วิทยานิพนธ์

การออกแบบระบบควบคุมแขนหุ่นยนต์ผ่านอินเตอร์เน็ต

ROBOT ARM CONTROL SYSTEM DESIGN VIA INTERNET

นายสกล กงแก้ว

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า)

พ.ศ. 2551

ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า)

ปริญญา

วิศวกรรมไฟฟ้า	วิศวกรรมไฟฟ้า
สาขา	**ภาควิชา**

เรื่อง การออกแบบระบบควบคุมแขนหุ่นยนต์ผ่านอินเตอร์เน็ต

Robot Arm Control System Design via Internet

นามผู้วิจัย นายสกล กงแก้ว

ได้พิจารณาเห็นชอบโดย

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก .....

(ผู้ช่วยศาสตราจารย์พีระยศ แสนโภชน์, D.Sc.)

อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม .....

(อาจารย์จันทนี รุ่งเรืองพิทยากุล, D.Sc.)

อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม .....

(อาจารย์ยศวีร์ วีระกำแหง, Ph.D.)

หัวหน้าภาควิชา .....

(รองศาสตราจารย์มงคล รักษาพัชรวงศ์, Ph.D.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

.....

(รองศาสตราจารย์กัญจนา ธีระกุล, D.Agr.)

คณบดีบัณฑิตวิทยาลัย

วันที่ ..... **เดือน** ..... **พ.ศ.** 2551

วิทยานิพนธ์

เรื่อง

การออกแบบระบบควบคุมแขนหุ่นยนต์ผ่านอินเตอร์เน็ต

Robot Arm Control System Design via Internet

โดย

นายสกล กงแก้ว

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์
เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า)
พ.ศ. 2551

สกล กงแก้ว 2551: การออกแบบระบบควบคุมแขนหุ่นยนต์ผ่านอินเตอร์เน็ต ปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า) สาขาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก: ผู้ช่วยศาสตราจารย์พีรยศ แสนโภชน์, D.Sc. 174 หน้า

งานวิจัยนี้เป็นการนำเสนอการออกแบบและสร้างต้นแบบตัวควบคุมระบบควบคุมเซอร์โวมอเตอร์ของหุ่นยนต์ เพื่อทดแทนระบบเดิมที่มีการป้องกัน source code และ Interface แบบ ISA bus ทำให้ยากในการนำไปศึกษาการทำงานของแขนหุ่นยนต์ เป้าหมายของงานวิจัยนี้คือ สร้างวงจรขับมอเตอร์ขึ้นมาใหม่ด้วยไมโครคอนโทรเลอร์ 16 บิตตระกูล dsPIC เบอร์ dsPIC30F2010 และใช้การควบคุมแบบ PID ควบคุมมอเตอร์ทั้ง 6 ตัวให้ทำงานอิสระต่อกัน โดยรับค่า set-point จากที่เดียวคือ โปรแกรมประยุกต์ที่พัฒนาขึ้นเอง เพื่อคำนวณหาค่ามุมของเซอร์โวมอเตอร์แต่ละตัวด้วย Inverse Kinematics และสร้างเส้นทางเดินของหุ่นยนต์ให้เคลื่อนที่ไปยังตำแหน่งที่ต้องการ ผ่านทางพอร์ตอนุกรม

สกล กงแก้ว ลายมือชื่อนิสิต

ลายมือชื่ออาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

29 / มี.ค. / 51

Sakol kongkaew 2008: Robot Arm Control System Design via Internet.
Master of Engineering (Electrical Engineering), Major Field: Electrical Engineering,
Department of Electrical Engineering. Thesis Advisor:
Assistant Professor Peerayot Sanposh, D.Sc. 174 pages.

The propose of this is to design a controller prototype of a robot arm servo motor to replace the previous system that protect the source code and interface algorithm via ISA bus. This makes it is difficult to study the robot arm. The objective of this research is to build the servo motors drive circuit implemented on dsPIC no. dsPIC30F2010, and use PID to control 6 motors independently by receiving set-points from the user by computer program to calculate to angle values of servo motors using Inverse Kinematics, and also generate robot trajectory and send data to serial port.

SAKOL KONGKAEW — Student's signature

Peerayot Sanposh — Thesis Advisor's signature

29 / Apr. / 2008

กิตติกรรมประกาศ

ข้าพเจ้าขอขอบคุณ ผศ.ดร.พีระยศ แสนโภชน์ อาจารย์ที่ปรึกษาวิทยานิพนธ์หลักเป็นอย่างสูง ที่ให้คำปรึกษาและช่วยชี้แนะแนวทางในการแก้ไขปัญหาต่างๆ

ขอขอบคุณ ผศ.ดร.ยอดเยี่ยม ทิพย์สุวรรณ อาจารย์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ บางเขน ที่ให้การสนับสนุน แขนหุ่นยนต์ ที่ใช้ในงานวิจัย

สุดท้ายต้องขอกราบขอบพระคุณ บิดา มารดา เป็นอย่างสูงที่เป็นกำลังใจสำคัญที่ทำให้การเรียนระดับปริญญาโทประสบผลสำเร็จลุล่วงได้ด้วยดี

สกล กงแก้ว

เมษายน 2551

สารบัญ

หน้า

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

คำอธิบายสัญลักษณ์และคำย่อ

$\mathbf{P_x}$ = เวคเตอร์ย่อยของเวคเตอร์ P ที่แยกออกมาในแกน x

$\mathbf{P_y}$ = เวคเตอร์ย่อยของเวคเตอร์ P ที่แยกออกมาในแกน y

$\mathbf{P_z}$ = เวคเตอร์ย่อยของเวคเตอร์ P ที่แยกออกมาในแกน z

${}^{A}\mathbf{P}$ = เวคเตอร์ P ที่จุดเริ่มต้น A

$\hat{\mathbf{X}}_\mathbf{B}$ = เซต $\{B\}$ เกิดจากการหมุนเฟรมของเซต$\{A\}$ รอบแกน x

$\hat{\mathbf{Y}}_\mathbf{B}$ = เซต $\{B\}$ เกิดจากการหมุนเฟรมของเซต$\{A\}$ รอบแกน y

$\hat{\mathbf{Z}}_\mathbf{B}$ = เซต $\{B\}$ เกิดจากการหมุนเฟรมของเซต$\{A\}$ รอบแกน z

${}^{A}_{B}\mathbf{R}$ = การหมุนเฟรม $\{A\}$ ไปหาเฟรม$\{B\}$

${}^{A}\mathbf{P}_{org}$ = เวคเตอร์ตำแหน่งอ้างอิงจากจุด org

${}^{A}_{D}\mathbf{T}$ = เคลื่อนที่และเปลี่ยนทิศทางของเฟรมหลายๆครั้ง

DOF = จุดที่สามารถเคลื่อนที่ได้ เราจะเรียกว่า degree -of- freedom

a_i = มุมระหว่าง$\mathbf{Z_i}$ กับ $\mathbf{Z_{i+1}}$รอบ $\mathbf{X_i}$

a_i = ระยะระหว่าง$\mathbf{Z_i}$ กับ$\mathbf{Z_{i+1}}$ตามแกน $\mathbf{X_i}$

d_i =ระยะระหว่าง $\mathbf{X_{i+1}}$ กับ$\mathbf{X_i}$ ตามแกน$\mathbf{Z_i}$

L_1 = ความยาวของแขนท่อนที่ 1

L_1 = ความยาวของแขนท่อนที่ 2

L_1 = ความยาวของแขนท่อนที่ 3

$c1 = \cos(q_1)$

$c2 = \cos(q_2)$

$c3 = \cos(q_3)$

$c4 = \cos(q_4)$

$c5 = \cos(q_5)$

$s1 = \sin(q_1)$

$s2 = \sin(q_2)$

$s3 = \sin(q_3)$

$s4 = \sin(q_4)$

$s5 = \sin(q_5)$

การออกแบบระบบควบคุมแขนหุ่นยนต์ผ่านอินเตอร์เน็ต

Robot Arm Control System Design via Internet

คำนำ

งานวิจัยนี้เป็นการนำเสนอการออกแบบและสร้าง บอร์ดควบคุมเซอร์โวมอเตอร์ให้กับแขนหุ่นยนต์ เพื่อใช้แทนระบบเดิมที่มีการป้องกัน Source code และ Interface แบบ ISA bus ซึ่งเป็นระบบ bus ที่ส่วนมากใช้ในคอมพิวเตอร์รุ่นเก่าที่ทำงานได้ช้ากับโปรแกรมใหม่ๆ ส่วนคอมพิวเตอร์รุ่นใหม่ที่มี ISA bus ปัจจุบันนั้นหายากและมีราคาแพงและต้องสั่งซื้อพิเศษ ทำให้ยุ่งยากสิ้นเปลืองงบประมาณ เป้าหมายของงานวิจัยนี้คือ สร้างวงจรขับมอเตอร์ขึ้นมาใหม่ด้วยไมโครคอนโทรเลอร์ 16 บิตตระกูล dsPIC เบอร์ dsPIC30F2010 2 ตัว โดยเลือกใช้ตัวควบคุมแบบ PID ควบคุมมอเตอร์ทั้ง 6 ตัว ให้ทำหน้าที่ควบคุมมอเตอร์แต่ละตัวอย่างอิสระ ใส่แทนวงจรขับมอเตอร์ชุดเดิม ทำให้สามารถอับเกรดหรือซ่อมบำรุงได้ง่ายและเปลี่ยนการเชื่มต่อกับคอมพิวเตอร์จาก ISA bus เป็นการเชื่อมต่อด้วยพอร์ตอนุกรมซึ่งมีอยู่ในคอมพิวเตอร์ทั่วไปทังรุ่เกาและรุ่นใหม่อีกทั้ง ยังสามารถให้ร่วมกับโมดูล USB to RS232 ทำให้ใช้กับพอร์ต USB ได้ด้วยสำหรับเครื่องที่ไม่มีพอร์ตอนุกรม ส่วนโปรแกรม ServoRobot คือโปรแกรมประยุกต์ที่พัฒนาขึ้นใช้เองเขียนขึ้นด้วยโปรแกรม VB.2005 ทำหน้าที่ควบบคุมการทำงานของแขนหุ่นยนต์ และสามารถที่จะควบคุมระยะไกลผ่านเวปได้ด้วย ทำให้สามารถใช้โปรแกรม Internet explorer ทำให้ใช้กับคอมพิวเตอร์ทั่วๆไปควบคุมการทำงานระยะไกล ผ่านเครือข่ายอินเตอร์เน็ตได้

วัตถุประสงค์

1. เพื่อสร้างชุดทดลองและควบคุม Robot arm
2. เพื่อใช้ในการศึกษาและทดสอบทฤษฎีควบคุม
3. เพื่อใช้ในการศึกษา การทำงานของ Browser web
4. สามารถนำชุดทดลองที่ได้มาศึกษาและออกแบบการควบคุมระบบในกระบวนการผลิตแบบใหม่ๆได้ ซึ่งจะเป็นการพัฒนาความรู้ในทางด้านวิศวกรรมควบคุมต่อไป

การตรวจเอกสาร

รายงานผลงานวิจัยที่ผ่านมา

ในปี ค.ศ.2002 สิทธิกร ลาภาพงศ์ ได้ศึกษาระบบการผลิตผ่านเว็บ เป็นการนำข้อดีของอินเตอร์เน็ทมาใช้งานเพื่อทำให้ผู้ใช้สามารถที่เข้าถึงระบบได้จากระยะไกล โดยผู้ใช้ มีเพียงเว็บบราวเซอร์และระบบอินเตอร์เน็ทก็สามารถที่จะล็อกอินเพื่อ เข้าไปสู่ระบบและควบคุมการทำงานของระบบการผลิตได้ โดยระบบ การผลิตในงานวิจัยนี้จะประกอบด้วยแขนหุ่นยนต์ซึ่งมี 5 องศาอิสระ ระบบ สายพานลำเลียง โต๊ะหมุน ต้น โดยโปรแกรมที่ใช้ในการติดต่อและ ควบคุมระบบทั้งหมดได้รับการพัฒนาด้วยภาษาซี พลัสพลัส จาวา และ แอสเซมบลี นอกจากนี้ยังมีการถ่ายทอดสดของการทำงานของระบบ มายังผู้ใช้อีกด้วย

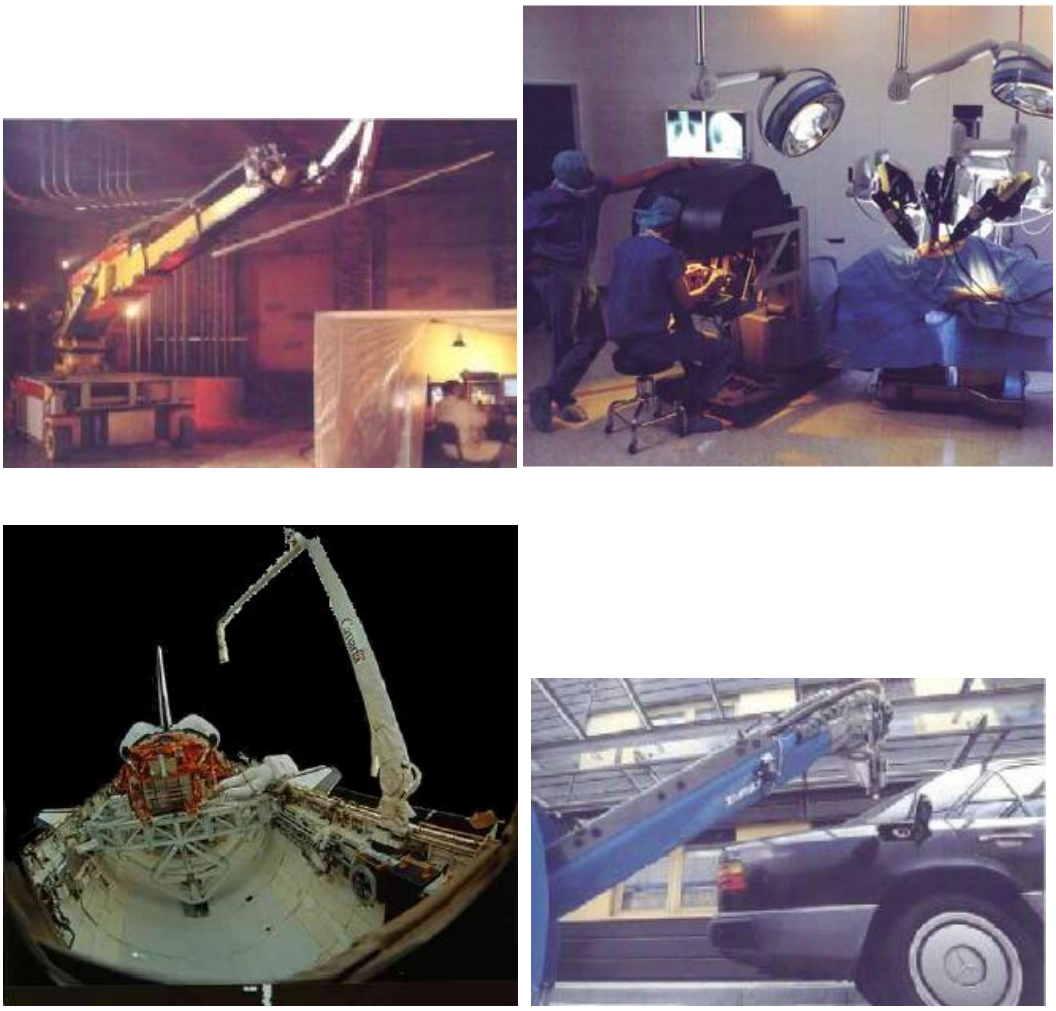
ในปี ค.ศ.2005 Mitja Trunti (June 20-23, 2005) ได้ศึกษาเกี่ยวกับปัญหาของการเปลี่ยนแบบจำลองของหุ่นยนต์ที่มีผลต่อการควบคุมแบบป้อนกลับ เพื่อสร้างต้นแบบการควบคุมหุ่นยนต์ และ หาแบบจำลองการทำงานของหุ่นยนต์เพื่อใช้วิเคราะห์ปัญหาและสามารถนำมาสร้างการควบคุมการทำงานระยะไกลได้โดยใช้ เครือข่ายอินเตอร์เน็ท

ในปี ค.ศ. 1997 Gaetano Borriello (1997 IEEE) ได้นำเสนอการควบคุมอุปกรณ์ ผ่านหน้าเวปเพจโดยใช้ซอฟแวร์ต้นทุนต่ำอย่างภาษาจาวา ทำให้ผู้ใช้สามารถสลักกันควบคุมหุ่นยนต์ได้ และแลกเปลี่ยนข้อมูลกันและกันได้ด้วย ตัวอย่างที่ได้นำมาประยุกต์ใช้คือ เชื่อมต่อกับอุปกรณ์จำพวกกล้องดิจิตอลและหุ่นยนต์ก็จะ สามารถควบคุมการทำงานของหุ่นยนต์ได้แบบป้อนกลับจากระยะไกลได้

ในปี ค.ศ. 2001 Kaan Erkorkmaz ได้ศึกษาการสร้างเรื่องเส้นทางเดิน ของเครื่องจักร CNC แบบความเร็วสูงโดยต้องการให้รูปแบบการเคลื่อนที่และไม่มีการกระตุกของเครื่องจักรหรือ servo มอเตอร์ เขาจึงเลือกที่จะใช้การสร้างเส้นทางเดินแบบ spline เทคนิคที่มีการใช้งานกันอย่างแพร่หลาย เพราะว่าจะทำให้อัตราการ เคลื่อนที่ของเครื่องมือมีความต่อเนื่อง การสร้างเส้นทางเดินแบบ spline จะประกอบไปด้วยเส้นตรงและเส้นโค้ง ซึ่งทำให้ผลที่ได้ออกมาราบเรียบดีที่สุด

ความหมายของแขนหุ่นยนต์

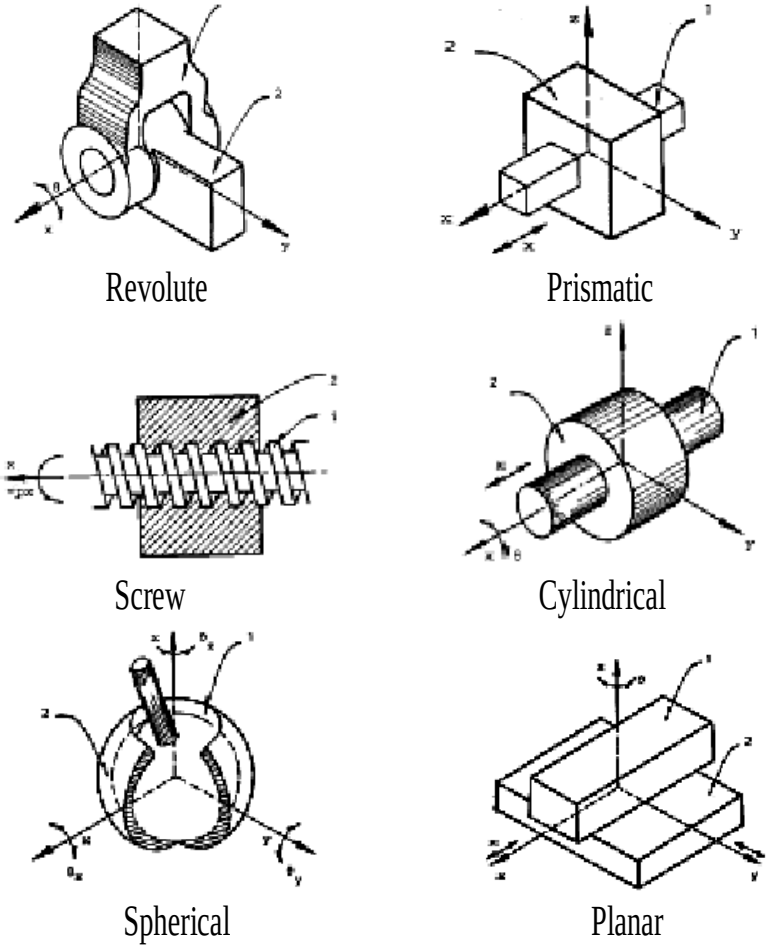
คำจำกัดความของหุ่นยนต์นั้น กำหนดว่าเป็นวัตถุชิ้น เดียวเพื่อความสะดวกในการวิเคราะห์ แต่ในความเป็นจริงนั้นหุ่นยนต์ไม่ได้สร้างจากวัตถุชิ้นเดียวแต่ประกอบด้วยชิ้นส่วนต่างๆ หลายชิ้น ที่เคลื่อนที่สัมพันธ์กัน นอกเหนือจากหุ่นยนต์ที่เคลื่อนที่ได้แล้ว ยังมีหุ่นยนต์อีกประเภทที่สามารถ พบ เห็นได้ในโรงงานประ กอบชิ้นส่วน นั่นคือแขนหุ่นยนต์ ปัจจุบันแขนหุ่นยนต์ได้มีบทบาทข้าม ขอบเขตของโรงงานเข้ามา ในชีวิตของเราในแง่อื่นๆ เพิ่มมากขึ้นเรื่อยๆ ยกตัวอย่างจากภาพที่ 1 ตัวอย่างการประยุกต์ใช้แขนหุ่นยนต์ เช่น ใช้ในโรงงานประกอบชิ้นส่วน ใช้ในทางการแพทย์ อวกาศ และในอุตสาหกรรมรถยนต์ตามลำดับ



ภาพที่ 1 ตัวอย่างการประยุกต์ใช้แขนหุ่นยนต์

ประเภทของข้อต่อ

แขนหุ่นยนต์คือหุ่นยนต์ที่ประกอบไปด้วยท่อนแขน (link) ที่นำมาประกอบกันด้วยข้อต่อ (joint) ข้อต่อมีหลายแบบ แต่ละ แบบจะอนุญาตให้เกิดการเคลื่อนที่ของท่อนแขนที่แตกต่างกันไป ภาพที่ 2 แสดงข้อต่อแบบต่างๆ ที่นิยมใช้ ลองพิจารณาว่าข้อต่อละแบบบังคับการเคลื่อนที่ของท่อนแขนสองท่อนที่มาเชื่อมกันอย่างไร

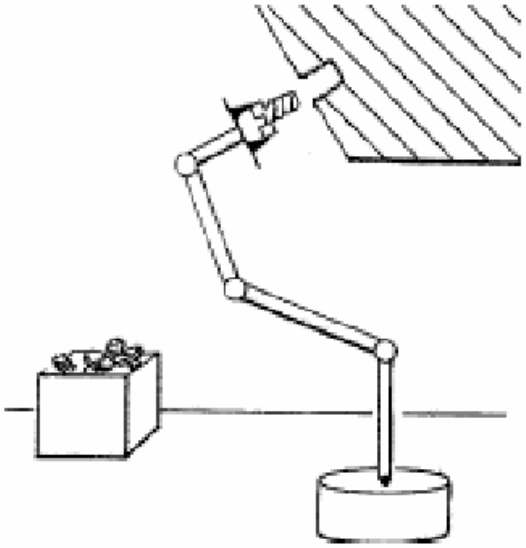


ภาพที่ 2 ข้อต่อแบบ ต่างๆ

ในการสร้างแขนหุ่นยนต์โดยทั่วไป ข้อต่อที่ นิยมใช้มากที่สุดคือ ข้อต่อแบบหมุน (Revolute joint) และข้อต่อแบบเลื่อน (prismatic joint) สำหรับข้อต่อแบบหมุน ท่อน แขนสองท่อนถูกยึดติดกันที่จุดหมุนซึ่งอยู่บนท่อนแขน โดยแต่ละท่อนสามารถหมุนได้รอบจุดหมุนนี้ เราสามารถบอกตำแหน่งของสองท่อนแขนที่สัมพันธ์กัน ด้วยมุมที่ท่อนแขนหมุน ส่วนข้อต่อแบบเลื่อนนั้น ท่อนแขนสองท่อนติดอยู่ด้วยกันในลักษณะเดียวกันกับเสาอากาศวิทยุรถยนต์ที่ยืดหดได โดยท่อนแขนแต่ละท่อนสามารถเลื่อนเข้าออกได้ในหนึ่งทิศทาง เราสามารถระบุ ตำแหน่ง ที่

สัมพันธ์กันของสอง ท่อนแขนได้จากระยะเลื่อนเข้าออกดังกล่าว จะเห็นได้ว่า ข้อต่อแบบหมุนและข้อต่อแบบเลื่อนมีระดับเสรีของการเคลื่อนที่ เป็นหนึ่ง เราเรียกตัวแปรที่กำหนดการเคลื่อนที่นี้ซึ่งได้แก่มุมหมุนของ ข้อต่อแบบหมุน และระยะเลื่อนของข้อต่อแบบ เลื่อนว่าเป็นพารามิเตอร์ของ ข้อต่อการมีระดับเสรีของการเคลื่อนที่เป็นหนึ่งทำให้ง่านในการออกแบบและวิเคราะห์ ข้อต่อทั้งสองแบบจึงถูกใช้มากที่สุดในการสร้างแขนหุ่นยนต์ โดยแขนหุ่นยนต์ที่มีระดับเสรีสูงๆ ก็สามารถสร้างขึ้นได้ โดยการ ประกอบท่อนแขนหลายท่อนด้วยข้อต่อสองแบบนี้

แขนหุ่นยนต์ทำงานด้วยการเคลื่อนที่ของท่อนแขนที่สัมพันธ์กันเพื่อให้ปลายแขน (End effecter) ไปอยู่ในตำแหน่งและทิศทางที่เหมาะสม เพื่อเครื่องมือที่ติด อยู่ที่ปลายแขนจะได้ทำงานที่ต้องการได้โดยสะดวกและมีประสิทธิภาพ ตัวอย่างในภาพที่ 3 แสดงให้เห็นถึงความจำเป็นที่ต้องจัดการให้ปลายแขนอยู่ในตำแหน่ง และทิศทางที่เหมาะสม



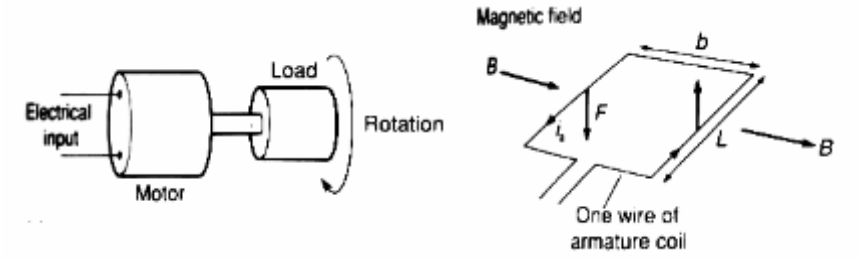
ภาพที่ 3 แขนหุ่นยนต์กำลังไขสกรูลงในเกลียวที่ต้องการ

การคำนวณ ว่าปลายแขนจะอยู่ที่ตำแหน่ง และทิศทางใดจึงเป็นเรื่องสำคัญ การคำนวณดังกล่าวอาศัยการกำหนดให้ท่อนแขนแต่ละท่อนมีพิกัดส่วนตัว ที่เราจะเรียกว่าเฟรม เฟรมประกอบไปด้วยจุดกำเนิดและ เวคเตอร์แกน โดยเฟรมที่กล่าวถึงจะอยู่ติด แน่น กับท่อนแขนที่เป็นเจ้าของเสมอ หรืออีกนัยหนึ่งก็คือแต่ละท่อนแขนจะอยู่นิ่งไม่ขยับเขยื้อน เมื่อเทียบกับเฟรมของมัน สำหรับท่อนแขนที่เกิดจากการเรียง ต่อกันไปเรานิยมเรียกท่อนแขนที่อยู่นิ่งยึดติดกับพื้น ว่าฐาน (Base) และเรียกท่อนถัดมาตามชื่อ ส่วนของแขนได้แก่ ไหล่(shoulder) ข้อศอก(elbow) แขนท่อนบน (forearm) และ ข้อมือ (wrist) เป็น ต้น ตำแหน่งและทิศทางการวางตัวของท่อนแขนหนึ่งๆ เมื่อ

เทียบกับเฟรมของฐานจึงขึ้นอยู่กับตำแหน่งและทิศทางของท่อนแขนก่อนๆ ด้วย เราสามารถคำนวณตำแหน่งและทิศทางของปลายแขนได้ด้วยการใช้การแปลงเอกพันธ์ที่ได้ศึกษาไปแล้ว โดยทำการคูณเมทริช์การแปลงแบบซ้ายไปขวาพิจารณาจากฐานไปจนถึงปลายแขน (อรรถวิทย์ สุดแสง)

มอเตอร์ไฟฟ้ากระแสตรง

เป็นอุปกรณ์ที่ใช้เปลี่ยนพลังงานไฟฟ้าเป็นพลังงานกล หรือจะกล่าวว่าเป็นระบบที่มีสัญญาณไฟฟ้าเป็นอินพุท และมีเอาท์พุทเป็นพลังงานกลก็ได้ โดยทั่วไปมอเตอร์จะประกอบด้วยขดลวดที่ส่วนหมุน หรือ Armature coil ซึ่งสามารถที่จะหมุนไปได้อย่างอิสระขดลวดนี้จะวางอยู่ในสนามแม่เหล็ก ซึ่งอาจจะเป็นแม่เหล็กถาวร หรือส่วนมากจะเป็นแม่เหล็กไฟฟ้าที่สร้างจากกระแสไฟฟ้าผ่าน Field coils. เมื่อมีกระแสไฟฟ้า i_a ไหลผ่าน armature coil ซึ่งวางอยู่ในสนามแม่เหล็ก ก็จะทำให้เกิดแรงผลักทำให้ armature นี้เกิดการหมุน ตามที่แสดงในภาพที่ 4

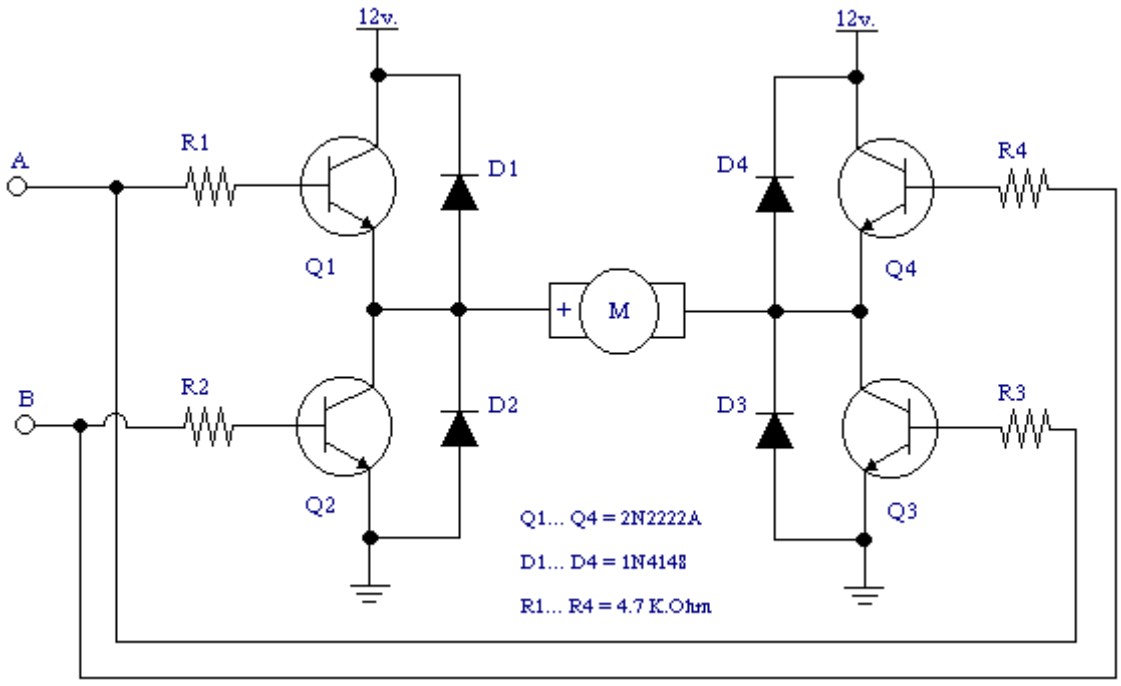


ภาพที่ 4 มอเตอร์กระแสตรง

1. **การควบคุมทิศการหมุนของมอเตอร์กระแสตรง**

การควบคุมทิศการหมุนของมอเตอร์กระแสตรงสามารถทำได้ โดยการควบคุมทิศทางการไหลของกระแสไฟฟ้าที่ไหลผ่านขดลวดอาเมเจอร์ ซึ่งสามารถทำได้โดยการกลับขั้วไฟฟ้าที่ป้อนให้กับขั้วของมอเตอร์ โดยการควบคุมการไหลของกระแสไฟฟ้านั้นส่วนใหญ่แล้วจะใช้วิธีการต่อวงจรอิเล็กทรอนิกส์ที่เรียกว่า H- Bridgeเข้ากับมอเตอร์ ดังภาพที่ 5 มีหลักการทำงานพื้นฐานดังนี้ เมื่อมีการจ่ายไบอัสเข้าที่จุด A ทำให้มีกระแสไหลผ่านR1 เข้าสู่ขา base ของ Q1 และมีกระแสไหลผ่าน R3 เข้าสู่ขา base ของ Q3 ทำให้ Q1 และ Q3 ทำงาน (ON) เปรียบเสมือนสวิทช์ปิดวงจร ส่งผลให้มีกระแสไหลจากแหล่งจ่าย 12 V ผ่านขา Collector และ Emitter ของ Q1 ผ่านเข้าสู่ขาบวกของมอเตอร์ผ่านไปยัง Collector และ Emitter ของ Q3 ทำให้มีกระแสไหลผ่านมอเตอร์ในทิศทางบวก

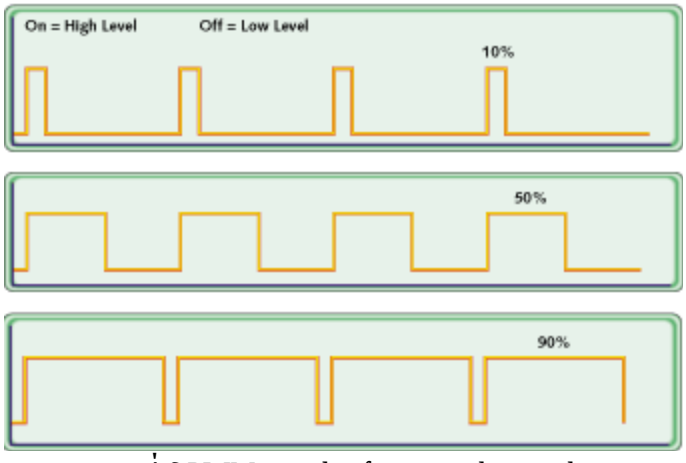
และครบวงจร จึงทำให้มอเตอร์สามารถหมุน ในทิศทาง Forward และในกรณีของการหมุนแบบ Reward คือ เมื่อมีการจ่ายไบอัสเข้าที่จุด B ทำให้มีกระแสไหลผ่าน R2 เข้าสู่ขา base ของ Q2 และมีกระแสไหลผ่าน R4 เข้าสู่ขา base ของ Q4 ทำให้ Q2 และ Q4 ทำงาน (ON) เปรียบเสมือนสวิทช์ปิดวงจร ส่งผลให้มีกระแสไหลจากแหล่งจ่าย 12 V ผ่านขา Collector และ Emitter ของ Q4 ผ่านเข้าสู่ขาลบของมอเตอร์ผ่านไปยัง Collector และEmitter ของ Q2 ทำให้มีกระแสไหลผ่านมอเตอร์ในทิศทางลบ และครบวงจร ข้อควรระวังก็คืออย่าให้ทรานซิสเตอร์ 4 ตัว ทำงานพร้อมกันเด็ดขาด เพราะจะทำให้เกิดการลัดวงจรถึงแม้ว่าการสร้างวงจร H-Bridge Switching จาก Transistor นั้นจะไม่มีปัญหารบกวนจากอำนาจสนามแม่เหล็กและยังตอบสนองการทำงานได้เร็วมาก แต่ก็ยังมีอุปกรณ์หลายตัว ทำให้วงจรมีขนาดใหญ่ จึงมีวงจร H-Bridge Switching ในรูปของ IC ที่สามารถนำมาใช้งานได้ง่าย และสะดวกกว่า คือ (DUAL FULLBRIDGEDRIVER) L298



ภาพที่ 5 วงจร H – Bridge

2. การควบคุมความเร็วของมอเตอร์กระแสตรง

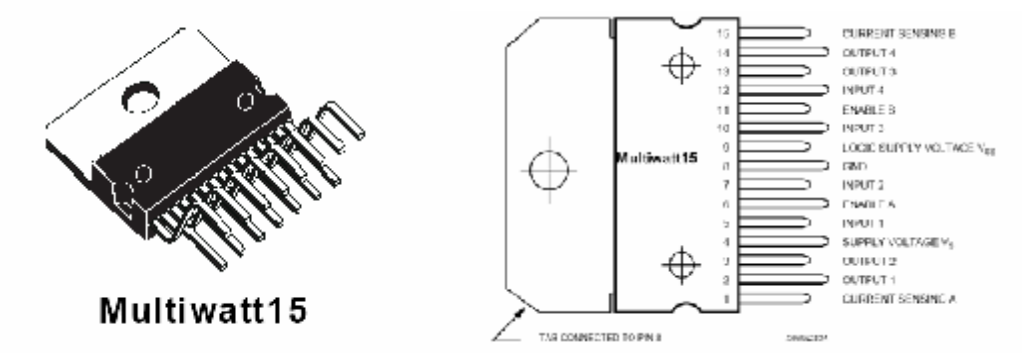
เนื่องจากส่วนของ Stator เป็นแม่เหล็กถาวร การควบคุมความเร็วจึงทำได้โดยการเปลี่ยนค่า armature voltage ซึ่งค่าความต่างศักย์นี้จะแปรผันตรงกับความเร็วในการหมุนของมอเตอร์ วิธีที่จะเปลี่ยนระดับความต่างศักย์ไฟฟ้า จะใช้คลื่นรูปสี่เหลี่ยมที่สามารถเปลี่ยนแปลงช่วงเวลาในการเปิดและปิดแหล่งจ่ายได้ซึ่งจะเป็นการเปลี่ยนแปลงค่าเฉลี่ยของแรงดัน เรียกว่า Pulse Width Modulation ภาพที่ 6 PWM signals of varying duty cycles เป็นสัญญาณที่สร้างจาก ไมโครคอนโทรลเลอร์ให้ควบคุม %duty cycle ของ Pulse Width ถ้า %duty cycle มาก ก็จะทำให้ค่าเฉลี่ยของแรงดันมากหรือทำให้มอเตอร์หมุนเร็ว หรือถ้า %duty cycle น้อย ก็จะทำให้ค่าเฉลี่ยของแรงดันน้อยหรือทำให้มอเตอร์หมุนช้านั้นเอง



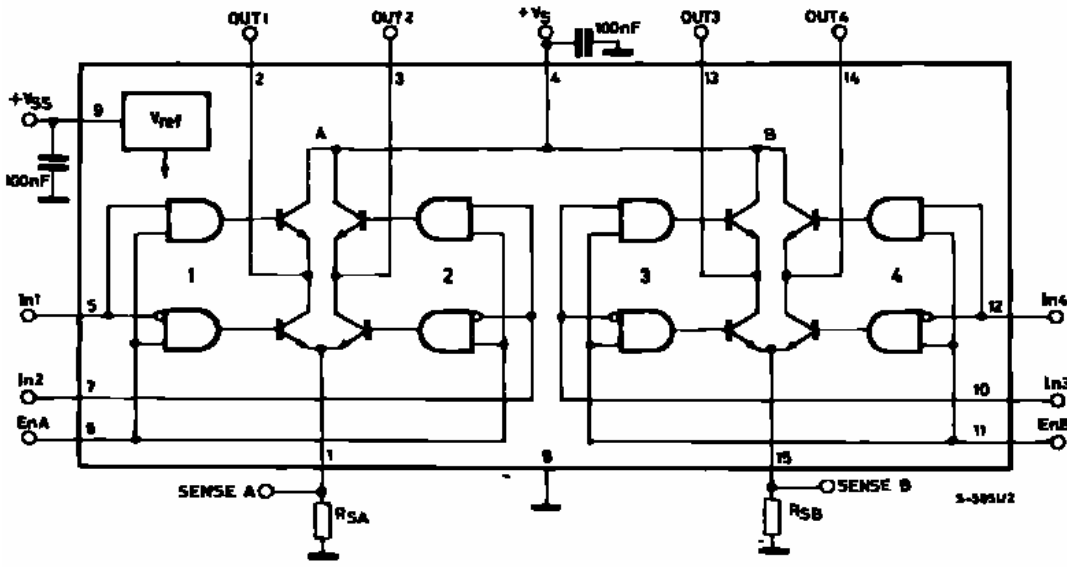
ภาพที่ 6 PWM signals of varying duty cycles

3. ชุดควบคุมมอเตอร์

การเปลี่ยนทิศทางการหมุนของดีซีมอเตอร์แบบแม่เหล็กถาวรทำได้โดยสลับขั้วของแหล่งจ่ายไฟฟ้าที่ต่อเข้ากับมอเตอร์ และในทางปฏิบัติจะใช้วงจรอิเล็กทรอนิกส์ที่เรียกว่า H Bridge เป็นตัวจัดการการทำงานในส่วนของโครงงานนี้ได้ศึกษาDual Full-Bridge Driver L298 ซึ่งมีตัวถังแสดงดังภาพที่ 7 การจัดขาของ IC L298



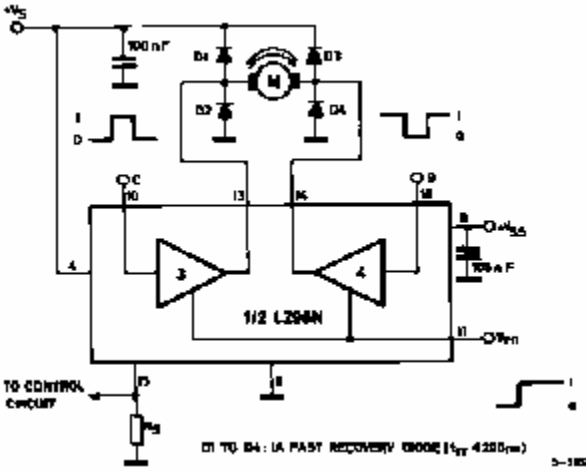
ภาพที่ 7 การจัดขาของ IC L298



ภาพที่ 8 วงจรภายในของ IC L298

จากภาพที่ 8 วงจรภายในจะเห็นว่า L298 สามารถขับโหลดได้ 2 ช่อง และสามารถรับสัญญาณควบคุมแบบ TTL (Transistor Transistor logic) เพื่อที่จะควบคุมทิศทางการไหลของกระแส และมีขา Enable เพื่อป้อนสัญญาณPulse Width Modulation เพื่อควบคุมความเร็วขา Emitter ของ Transistor ทั้งสองข้างของแต่ละ Bridge จะต่อกับตัวต้านทานภายนอกเพื่อที่จะใช้ในการกำหนดค่ากระแสไฟฟ้าที่ไหลได้โดยหากเกินกว่าที่วงจรและโหลดต่ออยู่สามารถที่จะรับได้ก็อาจจะมีวงจรเพื่อที่ทำการ Disable การทำงานของโหลดได้

เนื่องจากโหลดที่ใช้เป็นตัวเหนี่ยวนำค่ากระแสไฟฟ้า ไม่สามารถเปลี่ยนเป็นศูนย์ได้ในทันทีทันใด(คล้ายกับตัวเก็บประจุที่ไม่สามารถเปลี่ยนค่าความต่างศักย์ได้อย่างทันทีทันใด) จึงต้องมีการต่อไดโอดอยู่ภายนอกเพื่อให้กระแสไฟฟ้าไหลได้ดังภาพที่ 9 การต่อใช้งานของ IC L298 แบบ 1 ช่อง

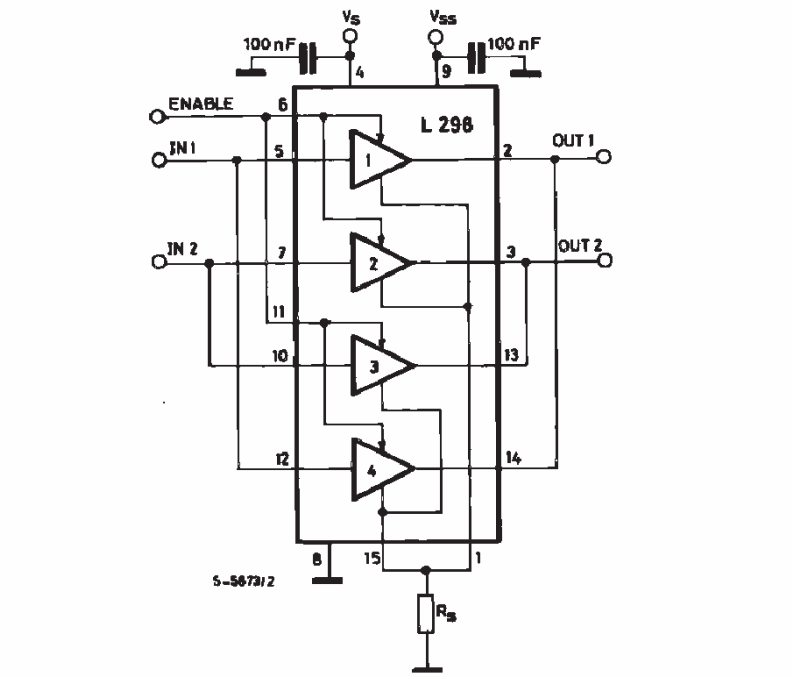


Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

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

ภาพที่ 9 การต่อใช้งานของ IC L298 แบบ 1 ช่อง

ในทางปฏิบัตินั้นมอเตอร์ต้องการกระแสสูงเพื่อไปขับโหลดที่หนักๆซึ่งสามารถต่อวงจรแต่ละChannel ขนานกันเพื่อให้วงจรขับมอเตอร์ได้จ่ายกระแสไปยังมอเตอร์ได้มากขึ้นดังภาพที่ 10

