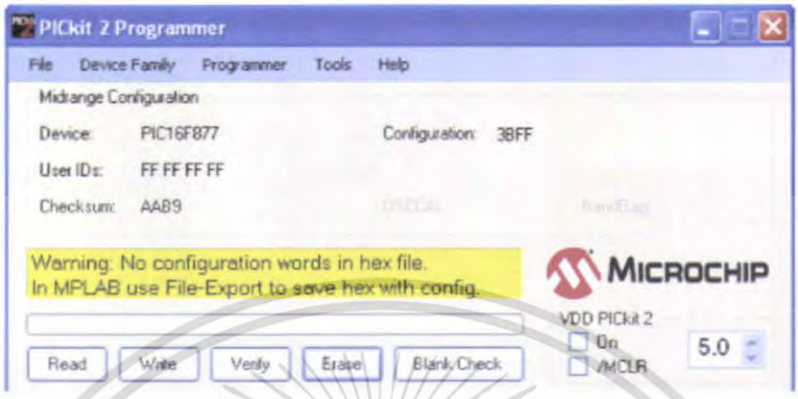
แนวทางการแก้ไข

- ตรวจสอบการใส่ไอซี ใน Text Tool ว่าใส่ถูกต้องหรือไม่ ขา 1 ของไอซีใส่ในตำแหน่งที่ถูกต้องหรือไม่

- กรณีการโปแกรมด้วยโมดูล Emulator ให้เช็คสายสัญญาณที่เชื่อมต่อว่าอยู่ในสภาพดีหรือไม่ และ เช็คแรงดันที่ Target Board ว่ามีการจ่ายแรงดันหรือไม่

ปัญหา

ปัญหาจาก Hex File ที่ Import เข้าไม่มีค่า Configuration รวมอยู่ด้วย ซึ่งปัญหานี้เกิดขึ้นในขั้นต้อนของการออกแบบ และ คอมไพล์โปรแกรม



แนวทางการแก้ไข

- ทำการกำหนดค่า Configuration ให้เรียบร้อยในขั้นตอนของการออกแบบและสร้างโปแกรม

ภาคผนวก ง.

ข้อมูลไอซี PIC 16F877

MICROCHIP

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

Devices Included in this Data Sheet:

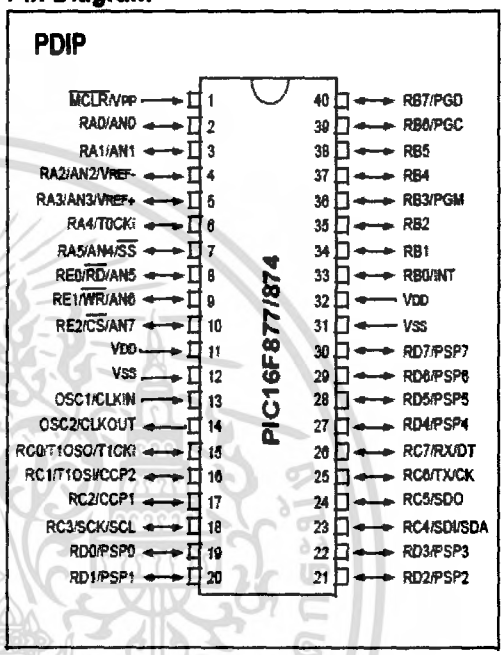
- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and
  Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