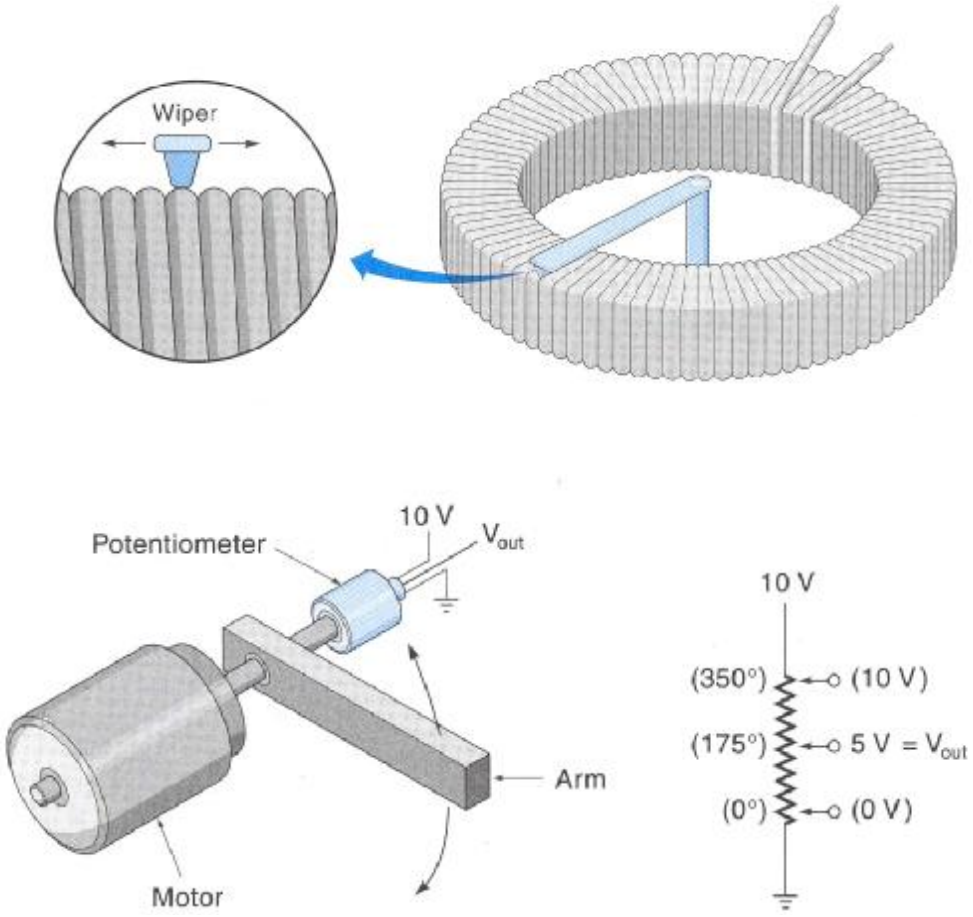


ภาพที่ 10 การต่อใช้งานแบบขนาน

การควบคุมตำแหน่งของมอเตอร์กระแสตรง

การควบคุมตำแหน่งมอเตอร์ (Position Control) มีหลักการคือจะทำการเปลี่ยนจากตำแหน่งการหมุนเป็นค่าแรงดันไฟฟ้าโดยใช้หลักการของการแบ่งแรงดัน (Voltage divider) มีชื่อเรียกว่า โพเทนชิออมิเตอร์ (Pot) โดยที่ตัว Pot จะต่อกับเพลาของมอเตอร์ และตัวถังจะยึดอยู่กับที่ เมื่อมีการหมุนก็คือเปลี่ยนค่าความต้านทานเปลี่ยนเป็นค่าแรงดันไฟฟ้า ค่าแรงดันที่ได้นี้จะนำไปแปลงเป็นสัญญาณดิจิตอล โดยผ่านวงจร (Analog Digital Converter) ซึ่งข้อมูลดิจิตอลที่ได้จะเรียกว่าข้อมูลตำแหน่ง Data position และจะนำไปประมวลผลโดยใช้ไมโครคอนโทรลเลอร์ต่อไป

ตัว Pot จะใช้เป็นชนิด โพเทนชิออมิเตอร์แบบขดลวด ดังภาพที่ 11 เพราะมีการเปลี่ยนค่าความต้านทานเป็นเชิงเส้น และมีขอบเขตการหมุน 350 องศา เนื่องจากตัวโครงสร้างของมัน

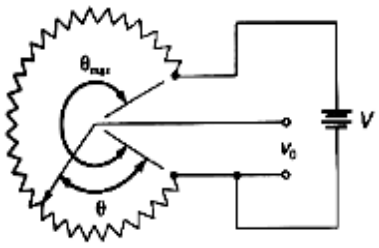


ภาพที่ 11 การวัดตำแหน่งการหมุนของมอเตอร์

1. โพเทนชิออมิเตอร์

เป็นอุปกรณ์ที่ใช้ปรับแรงเคลื่อนไฟฟ้า ดังนั้นจะมีความสัมพันธ์ระหว่างการขจัดและแรงเคลื่อนไฟฟ้า สำหรับ โพเทนชิออมิเตอร์ นั้น โดยปกติจะมี 2 แบบ คือ แบบเชิงเส้นกับแบบเชิงมุม ซึ่งแบ่งตามลักษณะการเคลื่อนที่ โดยแบบเชิงเส้นตัวชี้จะมีการเคลื่อนที่เป็นเส้นตรง ในขณะที่แบบเชิงมุมตัวชี้จะหมุนรอบแกนแกนหนึ่ง

พิจารณา โพเทนชิออมิเตอร์ แบบเชิงมุม ดังแสดงในภาพที่ 11



ภาพที่ 12 โพเทนชิออมิเตอร์

หากพิจารณา Voltage divided circuit เราจะได้

$$\frac{v_0}{v} = \frac{q}{q_{\max}} \tag{1}$$

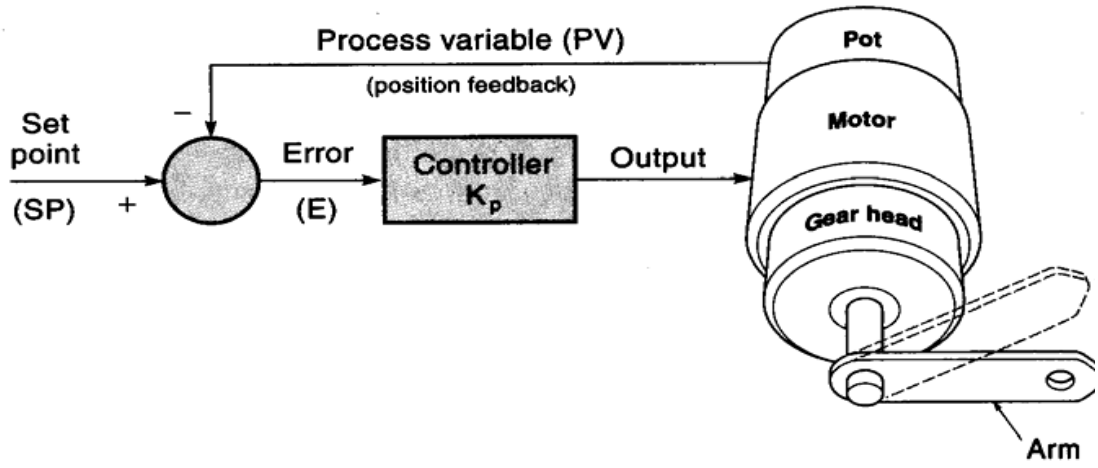
เมื่อ v คือความต่างศักย์ที่ตกคร่อมทั้งหมดตลอดความยาวของ โพเทนชิออมิเตอร์ และ $q_{\max}$ คือมุมหมุนทั้งหมดของ โพเทนชิออมิเตอร์ , ดังนั้นอัตราส่วนของเอาท์พุท v_0 กับอินพุท q จะเป็น

$$\frac{output}{input} = \frac{v_0}{q} = \frac{v}{q_{\max}} = \text{ค่าคงที่} \tag{2}$$

2. การควบคุมแบบป้อนกลับ

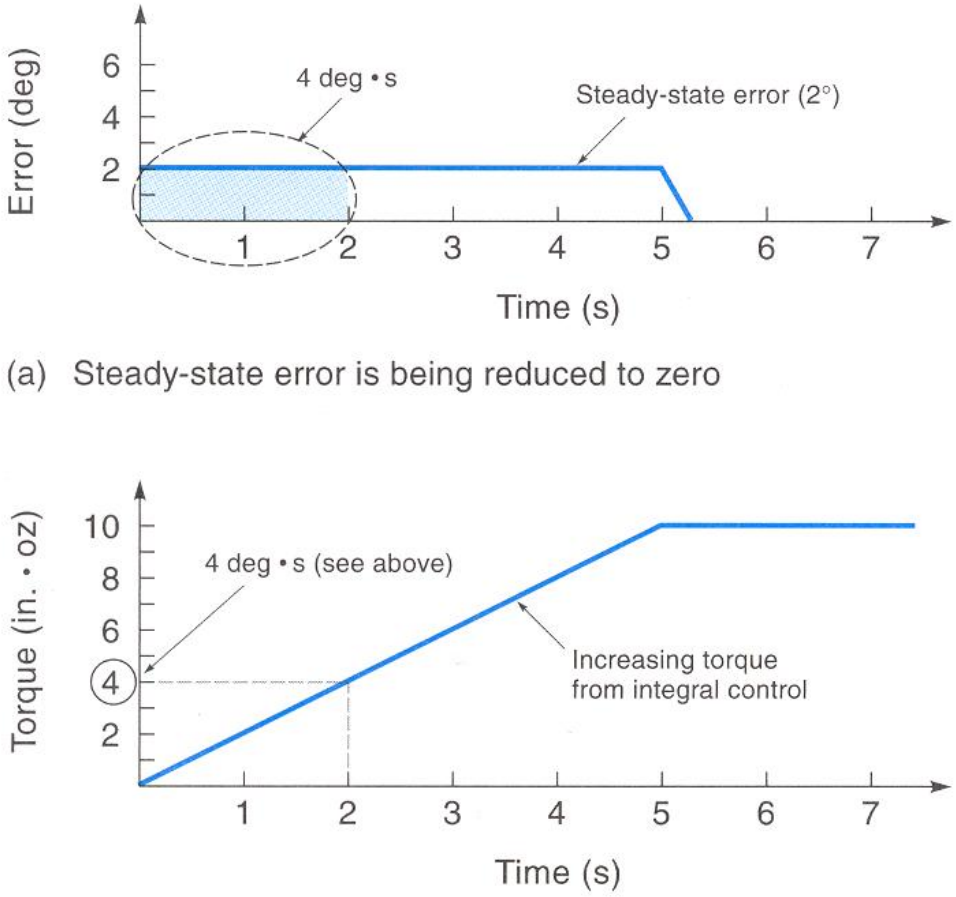
ระบบการควบคุมแบบป้อนกลับเป็นระบบการควบคุมที่นิยมใช้ในงานอุตสาหกรรมต่าง ๆ เป็นอย่างมากเนื่องจากคุณสมบัติที่ว่า ระบบการควบคุมแบบนี้สามารถปรับค่าการควบคุมได้เมื่ออุปกรณ์ที่ต้องการควบคุมเปลี่ยนไป ซึ่งประสิทธิภาพในการควบคุมก็ให้ผลดีเป็นที่น่าพอใจ อีกทั้งการควบคุมก็ทำได้ง่ายและไม่ซับซ้อนจนเกินไปซึ่งเราสามารถแบ่งการควบคุมออกเป็นส่วนๆ ดังนี้

2.1. ระบบควบคุมแบบสัดส่วน เป็นการนำสัญญาณความคลาดเคลื่อน E ระหว่างสัญญาณเข้า SPและสัญญาณขาออก PVไปคูณกับค่าคงที่ของการควบคุมของการควบคุมแบบสัดส่วน (Proportional Gain) แล้วส่งสัญญาณที่ได้ไปขับอุปกรณ์ ซึ่งจะสังเกตได้ว่าหากสัญญาณทั้งสองมีความแตกต่างกันมากๆ แล้วระบบจะทำให้เกิดสัญญาณควบคุมมาก จะทำให้อุปกรณ์อยู่ในสภาวะที่เราต้องการได้อย่างรวดเร็ว อย่างไรก็ตามการควบคุมแบบนี้จะทำให้เกิดสภาวะของการสั่นรอบๆจุดของสัญญาณที่เราต้องการ ซึ่งเรียกว่าการเกิด Overshoot โดยหากเราเพิ่มค่าคงที่ของการควบคุมแบบสัดส่วนมากๆ นั้นระบบจะเร็วขึ้นก็จริงแต่อาจเกิดการแกว่งของสัญญาณรอบๆ จุดที่ต้องการได้ variable ข้อมูลตำแหน่งจะถูกป้อนกลับ ซึ่งเรียกว่า Process variable (PV) และมาลบกับค่า set point (SP) ผลต่างระหว่าง SP กับ PV เรียกว่า error E = SP ตามภาพที่ 13 บล็อกไดอะแกรมของระบบควบคุมแบบพี



ภาพที่ 13 บล็อกไดอะแกรมของระบบควบคุมแบบพี

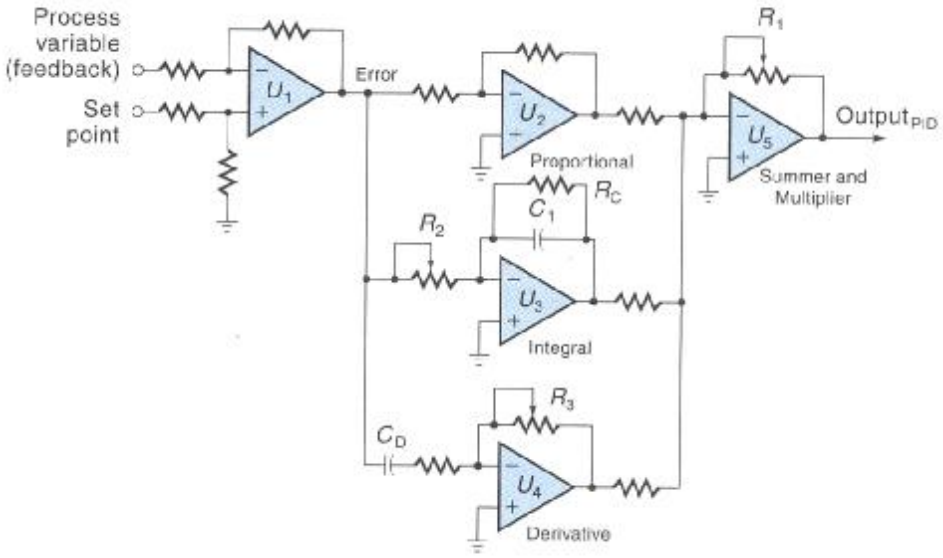
2.2. ระบบควบคุมแบบสะสม ในระบบการควบคุมนั้น บางครั้งระบบการควบคุมไม่อาจควบคุมอุปกรณ์ให้ไม่มีความคลาดเคลื่อนของสัญญาณขาเข้าและสัญญาณขาออกได้จึงได้มีการเพิ่มระบบการควบคุมแบบสะสมค่าความคลาดเคลื่อนแล้วนำไปคูณกับค่าคงที่ของการควบคุมแบบสะสม เพื่อให้เกิดค่าความผิดพลาดน้อยที่สุด ตามภาพที่ 14 หลักการทำงานของตัวควบคุมแบบสะสม



ภาพที่ 14 หลักการทำงานของตัวควบคุมแบบสะสม

2.3. ระบบควบคุมแบบความแตกต่าง ระบบการควบคุมแบบนี้จะหาว่าค่าความคลาดเคลื่อน ในอดีตกับปัจจุบันมีความแตกต่างกันมากเพียงใดซึ่งหากมีความแตกต่างมากเมื่อเรานำสัญญาณความแตกต่างไปคูณกับค่าคงที่ของการควบคุมแบบสะสมแล้วก็จะได้สัญญาณที่จะทำให้ระบบนั้นเร็วขึ้นได้

ระบบ Control ส่วนมากจะรวมระบบ 3 ส่วนนี้เข้าด้วยกันเพื่อปรับปรุงระบบให้มีประสิทธิภาพมากขึ้นเรียกว่า Proportional + Integral + Derivative (PID) Control ดังภาพที่ 15 วงจรของตัวควบคุมแบบพีไอดี

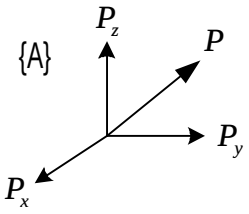


ภาพที่ 15 วงจรของตัวควบคุมแบบพีไอดี

การบอกตำแหน่งของวัตถุในแบบ 3 **มิติ**

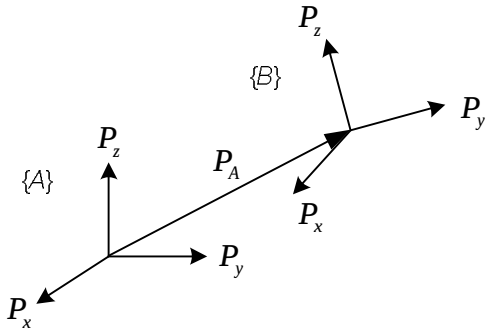
การกำหนดตำแหน่งของวัตถุใน 3 มิติ นั้นสามารถกำหนดตำแหน่งของจุด 1 จุด ด้วย $\mathbf{P}_x$ $\mathbf{P}_y$ และ $\mathbf{P}_z$ ส่วนใหญ่จะเขียนด้วยสัญญาลักษณ์ ${}^{A}\mathbf{P}$ ซึ่งแสดงขนาดของแต่ละแกนตาม ตัวอย่างในภาพที่ 16 การบอกตำแหน่งของวัตถุใน 3 มิติ

$$ {}^{A}\mathbf{P} = \begin{bmatrix} \mathbf{P_x} \\ \mathbf{P_y} \\ \mathbf{P_z} \end{bmatrix} $$



ภาพที่ 16 การบอกตำแหน่งของวัตถุใน 3 มิติ

การกำหนดทิศทางของวัตถุใน 3 มิติ โดยปรกติจะบอกว่ามีการหมุนเฟรมรอบแกนใดไปทางทิศใด ถ้ามี ${}^{A}\mathbf{P}$ เป็นเวคเตอร์ดูตัวอย่างในภาพที่ 17 การบอกตำแหน่งของวัตถุใน 3 มิติ



ภาพที่ 17 การบอกตำแหน่งของวัตถุใน 3 มิติ

$\hat{\mathbf{X}}_{\mathbf{B}}$ หมายถึง เซต $\{B\}$ เกิดจากการหมุนเฟรมของเซต $\{A\}$ รอบแกน x

$\hat{\mathbf{Y}}_{\mathbf{B}}$ หมายถึง เซต $\{B\}$ เกิดจากการหมุนเฟรมของเซต $\{A\}$ รอบแกน y

$\hat{\mathbf{Z}}_{\mathbf{B}}$ หมายถึง เซต $\{B\}$ เกิดจากการหมุนเฟรมของเซต $\{A\}$ รอบแกน z

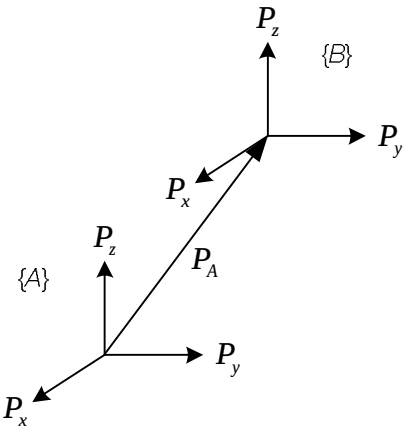
ซึ่งถ้านำการหมุนของทั้ง 3 แกนมาเขียนในรูปของ เมททริก ขนาด 3x3 จะเรียกว่า Rotation matrix ${}^{A}_{B}\mathbf{R}$ หมายถึงหมุนเฟรม $\{A\}$ ไปหาเฟรม $\{B\}$

$$ {}^{A}_{B}\mathbf{R} = [{}^{A}\hat{\mathbf{X}}_{\mathbf{B}} \quad {}^{A}\hat{\mathbf{Y}}_{\mathbf{B}} \quad {}^{A}\hat{\mathbf{Z}}_{\mathbf{B}}] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{3} $$

ผลรวมของเวคเตอร์ทั้ง 3 แกนใช้กำหนดทิศการหมุนจากจุดหนึ่งไปอีกจุดหนึ่ง ประกอบกันเป็น เมททริกขนาด 3x3 Rotation matrix มีคุณสมบัติทางคณิตศาสตร์ดังนี้

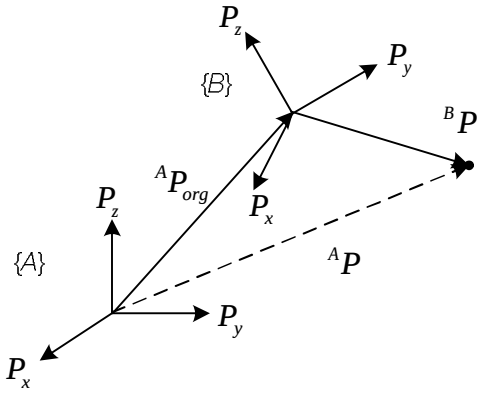
$$ {}^{A}_{B}\mathbf{R} = {}^{B}_{A}\mathbf{R}^{T} = {}^{B}_{A}\mathbf{R}^{-1} \tag{4} $$

$$ {}^{A}_{B}\mathbf{R}\,{}^{B}_{A}\mathbf{R}^{T} = \mathbf{I}_3 \tag{5} $$



ภาพที่ 18 การเคลื่อนที่ทางขนาด

การเคลื่อนที่ทางขนาดอย่างเดียวตัวอย่างเช่น เวคเตอร์ ${}^{A}\mathbf{P}_{org}$ คือเวคเตอร์ตำแหน่งอ้างอิงจากจุด org เฟรม{A} ไปยังเฟรม {B} โดยที่ไม่มีการเปลี่ยนการหมุนของเฟรมตามตัวอย่างในภาพที่ 18 การเคลื่อนที่ทางขนาด



ภาพที่ 19 การเปลี่ยนแปลงเวคเตอร์แบบทั่วๆไป

ถ้าการตำแหน่งของเวคเตอร์ ${}^{A}\mathbf{P}_{org}$ จากเฟรม {A} ไปยังเฟรม {B} และมีการเปลี่ยนการหมุนของเฟรมและมีเวคเตอร์ ${}^{B}\mathbf{P}$ ต่อจากเฟรม {B}อีกครั้ง9k, ภาพที่ 19 การเปลี่ยนแปลงเวคเตอร์แบบทั่วๆไป จะได้ผลรวมของเวคเตอร์เป็น ${}^{A}\mathbf{P}={}^{A}_{B}\mathbf{R}\,{}^{B}\mathbf{P}+{}^{A}\mathbf{P}_{org}$ ซึ่งสามารถเขียนให้อยู่ในรูปของเมททริกขนาด 4x4 ได้ เราเรียกว่า homogeneous transform

$$\begin{bmatrix} {}^{A}\mathbf{P} \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{A}_{B}\mathbf{R} & {}^{A}\mathbf{P}_{org} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{B}\mathbf{P} \\ 1 \end{bmatrix} \tag{6}$$

$$\begin{bmatrix} {}^{A}\mathbf{P} \\ 1 \end{bmatrix} = {}^{A}_{B}\mathbf{T} \begin{bmatrix} {}^{B}\mathbf{P} \\ 1 \end{bmatrix} \tag{7}$$

Compound Transformation คือการเคลื่อนที่และเปลี่ยนทิศทางของเฟรมหลายๆครั้ง เราสามารถเขียนให้อยู่ในรูปของ homogeneous transformation ได้ดังนี้

$$^{A}_{D}\mathbf{T}=^{A}_{B}\mathbf{T}^{B}_{C}\mathbf{T}^{C}_{D}\mathbf{T} \qquad (8)$$

โปรแกรม VisuaI Basic 2005

โปรแกรม Vlsual Baslc 2005 คือภาษาโปรแกรมภาษาหนึ่งที่ใช้สำหรับเขียนโปรแกรมเพื่อให้ทำงานภายใต้ NET Framework หรือกล่าวได้วาโปรแกรม Vlsual Baslc 2005 ก็คือภาษาโปรแกรมขคง NET(NET language) ภาษาหนึ่งนั่นเอง

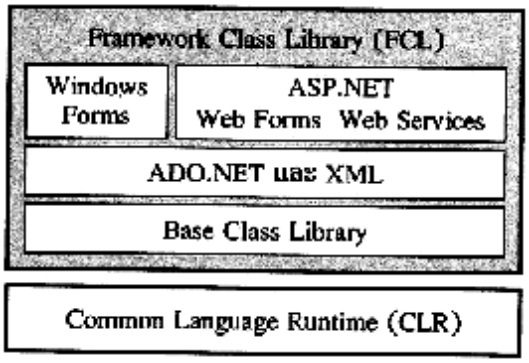
ภาษา Vlsual Baslc มีวิวัฒนาการมาจากภา BASIC (Beginner's All purpose symbolic lnstruction code) ซึ่งในยุคหนึ่งคือภาษาที่ใช้เขียนโปรแกรมบนระบบปฏิบัติการ MS DOS ต่อมามาเป็น visual Basic เพื่อให้เป็นภาษาสำหรับสร้างโปรแกรมที่แสดงผลแบบกราฟิก โดยมีสภาพแวดล้อมในการพัฒนาแบบ Visual programming (จริงๆ แล้วถ้าพูดให้ถูกต้องคือ Visual Basic คือชื่อของภาษาโปรแกรมและเครื่องมือที่ใช้พัฒนาที่ร่วมอยู่ในชุดเดียวกัน) ภาษา Visual Basic เริ่มเป็นทีรู้จักแพร่หลายในเวอร์ชั่น 3 และหลังจากนั้นเป็นต้นมา Visual Basic ก็ถือได้ว่าเป็นภาษาที่มีคนใช้งานมากที่สุดในโลกภาษาหนึ่ง ด้วยจุดเด่นคือสามารถใช้สร้างโปรแกรมได้อย่างสะดวกรวดเร็ว และมี productivity สูงกว่าภาษาอื่นๆ หมายความว่าในเวลาเท่ากัน ถ้าเขียนด้วย visual Basic จะได้งานมากกว่า แต่จุดอ่อนของ visual Basic ในยุคนั้นก็คือโปรแกรมที่เขียนขึ้นมาจะทำงานได้ช้า (แม้ว่าในเวอร์ชั่นหลังๆจะได้รับการปรับปรุงในเรื่องนี้ไปมากแล้วก็ตาม) อีกทั้งยังไม่สนับสนุนการเขียนโปรแกรมในรูปแบบเชิงวัตถุ (object-oriented programming-oop) อย่างสมบูรณ์

ภาษา Visual Basic ถูกปรับปรุงครั้งใหญ่เมื่อไมโครซอฟท์คิดโครงการ .NET ขึ้นมา และต้องการให้ภาษานี้เป็นทางเลือกหนึ่งสำหรับการพัฒนาโปรแกรมภายใต้ .NET ด้วย โดยใน .NET Framework เวอร์ชั่น 1.0 ไมโครซอฟท์เรียกชื่อภาษาโปรแกรมทีปรับปรุงมาจาก Visual Basic นี้ว่า Vsual Basic.NET (VB.NET) ถึงแม้ว่าคำสั่งส่วนใหญ่ใน visual Basic ยังคงมีให้ใช้ใน Visual Basic.NET และไวยากรณ์ (syrtax) ของทั้งสองภาษาจะเหมือนกัน แต่การเปลี่ยนจาก Visual Basic

มาเป็น Visual Basic.NET ต้องถือว่าเป็นการพลิกหน้ามือเป็นหลังมือ เพราะนอกจาก Visual Basic.NET จะเป็นภาษาที่ใช้สร้าง managed applicatlon (โปรแกรมที่รันอยู่ภายใต้การควบคุมของ .NET Framework) แล้ว ยังสนับสนุนการเขียนโปรแกรมเชิงวัตถุอย่างสมบูรณ์เช่นเดียวกับภาษายอดนิยมอื่นๆ เช่น Java และ c++ รวมทั้งมีคุณสมบัติเพียบพร้อมสำหรับการพัฒนาโปรแกรมในหน่วยงานขนาดใหญ่ดังนั้นในวันนี้ visual Basic (.NET) จึงไมใชภาษาโปรแกรมที่ใครจะมดูถูกได้อีกต่อไป

1. **ส่วนประกอบของ** .NET Framework

.NET Framework ประกอบด้วยส่วนหลักๆ 2 ส่วน ได้แก common Language Runtime (CLR) และ Framework class Library (FCL) ดังภาพที่ 16 ส่วนประกอบของ .NET Framework



ภาพที่ 20 ส่วนประกอบของ .NET Framework

1.1. CLR จะรับผิดชอบในเรื่องการรันโปรแกรม จึงเรียกได้วาเป็น execution engine ของ .NET Framework

1.2. FCL คือส่วนของไลบรารี (Library) ที่ใช้ทำงานด้านต่างๆ เช่นแสดงข้อความบนนน้าจออ่านหรือเขียนไฟล์ ติดต่อกับฐานข้ฐมูล สร้างส่วนติดต่อกับผู้ใช้ (วินโดว์และคอนโทรลต่างๆ) เป็นต้น โดย FCL เป็นไลบรารี่ในรูปแบบที่เรียกว่า คลาสไลบรารี (C1ass library)

2. **การคอมไพล์และรันโปรแกรมภายใต้** .NET Framework

ในการคอมไพล์โปรแกรมภายใต้ .NET Framework นั้นโปรแกรมที่เราเขียนจะไม่ได้ถูกแปลไปเป็นคำสังภาษาเครื่องโดยตรงตามที่อธิบายก่อนหน้านี้ แต่จะถูกแปลไปเป็นคำสั่งภาษา MSIL (Microsoft lntermediate Language) หลังจากนั้นในช่วงรันโปรแกรม โค้ด MSIL เหล่านี้จึงจะถูกแปลไปเป็นคำสังภาษาเครื่องอีกทีหนึ่งโดยความรับผิดชอบของ CLR ใน .NET Framework

การคอมไพล์โค้ดโปรแกรมไปเป็น Intermediate code แทนที่จะเป็นคำสั่งภาษาเครื่องส่งผลให้โปรแกรมที่คอมไพล์แล้วไม่ยึดติดกับซีพียูและฮาร์ดแวร์แบบใดแบบหนึ่ง เรียกว่าไม่ยึดติดหรือไม่ขึ้นกับแพลตฟอร์ม (Platform Independent) โปรแกรมสามารถทำงานได้กับเครื่องทุกแบบที่มี .NET Framework ติดตั้งและทำงานอยู่ อย่างไรก็ตาม ปัจจุบัน .NET Framework ที่ทำงานได้อย่างสมบูรณ์จริงๆ มีแค่บนระบบปฏิบัติการ Windows ของไมโครซอฟท์เท่านั้น

ก่อนหน้าที่จะมี .NET นั้นโปรแกรมที่คอมไพล์แล้วจะเป็นคำสั่งภาษาเครื่องที่ซีพียูสามารถประมวลผลได้โดยตรง การทำงานของโปรแกรมจึงไม่ได้อยู่ภายใต้สภาพแวดล้อมที่มีการควบคุม (เรียกว่า unmanaged applicaton) ซึ่งอาจก่อให้เกิดปัญหาตามมามากมาย เช่นปัญหาในเรื่องการจัดการหน่วยความจำ และปัญหาความไม่ปลอดภัยจากโปรแกรมไมพึงประสงค์ทั้งหลาย (ไวรัส สปายแวร์ ฯลฯ) เป็นต้น

ไมโครคอนโทรเลอร์

ไมโครคอนโทรลเลอร์ (Microcontroller) เป็นอุปกรณ์ไอซี (IC: Integrated Circuit) ที่สามารถโปรแกรมการทำงานได้หลายครั้ง สามารถรับข้อมูลในรูปสัญญาณดิจิตอลเข้าไปทำการประมวลผล แล้วส่งผลลัพธ์ข้อมูลดิจิตอลออกมาเพื่อนำไปใช้งานตามที่ต้องการได้ไมโครคอนโทรลเลอร์ หรืออาจจะเรียกได้ว่าไมโครโพรเซสเซอร์ ชิปเดี่ยว(Single-Chip Microprocessor) เป็นไมโครโพรเซสเซอร์ชนิดหนึ่งเช่นเดียวกับหน่วยประมวลผลกลาง (CPU: Central Processing Unit) ที่ใช้ในคอมพิวเตอร์ แต่ได้รับการพัฒนาแยกออกมาภายหลังเพื่อนำไปใช้ในวงจรทางด้านงานควบคุมคือ แทนที่ในการใช้งานจะต้องต่อวงจรภายนอกต่างๆเพิ่มเติมเช่น เดียวกับไมโครโพรเซสเซอร์ ก็จะทำการรวมวงจรที่จำเป็น เช่น หน่วยความจำ ส่วนอินพุต/เอาท์พุต บางส่วนเข้าไปในตัวไอซีเดียวกัน และเพิ่มวงจรบางอย่างเข้าไปด้วยเพื่อให้มีความสามารถเหมาะกับการใช้ในงานควบคุมเช่น วงจรตั้งเวลาวงจรการสื่อสารอนุกรมเป็นต้น ดังนั้นไมโครคอนโทรลเลอร์สามารถจะทำงานได้เสมือนกับเป็นคอมพิวเตอร์เล็กๆเครื่องหนึ่ง

1. ส่วนประกอบของไมโครคอนโทรลเลอร์

1.1 หน่วยประมวลผล (Central Processor Unit) เป็นส่วนที่ตัดสินใจเกี่ยวกับการทำงานต่างๆ

1.2 หน่วยความจำ (memory) เป็นตัวที่จะเก็บข้อมูลต่างๆที่ต้องใช้ในไมโครคอนโทรลเลอร์ ไม่ว่าจะเป็นหน่วยความจำโปรแกรมหรือหน่วยความจำข้อมูล โดยหน่วยความจำที่ใช้ได้แก่ ROM, EPROM, EEPROM, RAM รวมทั้งหน่วยความจำแบบ FLASH

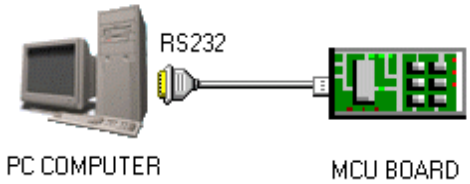
1.3 พอร์ตสัญญาณเข้าและสัญญาณออก (Input & output port) เป็นส่วนที่ใช้ติดต่อกับอุปกรณ์ภายนอก

1.4 คุณสมบัติอื่นๆ ไมโครคอนโทรลเลอร์จะมีฟังก์ชันพิเศษเพิ่มเติม เช่น Timer counter, Analog to Digital Converter, PWM, Analog Comparator, UART/USART สามารถทำงานได้กว้างและใช้อุปกรณ์ต่อร่วมกับภายนอกน้อยมาก และสามารถประมวลคำสั่งได้ใน 1 Clock

สำหรับโครงงานนี้จะใช้ไมโครคอนโทรลเลอร์ 16 บิต dsPIC30F2010 จาก Microchip Technology Inc. ผู้ผลิตไมโครคอนโทรลเลอร์ PIC ซึ่งรู้จักกันเป็นอย่างดีในแวดวงนักพัฒนาระบบไมโครคอนโทรลเลอร์ โดยMicrochip Technology ได้กำหนดชื่ออย่างเป็นทางการสำหรับไมโครคอนโทรลเลอร์อนุกรมใหม่นี้ว่า Digital Signal Controller หรือ DSC นั่นหมายความว่า dsPIC เป็นไมโครคอนโทรลเลอร์ที่ได้รับการออกแบบมาเป็นพิเศษเพื่องานประมวลผลสัญญาณดิจิตอลสำหรับสร้างระบบควบอัตโนมัติที่มีความสามารถสูง

การติดต่อพอร์ตอนุกรม

งานวิจัยนี้จะทำการควบคุมตำแหน่งของมอเตอร์กระแสตรงโดยคอมพิวเตอร์จะส่งข้อมูลจากพอร์ตอนุกรมเพื่อที่ไมโครคอนโทรลเลอร์จะได้นำข้อมูลไปประมวลผล งานวิจัยจึงจะกล่าวถึงการสื่อสารในลักษณะนี้พอสังเขป ภาพที่ 21 การติดต่อพอร์ตอนุกรมกับบอร์ด MCU



ภาพที่ 21 การติดต่อพอร์ตอนุกรมกับบอร์ด MCU

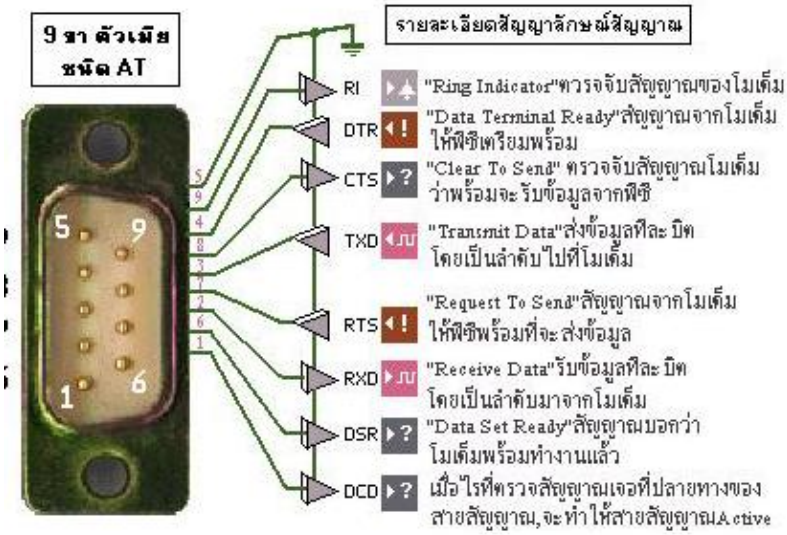
การสื่อสารแบบอนุกรม นับว่ามีความสำคัญ ต่อการใช้งาน ไมโครคอนโทรลเลอร์มาก เพราะสามารถใช้แป้นพิมพ์ และจอภาพของ PC เป็น อินพุต และ เอาต์พุต ในการติดต่อ หรือควบคุมไมโครคอนโทรลเลอร์ ด้วยสัญญาณอย่างน้อย เพียง 3 เส้นเท่านั้น คือ

- สายส่งสัญญาณ TX
- สายรับสัญญาณ RX
- และสาย GND

โดยปกติพอร์ตอนุกรม RS-232C จะสามารถต่อสายได้ยาว 50 ฟุตโดยประมาณ ขึ้นอยู่กับชนิดของ สายสัญญาณ, ระยะทาง, และ ปริมาณ สัญญาณ รบกวน

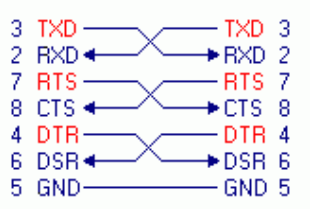
พอร์ตอนุกรมของ PC DB9 ตัวผู้ (Male)

พอร์ตอนุกรมของอุปกรณ์ภายนอก DB9 ตัวเมีย (Female)

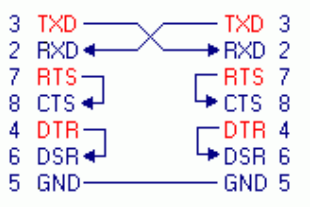


ภาพที่ 22 รายละเอียดของขาสัญญาณพอร์ตอนุกรม

การเชื่อมต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์ด้วยสาย DB9



ภาพที่ 23 การเชื่อมต่ออุปกรณ์ภายนอกผ่าน DB9 แบบ Null modem



ภาพที่ 24 การต่ออุปกรณ์ภายนอกผ่าน DB9 แบบ 3 เส้น

1. **การทำงานของขาสัญญาณ** DB9

TXD เป็นขาที่ใช้ส่งข้อมูล

RXD เป็นขาที่ใช้รับข้อมูล

DTR แสดงสภาวะพอร์ตว่าเปิดใช้งาน, DSR ตรวจสอบว่าพอร์ต ที่ติดต่อด้วย เปิดอยู่หรือไม่

1. เมื่อเปิดพอร์ตอนุกรม ขา DTR จะ ON เพื่อให้อุปกรณ์ได้รับทราบว่าต้องการติดต่อด้วย

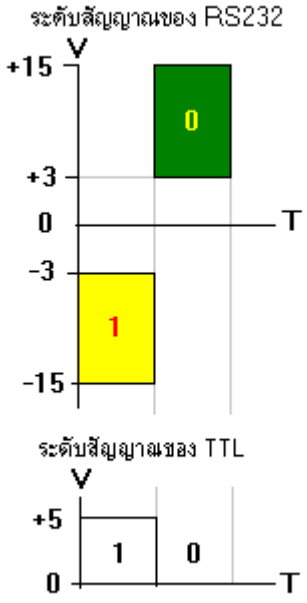
2. ในขณะเดียวกันก็จะตรวจสอบขา DSR ว่าอุปกรณ์พร้อมหรือไม่

RTS แสดงสภาวะพอร์ตว่าต้องการส่งข้อมูล, CTS ตรวจสอบว่าพอร์ตที่ติดต่ออยู่ ต้องการส่งข้อมูลหรือไม่

1. เมื่อต้องการส่งข้อมูลขา RTS จะ ON และจะส่งข้อมูลออกที่ขา TXD เมื่อส่งเสร็จก็จะOFF

2. ในขณะเดียวกันก็จะตรวจสอบขา CTS ว่าอุปกรณ์ต้องการที่จะส่งข้อมูลหรือไม่

GND ขา ground



ภาพที่ 25 ระดับสัญญาณของ RS232C และระดับสัญญาณของ TTL

3. สัญญาณรบกวนที่เกิดขึ้น ในสายนำสัญญาณ มักจะมีแรงดันเป็นบวกเมื่อเทียบกับกราวน์

4. เพื่อป้องกันสัญญาณรบกวนนี้ จึงออกแบบแรงดัน ของโลจิก "1" เป็นลบ คืออยู่ในช่วง -3V ถึง -15V ส่วนแรงดัน ของโลจิก "0" อยู่ในช่วง +3V ถึง +15V

5. และเหตุที่ ระดับสัญญาณ ของ RS232 อยู่ในช่วง +15V ถึง -15V ก็เพื่อให้ต่อสายสัญญาณไปได้ไกลขึ้น ดังนั้นจึงจำเป็นจะต้องมีวงจรเปลี่ยนระดับแรงดันของ RS232 มาเป็นระดับแรงดันของ TTL

2. **อัตราการส่งข้อมูล** (Baud rate)

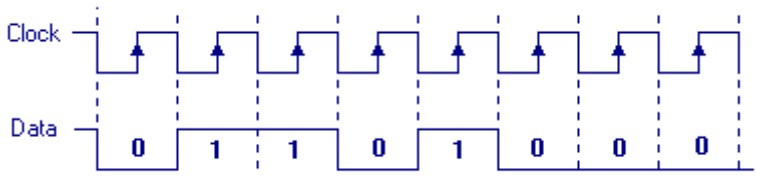
1. คือความเร็วของการรับ-ส่งข้อมูล เป็นจำนวนบิตต่อวินาทีเช่น 300, 1,200, 2,400, 4,800, 9,600, 14,400, 19,200, 38,400, 56,000 เป็นต้น

2. การเลือกอัตราการส่งข้อมูลขึ้นอยู่กับ ชนิดของสายสัญญาณ, ระยะทาง, และปริมาณสัญญาณรบกวน

3. **รูปแบบการสื่อสารแบบอนุกรม**

มีด้วยกันอยู่ 2 แบบ คือแบบซิงโครนัส (Synchronous) และแบบอะซิงโครนัส (Asynchronous)

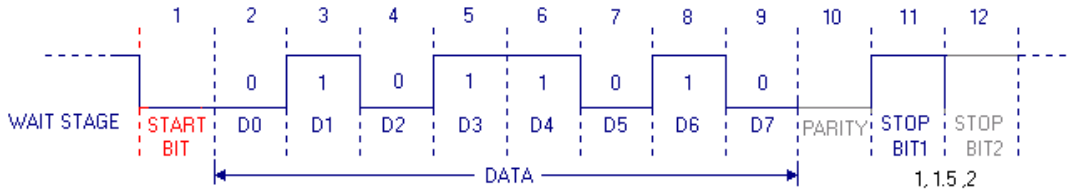
3.1 การสื่อสารแบบซิงโครนัส การรับส่งข้อมูล จะมีสัญญาณนาฬิกา ซึ่งเป็นตัวกำหนดจังหวะเวลา การส่งข้อมูล ร่วมอยู่ด้วยอีกเส้นหนึ่ง ใช้คู่กับสัญญาณข้อมูล ตัวอย่างเช่น การส่งสัญญาณจากคีย์บอร์ด