Pin Diagram

PDIP

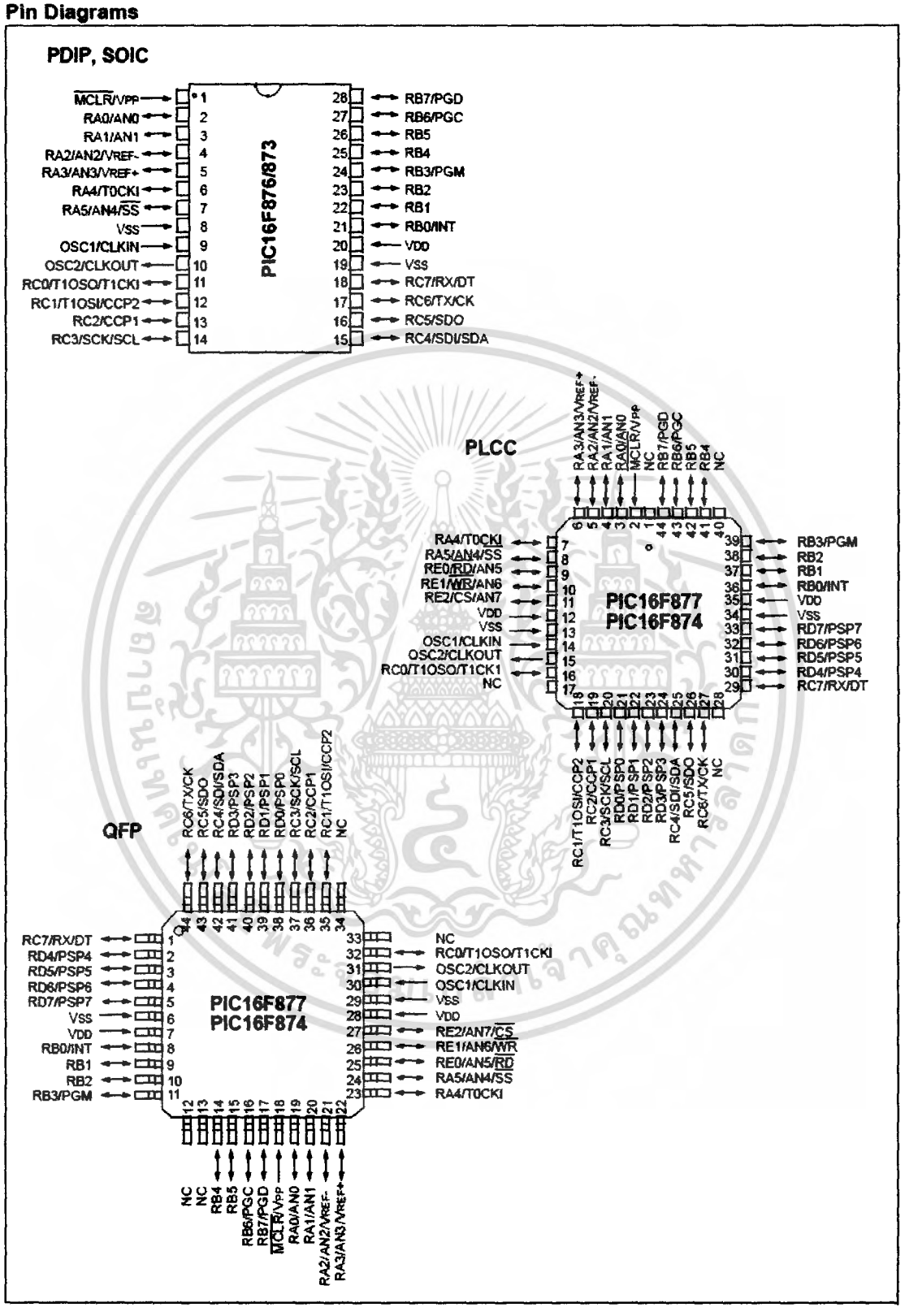


Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I²C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

PIC16F87X

Pin Diagrams



Key Features PICmicro™ Mid-Range Reference Manual (DS33023)	PIC16F873	PIC16F874	PIC16F876	PIC16F877
Operating Frequency	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz
RESETS (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
FLASH Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory	128	128	256	256
Interrupts	13	14	13	14
I/O Ports	Ports A,B,C	Ports A,B,C,D,E	Ports A,B,C	Ports A,B,C,D,E
Timers	3	3	3	3
Capture/Compare/PWM Modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Instruction Set	35 instructions	35 instructions	35 instructions	35 instructions