ภาพที่ 26 การสื่อสารแบบซิงโครนัส (Synchronous)

3.2. การสื่อสารแบบอะซิงโครนัส การรับส่งข้อมูล โดยที่ไม่จำเป็นต้อง มีสัญญาณนาฬิการ่วมด้วย แต่จะใช้ให้ตัวส่ง และตัวรับ มีอัตราส่งข้อมูล ที่เท่ากัน รูปแบบข้อมูลแบบอะซิงโครนัส ประกอบด้วย 4 ส่วนคือ

1. บิตเริ่มต้น (Start bit) มีขนาด1 บิต
2. บิตข้อมูล (Data) มีขนาด 5,6,7 หรือ 8 บิต
3. บิตตรวจสอบพาริตี้ (Parity bit) มีขนาด 1 บิตหรือไม่มี
4. บิตหยุด (Stop bit) มีขนาด 1, 1.5, 2 บิต



ภาพที่ 27 การสื่อสารแบบอะซิงโครนัส (Asynchronous)

5. เมื่อไม่มีการส่งข้อมูล ขา data จะมีสถานะเป็นโลจิก "1" หรือ สถานะหยุดรอ (Waiting stage)

6.เมื่อเริ่มต้นส่งข้อมูลจะให้ขา data เป็นโลจิก "0" เป็นจำนวน 1 บิต เรียกว่าบิตเริ่มต้น (Start bit)

7. จากนั้นก็จะเริ่มต้นส่งข้อมูล โดยส่งบิตต่ำไปก่อน (LSB)

8. แล้วตามด้วยพาริตี้บิต (จะมีหรือไม่ก็ได้ ขึ้นอยู่กับการติดตั้งค่าของทั้งสองฝ่าย)

9. สุดท้ายตามด้วยโลจิก "1" อย่างน้อย 1 บิต (มีขนาด 1, 1.5, หรือ 2 บิต) เพื่อแสดงว่าสิ้นสุดข้อมูล

4. การรับและส่งข้อมูลแบบอนุกรมยังแบ่งออกเป็นลักษณะการใช้งานได้ 3 แบบคือ

1. แบบซิมเพลกซ์ (Simplex) เป็นการส่ง หรือรับข้อมูล แบบทิศทางเดียว เท่านั้น

2. แบบฮาล์ฟดูเพลกซ์ (Half Duplex) เป็นการส่งและรับข้อมูลแบบสลับกันคือเมื่อด้านหนึ่งส่ง อีกด้านหนึ่ง เป็นฝ่ายรับ สลับกัน ไม่สามารถรับ-ส่งในเวลาเดียวกันได้

3. แบบฟลูดูเพลกซ์ (Full Duplex) สามารถรับ-ส่งข้อมูลในเวลาเดียวกันได้

ICD2 (In-Circuit Debugger & Programmer for dsPIC and PIC microcontroller)

เป็นเครื่องมือสำหรับทำการดีบักและโปรแกรมสำหรับไมโครคอนโทรลเลอร์ PIC และ dsPIC มีความสามารถทัดเทียมกับ ICD2 ของ Microchip

1. **คุณสมบัติทางเทคนิค**

1. เชื่อมต่อกับคอมพิวเตอร์ทางพอร์ต USB

2. สามารถดีบักเพื่อตรวจสอบการทำงานของไมโครคอนโทรลเลอร์ PIC แบบแฟลช และ dsPIC ได้แบบรีลไทม์หรือเวลาจริง

3. สามารถใช้งานเป็นเครื่องโปรแกรมไมโครคอนโทรลเลอร์ PIC แบบแฟลชและ dsPIC ได้ โดยมีฟังก์ชั่นการทำงานที่สมบูรณ์ทั้งอ่าน เขียน ตรวจสอบ และป้องกันการอ่านได้

4. ใช้งานร่วมกับ MPLAB IDE V6.2 ขึ้นไป

5. มีฟังก์ชั่นทั้งหมดที่ต้องใช้ในการดีบัก ไม่ว่าจะเป็นการกำหนดbreakpoint การรันโปรแกรมแบบ Single step หรือรันโปรแกรมแบบเต็ม พร้อมทั้งแสดงผลการทำงานผ่านทางหน้าต่างของซอฟต์แวร์ด้วย

6. สามารถอัปเกรดเพื่อเพิ่มเติมเบอร์ของไมโครคอนโทรลเลอร์ได้ด้วยตนเองผ่านทางคอมพิวเตอร์

7. มี LED แสดงผลการทำงาน Power, Busy และ Error

8. สามารถอ่านและเขียนข้อมูลในหน่วยความจำข้อมูลอีอีพรอมภายในไมโครคอนโทรลเลอร์ที่ต้องการได้

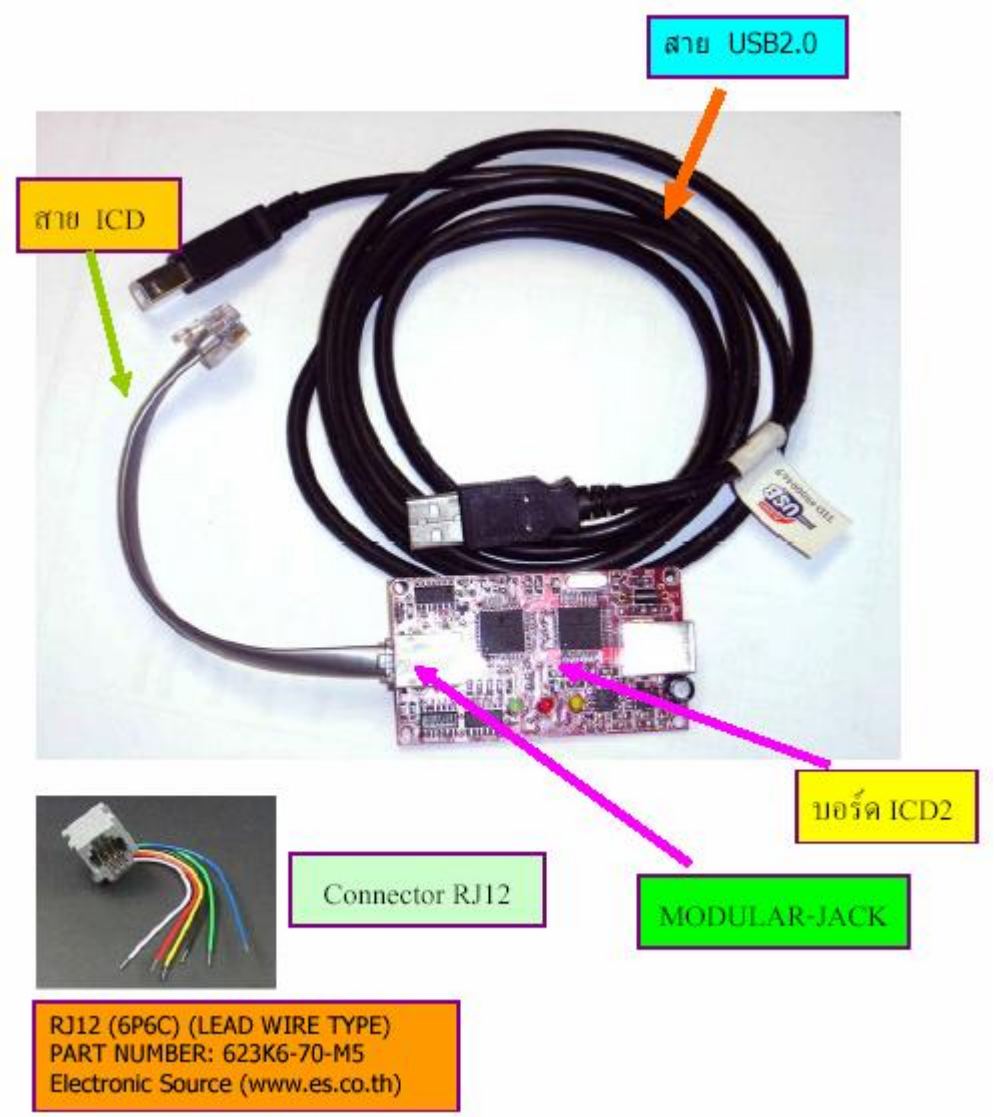
9. สามารถโปรแกรมบิตกำหนดการทำงานทางฮาร์ดแวร์หรือ configuration bits ได้

10. ใช้แจ๊กแบบโมดูล่าร์ 6 ขา (modular jack) สำหรับต่อสายสัญญาณเพื่อติดต่อกับบอร์ดเป้าหมาย โดยมีการจัดขาตรงกับแจ๊ก ICD ของ ICD2 ของ Microchip ทุกประการ

11. ใช้งานได้กับไมโครคอนโทรลเลอร์ PIC แบบแฟลช (อนุกรม PIC12F/16F/18F) และ dsPIC ทุกเบอร์ที่รองรับการดีบักในวงจรและการโปรแกรมแบบ ICSP และสามารถเพิ่มเติมได้ในอนาคต

2. ในชุดประกอบด้วย

1. ICD2
2. สาย USB2.0
3. สาย ICD
4. MODULAR-JACK
5. Connector RJ12
6. CDคู่มือใช้งาน



ภาพที่ 28 ส่วนประกอบของ ICD2

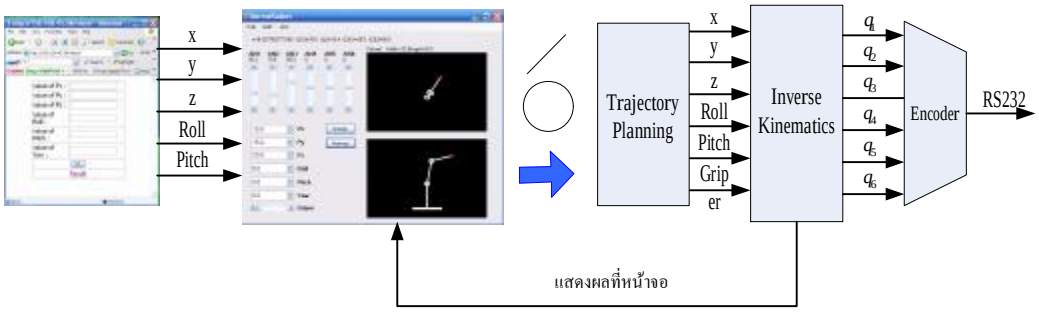
อุปกรณ์และวิธีการ

อุปกรณ์

1. คอมพิวเตอร์ 2 ชุด
2. แขนหุ่นยนต์ 5DOF 1 ชุด
3. บอร์ด dsPIC30F2010 1 ชุด
4. Power supply 12V 1 ชุด
5. ICD2 1 ชุด
6. โปรแกรม Visual Basic 2005
7. โปรแกรม MPLAB IDE V7.62
8. โปรแกรม MPLAB C30 V3.0
9. โปรแกรม Potel 99 SE
10. โปรแกรม Apache2.2

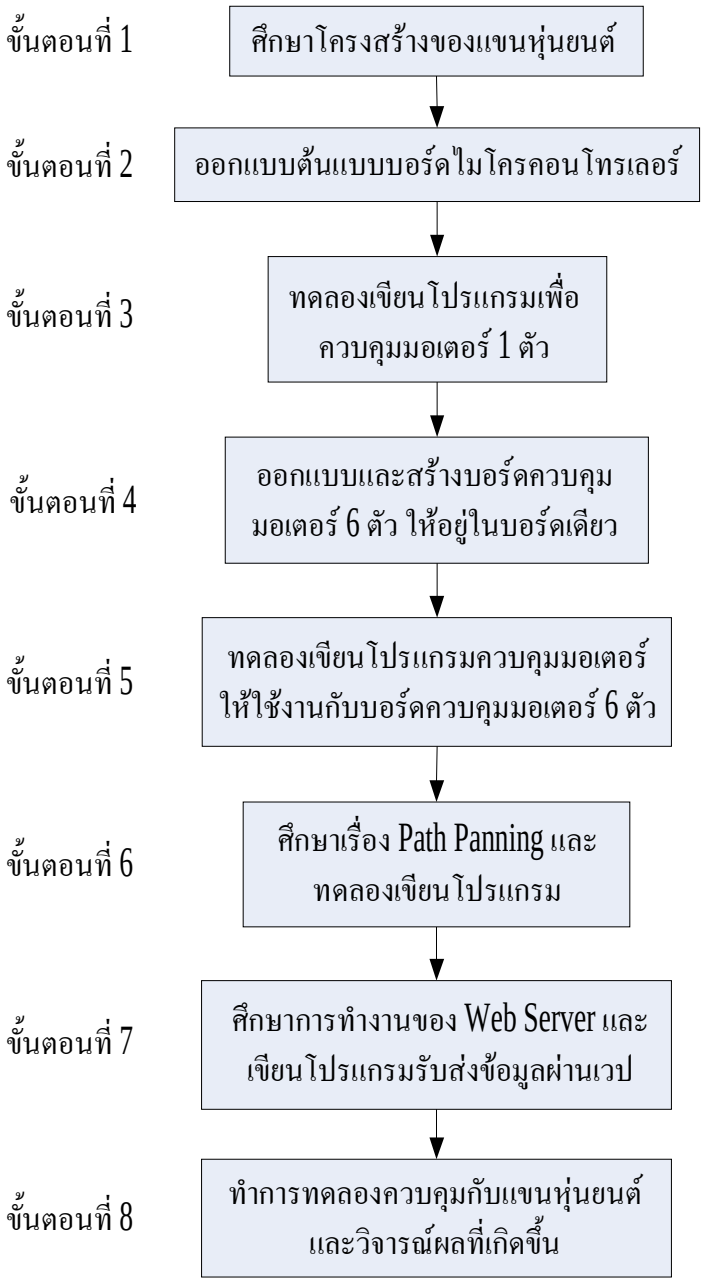
วิธีการ

วิธีการทำงานของการทำงานของการควบคุมแขนหุ่นยนต์ผ่านอินเตอร์เน็ทดูตามภาพที่ 29 นั้นผู้ใช้สามารถที่จะควบคุมได้จาก 2 แหล่ง คือการควบคุมผ่านอินเตอร์เน็ท ผู้ใช้จะสามารถใช้เครื่องคอมพิวเคอร์เครื่องไหนก็ได้ที่ต่ออินเตอร์เน็ทอยู่โดยใช้โปรแกรม Internet Explorer โดยพิมพ์ที่ช่อง Address http://158.108.45.58/remode ก็จะปรากฏหน้าเวปขึ้นมาให้ป้อนค่าตำแหน่งและทิศทางที่ต้องการให้แขนหุ่นยนต์เคลื่อนที่ไปแล้วกด OK แขนหุ่นยนต์ก็จะเลื่อนที่ ไปตามตำแหน่งที่ต้องการตามภาพที่ 29 โฟว์ชาดการทำงานของการควบคุมแขนหุ่นยนต์ผ่านอินเตอร์เน็ท



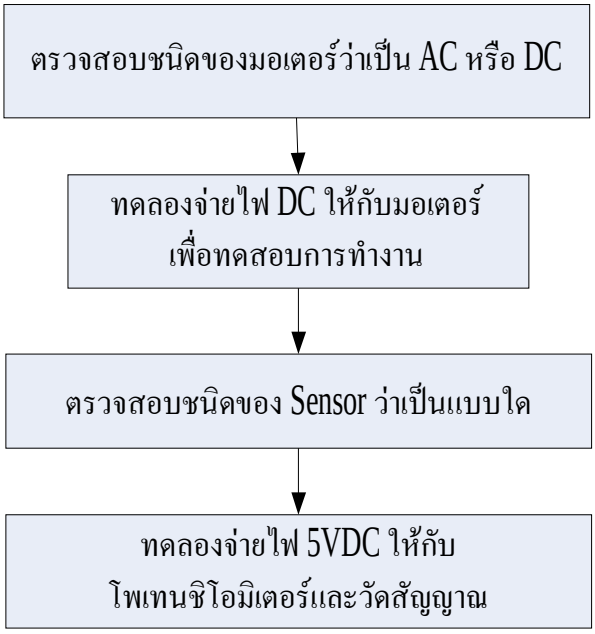
ภาพที่ 29 โฟว์ชาดการทำงานของการควบคุมแขนหุ่นยนต์ผ่านอินเตอร์เน็ท

การควบคุมแบบที่สองคือใช้โปรแกรม ServoRobot โปรแกรมนี้จะอยู่ที่เครื่องคอมพิวเตอร์ Server ซึ่งต่ออยู่กับแขนหุ่นยนต์โดยตรง ซึ่งมีความสามรถในการควบคุมที่มากกว่าการควบคุมผ่านอินเตอร์เน็ทคือ สามารถสร้างเส้นทางเดินให้กับตำแหน่งปลายแขนเป็นวงกลมหรือเส้นตรงได้ และสามารถควบคุมมุมของมอเตอร์แต่ละตัวโดยตรงได้ด้วย ขั้นตอนการทำงานสามารถแบ่ง ได้เป็น 8 ขั้นตอนตามภาพที่ 30 โฟลชาร์จแสดงขั้นตอนการทำงาน



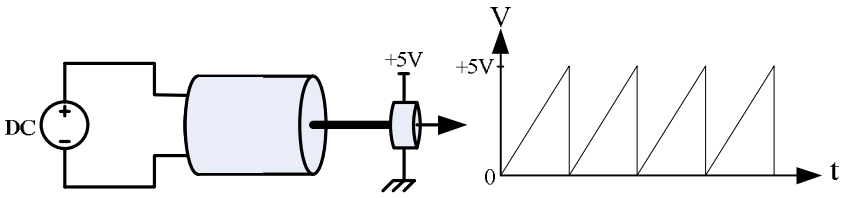
ภาพที่ 30 โฟลชาร์จแสดงขั้นตอนการทำงาน

ขั้นตอนที่ 1. **ศึกษาโครงสร้างของแขนหุ่นยนต์**



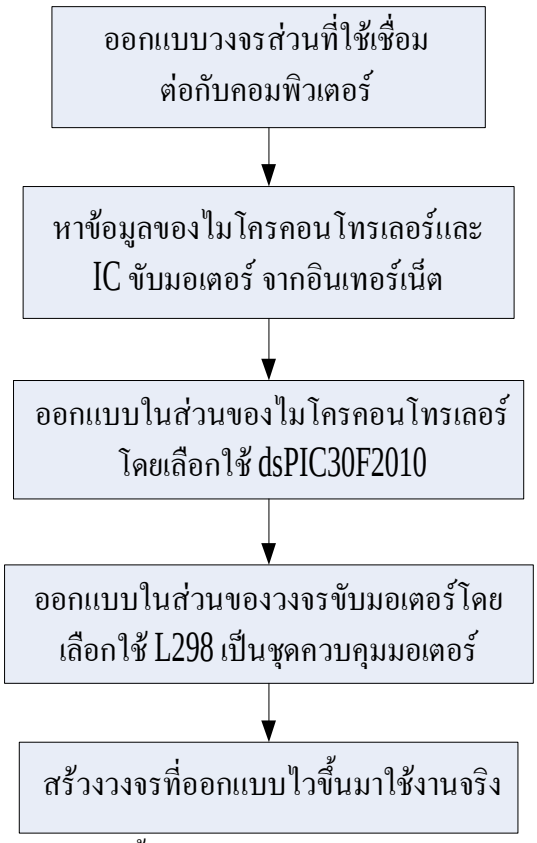
ภาพที่ 31 โฟลชาร์จแสดงขั้นตอนการศึกษาสร้างของโครงสร้างแขนหุ่นยนต์

หลังจากที่ได้แขนหุ่นยนต์มาแล้วก็ต้องมาศึกษาสร้างโครงสร้างและส่วนประกอบต่างๆ ของแขนหุ่นยนต์ ตามขั้นตอนในภาพที่ 31 เช่น ศึกษาข้อต่อที่ใช้เป็นข้อต่อแบบหมุนทั้งหมดซึ่งมีทั้งหมด 5 DOF ในส่วนของมอเตอร์ที่ใช้เป็นชนิด DC มอเตอร์และมีชุดเฟืองทดกำลังในตัว หลังจากที่ทราบว่าเป็น DC มอเตอร์ก็ทดลองจ่ายไฟ DC ให้กับมอเตอร์โดยใช้แรงดัน DC เหมือนกับแหล่งจ่ายของเดิมคือ 12 V DC เมื่อตรวจสอบการทำงานของมอเตอร์ทั้ง 6 ตัวแล้วทุกทำงานได้ปรกติ ก็ผ่านไปเรื่องของเซ็นเซอร์หุนยนต์ตัวนี้ใช้โพเทนชิออมิเตอร์อ่านค่ามุมของมอเตอร์เมื่อทดสอบการทำงามอเตอร์ผ่านแล้วก็ทดลอง โพเทนชิออมิเตอร์โดยต่อไฟ 5 V DC ให้กับโพเทนชิออมิเตอร์แล้วนำสัญญาณเอาพุดมาเปรียบเทียบหาความสัมพันธุ์กับมุมตามภาพที่ 32 การทดสอบมอเตอร์และโพเทนชิออมิเตอร์ ผลที่ได้คือมุมที่ได้ 0-360 องศา จะแปรผันตรงกับแรงดับเอาพุต 0-5 V



ภาพที่ 32 การทดสอบมอเตอร์และโพเทนชิออมิเตอร์

ขั้นตอนที่ 2. ออกแบบต้นแบบบอร์ดไมโครคอนโทรเลอร์

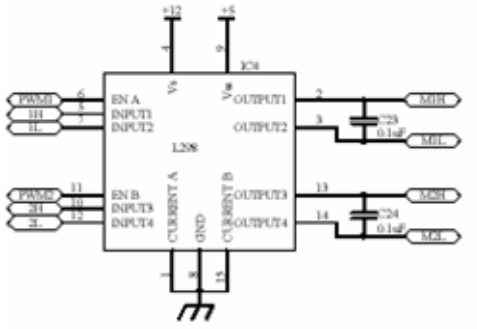


ภาพที่ 33 โฟลชาร์จแสดงขั้นตอนการแบบต้นแบบบอร์ดไมโครคอนโทรเลอร์

ในภาพที่ 33 เป็นขั้นตอนการออกแบบต้นแบบบอร์ดไมโครคอนโทรเลอร์ ที่ใช้สร้างวงจรต้นแบบบอร์ดควบคุมมอเตอร์เพื่อนำมาใช้ศึกษาและทดลองการทำงานของไมโครคอนโทรเลอร์เบอร์ dsPIC30F2010 ก่อนที่จะสร้างเป็นระบบควบคุมที่ใหญ่กว่านี้ เริ่มแรกต้องเริ่มจากกำหนดสเป็คของบอร์ดคล่าวๆ ก่อนคือ

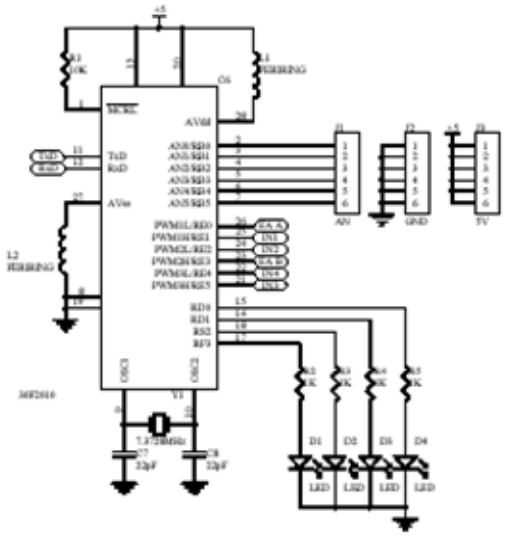
1. สามารถควบคุมมอเตอร์ DC ได้อย่างน้อย 1 ตัว
2. สามารถกลับทางหมุนได้
3. รองรับกระแสไฟได้อย่างน้อย 1 A
4. สามารถปรับความเร็วของมอเตอร์ได้
5. สามารถอ่านค่าสัญญาณไฟฟ้าที่ออกมาจากโพเทนชิออมิเตอร์ได้
6. เชื่อมต่อกับคอมพิวเตอร์ผ่านพอร์ตอนุกรมได้ (RS232)

ในขั้นแรกพิจารณาจากข้อ 1 – 3 เป็นวงจรตัวเดียวกันซึ่งหลังจากหาข้อมูลมาก็จะสรุปได้ว่าใช้เป็นแบบ IC ตัวเดียว IC เบอร์ L298 ซึ่งเป็น IC สำหรับควบคุมมอเตอร์โดยเฉพาะตามภาพที่ 34 วงจรควบคุมมอเตอร์ด้วย L298 ซึ่ง IC 1 ตัวประกอบไปด้วยวงจร half bridge จำนวณ 4 โมดูล แต่ละโมดูลจะรองรับกระแสได้ 2 A ถ้าต้องการควบคุมมอเตอร์แบบกลับทางหมุนได้ก็จะนำมาต่อเป็นวงจรแบบ full bridge ได้ 2 โมดูล ดังนั้นวงจรตามภาพที่ 34 จะสามารถใช้ควบคุมมอเตอร์ DC ได้ 2 ตัวแบบกลับทางหมุน



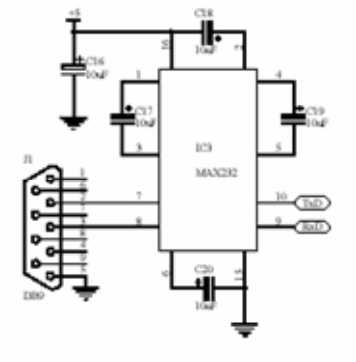
ภาพที่ 34 วงจรควบคุมมอเตอร์ด้วย L298 ที่ใช้งานจริง

เงื่อนไขข้อที่ 4 และ 5 ส่วนนี้ได้เลือกใช้ไมโครคอนโทรเลอร์เบอร์ dsPIC30F2010 เพราะว่า โทรเลอร์เบอร์ เบอร์นี้จะมีโมดูลสร้างสัญญาณ PWM อยู่ 3 โมดูล และมีพล๊อต ADC อยู่ 6 ช่องด้วยกัน ตามภาพที่ 35 วงจรของไมโครคอนโทรเลอร์ dsPIC30F2010 ที่ใช้งานจริง

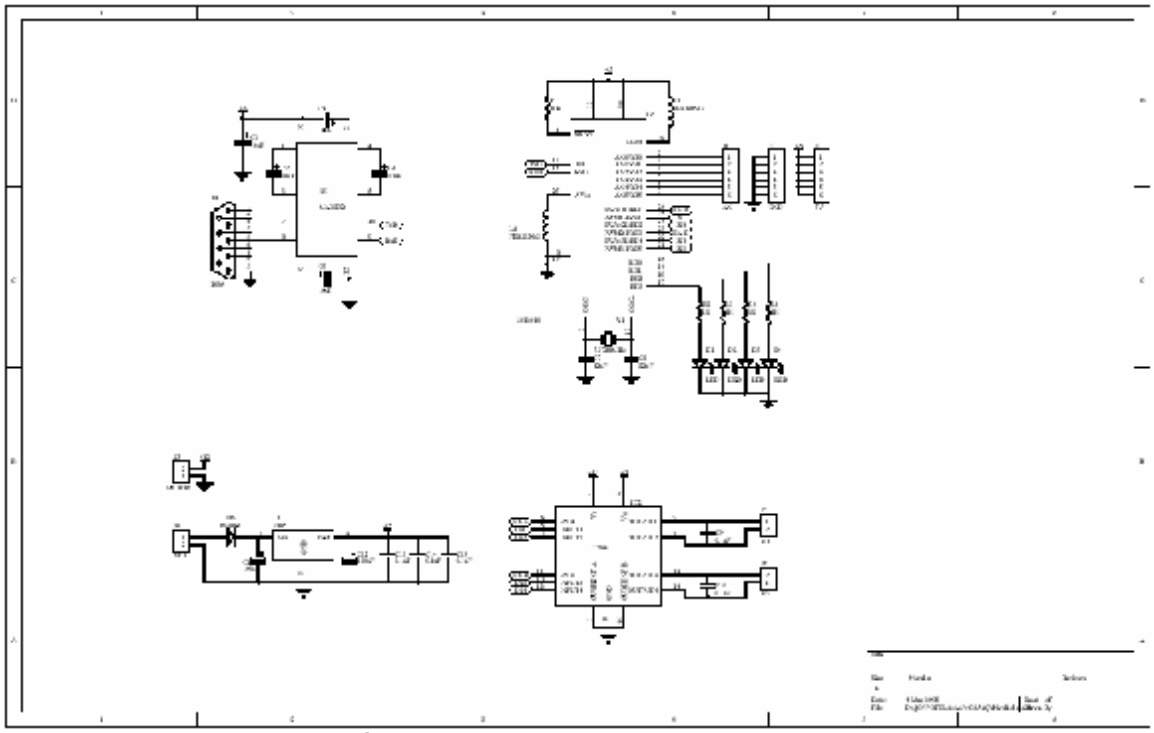


ภาพที่ 35 วงจรของไมโครคอนโทรเลอร์ dsPIC30F2010 ที่ใช้งานจริง

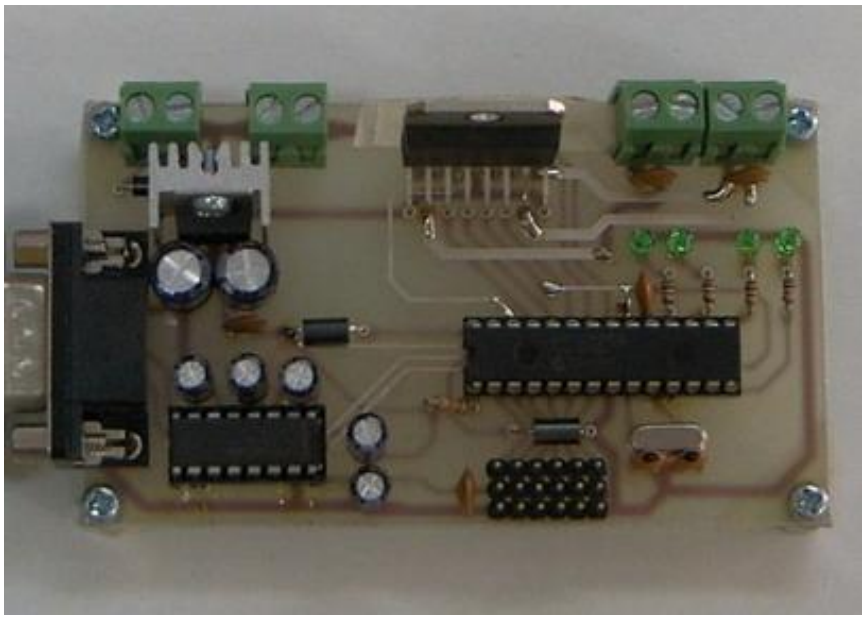
เงื่อนไขข้อสุดท้าย สามารถเชื่อมต่อกับคอมพิวเตอร์ผ่านพอร์ตอนุกรมได้ เนื่องจากพล็อต RS232 นั้นจะใช้ระดับของสัญญาณเป็น $\pm 15V$ แต่ในไมโครคอนโทรเลอร์จะใช้ 0-5V จึงต้องมีอุปกรณ์ตัวหนึ่งมาทำหน้าที่เปลี่ยนระดับของสัญญาณ ก็จะเลือกใช้เบอร์ที่นิยมใช้กันอย่างแพรหลายคือเบอร์ MAX232 ตามภาพที่ 36 วงจรMAX232 ที่ใช้งานจริงเมื่อได้วงจรครบทุกส่วนแล้วก็จะได้วงจรที่จะนำไปใช้งานจริงตามภาพที่ 37 วงจรต้นแบบบอร์ดควบคุมมอเตอร์ ในภาพที่ 38 เป็นหน้าตาของต้นแบบบอร์ดควบคุมมอเตอร์ ที่สร้างเสร็จแล้ว



ภาพที่ 36 วงจรMAX232 ที่ใช้งานจริง

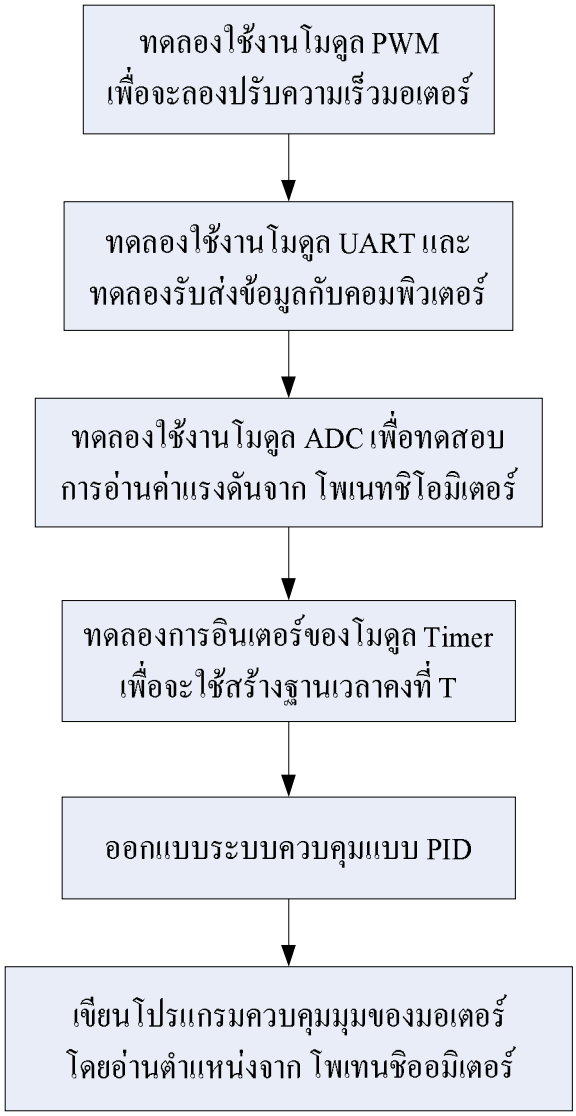


ภาพที่ 37 วงจรต้นแบบบอร์ดควบคุมมอเตอร์



ภาพที่ 38 วงจรต้นแบบบอร์ดควบคุมมอเตอร์ ที่สร้างเสร็จแล้ว

ขั้นตอนที่ 3. **ทดลองเขียนโปรแกรมเพื่อควบคุมมอเตอร์** 1 **ตัว**



ภาพที่ 39 โฟลชาร์จแสดงขั้นตอนการทดลองเขียนโปรแกรมเพื่อควบคุมมอเตอร์ 1 ตัว

1. **การใช้งานโมดูล** MCPWM **ใน** dsPIC30F2010 **ควบคุมมอเตอร์**

อีกหนึ่งโมดูลฟังก์ชั่นพิเศษที่ dsPIC มีและน่าสนใจอย่างยิ่งคือ โมดูลสร้างสัญญาณ PWM เพื่อควบคุมมอเตอร์หรือเรียกว่า โมดูล MCPWM (Motor Control PWM) ใน dsPIC บรรจุโมดูลนี้ไว้ตั้งแต่ 6 ถึง 8 ช่อง สำหรับ dsPIC30F2010 มี 6 ช่อง จึงสามารถขับมอเตอร์แบบเฟสเดี่ยวได้ 3 ตัว

และมอเตอร์ 3 เฟสได้ 1 ตัว ดังนั้นจึงเหมาะอย่างยิ่งที่จะนำไปใช้ควบคุมอินดักชั่นมอเตอร์3เฟส, สวิตช์รีลักแตนซ์มอเตอร์ (SR) มอเตอร์แบบไม่มีแปรงถานหรือบรัชเลสมอเตอร์ (BLDC) และในระบบเครื่องสำรองไฟฉุกเฉินหรือ UPS (Un-interrupted power supply)

1.1 คุณสมบัติโดยสรุปของโมดูล MCPWM

1. ความละเอียดของสัญญาณ PWM ที่สร้างขึ้นเท่ากับ Tcy/2

2. ในโมดูล MCPWM 1 ชุด มี 2 เอาต์พุต ใน dsPIC30F2010 มีโมดูล 3 ชุด จึงมีทั้งสิ้น 6 ช่อง

3. สามารถใช้งานเอาต์พุตของโมดูล MCPWM เรียกกันอย่างอิสระและร่วมกันเมื่อทำงานในแบบรวมกันหรือคอมพลีเมนตารีสามารถกำหนดค่าเวลาวิกฤต(deadtime)เพื่อช่วยให้การขับมอเตอร์ 3 เฟสเป็นไปอย่างมีประสิทธิภาพและ สามารถเลือกโหมดเอาต์พุตได้ 4 โหมด

3.1.โหมดปรับขอบสัญญาณ (Edge aligned mode)

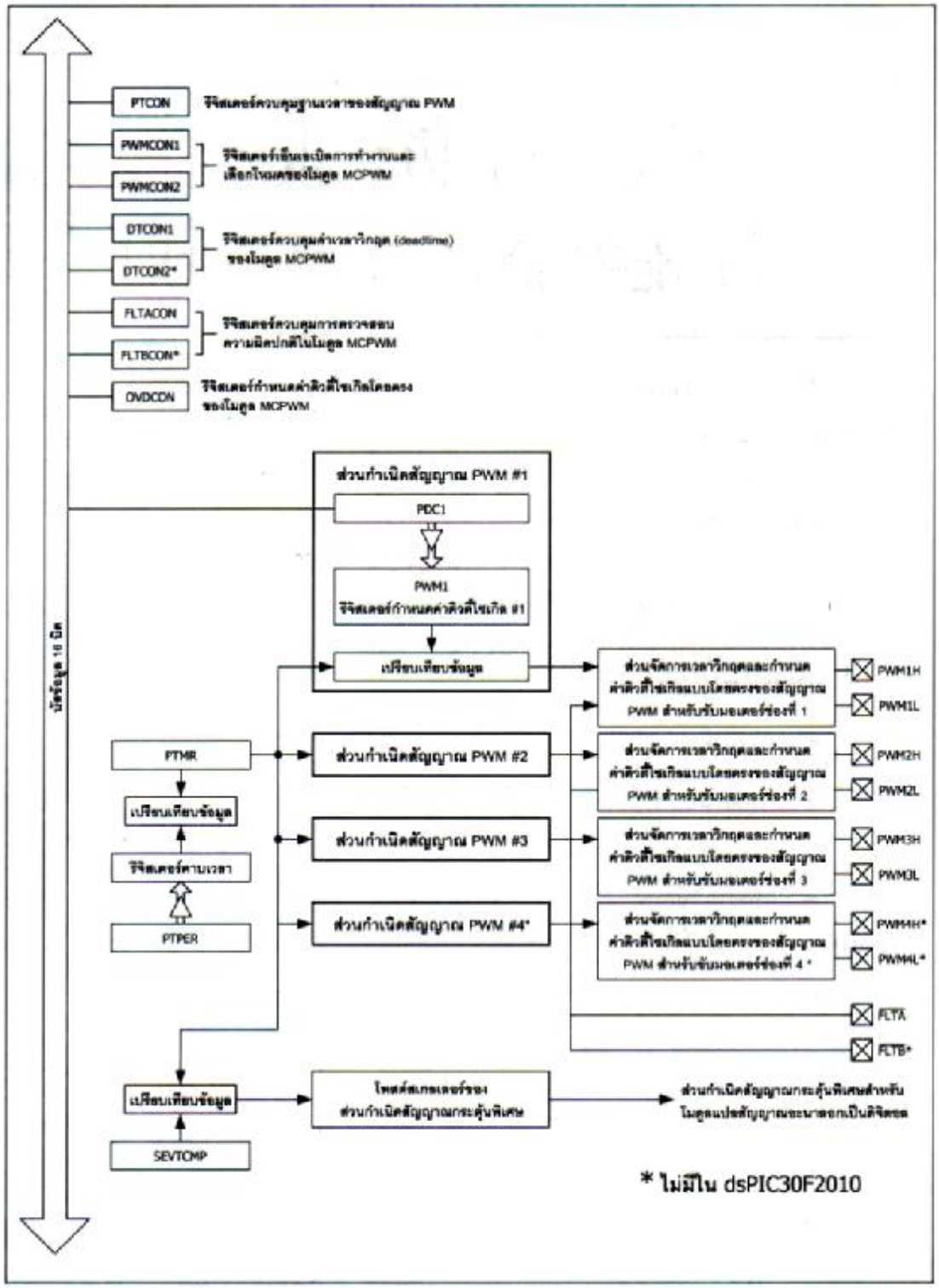
3.2.โหมดสัญญาณเดี่ยว (Single event mode)

3.3. โหมดปรับสัญญาณกึ่งกลาง (Center aligned mode)

3.4. โหมดปรับสัญญาณกึ่งกลางพร้อมปรับปรุงค่า (Center aligned mode with double updates)

4. มีอินพุตสำหรับตรวจจับความผิดพลาดในการทำงาน (FAULT) แบบโปรแกรมได้

5. สามารถสร้างสัญญาณกระตุ้นส่งไปยังโมดูลแปลงสัญญาณอะนาลอกเป็นดิจิตอลเพื่อกำหนดจังหวะการทำงานให้สัมพันธ์กัน



ภาพที่ 40 แสดงไดอะแกรมการทำงานของโมดูล MCPWM ในไมโครคอนโทรลเลอร์ dsPIC

ในภาพที่ 40 แสดงไดอะแกรมรมการทำงานของโมดูล MCPWM ส่วนประกอบหลักของโมดูลนี้คือส่วนกำเนิดสัญญาณ PWM 4ชุด โดยใน dsPIC30F2010 จะมี 3 ชุด ได้ค่าฐานเวลาจากรีจิสเตอร์ PTMRและ PTPER ส่วนค่าดิวตี้ไซเกิลของสัญญาณ PWM ในโมดูล MCPWM นี้สามารถกำหนดได้จากรีจีสเตอร์ดิวตี้ไซเกิลในส่วนกำเนิดสัญญาณ PWM แต่ละส่วนที่เป็นอิสระต่อกัน นอกจากนั้นยังสามารถกำหนดการทำงานของขาพอร์ตเอาต์พุตของโมดูล MCPWM โดยตรงผ่านทางรีจิสเตอร์ OVDCON

ส่วนกำเนิดสัญญาณ PWM สำหรับควบคุมมอเตอร์แต่ละชุดในโมดูล MCPWM สามารถกำหนดให้ทำงานเรียกจากกันเป็นอิสระ (Independent mode) หรือทำงานร่วมกัน (Complementary Mode) เพื่อขับมอเตอร์ 3 เฟสได้ โดยกำหนดผ่านรีจิสเตอร์ PWMCON1 และ PWMCON2 และเมื่อกำหนดให้ทำงานร่วมกันจะต้องมีการจัดการสัญญาณเพื่อให้มอเตอร์ในแต่ละเฟสสามรถทำงานได้อย่างต่อเนื่อง นั่นคือการจัดการค่าเวลาหน่วงเฟส (Datetime control) โดยใช้รีจิสเตอร์ DTCON1 และDTCON2 (ไม่มีใน dsPIC30F2010)

สัญญาณออกจากโมดูล MCPWM จะมีขาพอร์ต 2 ขาต่อช่องนั้นคือ ขาเอาต์พุตด้านแรงดันสูง - PWMxHและขาเอาต์พุตด้านแรงดันต่ำ - PWMxL (x คือหมายเลขของช่องเอาต์พุตมี 4 ค่าคือ 1-4 โดยใน dsPIC30F2010 มีเพียง 1-3) หรือเรียกว่า คู่เอาต์พุต นอกจากนั้นยังสามารถส่งสัญญาณเอาต์พุต ผ่านโพสต์สเกลเลอร์เพื่อสร้างเป็นสัญญาณกระต้นพิเศษให้แก่โมดูลแปลงสัญญาณอะนาลอกเป็น ดิจิตอล (ADC) ด้วยเพื่อให้โมดูล MCPWM สามารถทำงานสัมพันธ์กับโมดูล ADC ได้ด้วย

นอกจากนั้นในโมดูล MCPWM ยังมีอินพุตสำหรับรับสัญญาณตรวจสอบความผิดปกติหรือFAULT เพื่อป้องกันไม่ให้โมดูล MCPWM ทำงานผิดพลาดหรือเสียหายเมื่อเกิดความผิดปกติขึ้นในโมดูล MCPWM โดยในส่วนนี้มีพอร์ตอินพุตสำหรับรับสัญญาณ 2 ขา คือ FLTA และ FLTB สำหรับใน dsPIC30F2010 จะมีเพียงขา เดียวคือ FLTAโดยการทำงานในส่วนนี้ได้รับการควบคุมจากรีจิสเตอร์FLTACON สำหรับส่วนตรวจสอบความผิดปกติชุด A และ FLTBCON สำหรับส่วนตรวจสอบความผิดปกติชุด B (ไม่มีใน dsPIC30F2010)

1.2 รีจิสเตอร์ที่ใช้งานในโมดล MCPWM ในโมดูล MCPWM ของไมโครคอนโทรลเลอร์ dsPIC มีรีจิสเตอร์ที่ใช้ควบคุมและกำหนดค่าในการทำงานรวม 14 ตัว อันประกอบด้วย

PTCON รีจิสเตอร์ควบคุมฐานเวลาในการกำเนิดสัญญาณ PWM
PTMR รีจิสเตอร์กำหนดค่าฐานเวลาของการกำเนิดสัญญาณ PWM
PTPER รีจิสเตอร์กำหนดคาบเวลาของฐานเวลาสำหรับการกำเนิดสัญญาณ PWM
SEVTCMP รีจิสเตอร์เปรียบเทียบค่า
PWMCON1 รีจิสเตอร์ควบคม PWM #1
PWMCON2 รีจิสเตอร์ควบคุม PWM #2
DTCON1 รีจิสเตอร์ควบคุมค่าเวลาวิกฤตหรือ Dead Time #1
DTCON2 รีจิสเตอร์ควบคุมค่าเวลาวิกฤตหรือ Dead Time #2
FLTACON รีจิสเตอร์ควบคุมการตรวจจับความผิดปกติของการขับมอเตอร์ชุด A
FLTBCON รีจิสเตอร์ควบคุมการตรวจจับความผิดปกติของการขับมอเตอร์ชุด B
PDC1 รีจิสเตอร์กำหนดค่าดิวตี้ไซเกิลของโมดูลกำเนิดสัญญาณ PWM ชุดที่ 1
PDC2 รีจิสเตอร์กำหนดค่าดิวตี้ไซเกิลของโมดูลกำเนิดสัญญาณ PWM ชุดที่ 2
PDC3 รีจิสเตอร์กำหนดค่าดิวตี้ไซเกิลของโมดูลกำเนิดสัญญาณ PWM ชุดที่ 3
PDC4 รีจิสเตอร์กำหนดค่าดิวตี้ไซเกิลของโมดูลกำเนิดสัญญาณ PWM ชุดที่ 4

ใน dsPIC30F2010 มีใช้ทั้งสิ้น 12 ตัว ที่ไม่มีคือ รีจิสเตอร์ FLTBCONเนื่องจากโมดูล MCPWM ของ dsPIC30F2010 มีส่วนตรวจจับความผิดปกติหรือ FAULT เพียง 1 ชุดคือชุด A และรีจิสเตอร์ PDC4 เนื่องจาก dsPIC30F2010 มีโมดูล MCPWM 3 ชุด

1.2.1. PTCON (PWM Time Base controI Register): รีจิสเตอร์ควบคุมฐานเวลาในการกำเนิดสัญญาณ PWM

บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
PTEN	-	PTSIDL	-	-	-	-	-
R/W -0	U -0	R/W -0	U -0	U -0	U -0	U -0	U -0
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
PTOPS3	PTOPS2	PTOPS1	PTOPS0	PTCKPS1	PTCKPS0	PTMOD1	PTMOD0
R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0

บิต 15 - PTEN (PWM Time Base Timer Enable bit): บิตเอ็นเอเบิลการทำงานของฐานกลาง PWM

“0” = ปิดฐานเวลา
“1” = เปิดให้ฐานเวลาของสัญญาณ PWM ทำงาน

บิต 14, 12 ถึง 8 ไม่ใช้งาน กำหนดเป็น “0”

บิต 13 - PTSIDL (PWM Time Base stop in IDLE Mode bit): บิตเลือกการทำงานของส่วนฐานเวลา PWM ในโหมดประหยัดพลังงาน

“0” = ฐานเวลา PWM ยังคงทำงานต่อไป แม้ว่าซีพียูจะเข้าสู่โหมดประหยัดพลังงานก็ตาม
“1” = ฐานเวลา PWM หยุดทำงานทันทีที่เข้าสู่โหมดประหยัดพลังงาน

บิต 7 ถึง 4 - PTOPS3 ถึง PTOPS0 (PWM Time Base output postscale select bits): บิตเลือกการลดทอนทางเอาต์พุตของฐานเวลา PWM

“0000” = 1:1
“0001” = 1:2 ผ่านโพสต์สเกลเลอร์
.....
“1111” = 1:16 ผ่านโพสต์สเกลเลอร์

บิต 3 และ 2 - PTCKPS1 และ PTCKPS0 (PWM Time Base input clock presale select bits) : บิตเลือกอัตรา

“00” = คาบเวลาของสัญญาณนาฬิกาอินพุตเท่ากับ TCY (อัตราปรีสเกลเลอร์เท่ากับ 1:1)
“01” = คาบเวลาของสัญญาณนาฬิกาอินพุตเท่ากับ 4TCY (อัตราปรีสเกลเลอร์เท่ากับ 1:4)

“10” = คาบเวลาของสัญญาณนาฬิกาอินพุตเท่ากับ 16TCY (อัตราปรีสเกลเลอร์เท่ากับ 1:16)

“11” = คาบเวลาของสัญญาณนาฬิกาอินพุตเท่ากับ 64TCY (อัตราปรีสเกลเลอร์เท่ากับ 1:64)

บิต 1 และ 0 - PTMOD1 และ PTMOD0 (PWM Time Base Mode select bits): บิตเลือกโหมดการทำงานของฐานเวลา PWM

“00” = ฐานเวลา PWM ทำงานในโหมดเปลี่ยนแปลงค่าอิสระ

“01” = ฐานเวลา PWM ทำงานในโหมดทำงานครั้งเดียว

“10” = ฐานเวลา PWM ทำงานในโหมดนับค่าขึ้นลงอย่างต่อเนื่อง

“11” = ฐานเวลา PWM ทำงานในโหมดนับค่าขึ้นลงอย่างต่อเนื่องพร้อมปรับปรุงค่า PWM

1.2.2. PTMR (PWM Time Base Register): รีจิสเตอร์กำหนดค่าฐานเวลา PWM

บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
PTDIR	PTMR14	PTMR13	PTMR12	PTMR11	PTMR10	PTMR09	PTMR8
R -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
PTMR7	PTMR6	PTMR5	PTMR4	PTMR3	PTMR2	PTMR1	PTMR0
R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0

บิต 15 - PTDIR (PWM Time Base count Direction status): บิตแจ้งทิศทางการนับค่าของฐานเวลา PWM

“0” = ฐานเวลา PWM กำลังนับขึ้น

“1” = ฐานเวลา PWM กำลังนับลง

บิตนี้อ่านได้อย่างเดียว

บิต 14 ถึง 0 - PTMR14 ถึง PTMR0 (PWM Time Base Register count value bits): บิตกำหนดค่าฐานเวลาของการกำเนิดสัญญาณ PWM มีด้วยกัน 15 บิต เป็นค่าตัวเลขแบบไม่คิดเครื่องหมาย

PTPER (PWM Time Base period Register): รีจิสเตอร์กำหนดคาบเวลาของฐานเวลา PWM

บิต 15 ไม่ใช้งาน กำหนดเป็น “0”

บิต 14 ถึง 0 - PTPER14 ถึง 0 (PWM Time Base period value bits): บิตกำหนดค่าคาบเวลาของฐานเวลา PWM มีด้วยกัน 15 บิต เป็นค่าตัวเลขแบบไม่คิดเครื่องหมาย

1.2.3. SEVTCMP (special Event compare Register): รีจิสเตอร์เปรียบเทียบค่าสำหรับสร้างสัญญาณกระตุ้นพิเศษ

บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
SEVTDIR	SEVTCMP14	SEVTCMP14	SEVTCMP14	SEVTCMP14	SEVTCMP14	SEVTCMP9	SEVTCMP8
R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	U -0	U -0
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
SEVTCMP7	SEVTCMP6	SEVTCMP5	SEVTCMP4	SEVTCMP3	SEVTCMP2	SEVTCMP1	SEVTCMP0
R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0

บิต 15 - SEVTDIR (special Event Trigger Time Base Direction bit): บิตกำหนดการเกิดสัญญาณกระตุ้นพิเศษ (1)

“0” = เกิดสัญญาณกระตุ้นพิเศษเมื่อฐานเวลา PWM นับค่าขึ้น

“1” = เกิดสัญญาณกระตุ้นพิเศษเมื่อฐานเวลา PWM นับค่าลง

บิต 14 ถึง 0 - SEWCMP14 ถึง SEWCW0 (special Event compare value bit): บิตกำหนดค่าสำหรับเปรียบเทียบมีด้วยกัน 15 บิต เป็นค่าตัวเลขแบบไม่คิดเครื่องหมาย

หมายเหตุ (1) บิต SEVTDIR จะถูกเปรียบเทียบกับบิต PTDIR (บิต 15 ของรีจิสเตอร์ PTMR) เพื่อสร้างสัญญาณกระตุ้นพิเศษ ส่วนข้อมูลในบิต SEVCMP14 ถึง SEVTCMP0 จะถูกเปรียบเทียบกับบิต PTMR14 ถึง PTMR0 เพื่อสร้างสัญญาณกระตุ้นพิเศษ

1.2.4. PWMCON1 (PWM control Register 1): รีจิสเตอร์ควบคุม PWM #1

บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
-	-	-	-	PMOD4	PMOD3	PMOD2	PMOD1
U -0	U -0	U -0	U -0	R/W -0	R/W -0	R/W -0	R/W -0
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
PEN4H	PEN3H	PEN2H	PEN1H	PEN4L	PEN3L	PEN2L	PEN1L
R/W -1	R/W -1	R/W -1	R/W -1	R/W -1	R/W -1	R/W -1	R/W -1

บิต 15 ถึง 12 ไม่ใช้งาน กำหนดเป็น “0”

บิต 11 ถึง 8 - PMOD4 ถึง PMOD1 (PWM I/O pair Mode bits): บิตกำหนดโหมดการทำงานของเอาต์พุต

“0” = กำหนดให้คู่ของขาพอร์ตของโมดูล MCPWM ทำงานเป็นเอาต์พุตแบบคอมพลีเมนตารีหรือแบบทำงานร่วมกับคู่เอาต์พุตอื่น

“1” = กำหนดให้คู่ของขาพอร์ตของโมดูล MCPWM ทำงานเป็นเอาต์พุตแบบอิสระ

บิต 7 ถึง 4 - PEN4H ถึง PEN1H (PWMxH I/O Enable bits): บิตเอนเอเบลขาเอาต์พุตด้านแรงดันสูงของโมดูล MCPWM (1)

“0” = กำหนดให้ขา PWMxH ทำงานเป็นขาพอร์ตอินพุตเอาต์พุตปกติ

“1” = กำหนดให้ขา PWMxH เป็นขาเอาต์พุตด้านแรงดันสูงของโมดูล MCPWM

บิต 3 ถึง 0 - PEN4L ถึง PEN1L (PWMcL I/O Ehable bits): บิตเอ็นเอเบิลขาเอาต์พุตด้านแรงดันต่ำของโมดูล MCPWM (1)

“0” = กำหนดให้ขา PWMxL ทำงานเป็นขาพอร์ตอินพุตเอาต์พุตปกติ

“1” = กำหนดให้ขา PWMXL เป็นขาเอาต์พุตด้านแรงดันต่ำของโมดูล MCPWM

หมายเหตุ (1) หลังจากรีเซท สถานะของบิต PENxH จะขึ้นอยู่กับค่าทีกำหนดในรีจิสเตอร์ FBORPOR

1.2.5. PWMCON2 (PWM control Register 2): รีจิพเตอร์ควบคุม PWM #2

บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
-	-	-	-	SEVOPS3	SEVOPS2	SEVOPS1	SEVOPS0
U -0	U -0	U -0	U -0	R/W -0	R/W -0	R/W -0	R/W -0
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
-	-	-	-	-	-	OSYNC	UDIS
U -0	U -0	U -0	U -0	U -0	U -0	R/W 0	R/W -0

บิต 15 ถึง 12 ไม่ใช้งาน กำหนดเป็น “0”

บิต 11 ถึง 8 - SEVOPS3 ถึง SEVOPS0 (PWM Special Trigger output postscale select bits): บิตกำหนดอัตราโพสต์สเกลเลอร์ของสัญญาณกระตุ้นพิเศษ

“0000” = 1:1

“0001” = 1:2

.....

“1111” = 1:16

บิต 7 ถึง 2 ไม่ใช้งาน กำหนดเป็น “0”

บิต 1 - OSYNC (output override synchronization bit) : บิตกำหนดจังหวะการเปลี่ยนค่าสัญญาณเอาต์พุต

“0” = เปลี่ยนแปลงค่าของรีอิสเตอร์ VDCON ในไซเกิลของการทำงานถัดไป

“1” = เปลี่ยนแปลงค่าของรีอิสเตอร์ VDCON เมื่อจังหวะการทำงานตรงกับฐานเวลาของ PWM

บิต 0 - UDIS (PWM Update Disable bit): บิตดิสเอเบิลการปรับปรุงค่าดิวตี้ไซเกิลของสัญญาณ PWM

“0” = เลือกเพื่อยอมให้เกิดการปรับปรุงค่าดิวตี้ไซเกิลและรีจิสเตอร์บัฟเฟอร์ที่ใช้กำหนดคาบเวลา

“1” = เลือกเพื่อไม่ยอมให้เกิดการปรับปรุงค่าดิวตี้ไซเกิลและรีจิสเตอร์บัฟเฟอร์ที่ใช้กำหนดคาบเวลา

1.2.6. DTCON1 (Dead Time control Register # 1): รีจิสเตอร์ควบคุมค่าเวลาวิกฤตหรือ Dead Time #1

บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
DTBPS1	DTBPS1	DTB5	DTB4	DTB3	DTB2	DTB1	DTB0
R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
DTAPS1	DTAPS0	DTA5	DTA4	DTA3	DTA2	DTA1	DTA0
R/W -0	R/W -0	R/W -0	R/W -0	U -0	U -0	R/W -0	R/W -0

บิต 15 และ 14 - DTBPS1 และ DTBPS0 (Dead Time Unit B presale select bits): บิตเลือกอัตราปรีสเกลเลอร์ของส่วนกำเนิดค่าเวลาวิกฤตชุด B

“00” = เลือกคาบเวลาของสัญญาณนาฬิกาส่วนกำเนิดค่าเวลาวิกฤตชุด B เป็น Tcy

“01” = เลือกคาบเวลาของสัญญาณนาฬิกาส่วนกำเนิดค่าเวลาวิกฤตชุด B เป็น 2Tcy

“10” = เลือกคาบเวลาของสัญญาณนาฬิกาส่วนกำเนิดค่าเวลาวิกฤตชุด B เป็น 4Tcy

“11” = เลือกคาบเวลาของสัญญาณนาฬิกาส่วนกำเนิดค่าเวลาวิกฤตชุด B เป็น 8Tcy

บิต 13 ถึง 8 - DTB5 ถึง DTB0 (Unsigned 6-bit Dead Time value bits for Dead Time Unit B): บิตกำหนดค่าเวลาวิกฤตของส่วนกำเนิดค่าเวลาวิกฤตชุด B

มีด้วยกัน 6 บิต เป็นค่าตัวเลขแบบไม่คิดเครื่องหมาย กำหนดค่าได้ตั้งแต่ 000000-111111

ไม่มีใช้งานใน dsPIC30F2010

บิต 7 และ 6 -DTAPS1 และ DTAPS0 (Dead Time Unite presale select bits): บิตเลือกอัตราปรีสเกลเลอร์ของสวนจัดการค่าเวลาวิกฤตชุด A

“00” = เลือกคาบเวลาของสัญญาณนาฬิกาส่วนกำเนิดค่าเวลาวิกฤตชุด A เป็น Tcy

“01” = เลือกคาบเวลาของสัญญาณนาฬิกาส่วนกำเนิดค่าเวลาวิกฤตชุด A เป็น 2Tcy

“10” = เลือกคาบเวลาของสัญญาณนาฬิกาส่วนกำเนิดค่าเวลาวิกฤตชุด A เป็น 4Tcy

“11” = เลือกคาบเวลาของสัญญาณนาฬิกาส่วนกำเนิดค่าเวลาวิกฤตชุด A เป็น 8Tcy

บิต 5 ถึง 0 - DTA5 ถึง DTA0 (Unsigned 6-บิต Dead Time value bits for Dead Time Unit A): บิตกำหนดค่าเวลาวิกฤตของส่วนกำเนิดค่าเวลาวิกฤตชุด A

มีด้วยกัน 6 บิต เป็นค่าตัวเลขแบบไม่คิดเครื่องหมาย กำหนดค่าได้ตั้งแต่ 000000-111111

1.2.7. FLTACON (Fault A control Register): รีจิสเตอร์ควบคุมการตรวจจับความผิดปกติชุด A

บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
FAOV4H	FAOV4L	FAOV3H	FAOV3L	FAOV2H	FAOV2L	FAOV1H	FAOV1L
R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	U -0	U -0	U -0
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
FLTAM	-	-	-	FAEN4	FAEN3	FAEN2	FAEN1
R/W -0	U -0	U -0	U -0	U -0	R/W -0	R/W -0	R/W -0

บิต 15 และ 14 - FAOV4H-FAOV4L (Fault input A PWM override value bits for PWM4) บิตกำหนดลักษณะของสัญญาณ PWM ของโมดูล PWM4 จากอินพุตของส่วนตรวจจับความผิดปกติของการขับมอเตอร์ชุด A

“0” = ขาเอาต์พุตของสัญญาณ PWM เป็นสถานะ “ไม่แอกตีฟ” หรือเป็นสัญญาณแรงดันต่ำ เมื่อเกิดตรวจจับความผิดปกติได้ที่ขา FLTA

“1” = ขาเอาต์พุตของสัญญาณ PWM เป็นสถานะ “แอกตีฟ” หรือเป็นสัญญาณแรงดันสูง เมื่อเกิดตรวจจับความผิดปกติได้ที่ขา FLTA

ในการกำหนดค่าของขาเอาต์พุตคู่ (PWMxH และ PWMxL) ต้องแตกต่างกัน 2 บิดนี้ไม่ใช่งานใน dsPIC30F2010

บิต 13 ถึง 8- FAOV3H-FAOV1L (Fault input A PWM override value bits for PWM3 -PWM1) : บิตกำหนดลักษณะของสัญญาณ PWM ของโมดูล PWM3 ถึง PWM1 จากอินพุตของส่วนตรวจจับความผิดปกติของการขับมอเตอร์ชุด A

“0” = ขาเอาต์พุตของสัญญาณ PWM เป็นสถานะ “ไม่แอกตีฟ” หรือเป็นสัญญาณแรงดันต่ำ เมื่อเกิดตรวจจับความผิดปกติได้ที่ขา FLTA

“1” = ขาเอาต์พุตของสัญญาณ PWM เป็นสถานะ “แอกตีฟ” หรือเป็นสัญญาณแรงดันสูง เมื่อเกิดตรวจจับความผิดปกติได้ที่ขา FLTA ในการกำหนดค่าของขาเอาต์พุตคู่ (PWMxH และ PWMxL) ต้องแตกต่างกัน

บิต 7 - FLTAM (Fault a Mode bit): บิตเลือกโหมดของส่วนตรวจจับความผิดปกติของการขับมอเตอร์ชุด A

“0” = เลือกให้ขาพอร์ตเอาต์พุตทำงานตามที่กำหนดในบิต 15 ถึง 8 ของรีจิสเตอร์ FLTACONเมื่อตรวจจับความผิดปกติได้ที่ขา FLTA

“1” = เลือกให้ขาอินพุต FLTA ทำงานในโหมดไซเกิลต่อไซเกิล (cycle-by-cycle mode)

บิต 6 ถึง 4 ไม่ใช้งาน กำหนดเป็น “0”

บิต 3 -FAEN4 (Fault input A Enable bit for PWM4): บิตเอ็นเอเบิลการทำงานของส่วนตรวจจับความผิดปกติของการขับมอเตอร์ชุด A (FLTA)

“0” = เลือกให้ขา PWM4H/PWM4L ไม่ถูกควบคุมโดยอินพุต FLTA

“1” = เลือกให้ขา PWM4H/PWM4L ถูกควบคุมโดยอินพุต FLTA บิตนี้ไม่ใช้งานใน dsPIC30F2010

บิต 2 - FAEN3 (Fault input A Enable bit for PWM3): บิตเอ็นเอเบิลการทำงานของส่วนตรวจจับความผิดปกติของการขับมอเตอร์ชุด A (FLTA)

“0” = เลือกให้ขา PWM3H/PWM3L ไม่ถูกควบคุมโดยอินพุต FLTA

“1” = เลือกให้ขา PWM3H/PWM3L ถูกควบคุมโดยอินพุต FLTA

บิต 1 - FAEN2 (Fault input A Enable bit for PWM2): บิตเอ็นเอเบิลการทำงานของส่วนตรวจจับความผิดปกติของการขับมอเตอร์ชุด A (FLTA)

“0” = เลือกให้ขา PWM2H/PWM2L ไม่ถูกควบคุมโดยอินพุต FLTA

“1” = เลือกให้ขา PWM2H/PWM2L ถูกควบคุมโดยอินพุต FLTA

บิต 0 - FAEN1 (Fault input A Enable bit for PWM1): บิตเอ็นเอเบิลการทำงานของส่วนตรวจจับความผิดปกติของการขับมอเตอร์ชุด A (FLTA)

“0” = เลือกให้ขา PWM1H/PWM1L ไม่ถูกควบคุมโดยอินพุต FLTA
“1” = เลือกให้ขา PWM1H/PWM1L ถูกควบคุมโดยอินพุต FLTA

FLTBCON (Fault B control Register): รีจิสเตอร์ควบคุมการตรวจจับความผิดปกติของการขับมอเตอร์ชุด B

รีจิสเตอร์ตัวนี้ไม่มีใช้งานใน dsPIC30F2010 ลักษณะการกำหนดค่าเหมือนกับรีจิสเตอร์ FLTACON อย่างไรก็ตามถ้าหากมีการใช้งานส่วนตรวจจับความผิดปกติของการขับมอเตอร์ทั้ง 2 ชุดพร้อมกัน (ใน dsPIC เบอร์ที่สนับสนุนความสามารถนี้) การกำหนดที่ส่วนตรวจจับความผิดปกติของการขับมอเตอร์ชุด A จะมีนัยสำคัญสูงกว่าชุด B

1.2.8. OVDCON (override control Register): รีจิสเตอร์กำหนดการทำงานของขาเอาต์พุตของโมดูล MCPWM โดยตรง เป็นรีจิสเตอร์ที่ใช้กำหนดค่าสำหรับการสร้างสัญญาณ PWM ในการควบคุมมอเตอร์อีกตัวหนึ่งโดยการกำหนดค่าที่รีจิสเตอร์นี้ เป็นการกำหนดในลักษณะเข้าถึงโดยตรงที่ขาเอาต์พุตของโมดูลMCP

บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
POVD4H	POVD4L	POVD3H	POVD3L	POVD2H	POVD2L	POVD1H	POVD1L
R/W -1	R/W -1	R/W -1	R/W -1	R/W -1	R/W -1	R/W -1	R/W -1
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
POUT4H	POUT4L	POUT3H	POUT3L	POUT2H	POUT2L	POUT1H	POUT1L
R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0

บิต 15 และ 14 - POVD4H และ POVD4L (PWM output override bits for PWM4): บิตกำหนดการทำงานของขาเอาต์พุตของโมดูล PWM4 แบบโดยตรง

“1” = ขาพอร์ตของ PWM4 ถูกควบคุมโดยส่วนกำเนิดสัญญาณ PWM
“0” = ขาพอร์ตของ PWM4 ถูกควบคุมด้วยค่าของบิต POUTxx 2 บิดนี้ไม่ใช้งานใน dsPIC30F2010 กำหนดเป็น “0”

บิต 13 ถึง 8 - POVD3H ถึง POVD1L (PWM output override bits for PWM3-PWM1): บิตกำหนดการทำงานของขาเอาต์พุตของโมดูล PWM3 ถึง PWM1 แบบโดยตรง

“1” = ขาพอร์ตของ PWMxx (3H-1L) ถูกควบคุมโดยส่วนกำเนิดสัญญาณ PWM

“0” = ขาพอร์ตของ PWMxx (3H-1L) ถูกควบคุมด้วยค่าของบิต POUTxx

บิต 7 และ 6 - POUT4H และ POUT1L (PWM Manual output bits for PWM4): บิตกำหนดการทำงานของพอร์ตเอาต์พุต PWM4 แบบโดยตรง

“0” = ขาเอาต์พุตของสัญญาณ PWM เป็นสถานะ “ไม่แอกตีฟ” หรือเป็นสัญญาณแรงดันต่ำ โดยต้องกำหนดให้บิต POVD4H และ POD4L เป็น “0”

“1” = ขาเอาต์พุตของสัญญาณ PWM เป็นสถานะ “แอกตีฟ” หรือเป็นสัญญาณแรงดันสูง โดยต้องกำหนดให้บิต POVD4H และ POD4L เป็น “0”

บิต 5 ถึง 0 - POUT3H และ POUT1L (PWM Manual output bits for PWM3-PWM1): บิตกำหนดการทำงานของพอร์ตเอาต์พุต PWM3 ถึง PWM1 แบบโดยตรง

“0” = ขาเอาต์พุตของสัญญาณ PWM เป็นสถานะ “ไม่แอกตีฟ” หรือเป็นสัญษญาณแรงดันต่ำ โดยต้องกำหนดให้บิต POVD4H และ POD4L เป็น “0”

“1” = ขาเอาต์พุตของสัญญาณ PWM เป็นสถานะ “แอกตีฟ” หรือเป็นสัญญาณแรงดันสูง โดยต้องกำหนดให้บิต POVD4H และ POD4L เป็น “0”

1.2.9. PDCI ถึง PDC4 (PWM Duty cycle Register): รีจิสเตอร์กำหนดค่าดิวตี้ไซเกิลของโมดูลกำเนิดสัญญาณ PWM ชุดที่ 1 ถึง 4 เป็นรีจิสเตอร์ขนาด 16 บิตที่ใช้กำหนดค่าดิวตี้ไซเกิลของสัญญาณ PWM ในโมดูล MCPWM ทั้ง 4 ชุด โดยแยกกันอย่างอิสระคือ PDC1 สำหรับโมดูล PWM ชุดที่ 1 ไล่ไปตามลำดับจนถึง PDC4สำหรับโมดูล PWM ชุดที่ 4 อย่างไรก็ตามสำหรับ dsPIC30F2010 จะไม่มีรีจิสเตอร์ PDC4

1.2.10. FBORPOR (BOR AND POR Device configuration Register): รีจิสเตอร์กำหนดค่าสำหรับการเพาเวอร์-ออนรีเซตและ บราวเอาต์รีเซต บิตที่เกี่ยวข้องกับโมดูล MCPWM ประกอบด้วย

บิต 10 - PWMPIN (Motor control PWM Module pin Mode bit): บิตกำหนดการทำงานของขาพอร์ตในโมดูล MCPWM

“0” = ขาพอร์ตของโมดูล PWM ควบคุมมอเตอร์ถูกกำหนดให้เป็นขาเอาต์พุตสำหรับส่งสัญญาณPWMเพื่อขับมอเตอร์ (บิต 7 ถึง 0 ของ PWMCON1 = 0x00)

“1” = ขาพอร์ตของโมดูล PWM ควบคุมมอเตอร์ถูกกำหนดให้เป็นขาพอร์ตอินพุตเอาพุตปกติ (บิต 7 ถึง 0 ของ PWMCON1 = 0xFF)

บิต 9 - HPOL (Motor control PWM Module High side Polarity bit): บิตกำหนดขั้วของขาเอาต์พุตของโมดูลMCPWM ด้านแรงดันสูง

“0” = กำหนดให้ขาเอาต์พุตด้านแรงดันสูงของโมดูล PWM เป็นขั้วแรงดันต่ำ

“1” = กำหนดให้ขาเอาต์พุตด้านแรงดันสูงของโมดูล PWM เป็นขั้นแรงดันสูง

บิต 8 - LPOL (Motor control PWM Module Low side polarity bit): บิตกำหนดขั้วของขาเอาต์พุตของโมดูลMCPWM ด้านแรงดันต่ำ

“0” = กำหนดให้ขาเอาต์พุตด้านแรงดันต่ำของโมดูล PWM เป็นขั้วแรงดันต่ำ

“1” = กำหนดให้ขาเอาต์พุตด้านแรงดันต่ำของโมดูล PWM เป็นขั้วแรงดันสูง

ในภาพที่ 41 โปรแกรมย่อย InitMCPWM ที่ได้ทดลองเขียนขึ้นมามีหน้าที่เปิดใช้งาน โมดูล PWM 2 ทำงานเพียงตัวเดียว

```
void InitMCPWM(void)
{
        // FPWM = กำหนดความถี่ที่ต้องการ
        // FCY  7372800*16/4                    // xtal = 7.3728Mhz; PLLx16
        //pre อัตราส่วนปรีสเกลเลอร์
        PTPER = 29490;                          //PTPER =( FCY / (FPWM*pre) ) - 1;
         SEVTCMP = PTPER;
        PWMCON1bits.PMOD1 = 1;
        PWMCON1bits.PMOD2 = 1;          //กำหนดให้pwm2ทำงานแบบอิสระ
        PWMCON1bits.PEN1H =0;           //กำหนดให้เป็นเอาพุตอินพุตปรกติ
        PWMCON1bits.PEN2L =0;
        PWMCON1bits.PEN3L =0;
        PWMCON1bits.PEN3H =0;
        PWMCON1bits.PEN1L =1;           //กำหนดให้เป็นpwm
        PWMCON1bits.PEN2H =1;
        PWMCON2bits.SEVOPS = 0;         // 1:1 postscale value
        PWMCON2bits.UDIS    = 0;
        PWMCON2bits.OSYNC  = 0;
        OVDCON = 0xFFFF;                        // allow control using OVD
        PTCONbits.PTMOD  = 0;   //ฐานเวลา pwm ทำงานให โหมดเปลี่ยนเปลงอิสระ
        PTCONbits.PTCKPS  =0;   //อัตราส่วนลดทอนสัญญาณนาฬิกาอินพุตของฐานเวลา 1:1
        PTCONbits.PTOPS = 0;    //การลดทอนทางเวลาผ่านโพสสเกลเลอร์
        PTCONbits.PTSIDL = 0;
        PTCONbits.PTEN = 1;     //เปิดให้ pwm ทำงาน
/*
        PDC1 = PTPER*2*0.2;                     //PTPER*2 = (ดิวตี้ไซเคิล/100%)
*/
}
```

ภาพที่ 41 โปรแกรมย่อย InitMCPWM

2. **การใช้งาน โมดูล** UART **ใน** dsPIC30F2010

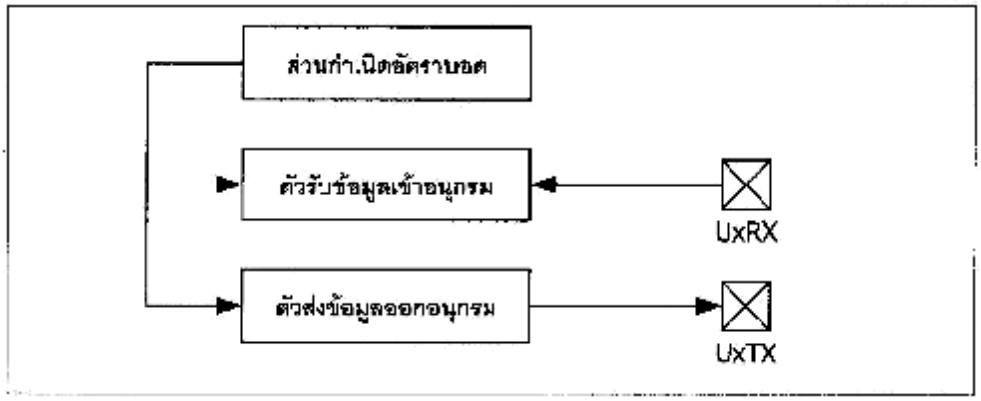
สำหรับบทนี้จะเป็นการเขียนโปรแกรมใช้งานโมดูล UARTเพื่อการสื่อสารข้อมูลอนุกรมระหว่าง dsPIC30F2010 กับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม

2.1 คุณสมบัติโดยสรุปของโมดูล UART ใน dsPIC30F2010

1. สื่อสารข้อมูลแบบสองทิศทาง (ฟูลดูเพล็กซ์ : full-duplex) ในแบบ 8 และ 9 บิต
2. เลือกการสื่อสารข้อมูลแบบตรวจสอบบิตพาริตี้คู่ (Even) หรือคี่ (odd) และไม่ตรวจสอบบิตพาริตี้ (None) สำหรับรูปแบบสื่อสารข้อมูลในแบบ 8 บิต มี บิตหยุด (Stop bit) 1 หรือ 2 บิต
3. มีส่วนกำเนิดอัตราบอด (Baud Rate Generator) ขนาด 16 บิต สำหรับกำหนดจังหวะ และ
4. อัตราเร็วโนการสื่อสารข้อมูลอนุกรมแยกอิสระเพื่อลดภาระการทำงานของโมดูลไทเมอร์
5. กำเนิดอัตราบอดได้ตั้งแต่ 38 บิตต่อวินาที (bps) ถึง 1.875 เมกะบิตต่อวินาที (Mbps)
6. บัฟเฟอร์ข้อมูลขาส่ง (TX) และขารับ (G) ขนาด 4 เวิร์ด แยกส่วนกัน
7. มีบิตแฟลก แจ้งข้อผิดพลาดในกรณีต่างๆ ของการสื่อสาร สามารถตรวจจับความผิดพลาดในการสื่อสารข้อมูลอนุกรม ได้แก่

    7.1. ความผิดพลาดทางพาริตี้ (parity Error : PE)
    7.2. รับข้อมูลไม่ทัน (Buffer overrun Error : OE)
    7.3. เฟรมข้อมูลผิดพลาด (Framing Error : FE)

8. สนับสนุนความสามารถในการอินเตอร์รัปต์แอสเดรส (ข้อมูลบิต 9 เป็น “1”)
9. อินเตอร์รัปต์เวกเตอร์แยกตำแหน่งกันระหว่างการส่งและรับข้อมูล(RX)
10. สามารถทำงานในโหมด Loop back



ภาพที่ 42 ไดอะแกรมแสดงส่วนประกอบหลักของโมดูล UART ในไมโครคอนโทรลเลอร์ dsPIC

2.2. รีจิสเตอร์ที่ใช้ในโมดูล UART มีทั้งสิ้น 5 ตัวคือ

1. UxMODE ใช้กำหนดโหมดการทำงานของโมดูล UART โดยตัวอักษร x เป็นเลข 1 หรือ 2เพราะใน dsPIC สามารถบรรจุโมดูล UART ได้2 ชุด สำหรับใน dsPIC30F2010 มี 1 ชุด จึงกำหนดเป็นU1MODE

2. UxSTA ใช้แสดงสถานะ การทำงานของโมดูลUART โดยตัวอักษรxเป็นเลข1หรือ2 เพราะใน dsPIC สามารถบรรจโมดูล UART ได้ 2 ชุดสำหรับในdsPIC30F2010 มี 1 ชุด จึงกำหนดเป็น U1STA

3. UxRXREG ใช้เก็บข้อมูลที่รับเข้ามาทางขาเชื่อมต่อพอร์ตอนุกรม โดยตัวอักษร x เป็นเลข 1 หรือ 2 เพราะใน dsPIC สามารถบรรจุโมดูล UART ได้ 2 ชุด สำหรับใน dsPIC30F2010 มี 1 ชุดจึงกำหนดเป็น U1RXREG

4. UxTXREG ใช้เก็บข้อมูลสำหรับส่งออกทางขาเชื่อมต่อพอร์ตอนุกรม โดยตัวอักษร x เป็นเลข 1 หรือ 2เพราะใน dsPIC สามารถบรรจโมดูล UART ได้ 2 ชุด สำหรับใน dsPIC30F2010 มี 1 ชุดจึงกำหนดเป็น U1TXREG

5. UxBRG ใช้เก็บค่าสำหรับกำหนดอัตราบอด โดยตัวอักษร x เป็นเลข 1 หรือ 2 เพราะใน dsPIC สามารถบรรจุโมดูล UART ได้ 2 ชุดสำหรับใน dsPIC30F2010 มี 1 ชุด จึงกำหนดเป็น U1BRG

2.2.1. U1MODE (UART Mode Register): รีจิสเตอร์กำหนดโหมดทำงานของโมดูล UART

บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
UARTEN	-	USIDL	-	-	ALTIO	-	-
R/W -0	U -0	R/W -0	U -0	U -0	R/W -0	U -0	U -0
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
WAKE	LPBACK	ABAUD	-	-	PDSEL1	PDSEL0	STSEL
R/W -0	R/W -0	R/W -0	U -0	U -0	R/W -0	R/W -0	R/W -0

บิต 15 - UARTEN (UART Enable bit): บิตเอ็นเอเบิลการทำงานของโมดูล UART

"0” = ดิสเอเบิลการทำงานของโมดูล UART

“1” = เอ็นเอเบิลให้โมดูลนี้ทำงาน ขาพอร์ตที่เกี่ยวข้องจะถูกต่อเข้ากับโมดูล UART

บิต 14 ไม่มีการใช้งาน กำหนดเป็น “0”

บิต 13 – USIDL (Stop in IDLE Mode bit): บิตบิตกำหนดให้โมดูลUARTหยุดทำงานในโหมดไอเดิล

“0” = ยังคงให้โมดูล UART ทำงานต่อไป แม้ว่าซีพียูจะเข้าสู่โหมดไอเดิลก็ตาม

“1” = หยุดการทำงานของโมดูล UART ทันทีที่ซีพียูเข้าโหมดไอเดิล

บิต 12 และ 11 ไม่มีการใช้งาน กำหนดเป็น “0”

บิต 10 - ALTIO (UART Alternate I/O selection bit): บิตเลือกใช้งานขาพอร์ต UART เสริม

“0” = เลือกใช้ขาพอร์ต U1TX และ U1RX ในการติดต่อกับพอร์ตอนุกรม

“1” = เลือกใช้ขาพอร์ต U1TX และ U1RX ซึ่งเป็นขาพอร์ต UART เสริมในการติดต่อกับพอร์ตอนุกรมแทนขาพอร์ตหลัก

บิต 9 และ 8 ไม่มีการใช้งาน กำหนดเป็น “0”

บิต 7 WAKE (Enable Wake-up on Startbit Detect during SLEEP Mode bit): บิตเอ็นเอเบิลให้โมดูล UARTทำงาน (เวกอัป) เมื่อตรวจพบบิตเริ่มต้นในขณะที่ซีพีอยู่ในโหมดสลีป

“0” = ดิสเอเบิลความสามารถนี้

“1” = เอ็นเอเบิลความสามารถนี้

บิต 6 - LPBACK (UART Loop back Mode select bit): บิตเลือกโหมดการทำงานแบบลูปแบ็กของโมดูล UART

“0” = ดิสเอเบิลความสามารถนี้

“1” = เอ็นเอเบิลความสามารถนี้

บิต 5 - ABAUD (Auto Baud Enable bit): บิตเอ็นเอเบิลการเลือกอัตราบอดอัตโนมิต

“1” = รับสัญญาณจากขา UxRX ป้อนเข้าสู่โมดูลตรวจจับสัญญาณอินพุต (IC)

“0” = รับสัญญาณจากขาพอร์ต ICx ป้อนเข้าสู่โมดูลตรวจจับสัญญาณอินพุต (IC)

บิต 3 และ 4 ไม่มีการใช้งาน กำหนดเป็น “0”

บิต 2 ถึง 1 - PDSEL1 และ PDSEL0 (Parity and Data Selection bits): บิตเลือกรูปแบบข้อมูลอนุกรม

“00” = 8 บิตข้อมูล ไม่มีบิตพาริตี้

“01” = 8 บิตข้อมูล บิตพาริตี้เป็นคู่

“10” = 8 บิตข้อมูล บิตพาริตี้เป็นคี่

“11” = 9 บิตข้อมูล ไม่มีบิตพาริตี้

บิต 0 - STSEL (STOP Selection bit): บิตเลือกจำนวนบิตปิดท้ายหรือบิตหยุด (Stop bit)

“1” = เลือกบิตปิดท้าย 2 บิต
“0” = เลือกบิตปิดท้าย 1 บิต

2.2.2. UISTA (UARTX Status and Control Register): รีจิสเตอร์ควบคุมและแสดงสถานะการส่งข้อมูลในโมดูล UART

บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
UTXISEL	-	-	-	UTXBRK	UTXEN	UTXBF	TRMT
R/W -0	U -0	U -0	U -0	R/W -0	R/W -0	R -0	R -1
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
URXISEL1	URXISEL0	ADDEN	RIDLE	PERR	FERR	OERR	URXDA
R/W -0	R/W -0	R/W -0	R -1	R -0	R -0	R/C -0	R -0

บิต 15 - UTXISEL (Transmission interrupt Mode Selection bit): บิตเลือกการอินเตอร์รัปต์ในการถ่ายทอดข้อมูล

“0” = เลือกให้เกิดการอินเตอร์รัปต์เมื่อข้อมูลถูกส่งผ่านไปยังชิฟต์รีจิสเตอร์ตัวส่งข้อมูล
“1” = เลือกให้เกิดการอินเตอร์รัปต์เมื่อข้อมูลถูกส่งผ่านไปยังชิฟต์รีจิสเตอร์ตัวส่งข้อมูลและข้อมูลถูกส่งออกไปเรียบร้อย จนบัฟเฟอร์ตัวส่งว่าง

บิต 14 ถึง 12 ไม่มีการใช้งาน กำหนดเป็น “0”
บิต 11 - UTXBRK (Transmit Break bit): บิตกำหนดการหยุดส่งข้อมูล

“0” = ขาพอร์ต U1TX ยังคงทำการส่งข้อมูลตามปกติ
“1” = ขาพอร์ต U1TX ถูกทำให้เป็นลอจิก “0” จึงหยุดการส่งข้อมูล

บิต 10 - UTXEN (Transmit Enable bit): บิตเอ็นเอเบิลการส่งข้อมูล

“0” = ดิสเอเบิลตัวส่งข้อมูลในโมดูล UART ขา U1TX ถูกควบคุมโดยวงจรควบคุมขาพอร์ต

“1” = เอ็นเอเบิลให้ตัวส่งข้อมูลในโมดูล UART ทำงาน ขา U1TX ถูกควบคุมโดยโมดูล UART

บิต 9 - UTXBF (Transmit Buffer Full Status bit): บิตแสดงสถานะของบัฟเฟอร์ตัวส่ง

“0” = บัฟเฟอร์ตัวส่งมีข้อมูลยังไม่เต็ม สามารถรับข้อมูลเพิ่มเติมได้อีกอย่างน้อย 1 เวิร์ด

“1” = บัฟเฟอร์ตัวส่งมีข้อมูลเต็มTransmit buffer is full บิตนี้สามารถอ่านได้เพียงอย่างเดียว การเซตหรือเคลียร์กระทำโดยกระบวนการภายในโมดูล UART

บิต 8 - TRMT (Transmit shift Register is Empty bit): บิตแสดงสถานะข้อมูลว่างของชิฟต์รีจิสเตอร์ตัวส่ง

“0” = ชิฟต์รีจิสเตอร์ตัวส่งในโมดูล UART ไม่ว่าง แสดงว่า การส่งผ่านข้อมูลยังไม่เสร็จสิ้น

“1” = ชิฟต์รีจิสเตอร์ตัวส่งและบัฟเฟอร์ตัวส่งในโมดูล UART ว่าง แสดงว่ากระบวนการส่งผ่านข้อมูลเสร็จสิ้น พร้อมสำหรับการถ่ายทอดข้อมูลรอบใหม่ บิตนี้สามารถอ่านได้เพียงอย่างเดียว การเซตหรือเคลียร์กระทำโดยกระบวนการภายในโมดูล UART

บิต 7 และ 6 - URXISEL1 และ URXlSEL0 (Receive interrupt Mode selection bit): บิตเลือกการอินเตอร์รัปต์ของการรับข้อมูลในโมดูล UART

“0x” = บิตแฟลกของการอินเตอร์รัปต์เซต เมื่อได้รับข้อมูลเสร็จสมบูรณ์

“10” = บิตแฟลกของการอินเตอร์รัปต์เซต เมื่อบัฟเฟอร์ตัวรับมีข้อมูลจำนวน 3/4 ของขนาดบัฟเฟอร์สูงสุด ซึ่งก็คือ 3 ตัวอักษร

“11”= บิตแฟลกของการอินเตอร์รัปต์เซตเมื่อบัฟเฟอร์ตัวรับมีข้อมูลเต็มเท่ากับมีข้อมูล4ตัวอักษร

บิต 5 - ADDEN (Address Character Detect): บิตเลือกการตรวจจับแอดเดรส (bit 8 of received data = 1)

“0” = ดิสเอเบิลความสามารถนี้

“1” = เอ็นเอเบิลการตรวจจับแอดเดรสเมื่อเลือกให้โมดูล UART ทำงานในโหมด 9บิต โดยบิตที่ใช้แสดงแอดเดรสคือ บิต 8 ต้องเป็น “1”แต่ถ้าไม่ได้กำหนดให้ทำงานในโหมด 9 บิต การกำหนดที่บิตนี้จะไม่ส่งผลใดๆ