1.0 DEVICE OVERVIEW

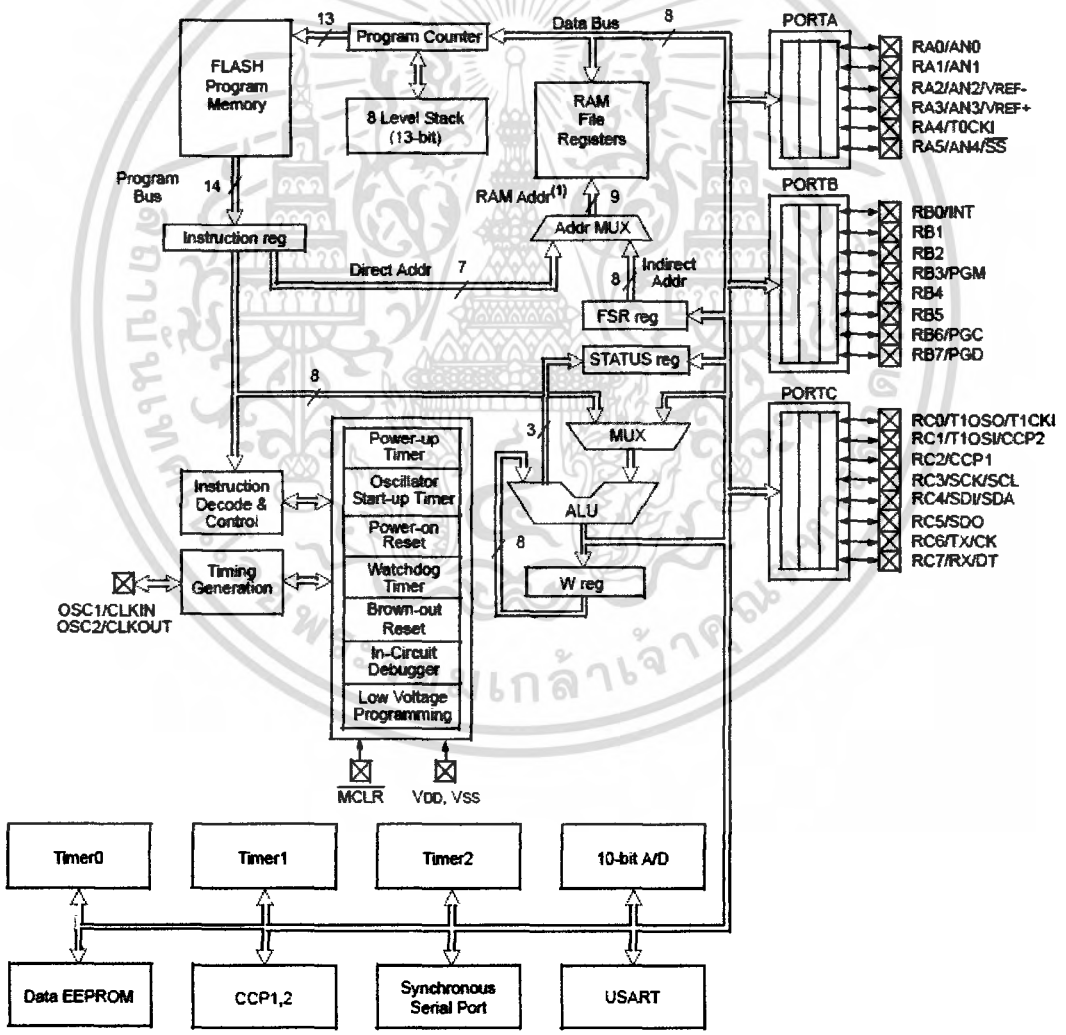
This document contains device specific information. Additional information may be found in the PICmicro™ Mid-Range Reference Manual (DS33023), which may be obtained from your local Microchip Sales Representative or downloaded from the Microchip website. The Reference Manual should be considered a complementary document to this data sheet, and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

There are four devices (PIC16F873, PIC16F874, PIC16F876 and PIC16F877) covered by this data sheet. The PIC16F876/873 devices come in 28-pin packages and the PIC16F877/874 devices come in 40-pin packages. The Parallel Slave Port is not implemented on the 28-pin devices.

The following device block diagrams are sorted by pin number; 28-pin for Figure 1-1 and 40-pin for Figure 1-2. The 28-pin and 40-pin pinouts are listed in Table 1-1 and Table 1-2, respectively.