บิต 4 - RIDLE (Receiver IDLE bit): บิตแสดงสถานะของตัวรับ (บิตนี้อ่านได้เพียงอย่างเดียว)

“0” = ตัวรับยังอยู่ในสภาวะการรับข้อมูล

“1” = ตัวรับอยู่ในสภาวะสงบ พร้อมทำงาน

บิต 3 - PERR (Parity Error Status bit): บิตแจ้งความผิดพลาดทางพาริตี้ (บิตนี้อ่านได้เพียงอย่างเดียว)

“0” = ไม่มีความผิดพลาดเกิดขึ้น

“1” = มีความผิดพลาดทางพาริตี้เกิดขึ้นในข้อมูลปัจจุบัน

บิต 2 - FERR (Framing Error Status bit): บิตแจ้งความผิดพลาดทางเฟรมข้อมูล (บิตนี้อ่านได้เพียงอย่างเดียว)

“0” = ไม่มีความผิดพลาดเกิดขึ้น

“1” = มีความผิดพลาดทางเฟรมข้อมูลเกิดขึ้นในข้อมูลปัจจุบัน

บิต 1 -OERR (Receive Buffer Overrun Error Status bit): บิตแจ้งความผิดพลาดเนื่องจากบัฟเฟอร์ตัวรับ รับข้อมูลเกิน

“0” = ไม่มีความผิดพลาดเกิดขึ้น

“1” = มีความผิดพลาดพลาดเนื่องจากบัฟเฟอร์ตัวรับ รับข้อมูลเกิน เกิดขึ้นในข้อมูลปัจจุบัน บิตนี้สามารถอ่านได้เพียงอย่างเดียว การเคลียร์สามารถกระทำโดยกระบวนทางซอฟต์แวร์

บิต 0 - URXDA (Receive Buffer Data Available bit): บิตแสดงสถานะข้อมูลในบัฟเฟอร์ตัวรับ

“0” = บัฟเฟอร์ตัวรับว่าง

“1” = บัฟเฟอร์ตัวรับยังมีข้อมูลอยู่อย่างน้อย 1 ตัวอักษร

2.2.3. U1XREG (UARTX Receive Register): รีจิสเตอร์ตัวรับข้อมูลของโมดูล UART

บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
-	-	-	-	-	-	-	URX8
U -0	U -0	U -0	U -0	U -0	U -0	U -0	R -0
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
URX7	URX6	URX5	URX4	URX3	URX2	URX1	URX0
R -0	R -0	R -0	R -0	R -0	R -0	R -0	R -0

บิต 15 ถึง 9 ไม่ใช้งาน กำหนดเป็น “0”

บิต 8 - URX8 (Data bit 8 of the Received Character): บิตข้อมูลที่ 9 หรือ บิต 8 เมื่อทำงานในโหมด 9 บิต

บิต 7 ถึง 0 - URX7 ถึง URX0 (Data bits 7-0 of the Received Character): บิตข้อมูลขนาด 8 บิต

2.2.4. U1TXREG (UARTX Transmit Register): รีจิสเตอร์ตัวส่งข้อมูลของโมดูล UART

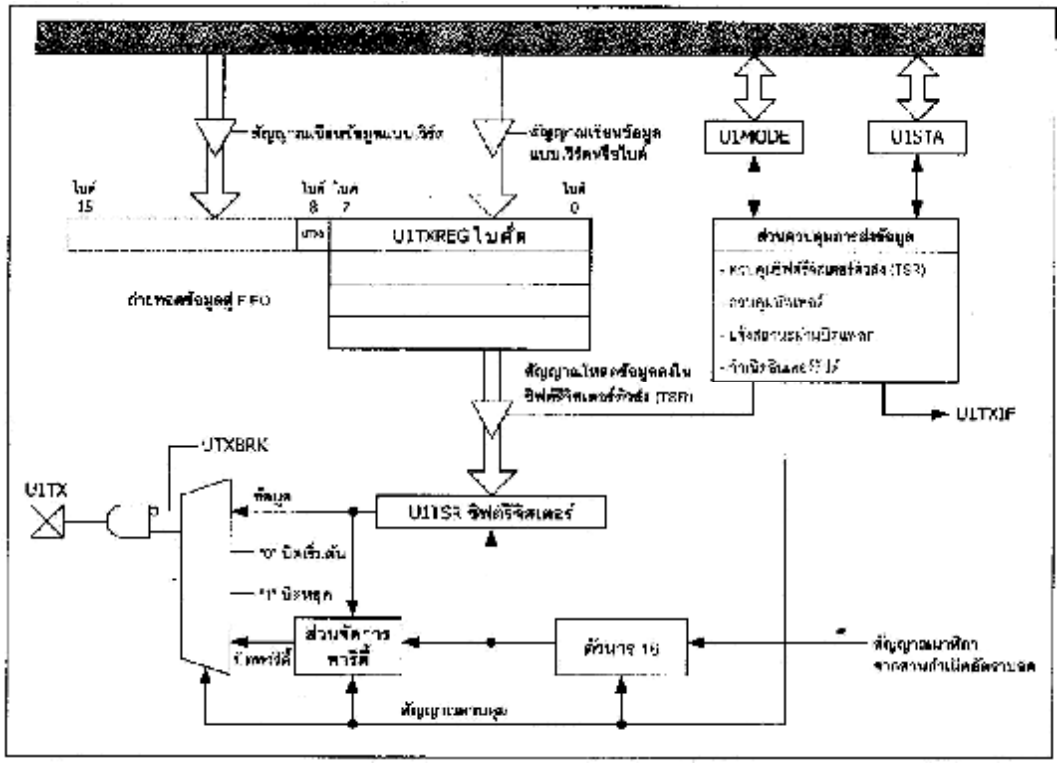
บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
-	-	-	-	-	-	-	UTX8
U -0	U -0	U -0	U -0	U -0	U -0	U -0	W -x
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
UTX7	UTX6	UTX5	UTX4	UTX3	UTX2	UTX1	UTX0
W -x	W -x	W -x	W -x	W -x	W -x	W -x	W -x

บิต 15 ถึง 9 ไม่ใช้งาน กำหนดเป็น “0”

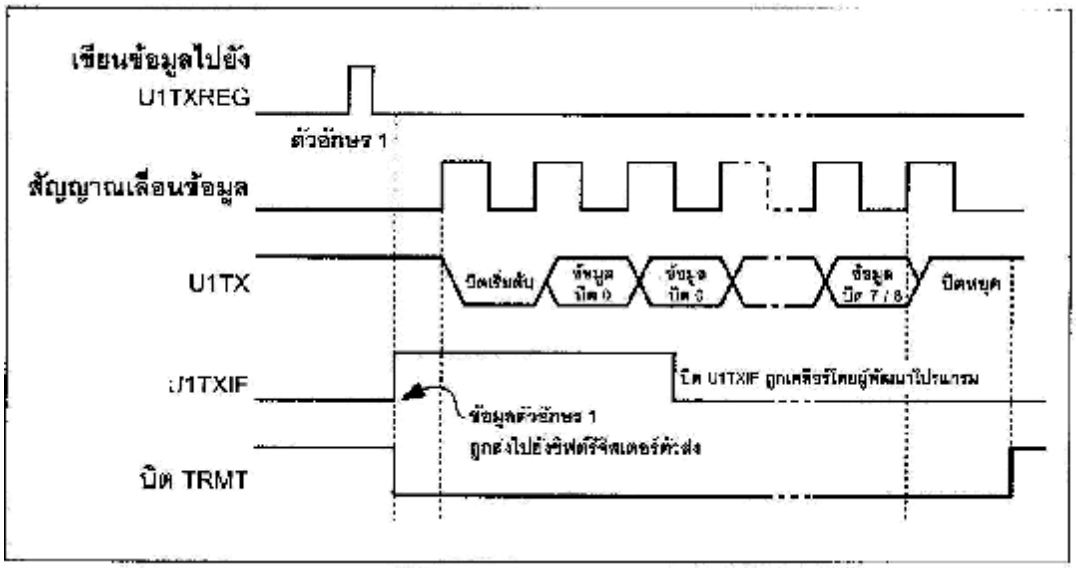
บิต 8 - UTX8 (Data bit 8 of the Character to be Transmitted in 9-bit mode): บิตข้อมูลที่ 9 หรือ บิต 8 เมื่อทำงานในโหมด 9 บิต

บิต 7 ถึง 0 - UTX7 ถึง UTX0 (Data bits 7-0 of the Character to be Transmitted): บิตข้อมูลขนาด 8 บิตที่ต้องการส่ง

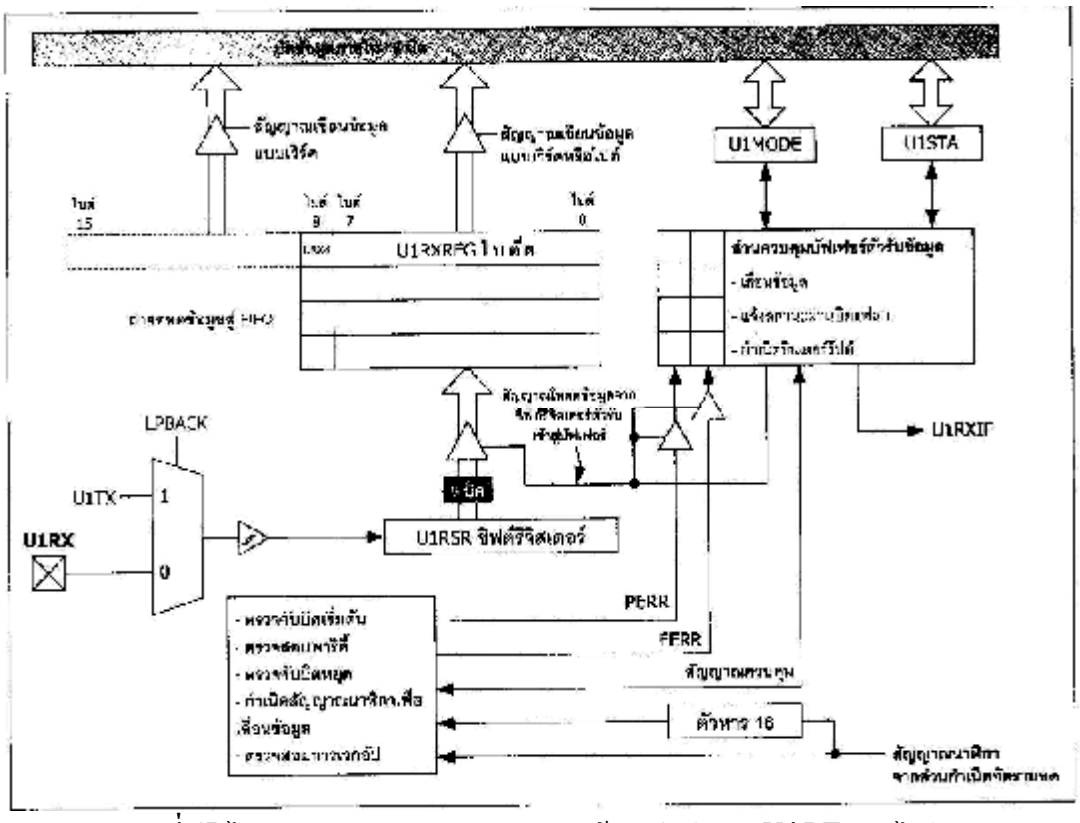
2.2.5 UXBRG (UARTX Baud Rate Register) เป็นรีจิสเตอร์ข้อมูลขนาด 16 บิต โดยข้อมูลที่ เขียนลงไปนั้นถูกใช้ในการคำนวณเพื่อกำหนดอัตราบอดให้แก่โมดูล UART



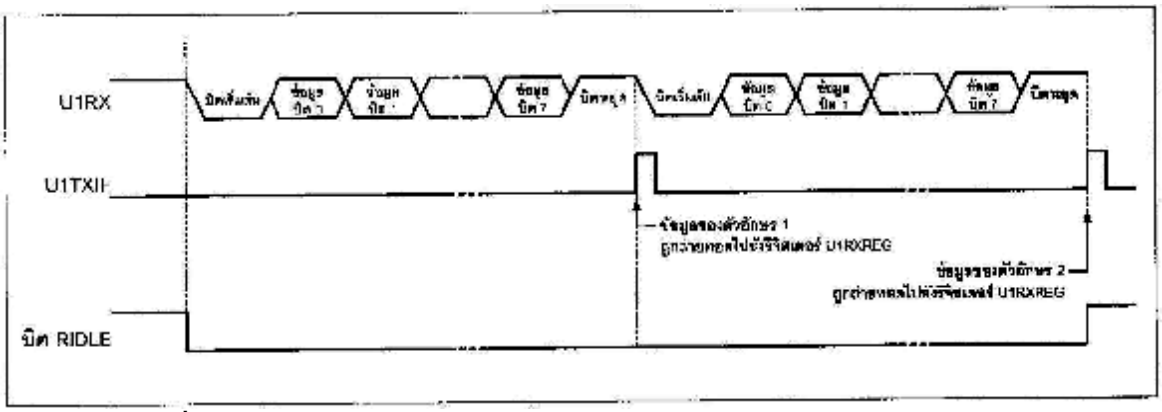
ภาพที่ 43 ไดอะแกรมการทำงานของตัวส่งข้อมูลในโมดูล UART ของไมโครคอนโทรลเลอร์ dsPIC



ภาพที่ 44 ไดอะแกรมเวลาของการส่งข้อมูลในโมดูล UART ของไมโครคอนโทรลเลอร์ dsPIC



ภาพที่ 45 ไดอะแกรมการทำงานของตัวรับข้อมูลในโมดูล UART ของไมโครคอนโทรลเลอร์ dsPIC



ภาพที่ 46 ไดอะแกรมเวลาของการรับข้อมูล 2 ชุดในโมดูล UART ของไมโครคอนโทรลเลอร์ dsPIC

2.3 การกำหนดให้โมดูล UART ทำงาน โมดูล UART ในไมโครคอนโทรลเลอร์ dsPIC ใช้รูปแบบข้อมูลในมาตรฐาน NRZ นั่นคือ มีบิตเริ่มต้น 1 บิต บิตข้อมูล 8 หรือ 9 บิต และบิตปิดท้าย 1 หรือ 2 บิต ส่วนบิตพาริตี้สามารถเลือกได้แบบคู่ (Even) คี่ (odd) หรือไม่มี (none) โดยปกติแล้วจะเลือกใช้รูปแบบ 8N1 คือ มีบิตเริ่มต้น 1 บิต,บิตข้อมูล 8 บิต และบิตปิดท้าย 1 บิต สำหรับ dsPIC30F2010 สามารถกำหนดได้ที่บิต PDSEL1, PDSEL0 และ STSEL ซึ่งเป็นบิต 2, 1 และ 0 ตามลำดับในรีจิสเตอร์ U1MODE ส่วนการกำหนด อัตราบอดนั้นกระทำผ่านรีจิสเตอร์ UXBRG ขนาด 16 บิต การรับฟังข้อมูลในโมดูล UART นั้นจะรับและส่งข้อมูลบิต LSB หรือบิตนัยสำคัญต่ำสุดก่อนโดยตัวรับและส่งข้อมูลในโมดูล UART ของ dsPIC จะทำงานเป็นอิสระแยกจากกัน โดยใช้อัตราบอดและรูปแบบข้อมูลที่เหมือนกัน จึงสามารถรับส่งข้อมูลได้ 2 ทิศทางพร้อมกันตลอดเวลา

2.3.1. การเอ็นเอเบิลโมดูล UART ใน dsPIC30F2010 ให้ทำงาน ทำได้โดยการเซตบิต UARTEN ซึ่งเป็นบิต 15 ในรีจิสเตอร์ U1MODE และเซตบิต UTXENซึ่งเป็นบิต 10 ในรีจิสเตอร์ U1STA ทันทีที่เอ็นเอเบิล ขา U1TX และ U1RX จะถูกกำหนดให้ทำงานเป็นขาพอร์ตเอาต์พุตและอินพุตตามลำดับ โดยไม่สนใจการกำหนดทิศทางที่เกิดขึ้นก่อนหน้านี้

2.3.2. การดิสเอเบิลโมดูล UART ใน dsPIC30F2010 ทำได้โดยการเคลียร์บิต UARTENซึ่งเป็นบิต 15 ในรีจิสเตอร์ U1MODE โดยปกติโมดูล UARTจะถูกดิสเอเบิลหลังจากที่เกิดการรีเซต ซึ่งเป็นการกำหนดสถานะในเบื้องต้น จากนั้นหากต้องการให้ทำงานจะต้องมีการเอ็นเอเบิลภายหลัง เมื่อโมดูล UART ถูกดิสเอเบิล ขา U1TX และ U1RX จะถูกปลดออกจากโมดูล UART สามารถนำไปใช้งานเป็นพอร์ตอินพุตเอาต์พุตทั่วไปได้ทันที นอกจากนั้นในภาวะที่โมดูล UART ถูกดิสเอเบิล ค่าในบัฟเฟอร์ทั้งหมดจะถูกเคลียร์ เช่นเดียวกับบิตแฟลกแจ้งสถานะความผิดพลาดทั้งหมดจะถูกเคลียร์ด้วย ส่วนตัวนับอัตราบอดจะอยู่ในภาวะรีเซตหลังจากที่ดิสเอเบิลโมดูล UART แล้วกลับมาเอ็นเอบิลใหม่ การทำงานทั้งหมดจะเริ่มต้นใหม่ทั้งหมด

2.3.3. ความสัมพันธ์ระหว่างสัญญาณนาฬิการะบบกับอัตราบอด ความสัมพันธ์ระหว่างความถี่สัญญาณนาฬิกาของระบบกับอัตราบอดในการสื่อสารข้อมูลอนุกรมนั้นสามารถแสดงในรูปของสมการดังนี้

$$\text{อัตราบอด} = \frac{F_{CY}}{16x(BRG+1)} \quad (9)$$

และ

$$F_{CY} = \frac{F_{OSC}}{4} \quad (10)$$

โดยที่ BRG คือ ค่าข้อมูลของรีจิสเตอร UxBRG (x เท่ากับ 1 หรือ 2 ขึ้นอยู่กับใช้โมดูล UART ชุดใด)

Fosc คือ ความถี่สัญญาณนาฬิกาหลัก ค่าอัตราบอดที่ใช้คือ 9,600 บิตต่อวินาที โดยกำหนดตัวแปรขึ้นมาตัวหนึ่งเพื่อเก็บค่าในที่นี้คือ Baudvalue โดยค่าที่ถูกกำหนดจะถูกโหลดไปยังรีจิสเตอร์ U1BRG อีกทอดหนึ่งสามารถคำนวณค่า Fcy และค่าของ BRG ได้ดังนี้

$$F_{CY} = \frac{F_{OSC}}{4} = \frac{7372800x4}{4} = 7372800 \quad (11)$$

$$\text{อัตราบอด} = \frac{F_{CY}}{16x(BRG+1)} \quad (12)$$

$$9600 = \frac{7372800}{16x(RGB+1)} \quad (13)$$

$$RGB = \frac{7372800}{(16x9600)} - 1 = 47 \quad (14)$$

ดังนั้นการกำหนดค่าเริ่มต้นของตัวแปร Baudvalue จึงใช้ค่าเป็น 47 ตามภาพที่ 47 โปรแกรมย่อย InitUART

```
void initUART()
{
 CloseUART1();   // Disable UART1 Before New Config
                        // Config UART1 Interrupt Control
 ConfigIntUART1(UART_RX_INT_EN &
        UART_RX_INT_PR2 &           // RX Interrupt Priority = 2
        UART_TX_INT_DIS &           // Disable TX Interrupt
        UART_TX_INT_PR6);           // TX Interrupt Priority = 3
                        // Open UART1 = Mode,Status,Baudrate
 OpenUART1(UART_EN    &             // Enable UART(UART Mode)
        UART_IDLE_STOP &            // Disable UART in IDLE Mode
        UART_ALTRX_ALTTX &          // Select U1TX=RC13,U1RX=RC14
        UART_DIS_WAKE &             // Disable Wake-Up
        UART_DIS_LOOPBACK &                 // Disable Loop Back
        UART_DIS_ABAUD &            // Disable Auto Baudrate
        UART_NO_PAR_8BIT &          // UART = 8 Bit, No Parity
        UART_1STOPBIT,                      // UART = 1 Stop Bit
                                            // Config UART1 Status
        UART_INT_TX_BUF_EMPTY&      // Select Interrupt After TX Complete
        UART_TX_PIN_NORMAL &        // Normal U1TX Mode
        UART_TX_ENABLE &            // Enable U1TX
        UART_INT_RX_CHAR &          // Flasg Set After RX Complete
        UART_ADR_DETECT_DIS &       // Disable Check Address
        UART_RX_OVERRUN_CLEAR,      // Clear Overrun Flag
        47);                        // Baudrate = 9600 BPS
}
```

ภาพที่ 47 โปรแกรมย่อย InitUART

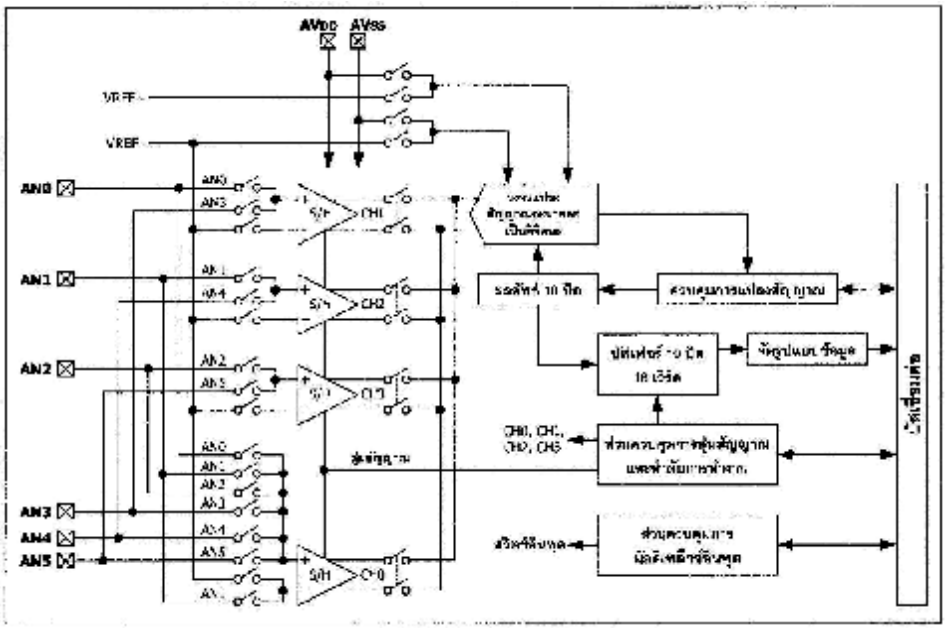
3. การใช้งานโมดูลแปลงสัญญาณอนาลอกเป็นดิจิตอลภายใน dsPIC

ในบทนี้นำเสนอตัวอย่างการเขียนโปรแกรมภาษา C เพื่อทดลองใช้งานโมดูลแปลงสัญญาณอนาลอกเป็นดิจิตอล (ADC) ภายในไมโครคอนโทรลเลอร์ dsPIC30F2010 ซึ่งบรรจุโมดูลแปลงสัญญาณอนาลอกเป็นดิจิตอลที่ มีความละเอียด 10 บิตจำนวน 6 ช่อง ที่ความละเอียดของการแปลงข้อมูล 10 บิต ทำให้ได้ข้อมูลดิจิตอลมีค่าตั้งแต่ 0 ถึง 1023 (2^10 หรือ 1,024 ค่า)

3.1. คุณสมบัติโดยสรุปของโมดูลแปลงสัญญาณอนาลอกเป็นดิจิตอล

1. เป็นโมดูลแปลงสัญญาณอนาลอกเป็นดิจิตอลที่มีความละเอียด 10 บิต จำนวน 6 ช่อง
2. ใช้วิธีการแปลงสัญญาณแบบประมาณค่าหรือซักเซสซีฟ แอ็ปพร็อกซิเมชั่น (Successive Approximation)
3. มีอัตราเร็วในการสุ่มสัญญาณสูงสุด 500 กิโลแซมเปิลต่อวินาที (ksps) หรือ 500,000 จุดตัวอย่างต่อวินาที
4. สามารถกำหนดให้ทำงานได้ขณะเข้าสู่โหมดสลีป (Sleep mode)
5. สามารถกำหนดระดับแรงดันอ้างอิงได้ทั้งจากภายในผ่านทางขา AVdd กับ AVss และภายนอกผ่านทางขา VREF+ และ VREF-

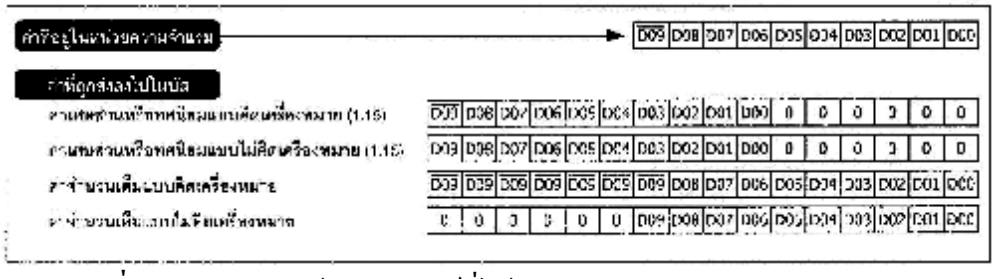
3.1.1 การทำงานเบื้องต้นของโมดูล ADC ใน dsPIC30F2010 ในภาพที่ 48 เป็นไดอะแกรมการทำงานของโมดูล ADC ใน dsPIC30F2010 ซึ่งมีขาพอร์ตอินพุตอนาลอกทั้งสิ้น 6 ขาคือ AN0-AN5 โดยมี 2 ขาที่สามารถใช้รับแรงดันอ้างอิง เพื่อขยายย่านของแรงดันอินพุต ภายในโมดูลมีวงจรสุ่มและเก็บค่าสัญญาณ (Sample and Hold: S/H) จำนวน 4 ชุด โดยทำงานร่วมกับส่วนควบคุมการมัลติเพล็กซ์สัญญาณอินพุตทำให้สามารถจัดสรรวงจร S/H ให้สามารถรองรับกับสัญญาณอินพุตอนาลอกทั้ง 6 ช่องได้ด้วยความเร็วสูงสุด



ภาพที่ 48 ไดอะแกรมการทำงานอย่างง่ายของโมดูล ADC ในไมโครคอนโทรลเลอร์ dsPIC30F2010

สัญญาณที่ผ่านจากวงจร S&H จะถูกป้อนเข้าสู่วงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลแบบซักเซสซีฟ แอ็ปพร็อกซิเมชั่น ขนาด 10 บิต ข้อมูลที่ได้จากการแปลงจะถูกพักไว้ในหน่วยความจำแรมจากนั้นจะได้รับการจัดรูปแบบตามที่ผู้พัฒนาโปรแกรมกำหนด ดังแสดงในภาพที่ 49 จากนั้นข้อมูลจะถูกถ่ายทอดลงบนบัสข้อมูลเพื่อส่งไปยังซีพียูต่อไป

อีกองค์ประกอบหนึ่งที่ทำให้โมดูล ADC สามารถแปลงสัญญาณได้รวดเร็วคือภายในโมดูล ADC มีบัฟเฟอร์ความจุ 16 เวิร์ด นั่นคือ สามารถรองรับข้อมูลที่ได้จากการแปลงสูงสุด 16 ชุดข้อมูล ดังนั้นเมื่อแปลงสัญญาณครั้งหนึ่งก็นำมาเก็บไว้ที่บัฟเฟอร์ หากบัฟเฟอร์ยังไม่เต็มก็สามารถกลับไปแปลงสัญญาณต่อได้ทันที โดยไม่ต้องรอให้การถ่ายทอดข้อมูลไปยังรีจิสเตอร์ที่ใช้เก็บค่าการแปลงเสร็จสิ้น



ภาพที่ 49 รูปแบบของข้อมูลผลลัพธ์ที่ได้จากการแปลงสัญญาณของโมดูล ADC

3.2. รีจิสเตอร์หลักที่ใช้ในโมดูล ADC

3.2.1. ADCON1 (A/D Control Register 1): รีจิสเตอร์ควบคุมโมดูล ADC ตัวที่ 1

บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
ADON	-	ADSIDL	-	-	-	FORM1	FORM0
R/W -0	U -0	R/W -0	U -0	U -0	U -0	R/W -0	R/W -0
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
SSRC2	SSRC1	SSRC0	-	SIMSAM	ASAM	SAMP	DONE
R/W -0	R/W -0	R/W -0	U -0	R/W -0	R/W -0	R/W -0	R/C -0

บิต 15 - ADON (A/D Operating Mode bit): บิตเลือกให้โมดูล ADC ทำงาน

"1" = เลือกให้โมดูล ADC ทำงาน

"0" = ปิดการทำงานของโมดูล ADC

บิต 14 ไม่ ใช้งาน กำหนดเป็น "0"

บิต 13 - ADSIDL (Stop in IDLE Mode bit): บิตกำหนดให้โมดูล ADC หยุดทำงานในโหมดไอเดิล

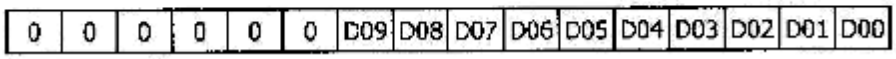
"0" = ยังคงให้โมดูล ADC ทำงานต่อไป แม้ว่าซีพียูจะเข้าสู่โหมดไอเดิลก็ตาม

"1" = หยุดการทำงานของโมดูล ADC ทันทีที่ซีพียูเข้าสู่โหมดไอเดิล

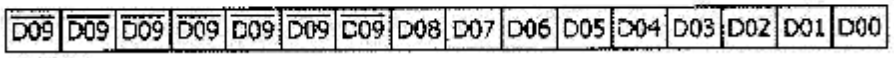
บิต 12 ถึง 10 ไม่ใช้งาน กำหนดเป็น "0"

บิต 9 และ 8 - FORM1 และ FORM0 (Data output Format bits): บิตเลือกรูปแบบข้อมูลเอาต์พุต

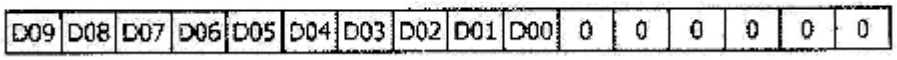
"00" – ข้อมูลเป็นเลขจำนวนเต็มแบบไม่คิดเครื่องหมาย (DOUT = 0000 00dd dddd dddd)



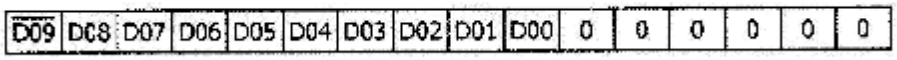
"01" = ข้อมูลเป็นเลขจำนวนเต็มแบบคิดเครื่องหมาย (DOUT = ssss sssd dddd dddd)



"10" = ข้อมูลเป็นเลขทศนิยมแบบไม่คิดเครื่องหมาย (DOUT = dddd dddd dd00 0000)



"11" = ข้อมูลเป็นเลขทศนิยมแบบคิดเครื่องหมาย (DOUT = sddd dddd dd00 0000)



บิต 7 ถึง 5 - SSRC2 ถึง SSRCO (Conversion Trigger Source Select bits): บิตเลือกแหล่งกำเนิดสัญญาณกระตุ้นให้โมดูล ADC แปลงสัญญาณ

"000" = เลือกให้กระตุ้นเมื่อเกิดการเคลียร์บิต SAMP

"001" = เลือกให้กระตุ้นเมื่อเกิดการเปลี่ยนแปลงสัญญาณที่ขา INT0

"010" = เลือกให้กระตุ้นเมื่อการเปรียบเทียบข้อมูลในไทเมอร์ 3 เสร็จสิ้น

"011" = เลือกให้กระตุ้นเมื่อเสร็จสิ้นการสุ่มสัญญาณจากโมดูล PWM ควบคุมมอเตอร์

"100" = สำรอง ไว้

"101" = สำรอง ไว้

"110" = สำรอง ไว้

"111" = เลือกให้กระตุ้นเมื่อตานับค่าภายในเสร็จสิ้นการสุ่มสัญญาณ และเริ่มต้นการแปลงสัญญาณ เป็นการกำหนดให้แปลงสัญญาณโดยอัตโนมัติ

บิต 4 ไม่ใช้งาน กำหนดเป็น "0"

บิต 3 - SIMSAM (Simultaneous sample select bit): บิตเลือกการสุ่มสัญญาณแบบทันทีทันใด จะใช้ก็ต่อเมื่อบิตCHPS = "01", "10" หรือ "11"

"0" = สุ่มสัญญาณเรียงลำดับตามหมายเลขช่องสัญญาณถ้าบิต CHPS = "10" หรือ "11"

“1” = เลือกสุ่มสัญญาณจากวงจร S&H ช่อง CH0, CH1, CH2, CH3 ทันทีทันใดถ้าบิต CHPS = “01”

“1” = เลือกสุ่มสัญญาณจากวงจร S&H ช่อง CH0 และ CH1 ทันทีทันใด

บิต 2 - ASAM (A/D Sample Auto-start bit): บิตกำหนดการเริ่มต้นสุ่มสัญญาณอัตโนมัติ

“0” = เริ่มสุ่มสัญญาณเมื่อบิต SAMP ถูกเซตเป็น “1”

“1” = เริ่มสุ่มสัญญาณทันทีที่การแปลงสัญญาณครั้งล่าสุดเสร็จสิ้นลง ทำให้บิตSAMPเซตอัตโนมัติ

บิต 1 - SAMP (A/D Sample Enable bit): บิตเอ็นเอเบิลการสุ่มสัญญาณของโมดูล ADC

“0” = เลือกให้พักการทำงานของวงจร S/H หรือดิสเอเบิลการสุ่มสัญญาณ

“1” = เลือกให้วงจร S/Hอย่างน้อยหนึ่งวงจรทำการสุ่มสัญญาณหรือเอ็นเอเบิลให้เกิดการสุ่มสัญญาณ

- เมื่อบิต ASAM = “0” การเขียนข้อมูล “1” มาที่บัดนี้จะเป็นการกระตุ้นให้เริ่มสุ่มสัญญาณ

- เมื่อบิต SSRC = “000” การเขียนข้อมูล “0” มาที่บิตนี้จะเป็นการกำหนดให้หยุดสุ่มสัญญาณแล้วเริ่มต้นกระบวนการ แปลง สัญญาณ

บิต 0 - DONE (A/D Conversion Status bit): บิตแสดงสถานะ การแปลงสัญญาณอนาลอกเป็นดิจิตอล

“0” = การแปลงสัญญาณยัง ไม่เสร็จสิ้น

“1" = การแปลงสัญญาณเสร็จสิ้น

บิตนี้สามารถเคลียร์ได้ด้วยกระบวนการทางซอฟต์แวร์หรือเมื่อเริ่มต้นการแปลงสัญญาณในรอบใหม่ นอกจากนั้นการเคลียร์บิตนี้จะไม่กระทบต่อการทำงานของวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลแต่อย่างใด

3.2.2. ADCON2 (A/D Control Register 2): รีจิสเตอร์ควบคุมโมดูล ADC ตัวที่ 2

บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
VCFG2	VCFG1	VCFG0	-	-	CSCNA	CHPS1	CHPS0
R/W -0	R/W -0	R/W -0	U -0	U -0	R/W -0	R/W -0	R/W -0
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
BUFS	-	SMPI3	SMPI2	SMPI1	SMPI0	BUFM	ALTS
R/W -0	U -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0

บิต 15 ถึง 13 - VCFG2 ถึง VCFG0 (Voltage Reference configuration bits) : บิตกำหนดแรงดันอ้างอิงสำหรับใช้ในโมดูล ADC

ค่าข้อมูล	แรงดันอ้างอิงด้านสูง V H	แรงดันอ้างอิงด้านตำ v
“000”	จากขา AVDD (ปกติมีค่า +5v)	จากขา AVSS (ปกติต่อลงกราวด์)
“001”	จากภายนอกผ่านขา VREF+	จากขา AVSS (ปกติต่อลงกราวด์)
“010”	จากขา AVDD (ปกติมีค่า +5v)	จากภายนอกผ่านขา VREF-
“011”	จากภายนอกผ่านขา VREF+	จากภายนอกผ่านขา VREF-
“1xx”	จากขา AVDD (ปกติมีค่า +5v)	จากขา AVSS (ปกติต่อลงกราวด์)

บิต 12 และ 11 ไม่ใช้งาน กำหนดเป็น “0”

บิต 10 - CSCNA (Scan input Selections for CH0+ S/H input for MUX A input Multiplexer Setting bit) : บิตเลือกการสแกนช่องสัญญาณของวงจร S/H ผ่านมัลติเพล็กเซอร์ A

“0” = ไม่มีการสแกนอินพุต
“1” = กำหนดให้สแกนอินพุต

บิต 9 และ 8- CHPS1 และ CHPS0 (Selects Channels utilized bits): บิตเลือกกลุ่มของช่องสัญญาณผ่านทางวงจร S/H

“00” = ต้องการแปลงวงจร S/H ช่อง CH1
“01” = ต้องการแปลงวงจร S/H ช่อง CH0 และ CH1
“1x” = ต้องการแปลงวงจร S/H ช่อง CH0, CH1, CH2 และ CH3

เมื่อบิต SIMSAM (บิต 3 ในรีจิสเตอร์ ADCON1) = “0” จะมีการสุ่มสัญญาณจากทุกช่อง
เมื่อบิต SMSAM = “1” การเลือกช่องสัญญาณจะกระทำผ่านบิต CHPS1 และ CHPS0.

บิต 7 - BUFS (Buffer Fill Status bit): บิตแสดงสถานะบัฟเฟอร์ จะมีการแสดงผลเกิดขึ้นเมื่อบิต BUFM ถูกเซตเป็น “1” (รีจิสเตอร์ ADRES แบ่งเป็น 2 ส่วน ส่วนละ 8 เวิร์ด)
“0” = แจ้งว่าขณะนี้โมดูล ADC กำลังเขียนข้อมูลลงในบัฟเฟอร์ที่ตำแหน่ง 0x0-0x7 ผู้พัฒนาโปรแกรมสามารถเข้าถึงข้อมูลในบัฟเฟอร์ที่ตำแหน่ง 0x8-0xF ได้
“1” = แจ้งว่าขณะนี้โมดูล ADC กำลังเขียนข้อมูลลงในบัฟเฟอร์ที่ตำแหน่ง 0x8-0xF ผู้พัฒนาโปรแกรมสามารถเข้าถึงข้อมูลในบัฟเฟอร์ที่ตำแหน่ง 0x0-0x7 ได้

บิต 6 ไม่ใช้งาน กำหนดเป็น “0”
บิต 5 ถึง 2 - SMPI3 ถึง SMPI0 (Sample/Convert Sequences Per interrupt Selection bits): บิตเลือกการเกิดอินเตอร์รัปต์ในกระบวนการสุ่มและแปลงสัญญาณในโมดูล ADC

“0000” = เกิดอินเตอร์รัปต์เมื่อการแปลงสัญญาณเสร็จสิ้นทุกลำดับของกระบวนการสุ่มและแปลงสัญญาณ

“0000” = เกิดอินเตอร์รัปต์เมื่อการแปลง สัญญาณเสร็จทุกๆลำดับที่ 2 ของกระบวนการสุ่มและแปลงสัญญาณ

“0001” = เกิดอินเตอร์รัปต์เมื่อการแปลงสัญญาณเสร็จทุกๆลำดับที่ 3 ของกระบวนการสุ่มและแปลงสัญญาณ

“0010” = เกิดอินเตอร์รัปต์เมื่อการแปลงสัญญาณเสร็จทุกๆลำดับที่ 4 ของกระบวนการสุ่มและแปลงสัญญาณ

“0011” = เกิดอินเตอร์รัปต์เมื่อการแปลงสัญญาณเสร็จทุกๆลำดับที่ 5 ของกระบวนการสุ่มและแปลงสัญญาณ

“0100” = เกิดอินเตอร์รัปต์เมื่อการแปลงสัญญาณเสร็จทุกๆลำดับที่ 6 ของกระบวนการสุ่มและแปลงสัญญาณ

“0101” = เกิดอินเตอร์รัปต์เมื่อการแปลงสัญญาณเสร็จทุกๆลำดับที่ 7 ของกระบวนการสุ่มและแปลงสัญญาณ

“0111” = เกิดอินเตอร์รัปต์เมื่อการแปลงสัญญาณเสร็จทุกๆลำดับที่ 8 ของกระบวนการสุ่มและแปลงสัญญาณ

“1000” = เกิดอินเตอร์รัปต์เมื่อการแปลงสัญญาณเสร็จทุกๆลำดับที่ 9 ของกระบวนการสุ่มและแปลงสัญญาณ

“1001” = เกิดอินเตอร์รัปต์เมื่อการแปลงสัญญาณเสร็จทุกๆลำดับที่ 10 ของกระบวนการสุ่มและแปลงสัญญาณ

“1010” = เกิดอินเตอร์รัปต์เมื่อการแปลงสัญญาณเสร็จทุกๆลำดับที่ 11 ของกระบวนการสุ่มและแปลงสัญญาณ

“1011” = เกิดอินเตอร์รัปต์เมื่อการแปลงสัญญาณเสร็จทุกๆลำดับที่ 12 ของกระบวนการสุ่มและแปลงสัญญาณ

“1100” = เกิดอินเตอร์รัปต์เมื่อการแปลงสัญญาณเสร็จทุกๆลำดับที่ 13 ของกระบวนการสุ่มและแปลงสัญญาณ

“1101” = เกิดอินเตอร์รัปต์เมื่อการแปลงสัญญาณเสร็จทุกๆลำดับที่ 14 ของกระบวนการสุ่มและแปลงสัญญาณ

“1110” = เกิดอินเตอร์รัปต์เมื่อการแปลงสัญญาณเสร็จทุกๆลำดับที่ 15 ของกระบวนการสุ่มและแปลงสัญญาณ

“1111” = เกิดอินเตอร์รัปต์เมื่อการแปลงสัญญาณเสร็จทุกๆลำดับที่ 16 ของกระบวนการสุ่มและแปลงสัญญาณ

บิต 1 - BUFM (Buffer Mode Select bit): บิตเลือกโหมดของบัฟเฟอร์

“0” = กำหนดให้บัฟเฟอร์มีความจุ 16 เวิร์ด มีชื่อเป็น ADCBUF0 ถึง ADCBUF15

“1” = แบ่งบัฟเฟอร์ใน 2 ส่วน ส่วนละ 8 เวิร์ด คือ รีจิสเตอร์ ADCBUF8 ถึง 15 และรีจิสเตอร์ADCBUF7 ถึง ADCBUF0

บิต 0 - ALTS (Alternate input Sample Mode Select bit): บิตเลือกโหมดการทำงานของอินพุตมัลติเพล็กเซอร์

“1” = เลือกใช้อินพุต A สำหรับการสุ่มสัญญาณครั้งแรก จากนั้นสลับกันระหว่างอินพุต A และ B

“0” = เลือกใช้อินพุต A ตลอดการทำงาน

3.2.3. ADCON3 (A/D ControI Register 3) : รีจิสเตอร์ควบคมโมดูล ADC ตัวที่ 3

บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
-	-	-	SAMC4	SAMC3	SAMC2	SAMC1	SAMC0
U -0	U -0	U -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
ADRC	-	ADCS5	ADCS4	ADCS3	ADCS2	ADCS1	ADCS0
R/W -0	U -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0

บิต 15 ถึง 13 ไม่ใช้งาน กำหนดเป็น “0”

บิต 12 ถึง 8 - SAMC4 ถึง SAMC0 (Auto-Sample Time bits) : บิตเลือกฆ่าเวลาในการสุ่มสัญญาณอัตโนมัติ

“00000” = 0 Tad (กำหนดได้ในกรณีที่เลือกใช้วงจร S/H มากกว่าหนึ่งวงจร)

“00001” = 1 Tad
“00010” = 2 Tad
.....
“11111” = 31 Tad

บิต 7- ADRC (A/D Conversion Clock Source bit) : บิตเลือกแหล่งกำเนิดสัญญาณนาฬิกาสำหรับการแปลงสัญญาณ

“0” = ใช้จากสัญญาณนาฬิกาหลักของระบบ
“1” = ได้จากวงจร RC ภายานโมดูล ADC

บิต 6 ไม่ใช้งาน กำหนดเป็น “0”

บิต 5 ถึง 0 - ADCS5 ถึง ADCS0 (A/D conversion clock select bits) : บิตเลือกฆ่าเวลาที่ใช้ในการแปลงสัญญาณ

“000000” = 0.5Tcy x (000000 + 1) = 0.5Tcy
“000001” = 0.5Tcy x (000001 + 1) = Tcy
.....
“111111” = 0.5Tcy x (111111 + 1) = 32 x Tcy

สัญญาณนาฬิกาของ ADC	เลือกสัญญาณนาฬิกา		ค่าเวลาในการแปลงสัญญาณของโมดูล ADC ที่ความถี่สัญญาณนาฬิกาต่างๆ				
	ADRC	ADCSS ADCS0	Fcy = 30MHz	Fcy = 30MHz	Fcy = 12.5MHz	Fcy = 6.25MHz	Fcy = 1MHz
Tcy/2	0	000000	16.67 nSec	20 nSec	40 nSec	80 nSec	500 nSec
Tcy/2	0	000001	33.3 nSec	40 nSec	80 nSec	160 nSec	1 nSec
Tcy/2	0	000011	66.66 nSec	80 nSec	160 nSec	320 nSec	2 nSec
Tcy/2	0	000111	133.32 nSec	160 nSec	320 nSec	640 nSec	4 nSec
Tcy/2	0	001111	533.32 nSec	320 nSec	640 nSec	1.28 nSec	8 nSec
Tcy/2	0	011111	533.28 nSec	640 nSec	1.28 uSec	2.56 uSec	16 nSec
Tcy/2	0	111111	1066.56 nSec	1280 nSec	2.56 uSec	5.12 uSec	32 nSec
Tcy/2	1	xxxxxx	200-400 nSec	200-400 nSec	200-400 nSec	200-400 nSec	200-400 nSec

ตารางที่ 1 แสดงการกำหนดค่าเวลาที่ใช้ในการแปลงสัญญาณในโมดูล ADC

3.2.4. ADCHS (A/D input Select Register): รีจิสเตอร์เลือกช่องของวงจร S/H ที่ต่อกับขาพอร์ตอินพุตอนาลอกที่ต้องการแปลงสัญญาณ

บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
CH123NB1	CH123NB0	CH123SB	CH0NB	CH0SB3	CH0SB2	CH0SB1	CH0SB0
R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
CH123NA1	CH123NA0	CH123SA	CH0NA	CH0SA3	CH0SA2	CH0SA1	CH0SA0
R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0

บิต 15 และ 14 - CH123NB1 และ CH123NB0 (channel 1, 2, 3 Negative input select for MUX B Multiplexer setting bits): บิตเลือกช่องอินพุตลบของวงจร S&H ช่อง CH1, CH2 และ CH3 สำหรับมัลติเพล็กเซอร์ B

“11” = อินพุตลบของ CH1 ต่อกับอินพุต AN9, อินพุตลบของ CH2 ต่อกับอินพุต AN10, อินพุตลบของ CH3 ต่อกับอินพุต AN11

“10” = อินพุตลบของ CH1 ต่อกับอินพุต AN6, อินพุตลบของ CH2 ต่อกับอินพุต AN7, อินพุตลบของ CH3 ต่อกับอินพุต AN8

“00” และ “01” = อินพุตลบของ CH1, CH2 และ CH3 ต่อกับ VREF- ให้ใช้ภาพที่ 48 ประกอบ

บิต 13 - CH123SB (channel 1, 2, 3 positive lnput select for MUX B Multiplexer setting bit): บิตเลือกช่องอินพุตบวกของวงจร S&H ช่อง CH1, CH2 และ CH3 สำหรับมัลติเพล็กเซอร์ B

“1” = CH1 positive input is AN3 , CH2 positive input is AN4 , CH3 positive input is AN5

“0” = อินพุตบวกของ CH1 ต่อกับอินพุต AN0 , อินพุตบวกของ CH2 ต่อกับอินพุต AN1 ,อินพุตบวกของ CH1 ต่อกับอินพุต AN2

บิต 12 - CH123SB (Channel 0 Negative lnput select for MUX B Multiplexer setting bit): บิตเลือกช่องอินพุตลบของวงจร S&H ช่อง CH0 สำหรับมัลติเพล็กเซอร์ B

“0” = อินพุตลบของ CH0 ต่อกับ VREF-
“1” = อินพุตลบของ CH0 ต่อกับ AN1

บิต 11 ถึง 8 CH0SB3 ถึง CH0SB0 (channel 0 positive input select for MUX B Multiplexer setting bits): บิตเลือกช่องอินพุตบวกของวงจร S&H ช่อง CH0 สำหรับมัลติเพล็กเซอร์ B

“0000” = อินพุตบวกของ CH0 ต่อกับขา AN0
“0001” = อินพุตบวกของ CH0 ต่อกับขา AN1
.....
“1110” = อินพุตบวกของ CH0 ต่อกับขา AN14
“1111” = อินพุตบวกของ CH0 ต่อกับขา AN15

บิต 7 และ 6 - CH123NA1 และ CH123NA0 (channel 1, 2, 3 Negative input select for MUX A Multiplexer setting bits): บิตเลือกช่องอินพุตลบของวงจร S&H ช่อง CH1, CH2 และ CH3 สำหรับมัลติเพล็กเซอร์ A

“11” = อินพุตลบของ CH1 ต่อกับอินพุต AN9, อินพุตลบของ CH2 ต่อกับอินพุต AN10, อินพุตลบของ CH3 ต่อกับอินพุต AN11
“10” = อินพุตลบของ CH1 ต่อกับอินพุต AN6, อินพุตลบของ CH2 ต่อกับอินพุต AN7, อินพุตลบของ CH3 ต่อกับอินพุต AN8
“00” และ “01” = อินพุตลบของ CH1, CH2 และ CH3 ต่อกับ VREF-

บิต 5 - CH123SA (channel 1, 2, 3 positive lnput select for MUX A Multiplexer setting bit): บิตเลือกช่องอินพุตบวกของวงจร S&H ช่อง CH1, CH2 และ CH3 สำหรับมัลติเพล็กเซอร์ A

“1” = CH1 positive input is AN3, CH2 positive input is AN4, CH3 positive input is AN5

“0” = อินพุตบวกของ CH1 ต่อกับอินพุต AN0, อินพุตบวกของ CH2 ต่อกับอินพุต AN1, อินพุตบวกของ CH3 ต่อกับอินพุต AN2

บิต 4 - CH0NA (channel 0 Negative input select for MUX A Multiplexer setting bit): บิตเลือกช่องอินพุตลบของวงจร S&H ช่อง CH0 สำหรับมัลติเพล็กเซอร์ A

“0” = อินพุตลบของ CH0 ต่อกับ VREF-
“1” = อินพุตลบของ CH0 ต่อกับ AN1

บิต 3 ถึง 0 - CH0SA3 ถึง CH0SA0 (channel 0 positive input select for MUX A Multiplexer setting bits): บิตเลือกช่องอินพุตบวกของวงจร S&H ช่อง CH0 สำหรับมัลติเพล็กเซอร์ A

“0000” = อินพุตบวกของ CH0 ต่อกับขา AN0
“0001” = อินพุตบวกของ CH0 ต่อกับขา AN1
.....
“1101” = อินพุตบวกของ CH0 ต่อกับขา AN13
“1110” = อินพุตบวกของ CH0 ต่อกับขา AN14
“1111” = อินพุตบวกของ CH0 ต่อกับขา AN15

หมายเหตุ ส่วนอินพุตมัลติเพล็กเซอร์ สามารถกำหนดการทำงานทางอินพุตได้ 2รูปแบบ คือ มัลติเพล็กเซอร์A (MUXA) และมัลติเพล็กเซอร์ B (MUXB) โดยบิต 15 ถึง 8 ของรีจิสเตอร์ ADCHS ใช้ตั้งค่าสำหรับ MUXBส่วนบิต 7 ถึง 0 ของรีจิสเตอร์ ADCHS ใช้ทั้งค่าสำหรับ MUX A

3.2.5. ADPCFG (A/D port configuration Register): รีจิสเตอร์กำหนดค่าทางฮาร์ดแวร์ของพอร์ตอินพุตอนาลอก

บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8
R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0
R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0

รีจิสเตอร์ตัวนี้สามารถรองรับการกำหนดค่าของพอร์ตอินพุตอนาลอกได้ครบทั้ง 16 ช่อง (กรณีใช้ dsPIC เบอร์ใหญ่ที่มีอินพุตอนาลอกครบ 16 ช่อง) โดยการกำหนดนี้จะอย่างกันอย่างอิสระ เริ่มจากบิต 15(PCFG15) สำหรับกำหนดค่าของอินพุต AN15 ไล่ไปตามลำดับจนถึงบิต0 (PCFG0) สำหรับกำหนดค่าของอินพุต AN0 สำหรับ dsPIC30F2010 มี 6 ช่อง จึงใช้งานเพียง 6 บิต คือ PCFG0-PCFG5

“0” = กำหนดให้อินพุตอนาลอกทำงานในโหมดอนาลอก

“1" = กำหนดให้อินพุตอนาลอกทำงานในโหมดดิจิตอล

3.2.6. ADCSSL (A/D input scan select Register) : รีจิสเตอร์เลือกช่องอินพุตที่ต้องการแปลง สัญญาณ

บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
CSSL15	CSSL14	CSSL13	CSSL12	CSSL11	CSSL10	CSSL9	CSSL8
R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
CSSL7	CSSL6	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0
R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0

รีจิสเตอร์ตัวนี้ใช้กำหนดให้วงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลมาอ่านค่าจากอินพุตแบบเรียงลำดับอัตโนมัติ ซึ่งสามารถเลือกได้ว่า ต้องการให้อ่านค่าจากช่องใดบ้าง การกำหนดค่าของรีจิสเตอร์ตัวนี้สามารถรองรับอินพุตอนาลอกได้ครบทั้ง 16 ช่อง (กรณีใช้ dsPIC เบอร์ใหญ่ที่มีอินพุตอนาลอกครบ 16 ช่อง) โดยการกำหนดนี้จะอย่างกันอย่างอิสระ เริ่มจากบิต 15

(CSSLI5) สำหรับกำหนดค่าของอินพุต AN15 ไล่ไปตามลำดับจนถึงบิต 0 (CSSL0) สำหรับกำหนดค่าของอินพุต AN0 สำหรับ dsPIC30F2010 มี 6 ช่อง จึงใช้งานเพียง 6 บิตคือ CSSL0-CSSL5

“0” = ไม่อ่านค่าจากอินพุตนี้
“1” = เลือกให้อ่านค่าอินพุต

3.2.7. บัฟเฟอร์เก็บผลลัพธ์จากการแปลงสัญญาณของโมดูล ADC มีชื่อเรียกว่า ADCBUF เป็นหน่วยความจำแรมขนาด 16 บิต มีทั้งสิ้น 16 ตัว หรืออาจกล่าวได้ว่า เป็นบัฟเฟอร์ 16 เวิร์ด จึงสามารถกำหนดชื่อเรียกได้เป็น ADCBUF0, ADCBUF1, ADCBUF2, ..., ADCBUFE,

ADCBUFF ใช้สำหรับเก็บค่าผลลัพธ์ที่ได้จากการแปลงสัญญาณของวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลชั่วคราว ก่อนที่จะส่งต่อไปจัดรูปแบบข้อมูล แล้วถ่ายทอดต่อไปยังบัสของระบบต่อไป

3.3. การกำหนดค่าเพื่อใช้งานโมดูล ADC มีขั้นตอนโดยสรุปดังนี้

3.3.1. ตั้งค่าของโมดูล ADC

1. เลือกขาพอร์ตให้ทำงานเป็นอินพุตอนาลอกที่รีจิสเตอร์ ADPCFG
2. เลือกแหล่งจ่ายแรงดันอ้างอิงให้เหมาะสมกับย่านแรงดันอนาลอกทางอินพุตที่บิต 15 ถึง 13 ของรีจิสเตอร์ ADCON2
3. คือสัญญาณนาฬิกาสำหรับการแปลงสัญญาณที่บิต 5 ถึง 0 ของรีจีสเตอร์ ADCON3
4. กำหนดจำนวนช่องของวงจร S/H ที่ต้องใช้ที่บิต 9 และ 8 ของรีจิสเตอร์ ADCON2และรีจิสเตอร์ ADPCFG
5. กำหนดวิธีการสุ่มสัญญาณที่บิต 3 ของรีจิสเตอร์ ADCON1 และรีจิสเตอร์ ADCSSL
6. กำหนดจำนวนอินพุตที่ต้องทำงานร่วมกับวงจร S/H ที่รีจีสเตอร์ ADCHS

7. เลือกลำดับการสุ่มและแปลงสัญญาณที่บิต 7 ถึง 0 ของรีจิสเตอร์ ADCON1 และบิต 12 ถึง 8 ของรีจีสเตอร์ ADCON3

8. เลือกรูปแบบของผลลัพธ์ที่ต้องการที่บิต 9 และ 8 ของรีจิสเตอร์ ADCON

9. เลือกการอินเตอร์รัปต์ที่บิต 9 ถึง 5 ของรีจิสเตอร์ ADCON2

10. เปิดการทำงานของโมดูล ADC ที่บิต 15 ของรีจิสเตอร์ ADCON1

3.3.2 กำหนดการอินเตอร์รัปต์ (ถ้าต้องการ)

1. เคลียร์บิต ADIF

2. เลือกระดับความสำคัญของการอินเตอร์รัปต์

ในภาพที่ 50 คือโปรแกรมย่อย InitADC โดยเปิดทัง 6 ช่องและใช้การอินเทอรัพท์ในการอ่านด้วย

```
void InitADC (void)
{
        ADCON1bits.FORM = 0;
        ADCON1bits.SSRC = 7;
        ADCON1bits.ASAM = 1;
        ADCON2bits.SMPI = 0;
        ADCON3bits.SAMC = 1;
        ADCON3bits.ADRC = 1;
        IPC2bits.ADIP = 1;
        ADCHS = 0x0;            //เลือก CH
        ADCSSL = 0x0000;        //Channel Scanning is disabled. All bits left to their default
state
    //เปิด adc ช่องที่ต้องการ
        ADPCFG = 0xFFFF;
        ADPCFGbits.PCFG5 = 0;
        ADPCFGbits.PCFG4 = 0;
        ADPCFGbits.PCFG3 = 0;
        ADPCFGbits.PCFG2 = 0;
        ADPCFGbits.PCFG1 = 0;
        ADPCFGbits.PCFG0 = 0;
```

ภาพที่ 50 โปรแกรมย่อย InitADC

4. การใช้งานไทเมอร์ใน dsPIC

ไทเมอร์เคาน์เตอร์หลักใน dsPIC มี 5 ตัวคือ ไทเมอร์ 1 ถึงไทเมอร์ 5 โดยแต่ละตัวมีขนาด 16 บิต สำหรับไทเมอร์ 2 และ 3 กับไทเมอร์ 4 และ 5 สามารถทำงานรวมกันเป็นไทเมอร์ขนาด 32 บิตเมื่อไทเมอร์แต่ละตัวทำงานแยกกันสามารถกำหนดการทำงานได้อีก 3 แบบตามลักษณะของฐานเวลาคือ ฐานเวลาแบบ A, B และ C ซึ่งจะได้กล่าวถึงในลำดับถัดไป

4.1. คุณสมบัติของไทเมอร์ ใน dsPIC30F2010 มีไทเมอร์/เคาน์เตอร์ขนาด 16 บิตให้ใช้งาน รวม 3 ตัว คือ ไทเมอร์ 1 (T1) , ไทเมอร์ 2 (T2) และไทเมอร์ 3 (T3)

4.1.1. คุณสมบัติของไทเมอร์ 1

1. รีจิสเตอร์ตัวนับความละเอียด 16 บิต
2. ทำงานได้ทั้งแบบซิงโครนัสและอะซิงโครนัสเคาน์เตอร์
3. ทำงานร่วมกับขาอินพุตประจำตัวของไทเมอร์ได้
4. มีปรีสเกลเลอร์สำหรับหารความถี่การนับ
5. สามารถกำหนดการอินเตอร์รัปต์จากการนับหรือจากการตรวจพบสัญญาณขอบขาลงที่ขาอินพุตของไทเมอร์

4.1.2. คุณสมบัติของไทเมอร์ 2 และ 3

1. ไทเมอร์ 2 และ3 เมื่อทำงานแยกอิสระต่อกัน มีคุณสมบัติคล้ายกับไทเมอร์ 1
2. เมื่อนำไทเมอร์ 2 และ 3 มาทำงานเริ่มกัน รีจิสเตอร์ตัวนับมีความละเอียดเพิ่มเป็น 32 บิต
3. ทำงานร่วมกับขาอินพุตประจำตัวไทเมอร์ได้ (ขา TxCKI)
4. มีปรีสเกลเลอร์สำหรับหารความถี่การนับ
5. สามารถกำหนดการอินเตอร์รัปต์จากการนับหรือจากการตรวจพบสัญญาณขอบขาลงที่ขาอินพุตของไทเมอร์
6. สามารถกำเนิดสัญญาณกระตุ้นการทำงานไปยังโมดูล ADC ได้

4.2. รีจิสเตอร์ในโมดูลไทเมอร์ของ dsPIC30F ในโมดูลไทเมอร์ของ dsPIC30F2010 มีรีจิสเตอร์ขนาด 16 บิตที่สำคัญอยู่ 3 กลุ่มคือ

กลุ่มที่ 1 TMRx (16-bit timer count register) เป็นกลุ่มของรีจิสเตอร์เก็บค่าไทเมอร์ กลุ่มนี้5 ตัวคือ TMR1, TMR2, TMR3, TMR4, และ TMR5 แต่สำหรับ dsPIC30F2010มี 3 ตัวคือ TMR1 ถึงTMR 3

กลุ่มที่ 2 PRx (16-bit period register associated with the timer) เป็นกลุ่มของรีจิสเตอร์คาบเวลาที่สัมพันธ์กับไทเมอร์ มี 5 ตัวเช่นกัน PR1 ถึง PR5 แยกกันตามไทเมอร์หลักทั้ง 5 ตัว (ไทเมอร์ 1 ถึงไทเมอร์ 5) แต่สำหรับ dsPIC30F2010 มี 3 ตัวคือ PR1 ถึง PR3

กลุ่มที่ 3 TxCON (16-bit control register associated with the timer) เป็นกลุ่มของรีจิสเตอร์ที่ใช้ควบคุมการทำงานของไทเมอร์ มี 5 ตัวเช่นกัน T1CON ถึง T5CON แยกกันตามไทเมอร์หลักทั้ง
ตัว สำหรับกลุ่มนี้จะมีความพิเศษตรงที่ยังสามารถแยกย่อยออกไปได้ 3 แบบภายใต้ชื่อรีจิสเตอร์เดียวกันเนื่องจากไทเมอร์สามารถทำงานได้อีก 3 แบบตามลักษณะของฐานเวลา จึงทำให้รีจิสเตอร์ TxCONสามารถระบุได้เป็น TxCON ในแบบ A, B และ C เวลาเขียนโปรแกรมเพื่อกำหนดค่าให้แก่รีจิสเตอร์TxCON จะต้องระมัดระวังในจุดนี้ด้วย

4.2.1. TxCON Type A Time Base Register:รีจิสเตอร์ควบคุมไทเมอร์ด้วยฐานเวลาแบบ A

บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
TON	-	TSIDL	-	-	-	-	-
R/W -0	U -0	R/W -0	U -0	U -0	U -0	U -0	U -0
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
-	TGATE	TCKPS1	TCKPS0	-	TSYNC	TCS	-
U -0	R/W -0	R/W -0	R/W -0	U -0	R/W -0	R/W -0	U -0

บิต 15 - TON (TImer on control bit): บิตควบคุมการเปิดไทเมอร์ให้ทำงาน

“0” = หยุดการทำงานของ ไทเมอร์
“1” = เลือกให้ไทเมอร์เริ่มทำงาน

บิต 14 ไม่ใช้งาน กำหนดเป็น “0”

บิต 13 - TSIDL (Stop in IDLE Mode bit): บิตกำหนดให้ไทเมอร์หยุดทำงานในโหมดไอเดิล

“0” = ยังคงให้ไทเมอร์ทำงานต่อไป แม้ว่าซีพียูจะเข้าสู่โหมดไอเดิลก็ตาม
“1” = หยุดการทำงานของไทเมอร์ทันทีที่ซีพียูเข้าโหมดไอเดิล

บิต 12 ถึง 7 ไม่ใช้งาน กำหนดเป็น “0”

บิต 6 - TGATE (Timer Gated Time Accumulah1on Enable bit): บิตเอ็นเอเบิลการเปิดเกตเพื่อรับสัญญาณจาก ภายนอก

“0” = ดิสเอเบิลการทำงานแบบนี้
“1” = เอ็นเอเบิลการทำงานแบบนี้

เมื่อเลือกบิตนี้เป็น “1” บิต TCS (บิต 1 ของรีจิสเตอร์ TxCON) ต้องกำหนดเป็น “0” อ่านค่าบิตนี้เป็น “0” ถ้าบิต TCS เป็น “1”

บิต 5 และ 4 - TCKPS1 และ TCKPS0 (Timer input clock presale select bits): บิตเลือกอัตราปรีสเกลเลอร์สัญญาณนาฬิกาของไทเมอร์

“00” = เลือกอัตรา 1:1
“01” = เลือกอัตรา 1:18
“10” = เลือกอัตรา 1:64
“11” = เลือกอัตรา 1:256

บิต 3 ไม่ใช้งาน กำหนดเป็น “0”

บิต2-TSYNC (Timer External Clock input Synchronization Select bit): บิตเลือกการซิงโครไนซ์ของสัญญาณนาฬิกาจากภายนอกกับการทำงานของไทเมอร์ต้องพิจารณาสถานะของบิต TCS ร่วมด้วย ดังนี้ เมื่อบิต TCS = “0”

บิตนี้ไม่ใช้งาน อ่านค่าได้เป็น “0” เนื่องจากเมื่อบิต TCS เป็น “0” หมายความว่า ไทเมอร์เลือกใช้สัญญาณนาฬิกาภายในในการกำหนดจังหวะการทำงานเมื่อบิต TCS = “1”

“0” = เลือกไม่ต้องการซิงโครไนซ์

“1” = เลือกให้เกิดซิงโครไนซ์สัญญาณนาฬิกาภายนอกกับการทำงานของไทเมอร์

บิต 1 - TCS (Timer Clock Source Select bit): บิตเลือกแหล่งกำเนิดสัญญาณนาฬิกาของไทเมอร์

“0” = เลือกใช้สัญญาณนาฬิกาภายใน ความถี่ FOSC/4

“1” = เลือกใช้สัญญาณนาฬิกาจากภายนอกผ่านทางขา TxCKI

บิต 0 ไม่ใช้งาน กำหนดเป็น “0”

4.2.2. TxCON Type B Time Base Register:รีจิสเตอร์ควบคุมไทเมอร์ด้วยฐานเวลาแบบ B

บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
TON	-	TSIDL	-	-	-	-	-
R/W -0	U -0	R/W -0	U -0	U -0	U -0	U -0	U -0
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
-	TGATE	TCKPS1	TCKPS0	T32	-	TCS	-
U -0	R/W -0	R/W -0	R/W -0	R/W -0	U -0	R/W -0	U -0

บิต 15 - TON (Timer on Control bit): บิตควบคุมการเปิดไทเมอร์ให้ทำงาน พิจารณาสถานะของบิต T32 ร่วมด้วย ดังนี้ เมื่อบิต T32 = “1” เป็นการเลือกให้ทำงานเป็นไทเมอร์ 32 บิต

“0” = หยุดการใช้งานคู่รีจิสเตอร์ 32 บิต TMRx และ TMRy

“1” = เริ่มต้นใช้งานคู่รีจิสเตอร์ 32 บิต TMRx และ TMRy

เมื่อบิต T32 = “0” เป็นการเลือกให้ทำงานเป็นไทเมอร์ 16 บิต

“0” = หยุดการทำงานของไทเมอร์ 16 บิต

“1” = เริ่มการทำงานของไทเมอร์ 16 บิต

บิต 14 ไม่ใช่งาน กำหนดเป็น “0”

บิต 13 - TSIDL (Stop in IDLE Mode bit): บิตกำหนดให้ไทเมอร์หยุดทำงานในโหมดไอเดิล

“0” = ยังคงให้ไทเมอร์ทำงานต่อไป แม้ว่าซีพียูจะเข้าสู่หมดไอเดิลก็ตาม

“1” = หยุดการทำงานของไทเมอร์ทันทีที่ซีพียูเข้าโหมดไอเดิล

บิต 12 ถึง 7 ไม่ใช้งาน กำหนดเป็น "0”

บิต 6 - TGATE (Timer Gated Time Accumulation Enable bit): บิตเอ็นเอเบิลการรับสัญญาณเกตจากภายนอก

“0” = ดิสเอเบิลการทำงานแบบนี้

“1” = เอ็นเอเบิลการทำงานแบบนี้

เมื่อเลือกบิตนี้เป็น “1” บิต TCS ต้องกำหนดเป็น “0” อ่านค่าได้เป็น “0” ถ้าบิต TCS เป็น “1”

บิต 5 และ 4 - TCKPS1 และ TCKPS0 (Timer input Clock Presale Select bits): บิตเลือก อัตราปรีสเกลเลอร์สัญญาณนาฬิกาของไทเมอร์

“00” = เลือกอัตรา 1:1

“01” = เลือกอัตรา 1:8

“10” = เลือกอัตรา 1:64

“11” = เลือกอัตรา 1:256

บิต 3 - T32 (32-bit Timer Mode Select bits): บิตเลือกโหมดไทเมอร์ 32 บิต

“0” = กำหนดให้ไทเมอร์ TMRx และ TMRy แยกการทำงานเป็นไทเมอร์ 16 บิต

“1” = กำหนดให้ไทเมอร์ TMRx และ TMRy ทำงานร่วมกันเป็นไทเมอร์ 32 บิต

บิต 2 ไม่ใช้งาน กำหนดเป็น “0”

บิต 1 - TCS (Timer clock source select bit): บิตเลือกแหล่งกำเนิดสัญญาณนาฬิกาของไทเมอร์

“0” = เลือกใช้สัญญาณนาฬิกาภายใน ความถี่ Fosc/4

“1” = เลือกใช้สัญญาณนาฬิกาจากภายนอกผ่านทางขา TxCKI

บิต 0 ไม่ใช่งาน กำหนดเป็น “0”

4.2.3. TxCON Type C Time Base Register: รีจิสเตอร์ควบคุมไทเมอร์ด้วยฐานเวลาแบบ C

บิต 15	บิต 14	บิต 13	บิต 12	บิต 11	บิต 10	บิต 9	บิต 8
TON	-	TSIDL	-	-	-	-	-
R/W -0	U -0	R/W -0	U -0	U -0	U -0	U -0	U -0
บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
-	TGATE	TCKPS1	TCKPS0	-	-	TCS	-
U -0	R/W -0	R/W -0	R/W -0	U -0	U -0	R/W -0	U -0

บิต 15 - TON (Timer on Control bit): บิตควบคุมการเปิดไทเมอร์ให้ทำงาน

“0” = หยุดการทำงานของไทเมอร์ 16 บิต

“1” = เริ่มต้นการทำงานของไทเมอร์ 16 บิต

บิต 14 ไม่ใช้งาน กำหนดเป็น “0”

บิต 13 - TSIDL (stop in IDLE Mode bit): บิตกำหนดให้ไทเมอร์หยุดทำงานในโหมดไอเดิล

“0” = ยังคงให้ไทเมอร์ทำงานต่อไป แม้ว่าซีพียูจะเข้าสู่โหมดไอเดิลก็ตาม
“1” = หยุดการทำงานของไทเมอร์ทันทีที่ซีพียูเข้าโหมดไอเดิล

บิต 12 ถึง 7 ไม่ใช้งาน กำหนดเป็น “0”

บิต 6 - TGATE (Timer Gated Time Accumulation Enable bit): บิตเอ็นเอเบิลการรับสัญญาณเกตจากภายนอก

“0” = ดิสเอเบิลการทำงานแบบนี้
“1” = เอ็นเอเบิลการทำงานแบบนี้

เมื่อเลือกบิตนี้เป็น “1” บิต TCS (บิต 1 ของรีจิสเตอร์ TxCON) ต้องกำหนดเป็น “0” อ่านค่าบิตนี้เป็น “0” ถ้าบิต TCS เป็น “1”

บิต 5 และ 4 - TCKPS1 และ TCKPS0 (Timer input Clock Presale Select bits): บิตเลือกอัตราปรีสเกลเลอร์สัญญาณนาฬิกาของไทเมอร์

“00” = เลือกอัตรา 1:1
“01” = เลือกอัตรา 1:8
“10” = เลือกอัตรา 1:64
“11” = เลือกอัตรา 1:256

บิต 3 และ 2 ไม่ใช้งาน กำหนดเป็น “0”

บิต 1 - TCS (Timer Clock Source Select bit): บิตเลือกแหล่งกำเนิดสัญญาณนาฬิกาของไทเมอร์

“0” = เลือกใช้สัญญาณนาฬิกาภายใน ความถี่ Fosc/4
“1” = เลือกใช้สัญญาณนาฬิกาจากภายนอกผ่านทางขา TxCKI

บิต 0 ไม่ใช้งาน กำหนดเป็น “0”

4.3. การทำงานของไทเมอร์ตามลักษณะของฐานเวลา มีทั้งสิ้น 3 แบบคือ ไทเมอร์ที่ทำงานด้วยฐานเวลาแบบ A, B และ C มีข้อมูลโดยสรุปดังนี้

4.3.1. ไทเมอร์ที่ทำงานด้วยฐานเวลาแบบ A เป็นการทำงานในแบบมาตรฐานของ dsPIC โดยไทเมอร์ที่ทำงานในแบบนี้คือ ไทเมอร์ 1 ซึ่งมีคุณสมบัติพิเศษเพิ่มเติมดังนี้

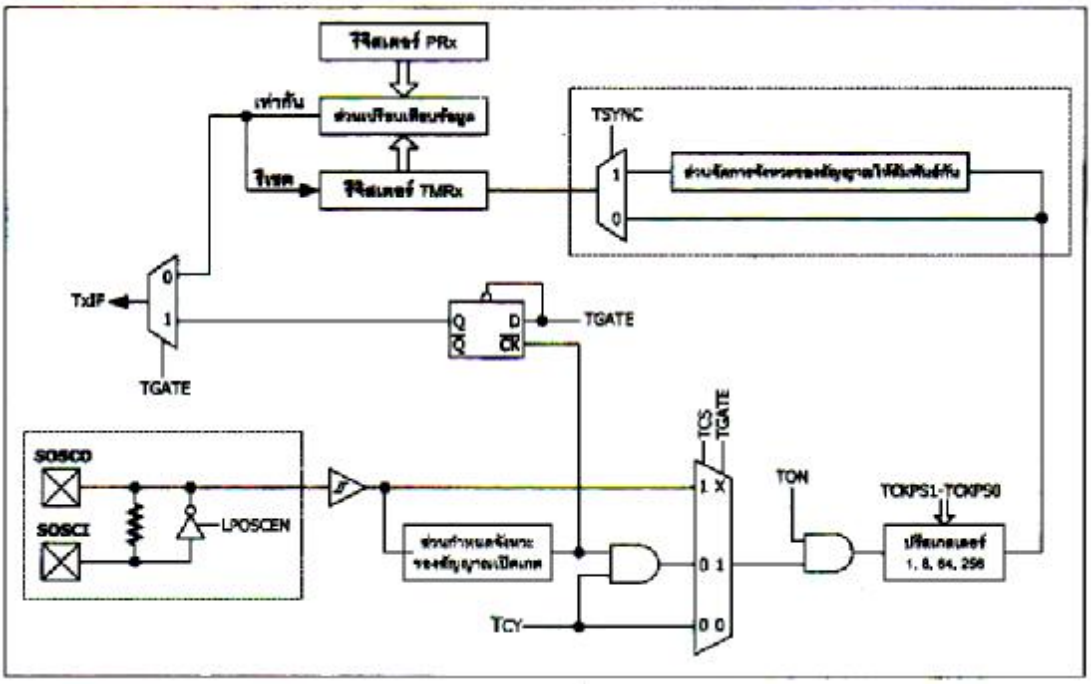
1. กำหนดให้ทำงานกับแหล่งจ่ายสัญญาณนาฬิกากำลังงานต่ำ 32kHz จากภายนอกได้
2. กำหนดทำงานในแบบอะซิงโครนัสโดยรับสัญญาณนาฬิกาจากภายนอกได้

ดังนั้นไทเมอร์ 1 จึงเหมาะอย่างยิ่งในการนำมาใช้สร้างฐานเวลานาฬิกาจริง (RTC) ในภาพที่ 51 แสดงไดอะแกรมการทำงานพื้นฐานของไทเมอร์ 1 ที่ทำงานด้วยฐานเวลาแบบ A

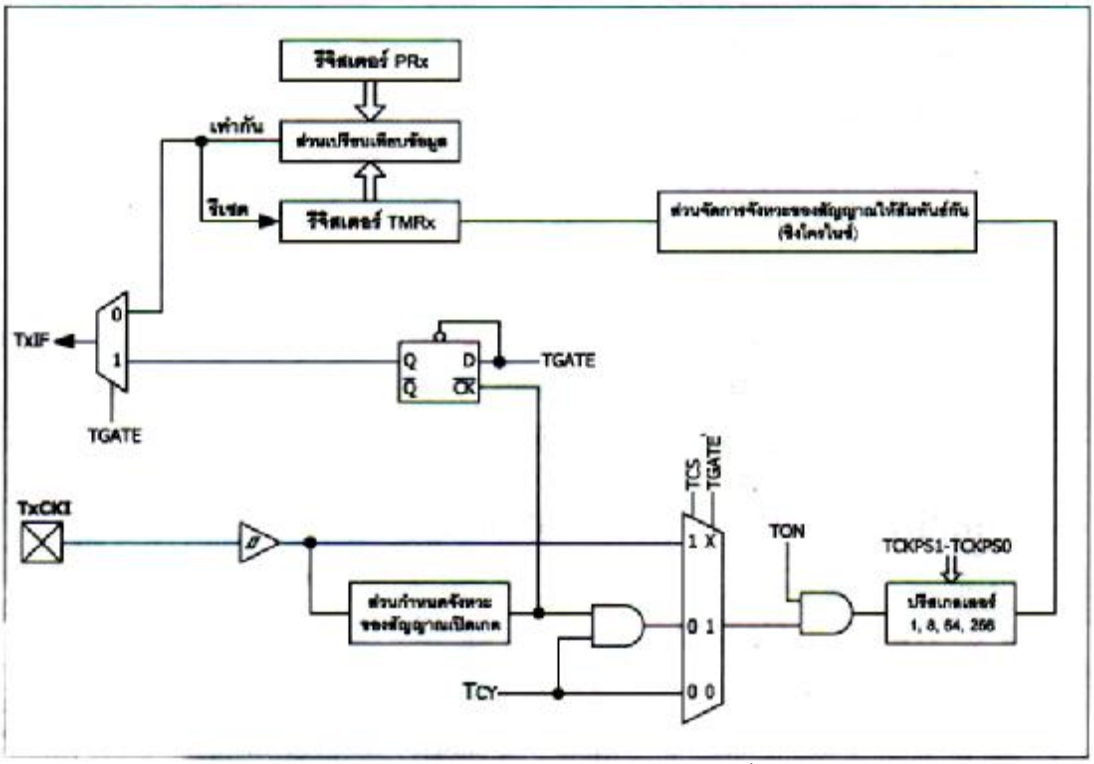
4.3.2 .ไทเมอร์ที่ทำงานด้วยฐานเวลาแบบ B ไทเมอร์ที่สามารถทำงานในลักษณะนี้ได้มี 2 ตัวคือ ไทเมอร์ 2 และไทเมอร์ 4 (เฉพาะไทเมอร์4 ไม่มีใน dsPIC30F2010) คุณสมบัติที่เพิ่มเติมเข้ามาสำหรับฐานเวลาแบบ B คือ

1. สามารถต่อกับไทเมอร์ฐานเวลาแบบ C เพื่อใช้งานเป็นไทเมอร์ขนาด 32 บิตได้ โดยการกำหนดสถานะที่บิต T32 ในรีจิสเตอร์ TxCON เมื่อทำงานด้วยฐานเวลาแบบ B ให้เป็นลอจิก “1”
2. มีวงจรกำหนดจังหวะของสัญญาณนาฬิกาให้สัมพันธ์กัน (ซิงโครไนซ์) หลังจากผ่านการปรับอัตราลดทอนความถี่ด้วยปรีสเกลเลอร์แล้ว

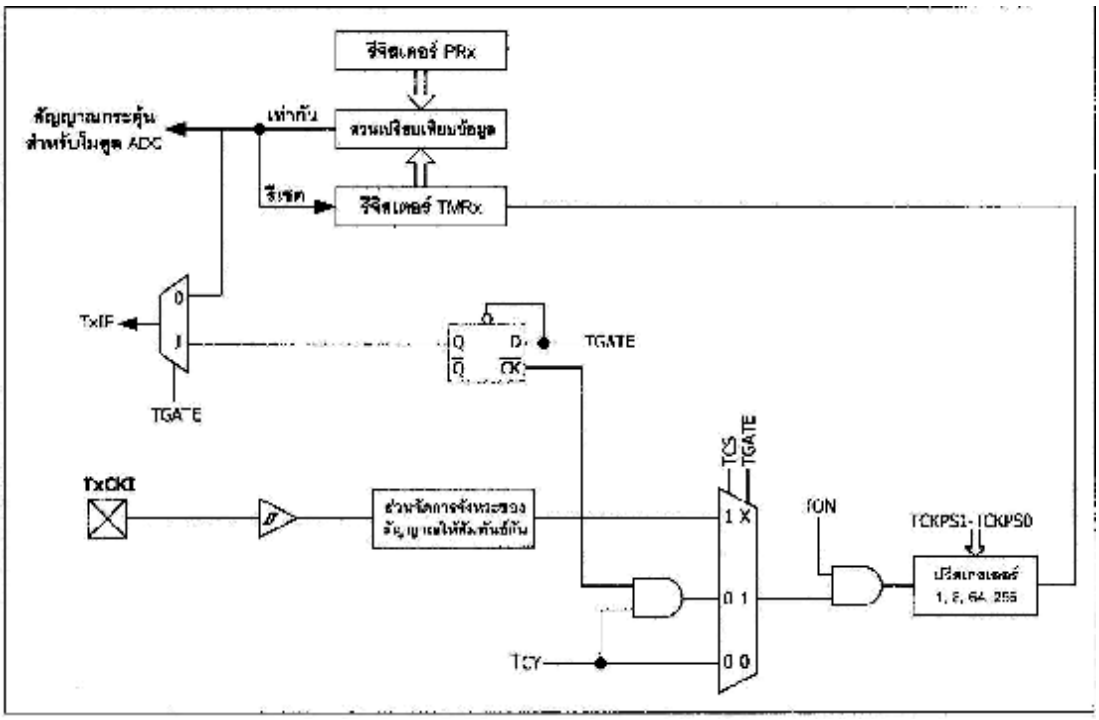
ในภาพที่ 52 แสดงไดอะแกรมการทำงานพื้นฐานของไทเมอร์ที่ทำงานด้วยฐานเวลาแบบ B



ภาพที่ 51 ไดอะแกรมการทำงานของไทเมอร์ 1 ซึ่งทำงานด้วยฐานเวลาแบบ A



ภาพที่ 52 ไดอะแกรมการทำงานของไทเมอร์ 2 และไทเมอร์ 4 ซึ่งทำงานด้วยฐานเวลาแบบ B



ภาพที่ 53 ไดอะแกรมการทำงานของไทเมอร์ 3 และไทเมอร์ 5 ซึ่งทำงานด้วยฐานเวลาแบบ C

4.3.3. ไทเมอร์ที่ทำงานด้วยฐานเวลาแบบ C ไทเมอร์ที่สามารถทำงานในลักษณะนี้ได้มี 2 ตัวคือ ไทเมอร์ 3 และไทเมอร์ 5 (เฉพาะไทเมอร์ 5 ไม่มีใน dsPIC30F2010) คุณสมบัติที่เพิ่มเติมเข้ามาสำหรับฐานเวลาแบบ C คือ

1. สามารถต่อกับไทเมอร์ฐานเวลาแบบ B เพื่อใช้งานเป็นไทเมอร์ 32 บิตได้ โดยการกำหนดสถานะที่บิต T32 ในรีจิสเตอร์ TxCON เมื่อทำงานด้วยฐานเวลาแบบ B ให้เป็นลอจิก “1”

2. สามารถสร้างสัญญาณกระตุ้นการทำงานของโมดูลแปลงสัญญาณอะนาลอกเป็นดิจิตอลได้ ทำให้ไทเมอร์และโมดูลแปลงสัญญาณอะนาลอกเป็นดิจิตอลทำงานสัมพันธ์กัน

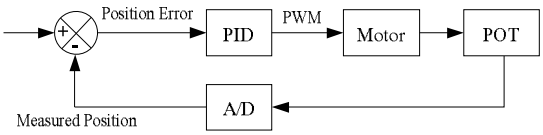
ในภาพที่ 53 แสดงไดอะแกรมการทำงานพื้นฐานของไทเมอร์ที่ทำงานด้วยฐานเวลาแบบ C และใน ภาพที่ 54 โปรแกรมย่อย InitTIMER1โดยกำหนดให้เกิดการอินเตอร์รัปท์ ทุกๆ 1ms.

```
void IntTimer1(void)
{       unsigned int match_value;
        ConfigIntTimer1(T1_INT_PRIOR_1 &        // Timer1 interrupt priority 1
        T1_INT_ON);                             // Enable interrupt for timer1
        WriteTimer1(0);                         // Clear count value at TMR1 register
        match_value = 29491;                    // Load value Interval 1 ms
        OpenTimer1(T1_ON &                      // Start timer1
                T1_GATE_OFF &                   // Disable gate pin for timer1
                T1_IDLE_STOP &                  // Stop timer in idle mode
                T1_PS_1_1 &                     // Presceler 1:1
                T1_SYNC_EXT_OFF &               // Disable sync external source
                T1_SOURCE_INT, match_value);
}
```

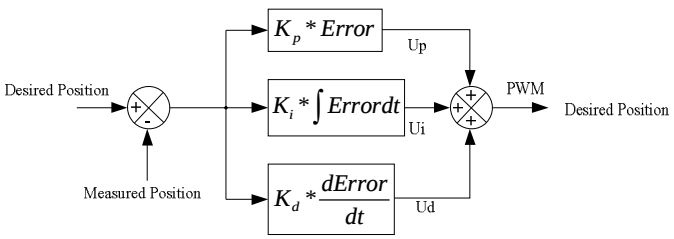
ภาพที่ 54 โปรแกรมย่อย InitTIMER1

5. ออกแบบระบบควบคุมแบบ พีไอดี

จากภาพที่ 55 เป็นระบบควบคุมแบบลิเนียร์ พีไอดี กำหนดให้ En = Set-point – ตำแน่งที่อ่านจากโพเทนชิออมิเตอร์ (Measured Position) แล้วส่งผลลัพธ์ ออกไปให้วงจรขับมอเตอร์ ในรูปแบบของความถี่ PWM ภาพที่ 56 บล๊อกอะแกรมของระบบควบคุมตำแหน่งของมอเตอร์แบบดิจิตอล พีไอดี



ภาพที่ 55 บล๊อกอะแกรมของระบบควบคุมตำแหน่งของมอเตอร์แบบลิเนียร์ พีไอดี



ภาพที่ 56 บล๊อกอะแกรมของระบบควบคุมตำแหน่งของมอเตอร์แบบดิจิตอล พีไอดี

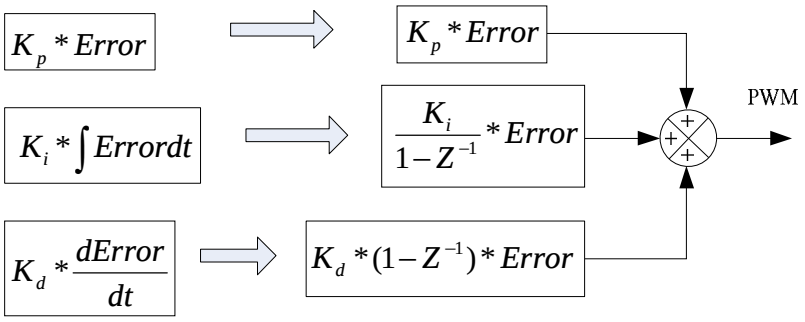
จากระบบควบคุมในภาพที่ 56 บล๊อกอะแกรมของระบบควบคุมตำแหน่งของมอเตอร์แบบลิเนียร์พีไอดี ต้องการออกแบบระบบควบคุมด้วย dsPIC นั้นต้องเริ่มจากเปลี่ยนจากระบบลิเนียร์พีไอดี เป็นแบบดิจิตอล พีไอดี เป็นดิจิตอล ก่อนเพราะว่าไม่โคคอนโทรเลอร์ทำงานในระบบดิจิตอล ด้วย Z ทรานฟอร์ม จะได้ สมการระบบใหม่ใน ภาพที่ 57 การแปลงลิเนียร์พีไอดีเป็นดิจิตอลพีไอดีด้วย Z ทรานฟอร์ม

$$Up(T) = Kp * Error(T) \tag{15}$$

$$Ui(T) = Ki * Error(T) + Ui(T-1) \tag{16}$$

$$Ud(T) = Kd * (Error(T) - Error(T-1)) \tag{17}$$

$$PWM(T) = Up + Ui + Ud \tag{18}$$



ภาพที่ 57 การแปลงลิเนียร์พีไอดีเป็น ดิจิตอลพีไอดีด้วย Z ทรานฟอร์ม

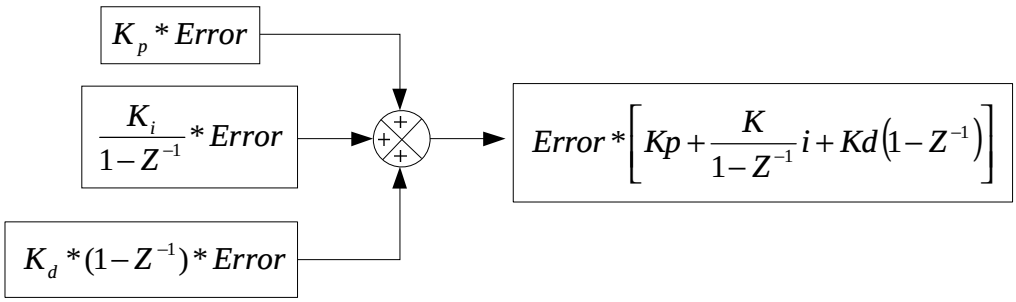
หลังจากได้ระบบ พีไอดี ในรูปแบบดิจิตอลแล้วก็ทำการจัดรูปสมการใหม่ จะได้

$$Error * \left[Kp + \frac{Ki}{(1-Z^{-1})} + Kd(1-Z^{-1}) \right] = PWM \tag{19}$$

$$Error * \frac{\left[Kp(1-Z^{-1}) + Ki + Kd(1-Z^{-1})^2 \right]}{(1-Z^{-1})} = PWM \tag{20}$$

$$Error * \frac{\left[(Kp + Ki + Kd) + (-Kp - 2 * Kd)Z^{-1} + KdZ^{-2} \right]}{(1-Z^{-1})} = PWM \tag{21}$$

$$Error * \frac{\left[(Kp + Ki + Kd) + (-Kp - 2 * Kd)Z^{-1} + KdZ^{-2} \right]}{(1-Z^{-1})} = PWM \tag{22}$$