FIGURE 1-1: PIC16F873 AND PIC16F876 BLOCK DIAGRAM

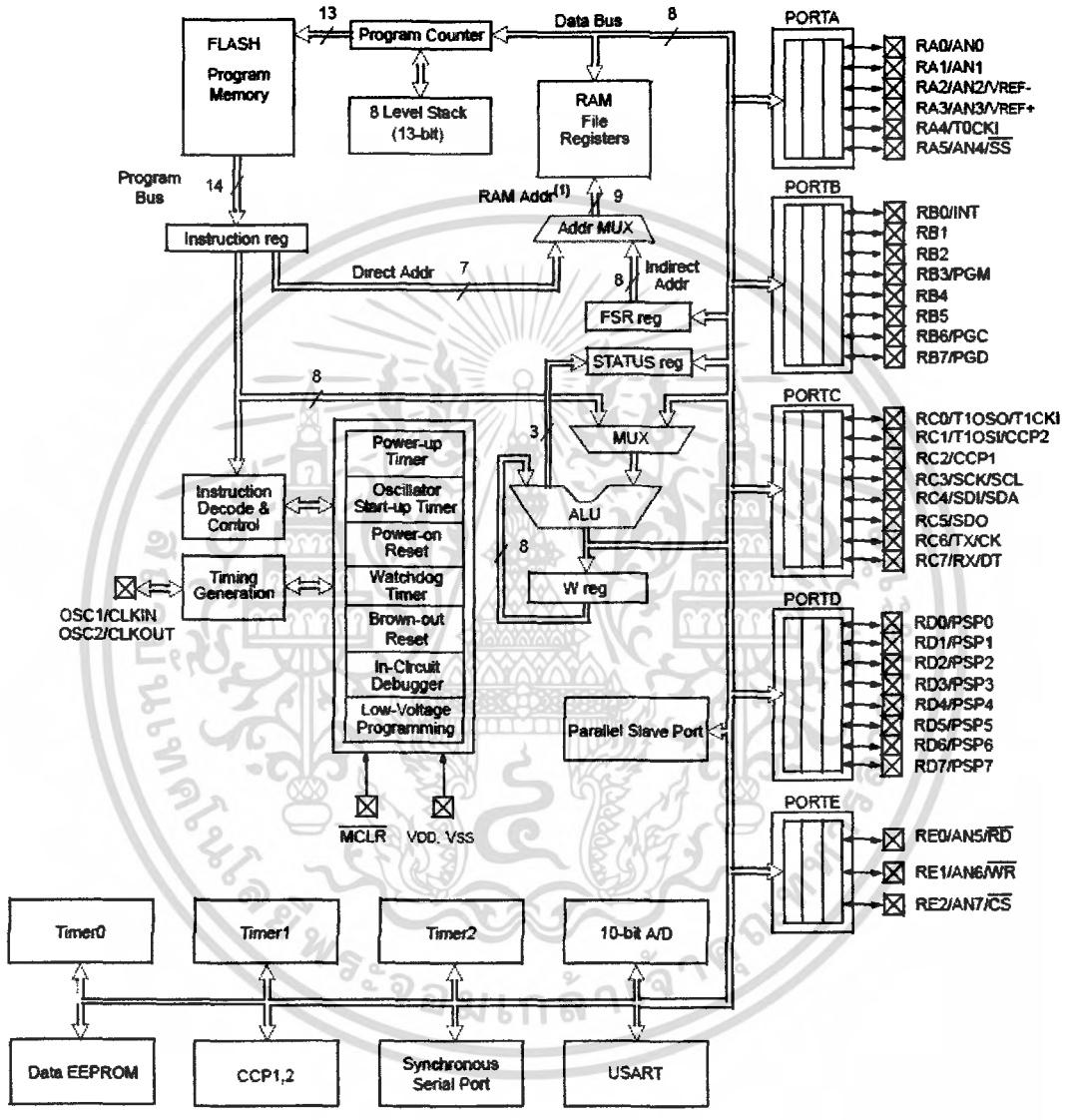
Device	Program FLASH	Data Memory	Data EEPROM
PIC16F873	4K	192 Bytes	128 Bytes
PIC16F876	8K	368 Bytes	256 Bytes



Note 1: Higher order bits are from the STATUS register.