ภาพที่ 58 ดิจิตอลพีไอดี

เมื่อจัดรูปสมการเสร็จแล้วก็ แทนค่า ตามสมการที่ 23-28 เพื่อที่จะสามารถนำไปเขียนเป็นโปรแกรมในไมโครคอนโทรเลอร์ได้

$$Error = Error(T) \tag{23}$$

$$Error * Z^{-1} = Error(T-1) \tag{24}$$

$$Error * Z^{-2} = Error(T-2) \tag{25}$$

$$Error * \frac{\left[(Kp + Ki + Kd) + (-Kp - 2 * Kd)Z^{-1} + KdZ^{-2}\right]}{(1-Z^{-1})} = PWM \tag{26}$$

$$\begin{aligned} PWM = {} & PWM(T-1) \\ & + Error(T) * K1 \\ & + Error(T-1) * K2 \\ & + Error(T-2) * K3 \end{aligned} \tag{27}$$

เมื่อ

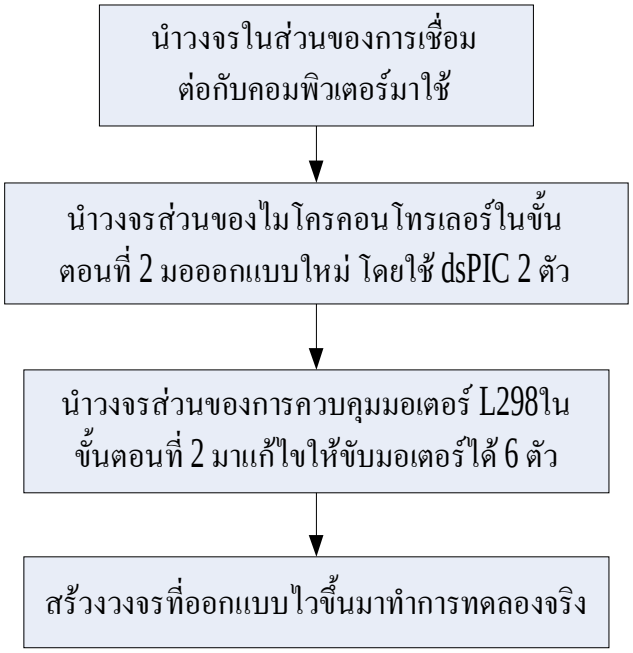
$$\begin{aligned} K1 &= Kp + Ki + Kd \\ K2 &= -Kp - 2 * Kd \\ K3 &= Kd \end{aligned} \tag{28}$$

หลังจากนั้นนำสมการที่ 26 ถึงสมการที่ 28 ไปเขียนลงในบอร์ด dsPIC30F2010 ตามภาพที่ 58 โปรแกรมย่อยคำนวณค่าพีไอดีเพื่อคำนวณหาค่า PWM ซึ่งในโปรแกรมนี้คือค่าในรีจิสเตอร์ PGC มีหน้าที่ไปควบคุมสัญญาณ PWM ในโมดูล และต้องมีการจำกัดขนาด $U \le abs(2PTPER)$ คือค่าทำให้ความถี่ โดยที่ U=2*PTPER คือ PWM = 100%

```
int PID(long ErrorT , long ErrorT_1 , long ErrorT_2 ,long UT_1 , long Kp , long Ki , long Kd)
{       int UT=0;
        long U32=0;
        long Up , Ui , Ud;
        int max,min;
        Up = ErrorT*(Kp+Ki+Kd);
        Ui = ErrorT_1*(-Kp-(2*Kd));
        Ud = ErrorT_2*(Kd);
        U32 = UT_1 + Up + Ui + Ud;
        //จำกัดขนาดของ Uไม่ให้เกินค่าของ Hardware pwm PTPER = 2048;
        max=PTPER*2;    min=PTPER*-2;
        if (U32 > max)      U32 = max;
        if (U32 < min )     U32 = min;
        UT = abs((long)U32);
        if (U32 < 0) UT = -1*abs((long)U32);
return UT;
}
```

ภาพที่ 58 โปรแกรมย่อยคำนวณค่าพีไอดี

ขั้นตอนที่ 4. ออกแบบและสร้างบอร์ดควบคุมมอเตอร์ 6 ตัว ให้อยู่ในบอร์ดเดียว



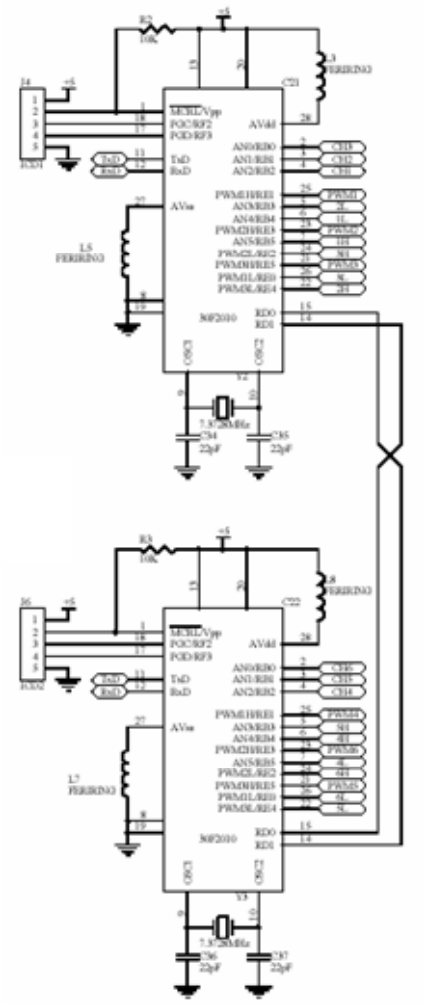
ภาพที่ 59 โฟลชาร์จแสดงขั้นตอนการออกแบบและสร้างบอร์ดควบคุมมอเตอร์ 6 ช่อง

หลังจากผ่านขั้นตอนที่ 2 การออกแบบต้นแบบบอร์ดไมโครคอนโทรเลอร์ และขั้นที่ 3 ทดลองเขียนโปรแกรมเพื่อควบคุมมอเตอร์ 1 ตัว แล้ว ต่อไปเป็นขั้นตอนที่ 4 การออกแบบและสร้างบอร์ดควบคุมมอเตอร์ 6 ช่อง ให้อยู่ในบอร์ดเดียวกันขั้นแรกต้องเริ่มจากกำหนดสเป็คของบอร์ดคล่าวๆ ก่อน

1. สามารถควบคุมมอเตอร์ DC ได้ 6 ช่อง อย่างอิสระต่อกัน
2. มอเตอร์ทุกๆ ตัวสามารถกลับทางหมุนได้
3. รองรับกระแสไฟได้อย่างน้อยตัวละ 1 A
4. สามารถปรับความเร็วของมอเตอร์ได้ 6 ช่อง
5. สามารถอ่านค่าสัญญาณไฟฟ้าที่ออกมาจากโพเทนชิออมิเตอร์ในข้อต่อได้ 6 ซ่อง
6. เชื่อมต่อกับคอมพิวเตอร์ผ่านพอร์ตอนุกรมได้ (RS232)

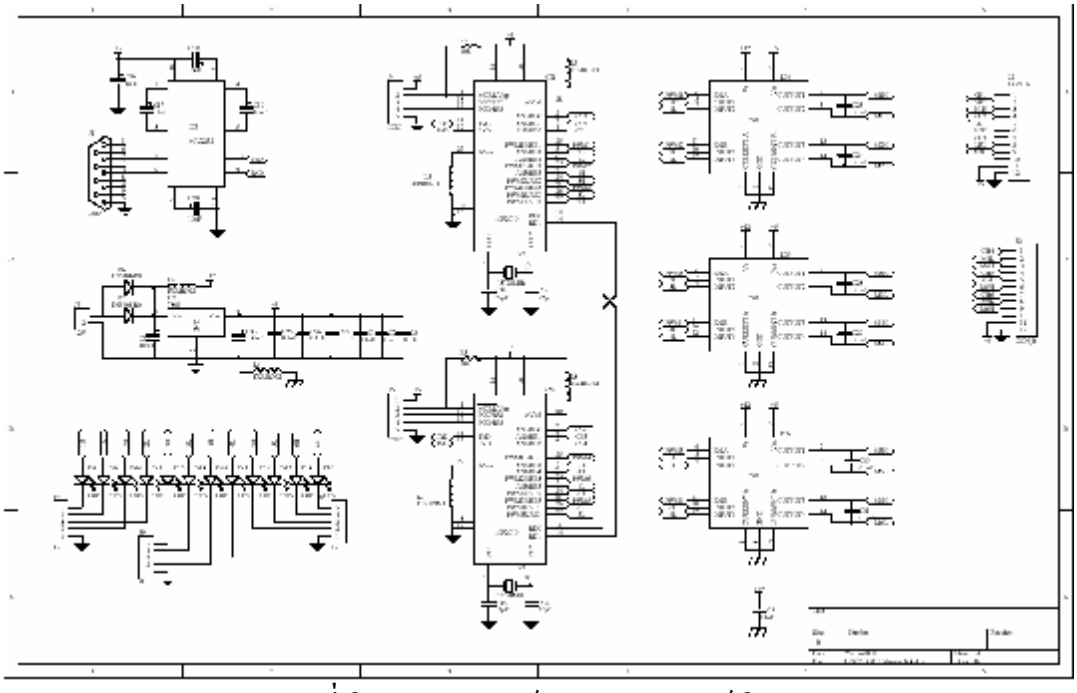
ในเงื่อนไขข้อ 1 – 3 เป็นวงจรแบบเดียวกันและคล้ายกับ ขั้นที่ 2 ซึ่งต่างกันที่จำนวณมอเตอร์ที่ต้องการควบคุมวงจรในภาพที่ 34 วงจรควบคุมมอเตอร์ด้วย L298 ได้โดยเพื่อมเป็น 3 ตัว เพื่อให้ควบคุมอมเตอร์ได้ 6 ช่อง

ข้อที่ 4 และ 5 ส่วนนี้ต้องใช้ไมโครคอนโทรเลอร์เบอร์ dsPIC30F2010 2 ตัว เพราะว่าไมโครคอนโทรเลอร์ 1 ตัว จะมีPWM 3 โมดูล ที่อิสระต่อกัน ในบอร์ดควบคุมตัวใหม่นี้จริงต้องใช้ 2 ตัวเพื่อให้ได้ PWM 6 โมดูล ตามภาพที่ 60 การต่อไมโครคอลโทรเลอร์ 2 ตัว โดยมี2 บิตที่ใช้สือสารกันระหว่างไมโครคอนโทรเลอร์ 2 ตัว เพื่อให้ทำงานสัมพันธุ์กัน

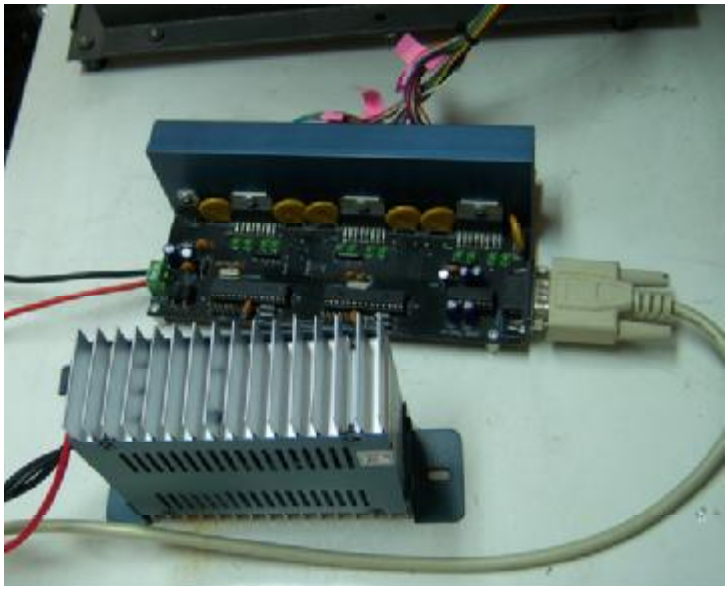


ภาพที่ 60 การต่อไมโครคอลโทรเลอร์ 2 ตัว

การเชื่อมต่อกับคอมพิวเตอร์ผ่านพอร์ดอนุกรมนั้นไม่ต้องเพิ่ม เติมแต่อย่างใดสามารถยกวงจรตาม ภาพที่ 36 วงจรMAX232 มาใช้ได้เลย และเมื่อได้วงจรครบทุกส่วนแล้วก็จะได้วงจรที่ใช้งานจริงตามภาพที่ 61 วงจรของบอร์ดควบคุมมอเตอร์ 6 ช่อง ในภาพที่ 62 เป็นหน้าตาของบอร์ดควบคุมมอเตอร์ 6 ช่อง ที่สร้างเสร็จแล้ว

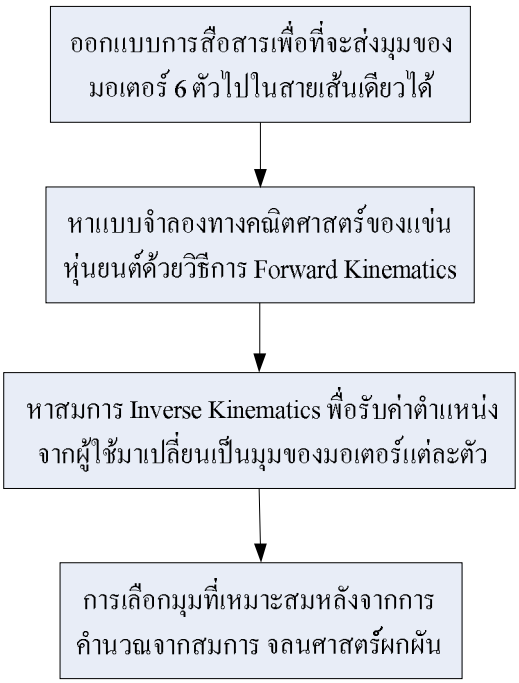


ภาพที่ 61 วงจรของบอร์ดควบคุมมอเตอร์ 6 ช่อง



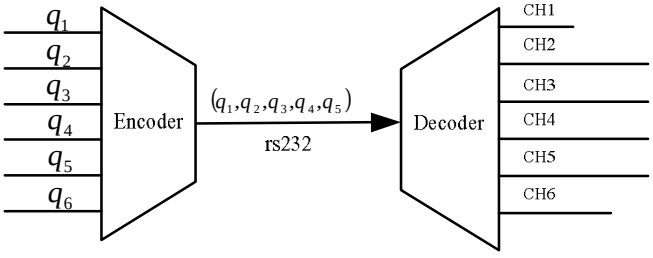
ภาพที่ 62 ของบอร์ดควบคุมมอเตอร์ 6 ช่อง ที่สร้างเสร็จแล้ว

ขั้นตอนที่ 5. **ทดลองเขียนโปรแกรมควบคุมมอเตอร์ ให้ใช้งานกับบอร์ดควบคุมมอเตอร์** 6 **ตัว**



ภาพที่ 63 โฟลชาร์จการเขียนโปรแกรมควบคุมมอเตอร์ ให้ใช้งานกับบอร์ดควบคุมมอเตอร์ 6 ตัว

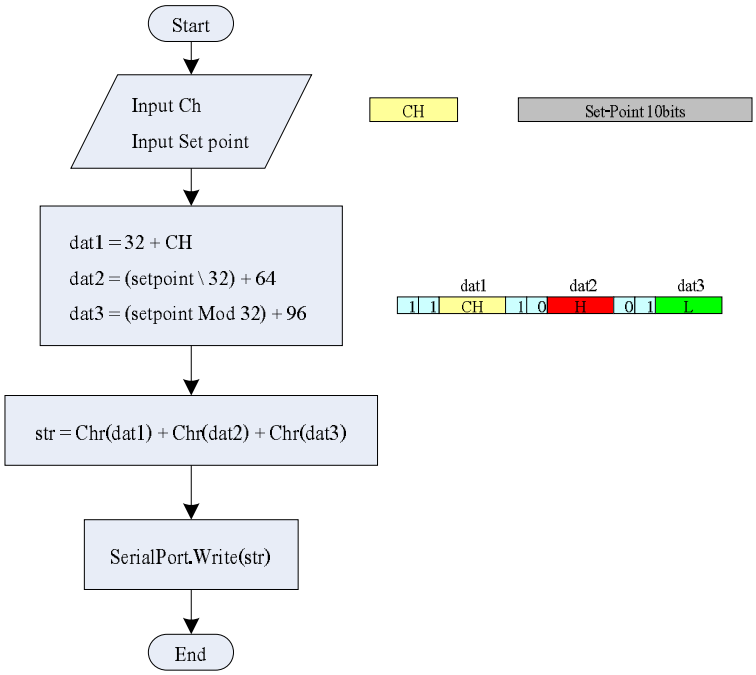
ขั้นแรกต้องออกแบบการสื่อสารเพื่อทำให้ส่งมุมของมอเตอร์ 6 ตัวไปในสายเส้นเดียวการออกแบบ Protocol ที่ใช้สื่อสารกันระหว่างบอร์ด dsPIC30F2010 กับ คอมพิวเตอร์ ผ่านทางพอร์ตอนุกรมตามภาพที่ 64 เป็นบล๊อกไดอะแกรมการสื่อสารข้อมูลระหว่างบอร์ด dsPIC กับคอมพิวเตอร์ เนื่องจากมีมอเตอร์ที่ต้องการควบคุม 6 ตัว แต่การเชื่อมต่อระหว่างบอร์ดกับคอมพิวเตอร์ใช้พอร์ตอนุกรมจึงต้องมีการออกแบบการเข้ารหัสข้อมูลแล้วส่งออกไปแบบอนุกรมให้ทางฝั่งรับถอดระหัสข้อมูลแล้วแยกออกมาเป็น 6 ตัว ให้มอเตอร์แล้วส่งให้กับส่วนควบคุมแบบป้อนกลับต่อไป



ภาพที่ 64 บล๊อกไดอะแกรมการสื่อสารข้อมูลระหว่างบอร์ด dsPIC กับ คอมพิวเตอร์

1. การเข้ารหัสข้อมูล

ขั้นตองการเข้าระหัสข้อมูลเริ่มจากรับข้อมูล CH คือข้อต่อที่ต้องการควบคุม data คือตำแหน่งมุมที่ต้องการแต่ถ้าต้องการควบคุมมากกว่า 1 ข้อต่อต้องส่ง CH1data แล้วต่อด้วย CH2 data ไปเรื่อยๆ วิธีการเข้ารหัสข้อมูลนั้นเริ่มจาก แยกข้อมูลออกเป็น 3 กลุ่มก่อนและใส่บิตตรวจสอบไปอีก 2 บิตไว้ด้านหน้าเพื่อให้ทางฝั่งรับตรวจสอบความถูกด้วยและส่ง กลุ่มข้อมูล ออกไปที่พอร์ดอนุกรมต่อไปตามภาพที่ 65 โฟลชาร์จการเข้ารหัสข้อมูล



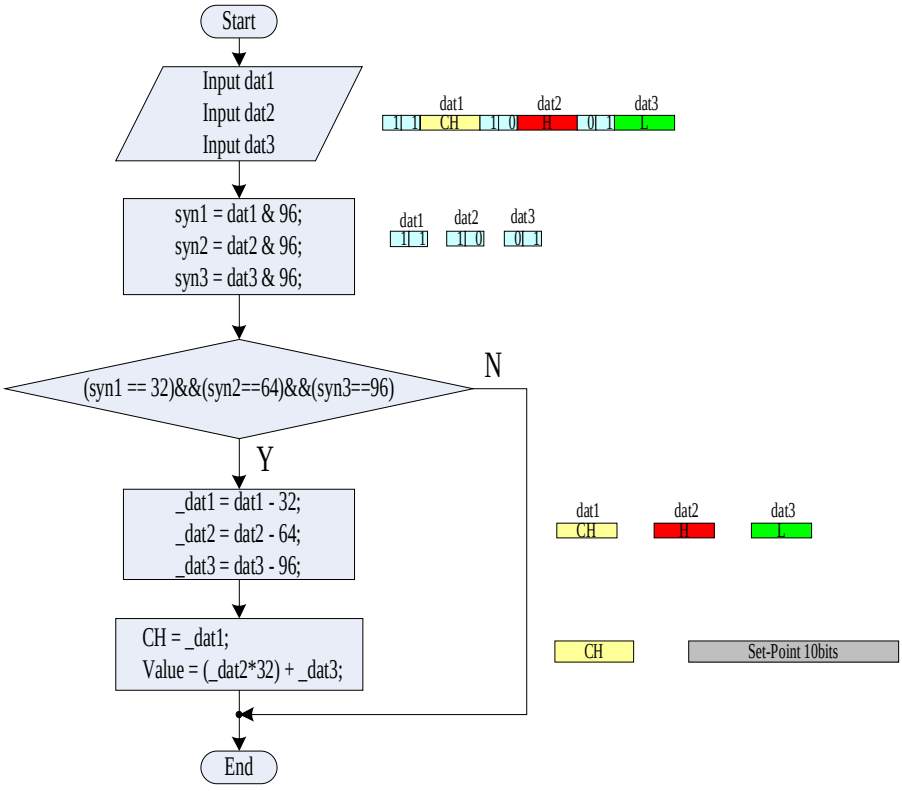
ภาพที่ 65 โฟลชาร์จการเข้ารหัสข้อมูล

2. การถอดรหัส ข้อมูล

หลังจากที่รับข้อมูลที่ถูกส่งจากคอมพิวเตอร์แล้วก็จะได้กลุ่มข้อมูลเป็น dat1-3 เป็นข้อมูล 1 คำสั่งการสั่งงานมอเตอร์ 1 ตัว ดูจากภาพที่ 66 โฟลชาร์จถอดเข้ารหัสข้อมูลประกอบ เมื่อข้อมูลเข้ามาแล้วจะต้องมาตรวจบิตตรวจสอบ 2 บิตก่อนว่าเป็นชุดเดียวกันหรือไม่ หรือการขอดหายของข้อมูล ถ้าไม่เป็นไปตามเงื่อนไขก็แสดงว่าข้อมูนนั้นยังผิดอยู่ก็ให้ปล่อยผ่านไป แต่ถ้าเข้ามาถูกต้องก็ทำการรวมข้อมูล CH และ data กลับคืนมาตามสมการที่ 29 และ 30 หลังจากนั้นก็ส่งไปให้ในส่วนของการควบคุมมอเตอร์ต่อไป

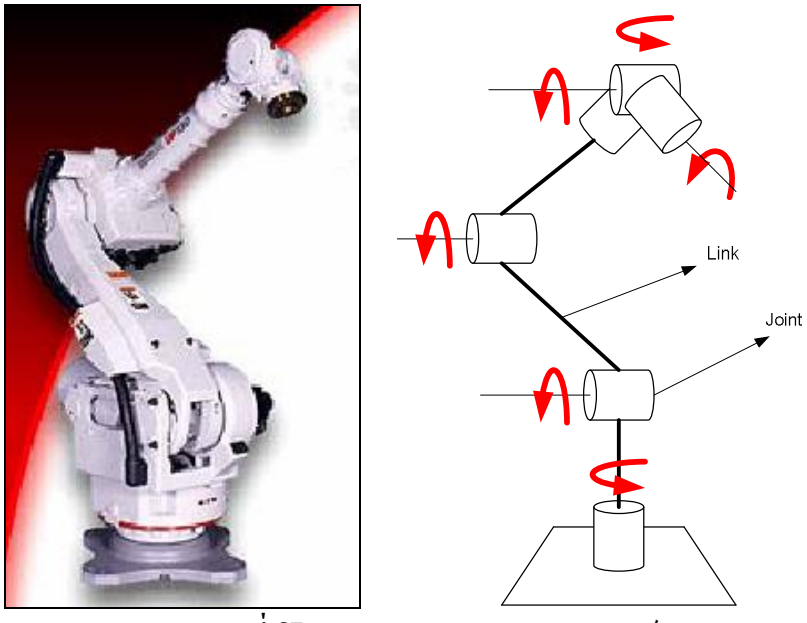
$$CH = \_dat1 \tag{29}$$

$$Value = (\_dat2*32) + \_dat3 \tag{30}$$



ภาพที่ 66 โฟลชาร์จถอดเข้ารหัสข้อมูล

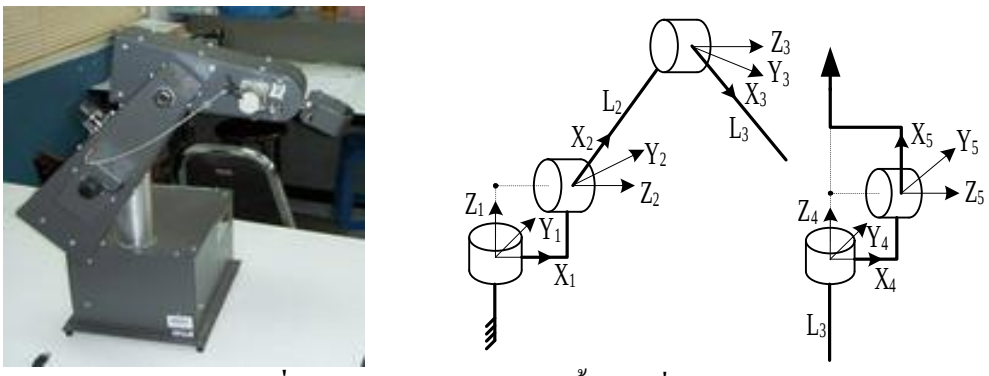
3. จลนศาสตร์ทางตรงของแขนกล



ภาพที่ 67 ส่วนประกอบของแขนหุ่นยนต์

จลนศาสตร์สนใจเกี่ยวกับ ตำแหน่ง ทิศทางโดยไม่คำนึงถึงแรงต้นเหตุ จลนศาสตร์ทางตรงของแขนกลว่าด้วยการคำนวณตำแหน่งในพิกัดที่ถือว่าหยุดนิ่ง ของจุดบนแขนกลในหัวข้อนี้เราจะศึกษาจลนศาสตร์ทางตรงของแขนกลในกรณีสามมิติ ตามภาพที่ 67 ส่วนประกอบของแขนหุ่นยนต์จะประกอบไปด้วย Links และ joint ต่างๆมากมาย ประกอบกันเป็นหุ่นยนต์ซึ่ง ในแต่ละจุดที่สามารถเคลื่อนที่ได้ เราจะเรียกว่า degree -of- freedom ในการที่เราจะสามารถให้หุ่นยนต์เคลื่อนที่ได้ทั่วพื้นที่ทำงานได้นั้นจะใช้ 5-6 DOF ซึ่งจะเขียนให้อยู่ในรูปแบบของ Compound Transform ได้ ${}^{A}_{D}\mathbf{T}={}^{A}_{B}\mathbf{T}\,{}^{B}_{C}\mathbf{T}\,{}^{C}_{D}\mathbf{T}$ ในการที่จะหาตำแน่ง (X, Y, Z) จากหุ่นยนต์ ที่มี หลาย DOF โดยวิธีการของ Denavit - Hartenberg เข้ามาช่วยซึ่งมีขั้นตอนดังนี้

จลนศาสตร์ทางตรงของแขนกลที่ใช้ในกานวิจัย แขนหุ่นยนต์ที่ใช้ในงานวิจัยเป็นแขนหุ่นยนต์ขนาด 5DOF



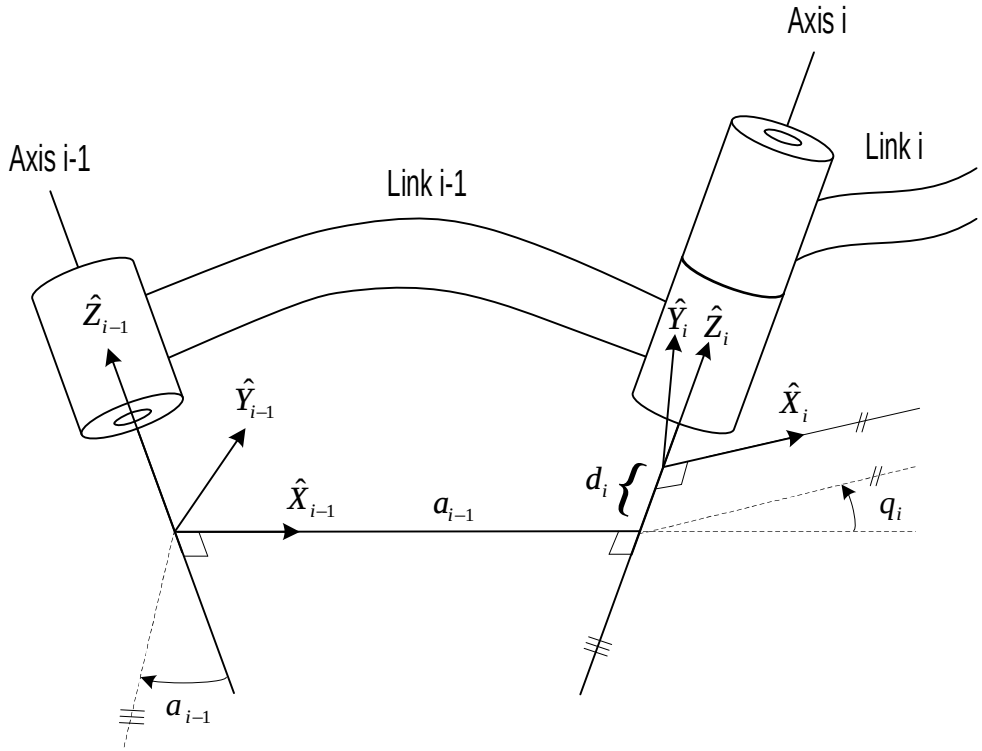
ภาพที่ 68 แขนหุ่นยนต์และการตั้งแกนที่ใช้งานวิจัย

3.1. กำหนดหมายเลขของ link จาก 0 ถึง 3 ตามภาพที่ 68 แขนหุ่นยนต์และการตั้งแกนที่ใช้งานวิจัย

3.2. กำหนดหมายเลขของ joint จาก 1 ถึง 5 กำหนดให้ตรงกับจุดที่เป็นจุดหมุนของแขนหุ่นยนต์

3.3. กำหนดแกน z_i ให้ชี้ไปในทิศของการหมุนตามกฎมือขวาตั้งแต่ joint 1 ถึง n

3.4. กำหนด x_i ให้ตั้งฉากร่วมระหว่าง z_i และ z_{i+1} โดยดูจากภาพที่ 69 ประกอบ



ภาพที่ 69 Link frame (i-1) ที่กระทำต่อ Link frame i

3.5. กำหนดแกน Y_i จากกฎมือขวา และ ทำการหาความยาวของแขนหุ่นยนต์ระหว่างจุดหมุน จะสังเกตเห็นว่าในงานวิจัยนี้มีแค่ L_2 กับ L_3 เพราะว่าเพื่อให้ง่ายในการคิด เราสามารถกำหนดให้ q_1 กับ q_2 อยู่ที่เดียวกันได้

3.6. หาค่าพารามิเตอร์

a_i = มุมระหว่าง $\mathbf{Z_i}$ กับ $\mathbf{Z_{i+1}}$ รอบ $\mathbf{X_i}$ คือมุมที่แขนบิดให้เทียบกับ joint ก่อนหน้า

a_i =ระยะระหว่าง $\mathbf{Z_i}$ กับ $\mathbf{Z_{i+1}}$ ตามแกน $\mathbf{X_i}$ ก็คือความยาวของแขนหุ่นยนต์ แต่ในกรณีของ joint 1,2 และ joint 4,5 ทับกันก็ให้ มีค่าเท่ากับ 0

q_i =มุมระหว่าง $\mathbf{X_{i+1}}$ กับ $\mathbf{X_i}$ รอบ $\mathbf{Z_i}$ คือมุมที่หมุนได้ในที่นี้คือที่มอเตอร์

d_i =ระยะระหว่าง $\mathbf{X_{i+1}}$ กับ $\mathbf{X_i}$ ตามแกน $\mathbf{Z_i}$ คือระยะห่างที่เกิดจากรูปร่างแขนที่ไม่เป็นเส้นตรงแต่ในงานวิจัยนี้ จุดหมุนทุกจุดวางอยู่ในแนวเดียวกันทุกจุดเลยมีค่าเท่ากับ 0หลังจากนั้นนำผลที่ได้ไปใส่ในตารางที่ 2 Denavit – Hartenberg

i	a_{i-1}	a_{i-1}	d_i	q_i
1	0	0	0	q_1
2	-90	0	0	q_2
3	0	L_2	0	q_3
4	-90	L_3	0	q_4
5	90	0	0	q_5

ตารางที่ 2 Denavit - Hartenberg ของแขนหุ่นยนต์ที่ใช้ในงานวิจัย

หลังจากได้ตาราง Denavit - Hartenberg ให้นำ a_{i-1} , a_{i-1}, d_i และ q_i มาใส่ในสมการที่ (9) homogeneous transform ตั้งแต่ 0 ถึง 5

$$\mathbf{{}^{i-1}_{i}T} = \begin{bmatrix} cq_i & -sq_i & o & a_{i-1} \\ sq_i ca_{i-1} & cq_i ca_{i-1} & -sa_{i-1} & -sa_{i-1}d_i \\ sq_i sa_{i-1} & cq_i sa_{i-1} & ca_{i-1} & ca_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{31}$$

$$\mathbf{{}^{0}_{1}T} = \mathbf{{}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T\,{}^{4}_{5}T} \tag{32}$$

$$
{}^{0}_{5}\mathbf{T} = \begin{bmatrix} cq_1 & -sq_1 & 0 & 0 \\ sq_1 & cq_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cq_2 & -sq_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -sq_2 & -cq_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cq_3 & -sq_3 & 0 & L_2 \\ sq_3 & cq_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{33}
$$

$$
\begin{bmatrix} cq_4 & -sq_4 & 0 & L_3 \\ 0 & 0 & 1 & 0 \\ -sq_4 & -cq_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cq_5 & -sq_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ sq_5 & cq_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{5}_{0}\mathbf{T} = \begin{bmatrix} r11 & r12 & r13 & x \\ r21 & r22 & r23 & y_i \\ r31 & r32 & r33 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{34}
$$

$$r11 = c5s1s4 + (c1c2c3 - c1s2s3)c4c5 + (-c1c2s3 - c1s2c3)s5 \tag{35}$$

$$r12 = -s5s1s4 - (c1c2c3 - c1s2s3)c4s5 + (-c1c2s3 - c1s2c3)c5 \tag{36}$$

$$r13 = (c1c2c3 - c1s2s3)s4 - s1c4 \tag{37}$$

$$r21 = ((c2c3 - s2s3)c4c5 + (-s2c3 - c2s3)s5)s1 - c5c1s4 \tag{38}$$

$$r22 = \left(-\left(c2c3 - s2s3\right)c4s5 + \left(-s2c3 - c2s3\right)c5\right)s1 + s5c1s4 \tag{39}$$

$$r23 = s1(c2c3 - s2s3)s4 + c1c4 \tag{40}$$

$$r31 = (-s2c3 - c2s3)c4c5 + (s2s3 - c2c3)s5 \tag{41}$$

$$r32 = -(-s2c3 - c2s3)c4s5 + (s2s3 - c2c3)c5 \tag{42}$$

$$r33 = (-s2c3 - c2s3)s4 \tag{43}$$

$$x = (c1c2c3 - c1s2s3)L3 + c1c2L2 \tag{44}$$

$$y = ((c2c3 - s2s3)L3 + c2L2)s1 \tag{45}$$

$$z = (-s2c3 - c2s3)L3 - s2L2 \tag{46}$$

โดยที่

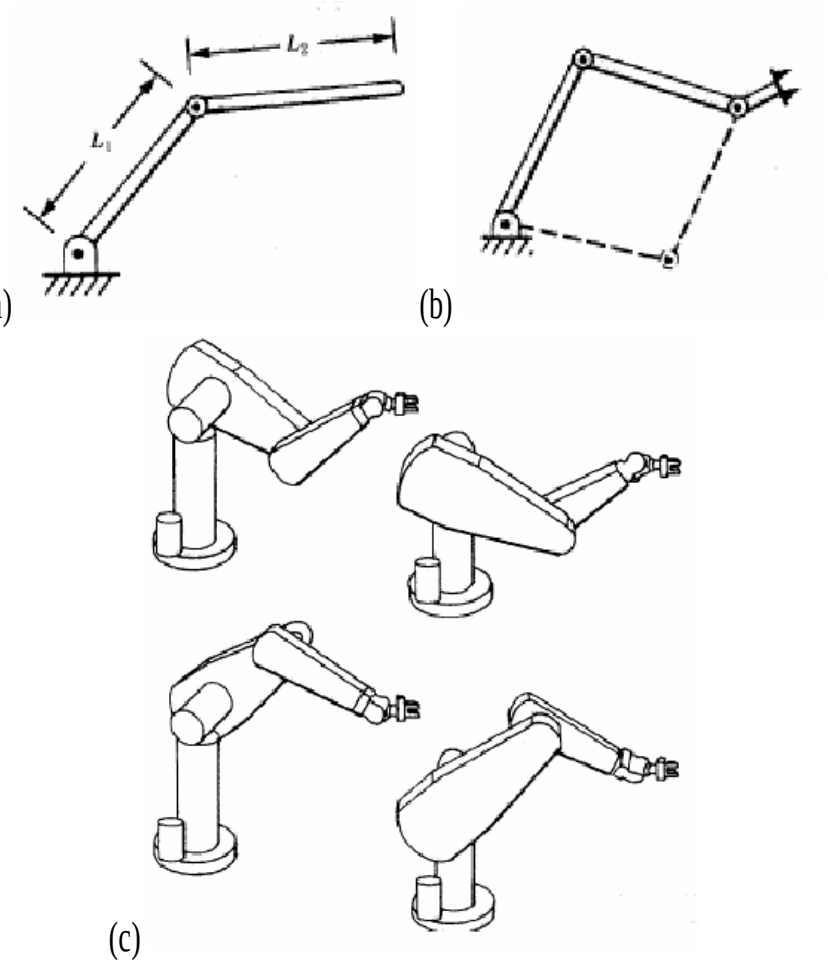
$$c1 = \cos(q_1) \tag{47}$$

$$s1 = \sin(q_1) \tag{48}$$

จากนั้นแทนค่าความยาวของแขนแต่ละชิ้น $L_1 = 18.5cm$ $L_2 = 16.5cm$ $L_3 = 15cm$ แล้วใช้สมการที่ 44, 45 และ46 เพื่อหาค่า (x,y,z) ของมือจับเทียบกับเบสเฟรม

4. **จลนศาสตร์ผกผันของแขนกล** (Inverse kinematics of manipulator)

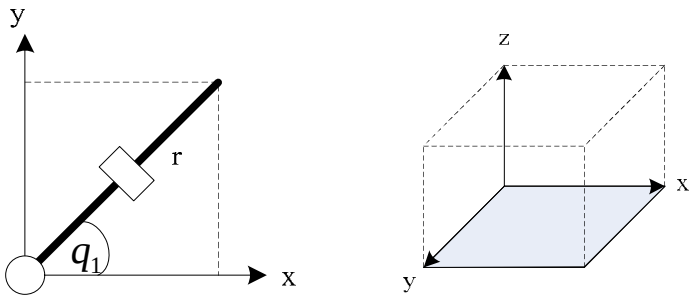
ในหัวข้อนี้เราจะศึกษาการคำนวณที่ตรงข้ามกับจลนศาสตร์ทางตรงของแขนกล นั่นก็คือ เมื่อกำหนดตำแหน่งและทิศทางในเฟรมที่หยุดนิ่งของปลายแขนให้ เราต้องการคำนวณพารามิเตอร์ของข้อต่อที่ทำให้ปลายแขนเรียงตัวในลักษณะที่กำหนด เราเรียกการคำนวณในลักษณะนี้ว่าการคำนวณจลนศาสตร์ผกผันของแขนกลการคำนวณจลนศาสตร์ผกผันของแขนกลต้องคำนึงถึงสองเรื่องหลัก เรื่องแรกคือว่าพารามิเตอร์ของข้อต่อที่ต้องการอาจไม่มี เรื่องที่สองคือพารามิเตอร์คำตอบที่เป็นไปได้อาจมีหลายชุดภาพที่ 70 (a) แสดงแขนกลที่มีข้อต่อแบบหมุนสองข้อต่อโดยท่อนแขนทั้งสองที่แสดงในรูปมีความยาวเป็น L_1 และ L_2 จะเห็นได้ว่าไม่มีพารามิเตอร์ของข้อต่อที่ทำ ให้ปลายแขนอยู่ห่างจุดหมุนที่ฐานเกิน $L_1 + L_2$ ไปได้ และหากกำหนดให้จุดปลายแขนอยู่ห่างจากจุดหมุนที่ฐาน น้อยกว่า $L_1 + L_2$ หากแต่ละข้อต่อสามารถหมุนได้ครบรอบก็จะได้ว่ามีพารามิเตอร์คำตอบอย่างมาสองชุด ในภาพที่ 70 (b) แสดงสองชุดของพารามิเตอร์ข้อต่อที่เป็นคำตอบสำหรับตำแหน่งและทิศทางของปลายแขนที่กำหนดให้การมีพารามิเตอร์คำตอบมากกว่าหนึ่งชุด ถึงแม้จะทำให้ขั้นตอนการหาคำตอบอาจยุ่งยากกว่าการมีคำตอบเดียว แต่คุณสมบัตินี้ทำให้เราสามารถเลือกชุดพารามิเตอร์ข้อต่อที่เหมาะสมกับงานเป้าหมาย เช่นเราอาจเลือกใช้ชุด พารามิเตอร์ที่อ้อมหลบสิ่งกีดขวาง แขนกลที่ใช้ในงานวิจัย ก็ได้ถูกออกแบบให้มีหลายชุดคำตอบ เช่น ตามแสดงในภาพที่ 70(c) แขนกลมีสี่ชุดคำตอบที่ให้ตำแหน่งและทิศทางเดียวกันของปลายแขน



ภาพที่ 70 (a) แขนกลสองข้อต่อ (b) แขนกลสามข้อต่อ (c) PUMA 560
ที่ปลายแขนตำแหน่งและทิศทางเดียวกัน

การหาจลนศาสตร์ผกผันของแขหุ่นยนต์ที่ใช้ในงานวิจัย พิจาณาจากภาพที่ 71 คือการมองลงมาจากด้านบน โดยไม่สนใจความสูงแกน z จะได้r เป็นเวคเตอร์ที่ชี้ไปที่มือจับ ดังนั้น

$$q_1 = A\tan 2(x, y) \tag{49}$$



ภาพที่ 71 การมองลงมาจากด้านบน โดยไม่สนใจความสูงแกน z

ในการหาค่า q_2 และq_3 ให้พิจารณาจากด้านข้างของแขนหุ่นยนต์โดยมองแกน x,y เป็นระนาบเดียวกัน แทนด้วย r และ s=z