FIGURE 1-2: PIC16F874 AND PIC16F877 BLOCK DIAGRAM

Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874	4K	192 Bytes	128 Bytes
PIC16F877	8K	368 Bytes	256 Bytes



Note 1: Higher order bits are from the STATUS register.

TABLE 1-1: PIC16F873 AND PIC16F876 PINOUT DESCRIPTION

Pin Name	DIP Pin#	SOIC Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	9	9	I	ST/CMOS(3)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	10	10	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, the OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{MCLR}$/VPP	1	1	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
					PORTA is a bi-directional I/O port.
RA0/AN0	2	2	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	3	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	4	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	5	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	6	I/O	ST	RA4 can also be the clock input to the Timer0 module. Output is open drain type.
RA5/$\overline{SS}$/AN4	7	7	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
					PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	21	21	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	22	22	I/O	TTL	
RB2	23	23	I/O	TTL	
RB3/PGM	24	24	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	25	25	I/O	TTL	Interrupt-on-change pin.
RB5	26	26	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	27	27	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	28	28	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.
					PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	11	11	I/O	ST	RC0 can also be the Timer1 oscillator output or Timer1 clock input.
RC1/T1OSI/CCP2	12	12	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output
RC2/CCP1	13	13	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/ PWM1 output.
RC3/SCK/SCL	14	14	I/O	ST	RC3 can also be the synchronous serial clock input/output for both SPI and I^2C modes.
RC4/SDI/SDA	15	15	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	16	16	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	17	17	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	18	18	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
VSS	8, 19	8, 19	P	—	Ground reference for logic and I/O pins.
VDD	20	20	P	—	Positive supply for logic and I/O pins.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

PIC16F87X

TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	I	ST/CMOS(4)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{MCLR}$/VPP	1	2	18	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	4	20	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	5	21	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	6	22	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	7	23	I/O	ST	RA4 can also be the clock input to the Timer0 timer/ counter. Output is open drain type.
RA5/$\overline{SS}$/AN4	7	8	24	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
						PORTB is a bi-directional I/O port. PORTB can be soft-ware programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3/PGM	36	39	11	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	37	41	14	I/O	TTL	Interrupt-on-change pin.
RB5	38	42	15	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	39	43	16	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	40	44	17	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION (CONTINUED)

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
						PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	RC0 can also be the Timer1 oscillator output or a Timer1 clock input.
RC1/T1OSI/CCP2	16	18	35	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	17	19	36	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	18	20	37	I/O	ST	RC3 can also be the synchronous serial clock input/ output for both SPI and I^2C modes.
RC4/SDI/SDA	23	25	42	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	24	26	43	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	25	27	44	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	26	29	1	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
						PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD0/PSP0	19	21	38	I/O	ST/TTL(3)	
RD1/PSP1	20	22	39	I/O	ST/TTL(3)	
RD2/PSP2	21	23	40	I/O	ST/TTL(3)	
RD3/PSP3	22	24	41	I/O	ST/TTL(3)	
RD4/PSP4	27	30	2	I/O	ST/TTL(3)	
RD5/PSP5	28	31	3	I/O	ST/TTL(3)	
RD6/PSP6	29	32	4	I/O	ST/TTL(3)	
RD7/PSP7	30	33	5	I/O	ST/TTL(3)	
						PORTE is a bi-directional I/O port.
RE0/$\overline{RD}$/AN5	8	9	25	I/O	ST/TTL(3)	RE0 can also be read control for the parallel slave port, or analog input5.
RE1/$\overline{WR}$/AN6	9	10	26	I/O	ST/TTL(3)	RE1 can also be write control for the parallel slave port, or analog input6.
RE2/$\overline{CS}$/AN7	10	11	27	I/O	ST/TTL(3)	RE2 can also be select control for the parallel slave port, or analog input7.
VSS	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
VDD	11,32	12,35	7,28	P	—	Positive supply for logic and I/O pins.
NC	—	1,17,28, 40	12,13, 33,34		—	These pins are not internally connected. These pins should be left unconnected.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.