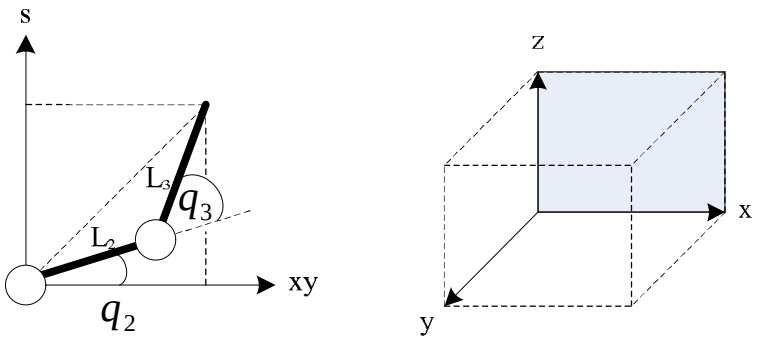
$$r = \sqrt{x^2 + y^2} \tag{50}$$

$$s = z \tag{51}$$

$$\cos q_3 = \frac{r^2 + s^2 - L_3{}^2 - L_3{}^2}{2L_2L_3} = D \tag{52}$$

$$q_3 = A\tan 2(D, \pm\sqrt{1 - D^2}) \tag{53}$$

$$q_2 = A\tan 2(r, s) - A\tan 2(L_2 + L_3c3, L_3s3) \tag{54}$$



ภาพที่ 72 ด้านข้างของแขนหุ่นยนต์โดยมองแกน x,y เป็นระนาบเดียวกัน แทนด้วย r และ s=z

5. การเลือกมุมที่เหมาะสมหลังจากการคำนวณจากสมการจลนศาสตร์ผกผัน

หลังจากการคำนวณสมการจลนศาสตร์ผกผันแล้วเราจะได้คำตอบของมุม q_2 และ q_3 มา 2 ค่าซึ่งเป็นค่าที่ถูกต้องทั้งคู่แต่การใช้งานจริงนั้นต้องการเพียง 1 ค่าเท่านั้นโดยดูจากความเหมาะสมเพื่อหลบสิ่งกีดขวาง หรือ ความต่อเนื่องในการเคลื่อนที่ ซึ่งในงานวิจัยนี้จะใช้ความต่อเนื่องเป็นหลักโดยเริ่มจากรับค่ามุมที่คำนวณได้มาจากสมการจลนศาสตร์ผกผันมาเปรียบเทียบกับค่าของมุมก่อนหน้าโดยเลือกเอามุมที่อยู่ใกล้มากที่สุดมาใช้งานตามภาพที่ 73 โฟลชาร์จการเลือกมุมที่เหมาะสมก่อนที่จะส่งไปทำการ แปลงค่าจาก $0-\frac{p}{2}rad$ เป็น 0-1023 ก่อนส่งให้กับบอร์ด dsPIC30F2010 ตามสมการที่ 55-60

$$M1 = -0.306q_1 + 345 \quad (55)$$

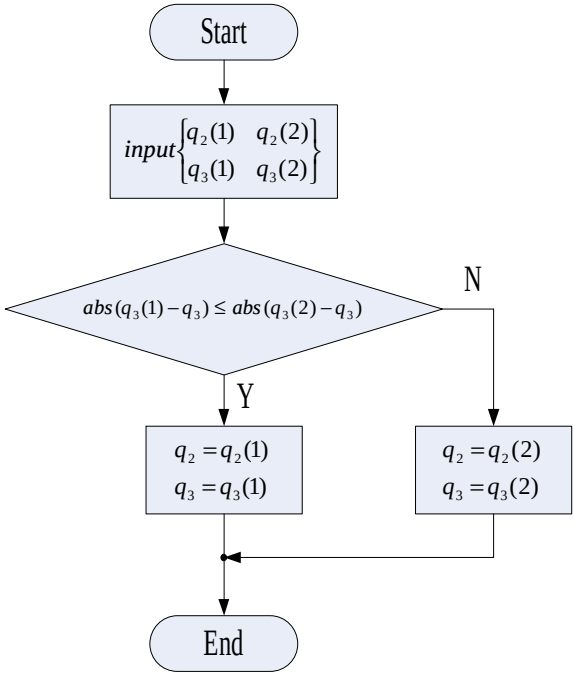
$$M2 = -0.313q_2 + 610 \quad (56)$$

$$M3 = -0.304q_3 + 382 \quad (57)$$

$$M4 = -0.304(q_5 + q_4) + 509 \quad (58)$$

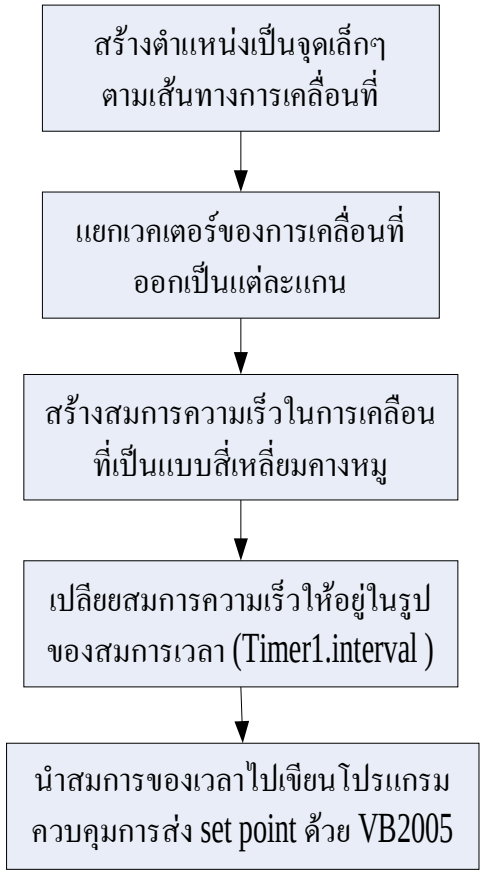
$$M5 = -0.304(q_5 - q_4) + 509 \quad (59)$$

$$M6 = -0.31q_6 + 512 \quad (60)$$



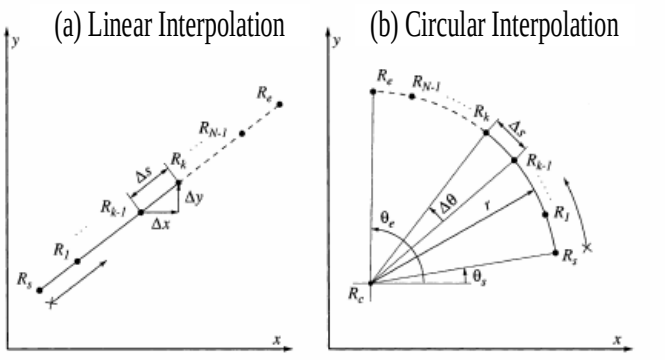
ภาพที่ 73 โฟลชาร์จการเลือกมุมที่เหมาะสม

ขั้นตอนที่ 6. **การสร้าง** Path Panning

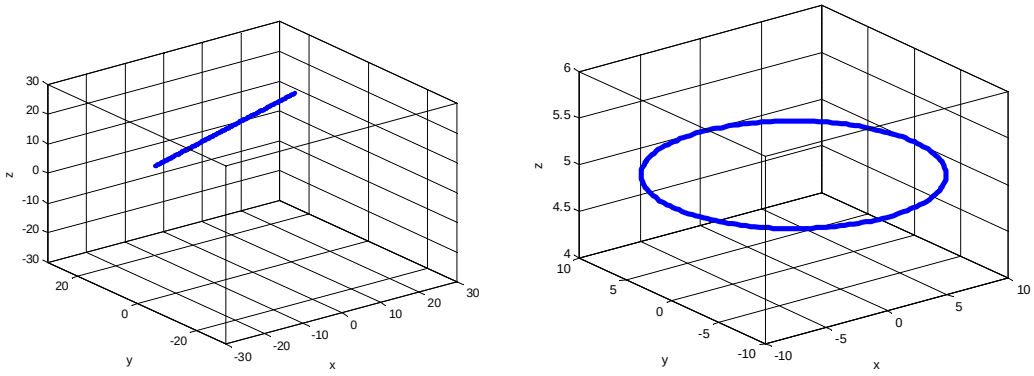


ภาพที่ 74 โฟลชาร์จแสดงขั้นตอนการสร้าง Path Panning

การสร้างเส้นทางเดิน คือการสร้างเส้นทางเดินให้กับปลายแขนหุ่นยนต์เลื่องที่ไปตามที่ต้องการ เช่น เส้นตรง หรือเส้นโค้งหลักการคือการสร้างจุดเล็กๆ ตามเส้นทางการเคลื่อนที่ที่ต้องการ จากภาพที่ 75 (a) แสดงการสร้างเส้นทางเดินแบบเส้นตรงและ ภาพที่ 75 (b) เป็นการสร้างเส้นทางเดินแบบเส้นโค้ง

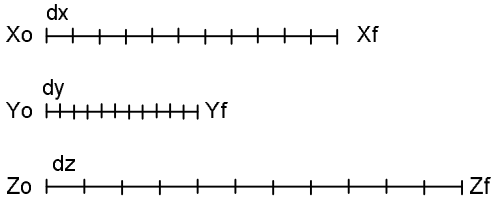


ภาพที่ 75 การสร้างเส้นทางเดินแบบเส้นตรงและเส้นโค้ง

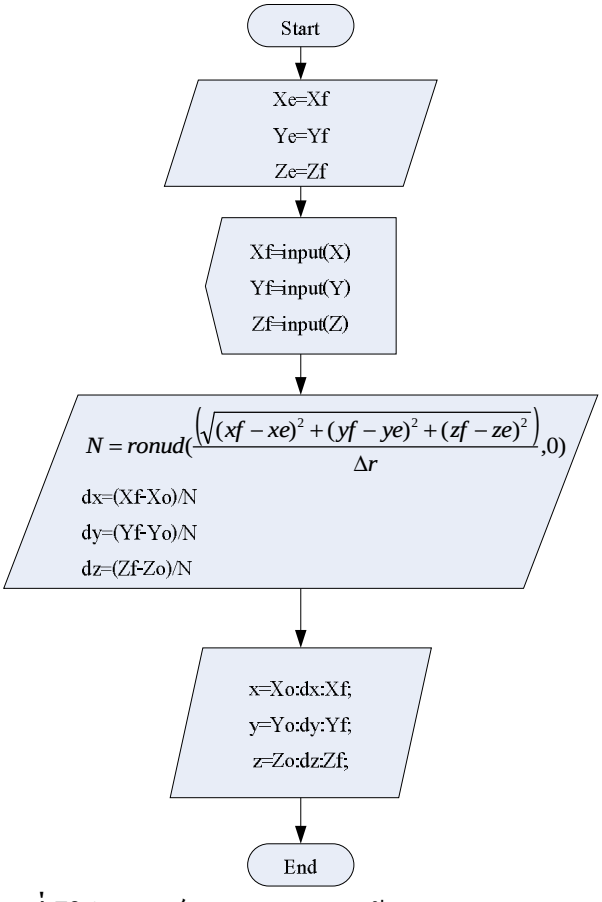


ภาพที่ 76 สร้างเส้นทางเดินของปลายแขนหุ่นยนต์แบบ 3 มิติ

ภาพที่ 76 แสดงสร้างเส้นทางเดินของปลายแขนหุ่นยนต์แบบ 3 มิติ ทางด้านซ้ายเป็นแบบเส่นตรงและด้านขวาเป็นเส้นโคง หลักการในการสร้างจุดเล็กๆ เริ่มจากการ แบ่งเวคเตอร์ของการเคลื่อนที่ออกเป็นส่วนเล็กๆแล้วแยกแกนออกเป็นแกน 3 แกน x,y และ z ตามภาพที่ 77 ผลที่ได้คือ แต่ละแกนจะ มีจำนวนช่องที่เท่ากันแต่ระยะทางเล็กๆที่ใช้แบ่งนั้น จะมีความยาวแตกต่างกันโดยมีวิธการ ตามภาพที่ 78 โฟลชาร์จแสดงการแบ่งเส้นตามแกน x,y และ z



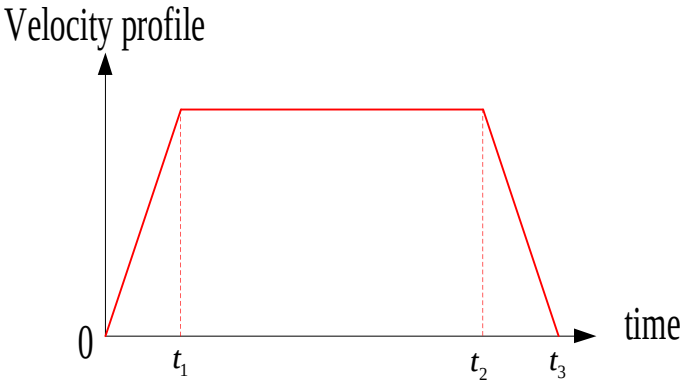
ภาพที่ 77 แสดงการแบ่งเส้นตามแกน x, y และ z



ภาพที่ 78 โฟลชาร์จแสดงการแบ่งเส้นตามแกน x, y และ z

1. การสร้างความเร็วในการเคลื่อนที่

ขั้นตอนการสร้างความเร็วในการเคลื่อนที่เริ่มต้นจากสร้างรูปแบบของความเร็วตามภาพที่ 79 มีลักษณะเป็นรูปสี่เหลี่ยมคางหมูโดยจะแบ่งเป็น 3 ส่วน ตามสมการที่ 61 ซึ่งเป็นสมการความเร็ว และเมื่อพิจารณาสมการที่ 62 ก็จะเห็นว่าความเร็วนั้นสร้างมาจากระยะทาง s และเวลา t ในงานวิจัยนี้จะกำหนดให้ ระยะทางนั้นคงที่แล้วใช้การเปลี่ยนค่าวเวลาเพื่อกำหนดความเร็ว จะได้สมการที่ 63 Timer1.interval คือค่าเวลาของTimer 1 ที่เอาไปใส่ในโปรแกรม visual2005 เพื่อตำแหน่ง x, y และ z เพื่อส่งไปให้ส่วนของ สมการ จลนศาสตร์ผกผัน คำนวณหามุมต่อไป ภาพที่ 80 โฟลชาร์จการสร้างความเร็วด้วยโปรแกรม Visual Basic 2005

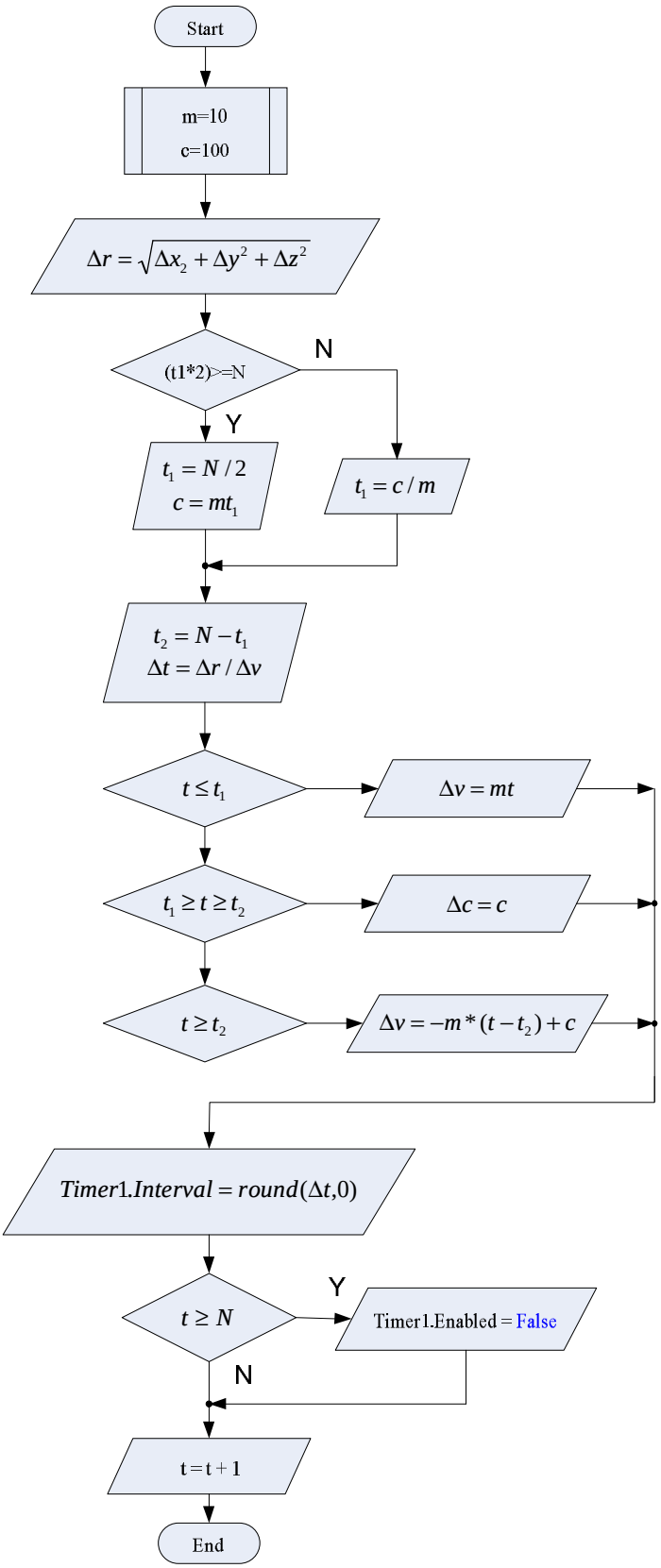


ภาพที่ 79 ความเร็วที่ใช้ในการเคลื่อนที่

$$v = \begin{cases} v = mt & ; \ t \le t_1 \\ v = c & ; \ t_1 < t < t_2 \\ v = -m(t - t_2) + c & ; \ t_2 \le t \end{cases} \tag{61}$$

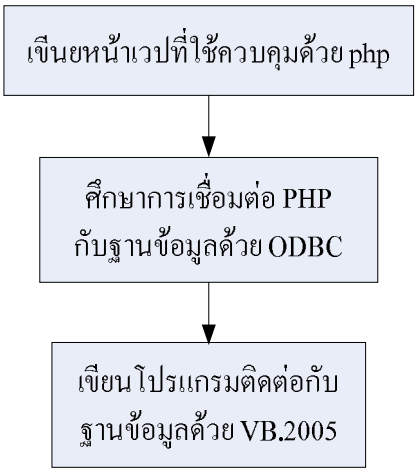
$$v = \frac{s}{t} \tag{62}$$

$$\text{Timer1.Interval} = \frac{\Delta r}{\Delta v} \tag{63}$$



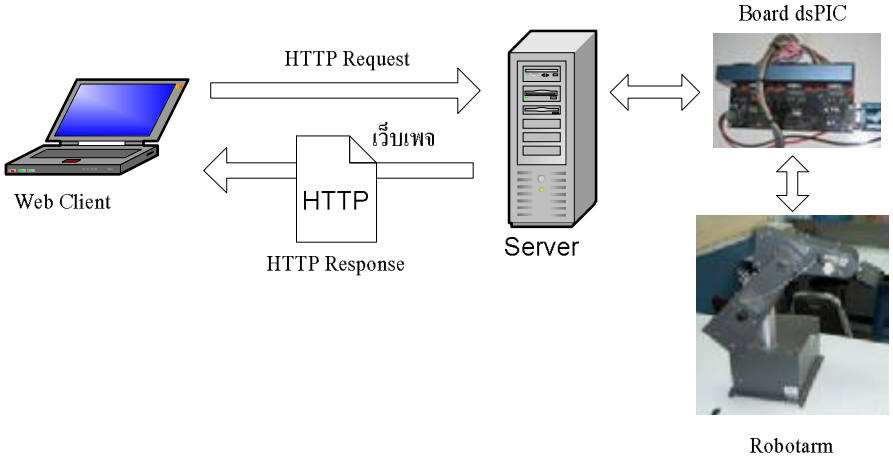
ภาพที่ 80 โฟลชาร์จการสร้างความเร็วในโปรแกรม Visual Basic 2005

ขั้นตอนที่ 7. ศึกษาการทำงานของ Web Server และ เขียนโปรแกรมรับส่งข้อมูลผ่านเวป

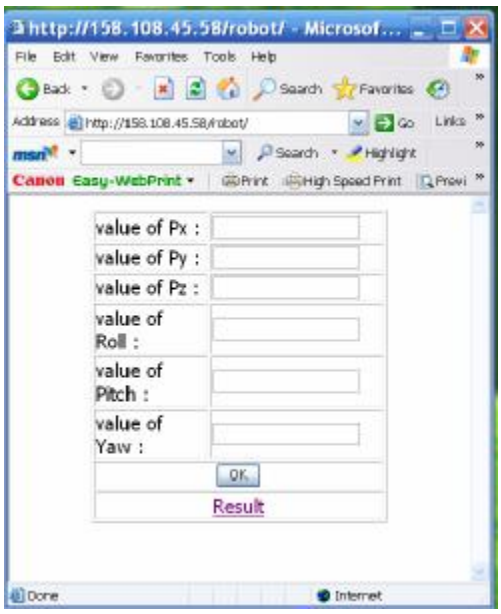


ภาพที่ 81 โฟลชาร์จขั้นตอนศึกษาการทำงานของ Web Server และ เขียนโปรแกรมควบคุมผ่านเวป

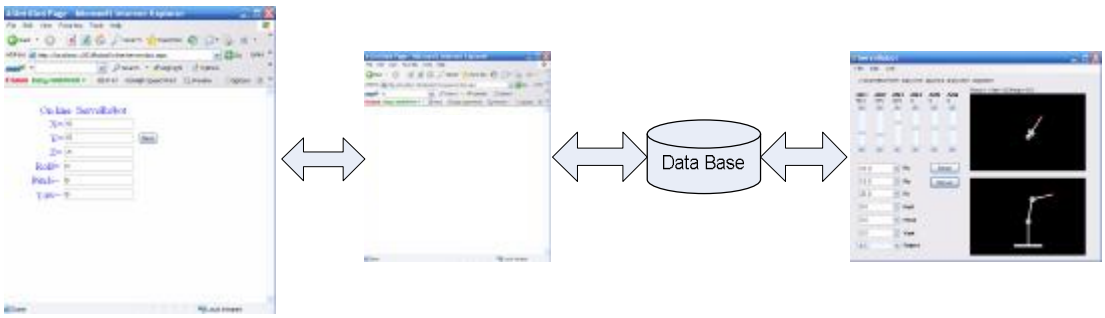
หลักการของ การควบคุมผ่านอินเตอร์เน็ท คือให้ผู้ใช้จะป้อนตำแหน่งที่ต้องเพื่อส่งไปให้โปรแกรม Servos Robot ผ่านทางฐานข้อมูล ตามภาพที่ 82 โฟลชาร์จเส้นทางของข้อมูลที่ส่งผ่านจากเวป ขั้นตอนเริ่มจากผู้ใช้เปิดหน้าเวปตามภาพที่ 83 หน้าเวปเพจที่ใช้ควบคุมแขนหุ่นยนต์แล้วป้อนตำแหน่งที่ต้องการให้ปลายแขนเคลื่อนที่ไปและกด OK ข้อมูลก็จะถูกส่งไปที่หน้าเวปอีกตัวก่อนซึ่งรันอยู่บน Server เวปตัวนี้ก็จะเข้าถึงฐานข้อมูลบนเครื่อง Server ผ่านทาง ODBC และโปแกรม Servo Robot ก็จะนำข้อมูลนั้นไปควบคุมแขนต่อไปตามภาพที่ 84 การเชื่อมต่อระหว่างหน้าเวปกับโปแกรม Servo Robotผ่านฐานข้อมูล



ภาพที่ 82 โฟลชาร์จเส้นทางของข้อมูลที่ส่งผ่านจากเวป



ภาพที่ 83 หน้าเวปเพจที่ใช้ควบคุมแขนหุ่นยนต์



ภาพที่ 84 การเชื่อมต่อระหว่างหน้าเวปกับโปแกรม Servo Robotผ่านฐานข้อมูล

1. การเชื่อมต่อ PHP กับฐานข้อมูลด้วย ODBC

ODBC ย่อมาจาก Open Database Connectivity เป็น Application Programming Interface (API) คือ มาตรฐานในการเข้าถึงข้อมูลที่ถูกกำหนดขึ้น ที่อนุญาตให้คุณเชื่อมต่อฐานข้อมูลอื่นๆ ได้ ซึ่ง ODBC เป็นเสมือนตัวกลางระหว่างโปรแกรมที่เราใช้งาน กับฐานข้อมูลเหล่านั้น เช่น MS Access

1.1. สร้างการเชื่อมต่อแบบ ODBC การเชื่อมต่อแบบ ODBC คุณสามารถเชื่อมต่อได้กับหลายฐานข้อมูลบนคอมพิวเตอร์ในเครือข่ายของคุณเองได้ ฐานข้อมูลส่วนใหญ่ที่ใช้การเชื่อมต่อแบบนี้ คือ MS Access มาดูขั้นตอนการเชื่อมต่อฐานข้อมูลกัน:

1. เปิด Control Panel แล้วเลือกที่ไอคอน Administrative Tools
2. ดับเบิ้ลคลิ้กที่ไอคอน Data Sources (ODBC)
3. เลือกแท็ป System DSN แล้วคลิ้กปุ่ม Add
4. เลือก Microsoft Access Driver แล้วคลิ้กปุ่ม Finish
5. ตั้งชื่อ Data Source และคำอธิบายลงไปในช่อง แล้วคลิ้ก OK
6 .ในหน้าถัดไป ให้คลิ้กปุ่ม Select เลือกฐานข้อมูลที่เราต้องการเชื่อมต่อ
7. หากมีการใช้ไฟล์บริหารกลุ่มงาน (MS Access Workgroup Administrator) เพื่อดูแลและกำหนดสิทธิของผู้ใช้ในการเข้าถึงข้อมูลและส่วนต่างๆในฐานข้อมูลแล้ว จะต้องมีการระบุชื่อไฟล์นั้นไว้ในการสร้าง DSN นี้ด้วย โดยเลือกออปชั่น Database ในกรอบ System Database
8. คลิ้กปุ่ม System Database เพื่อเลือกไฟล์บริหารกลุ่มงาน
9. คลิ้ก OK แล้วปิด ODBC MS Access Setup ด้วยการคลิ้ก OK อีกครั้ง จะปรากฏ Data Source ที่สร้างขึ้นใหม่
10. คลิ้ก OK เพื่อปิด ODBS Data Source Administrator เป็นอันจบกระบวนการ

การเซ็ทคอนฟิกูเรชั่นต้องทำบนเครื่องคอมพิวเตอร์ที่เว็บไซต์ตั้งอยู่เท่านั้น ถ้าคุณรัน IIS บนเครื่อง PC ของคุณเอง ก็ใช้วิธีที่กล่าวข้างบนได้ แต่ถ้าเว็บไซต์ของคุณอัพไปเซิร์ปเวอร์แล้ว ต้องไปเซ็ทที่ Control Panel ของเซิร์ปเวอร์นั้นๆ ซึ่งต้องถามทางโฮสติ้งที่คุณใช้อยู่

1.2. วิธีเชื่อมต่อ ODBC คล้ายๆ กับวิธีเชื่อมต่อฐานข้อมูล MySQL แต่ชื่อฟังก์ชั่นจะต่างกัน โดยจะใช้ฟังก์ชั่น odbc_connect () ในการเชื่อมต่อผ่าน ODBC จะมี 4 พารามิเตอร์ คือ ชื่อ Data Source, ชื่อผู้ใช้, รหัสผ่าน และชนิดของเคอร์เซอร์ [ไม่มีก็ได้] จะมี Syntax ดังนี้

```
ODBC_CONNECT ('Data Source Name',' Username ', 'Password');
```

ถ้าการเชื่อมต่อไม่สำเร็จจะคืนค่า Connection ID กลับมาเป็นค่า 0 หมายถึง False เมื่อเชื่อมต่อฐานข้อมูลได้แล้ว การดำเนินการใดๆ กับฐานข้อมูลจะใช้ฟังก์ชั่น odbc_exec() โดยระบุคำสั่ง SQL ลงไป ดังตัวอย่างจะใช้ DSN ชื่อว่า sanook

```
$con=odbc_connect('sanook','','');
$sql="SELECT * FROM customers";
$rs=odbc_exec($con,$sql);
```

1.3 ฟังก์ชั่นสำคัญต่างๆ ฟังก์ชั่น odbc_fetch_row () เป็นการดึงแถว หรือเรคอร์ดออกมา ถ้ามีค่าแสดงออกมาจะเป็น true ไม่เช่นนั้นจะเป็น false โดยมีพารามิเตอร์ 2 ตัว คือ ค่า ID ของผลลัพธ์ที่ได้จากฟังก์ชั่น odbc_exec () และหมายเลขแถวที่ต้องการดึงข้อมูล หากไม่ระบุจะทำการดึงแถวถัดไป ซึ่งมี syntax ดังนี้

```
odbc_fetch_row (result _id, row_number)
```

ฟังก์ชั่น odbc_result () เป็นการอ่านค่าในฟิลด์ของรายการผลลัพธ์ มี 2 พารามิเตอร์ คือ ค่า ID ของผลลัพธ์ที่ได้จากฟังก์ชั่น odbc_exec () และหมายเลขฟิลด์ที่ต้องการอ่านค่า โดยฟิลด์แรกจะมีหมายเลขเป็น 1 โค้ดข้างล่างจะให้ค่าฟิลด์แรกของเรคอร์ด

```
odbc_result ($rs,1);
```

หรือ คุณสามารถใส่ชื่อของฟิลด์แทนหมายเลขฟิลด์ ก็ได้ แบบนี้

```
odbc_result ($rs,"FirstName");
```

และจบด้วยปิดการเชื่อมต่อกับฐานข้อมูล ด้วยฟังก์ชั่น odbc_close ()

```
odbc_close ($con);
```

1.4. ตัวอย่างการเชื่อมต่อแบบ ODBC ตัวอย่างข้างล่างนี้จะแสดงการเชื่อมต่อกับฐานข้อมูลก่อน จนถึงการแสดงข้อมูลออกมาในรูปแบบตาราง HTML

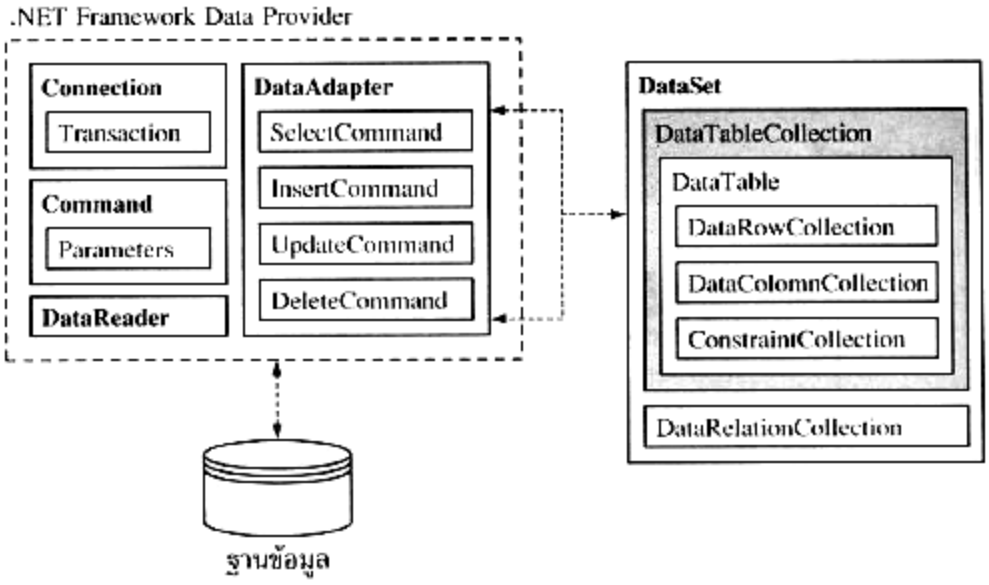
```
<html>
```

```
<body>
<?php
$con=odbc_connect('sanook',",");
if (!$conn)
  {exit("เชื่อมต่อฐานข้อมูลไม่ได้: " . $con);}
$sql="SELECT * FROM customers";
$rs=odbc_exec($con,$sql);
if (!$rs)
  {exit("คำสั่ง SQL ผิดพลาด");}
echo "<table><tr>";
echo "<th>FirstName</th>";
echo "<th>LastName</th></tr>";
while (odbc_fetch_row($rs))
{
  $firstname=odbc_result($rs,"FirstName");
  $lastname=odbc_result($rs,"LastName");
  echo "<tr><td>$firstname</td>";
  echo "<td>$lastname</td></tr>";
}
odbc_close($con);
echo "</table>";
?>
</body>
</html>
```

2. การติดต่อกับฐานข้อมูลด้วยADO.NET

ADO.NET คือเทคโนโลยีการเข้าถึงข้อมูลของ .NET ที่ช่วยให้เราเขียนโปรแกรมเพื่อเข้าถึงและทำงานกับข้อมูลแหล่งจากแหล่งข้อมูลประเภทต่างๆ ได้ง่ายอย่างเรียบง่ายแต่มีประสิทธิภาพ

ส่วนประกอบ ในการทำงานของ ADO.NET แบ่งออกเป็น 2 ส่วนได้แก่ DataSet และ Data Provider ดังรายละเอียดต่อไปนี้ ตามภาพที่ 85 ส่วนประกอบ ในการทำงานของ ADO.NET



ภาพที่ 85 ส่วนประกอบ ในการทำงานของ ADO.NET

2.1. Dataset คือออบเจ็คซึ่งเป็นตัวแทนของข้อมูลที่โปรแกรมของเราดึงมาจากฐานข้อมูล โดย DataSet จะเก็บสำเนา ของข้อมูลส่วนนั้นในฐานข้อมูลไว้ในหน่วยความจำของเครื่อง client (เครื่องที่โปรแกรมของเราทำงานอยู่) การทำงานต่างๆ กับข้อมูลใน Dataset จึงเป็นอิสระจากฐานข้อมูล และเมื่อถึงเวลาที่เหมาะสมเราสามารถนำข้อมูลจาก DataSet ไปอัพเดทลงฐานข้อมูลได้

การทำงานกับฐานข้อมูลโดยใช้ออบเจ็ค DataSet จะเป็นรูปแบบที่เรียกว่า disconnected data access หรือ “การเข้าถึงข้อมูลแบบไม่ต้องเชื่อมต่อกับฐานข้อมูล (ไว้ตลอดเวลา)” และเรียกข้อมูลใน Dataset ว่า disconnected data เนื่องจากโปรแกรมจะมีการเชื่อมต่อกับฐานข้อมูลเท่าที่จำเป็นเท่านั้น เช่น ในการดึงข้อมูล โปรแกรมจะเปิดการเชื่อมต่อกับฐานข้อมูลเพื่อดึงข้อมูลเข้ามาแล้วปิดการเชื่อต่อทันทีหลังจากดึงข้อมูลเสร็จ สำหรับการแก้ไข เพิ่ม หรือลบข้อมูลก็เช่นเดียวกัน โปรแกรมจะเชื่อมต่อไปยังฐานข้อมูลเพื่อรัน query ของคำสั่ง UPDATE, INSERT หรือ DELETE จากนั้นการเชื่อมต่อจะถูกปิดลงไป

ภายในออฟเจ็ค DataSet ยังมีออบเจ็คย่อยที่สำคัญ เช่น Data Table และ DataRow ที่เป็นๆ ตัวแทนของเทเบิล (ตารางข้อมูล) และเรคอร์ด (แถวข้อมูล) ตามลำดับ รวมถึง DataRelation ที่เป็นตัวแทนของความสัมพันธ์ระหว่างเทเบิลใน DataSet เป็นต้น ซึ่งจะได้กล่าวรายละเอียดในบทถัดไป ตอนนี้เรามาทำความรู้จักกับส่วนประกอบในการทำงาน ADO.NET อีกส่วนหนึ่งกันก่อน นั่นก็คือ Data Provider (หรือเรียกเต็มๆ ก็คือ .NET Framework Data Provider)

2.2. Data Provider คือส่วนที่ดูแลในเรื่องการเชื่อมต่อกับฐานข้อมูล และการส่ง query ไปประมวลผลยังฐานข้อมูล ทั้งนี้ Data Provider ไม่ใช่ออบเจ็คเดี่ยวๆ เหมือนอย่าง DataSet หากแต่เป็นกลุ่มของออบเจ็คที่เกี่ยวข้องสัมพันธ์กัน ซึ่งร่วมกันทำงานเพื่อเป็นตัวกลางในการรับ-ส่งข้อมูลระหว่างฐานข้อมูลกับ DataSet โดย ADO.NET 2.0 ที่ติดตั้งมาพร้อมกับ Visual Studio 2005 ได้จัดเตรียม Data Provider ไว้ 4 ชนิดแยกตามประเภทของฐานข้อมูล ได้แก่

1. SQL Server Data Provider (หรือ SQLClient Data Provider) เป็น Data Provider ที่ถูกออกแบบมาสำหรับฐานข้อมูล Microsoft SQL Server ตั้งแต่เวอร์ชั่น 7 ขึ้นไปโดยเฉพาะ

2. OLEDB Data Provider เป็น Data Provider สำหรับแหล่งข้อมูลใดๆ ก็ตามที่สนับสนุนการติดต่อผ่านทางอินเทอร์เฟซ OLEDB เช่นฐานข้อมูล Microsoft Access เป็นต้น

3. Oracle Data Provider (หรือ OracleClient Data Provider) เป็น Data Provider ที่ถูกออกแบบมาสำหรับฐานข้อมูล Oracle โดยเฉพาะ (Data Provider นี้ถูกเพิ่มขึ้นมาใน ADO.NET 1.1 ที่ติดตั้งมาพร้อมกับ Visual Studio.NET 2003)

4. ODBC Data Provider เป็น Data Provider สำหรับแหล่งข้อมูลใดๆ ก็ตามที่มี ODBC driver ให้ใช้ในการติดต่อ (Data Provider นี้ถูกเพิ่มเข้ามาใน ADO.NET 1.1 ที่ติดตั้งมาพร้อมกับ Visual Studio.NET 2003)

Data Provider แต่ละชนิดข้างต้นจะประกอบไปด้วยออบเจ็คต่างๆ ที่ทำงานร่วมกันดังนี้

5. ออบเจ็ค Connection รับผิดชอบในเรื่องการเชื่อมต่อกับฐานข้อมูล

6. ออบเจ็ค Command ทำหน้าที่ส่งคำสั่งไปประมวลผลยังฐานข้อมูล ซึ่งคำสั่งนั้นอาจจะเป็น query ในภาษา SQL หรือเป็นการเรียกใช้ stored procedure ในฐานข้อมูล เป็นต้น

7. ออบเจ็ค DataAdapter ทำหน้าที่ดึงข้อมูลจากฐานข้อมูลมาใส่ (fill) ลงใน DataSet และ นำข้อมูลจาก DataSet ไปอัพเดทลงฐานข้อมูล

8. ออบเจ็ค DataReader เป็นออบเจ็คที่ใช้ทำงานกับ result set ในลักษณะ forward-only (เลื่อนตำแหน่งเรคอร์ดไปข้างหน้าได้อย่างเดียว) และ read-only (อ่านข้อมูลได้อย่างเดียว แต่

เปลี่ยนแปลงแก้ไขไม่ได้) โดยระหว่างที่ใช้ออบเจ็คนี้ทำงานกับข้อมูลในฐานข้อมูล จะต้องเปิดการเชื่อมต่อกับฐานข้อมูลได้ตลอดเวลา จึงเป็นรูปแบบที่เรียกว่า connected data access ซึ่งตรงกันข้ามกับการทำงานของ DataSet แต่ข้อดีของ DataReader คือมันจะอ่นข้อมูลจากฐานข้อมูบเข้ามาในโปรแกรมได้รวดเร็วกว่า DataSet เนื่องจากเป็นการอ่านข้อมูลแบบไปข้างหน้าอย่างเดียว โดยไม่อนุญาตให้ย้อนกลับไปยังเรคอร์ดก่อนหน้าได้ DataReader จึงเหมาะสำหรับการอ่านข้อมูลเพื่อนำมาแสดงผลในโปรแกรมโดยไม่ให้ผู้ใช้แก้ไขข้อมูล หรือเพื่อนำข้อมูลมาสร้างรายงาน เป็นต้น

คลาสที่เป็นต้นแบบของงานออบเจ็คทั้งสี่นี้จะขึ้นอยู่กับชนิดของ Data Provider ดังรายการในตารางที่ 3 คลาสต้นแบบของงานออบเจ็คทั้งสี่นี้ที่ขึ้นอยู่กับชนิดของ Data Provider

ชนิดของ Data Provider	ชื่อคลาสต้นแบบของออบเจ็ค Connection, Command, DataAdapter และ DataReader ตามลำดับ
SQL Server Data Provider	SqlConnection, SqlCommand, SqlDataAdapter และ SqlDataReader โดยทั้งหมดอยู่ในเนมสเปซ System.Data.SqlClient
OLEDB Data Provider	OleDbConnection, OleDbCommand, OleDbDataAdapter และ OleDbDataReader โดยทั้งหมดอยู่ในเนมสเปซ System.Data.OleDb
Oracle Data Provider	OracleConnection, OracleCommand, OracleDataAdapter และ OracleDataReader โดยทั้งหมดอยู่ในเนมสเปซ System.Data.OracleClient
ODBC Data Provider	OdbcConnection, OdbcCommand, OdbcDataAdapter และ OdbcDataReader โดยทั้งหมดอยู่ในเนมสเปซ System.Data.Odbc

ตารางที่ 3 คลาสต้นแบบของงานออบเจ็คทั้งสี่นี้ที่ขึ้นอยู่กับชนิดของ Data Provider

2.3. การเชื่อมต่อกับฐานข้อมูล การเชื่อมต่อกับฐานข้อมูลเป็นหน้าที่ของออบเจ็ค Connection ดังที่อธิบายแล้ว คลาสที่ใช้สร้างออบเจ็ค Connection จะขึ้นอยู่กับชนิดของ Data Provider ที่ใช้ในการเชื่อมต่อฐานข้อมูล ซึ่งขึ้นอยู่กับประเภทของฐานข้อมูลที่ต้องการเชื่อมต่ออีกทีหนึ่ง เช่นสำหรับการเชื่อมต่อกับฐานข้อมูล Microsoft Access โดยทั่วไปจะใช้คลาส OleDbConnection ในเนมสเปซ System.Data.OleDb แต่ถ้าเป็นฐานข้อมูล Microsoft SQL Server ก็จะใช้คลาส SqlConnection ในเนมสเปซSystem.Data.SqlClient เป็นต้น

หลังจากสร้างออบเจ็คของคลาสที่เหมาะสมขึ้นมาแล้ว ให้กำหนดรายละเอียดการเชื่อมต่อกับฐานข้อมูลที่พร็อพเพอร์ตี้ ConnectionString ในที่นี้จะขอยกตัวอย่างการเชื่อมต่อกับฐานข้อมูล Northwind (ไฟล์ NWIND.mdb ที่เราก๊อปปี้มาไว้ในโฟลเดอร์ C:\ในบทก่อน) ดังนี้

```
Dim cn As New System.Data.OleDb.01eDbConnection()
Cn. ConnectionString = "Provider=Microsoft.jet.OLEDB.4.0;" & _
                     = "Data Source=C:\NWIND.MDB"
```

บรรทัดแรกคือการสร้างออบเจ็คของคลาส OleDbConnection ขึ้นมา ซึ่งอาจเขียนให้สั้นลงได้ว่า

```
Dim cn As New OleDb.OleDbConnection()
```

ทั้งนี้เพราะว่า Visual Studio จะอิมพอร์ตเนมสเปซ System.Data เข้ามาให้อยู่แล้ว หรือถ้าคุณต้องการเขียนโค้ดให้สั้นลงอีกโดยระบุเฉพาะชื่อคลาสหลังคำสั่ง New ก็ให้อิมพอร์ตเนมสเปซ System.Data.OleDb เข้ามาเอง โดยเพิ่มโค้ดไว้ที่ตอนต้นของฟอร์ม (ให้อยู่นอกการประกาศฟอร์ม ดังนี้

```
Imports System.Data.OleDb
```

สำหรับบรรทัดที่สองคือการกำหนดค่าสตริงที่เรียกว่า connection string ให้กับพร็อพเพอร์ตี้ ConnectionString ของออบเจ็ค Connection ที่สร้างขึ้น ซึ่งภายใน connection string เป็นการระบุค่าพารามิเตอร์ต่างๆที่เกี่ยวข้องกับการเชื่อมต่อานข้อมูล โดยให้ใช้เครื่องหมาย semicolon (;) ในการแยกแต่ละพารามิเตอร์ออกจากกัน ในตัวอย่างนี้ connection string ประกอบด้วยพารามิเตอร์ Provider และ Data Source (มีช่องว่างระหว่าง Data กับ Source) ซึ่งมีความหมายดังนี้

Provider	ใช้ระบุ provider ของประเภทฐานข้อมูลที่ต้องการเชื่อมต่อ ซึ่งฐานข้อมูล Microsoft Access มี provider ชื่อว่า Microsoft.Jet.OLEDB.4.0 (ตัวเลข 2 ตัวท้ายจะขึ้นอยู่กับเวอร์ชั่นของ provider)

Data Source ใช้ระบุชื่อพาธเต็ม (full path name) ของไฟล์ฐานข้อมูล

พารามิเตอร์ใน connection string ที่ต้องระบุค่าจะขึ้นอยู่กับประเภทของฐานข้อมูล รวมทั้งคุณสมบัติในการเชื่อมต่อที่คุณต้องการกำหนด

2.4.เปิดและปิดการเชื่อมต่อ ถึงแม้ว่าจะสร้างออบเจ็ค Connection และกำหนด connection string แล้ว แต่การเชื่อมต่อกับฐานข้อมูลจะยังไม่เกิดขึ้น จนกว่าจะเรียกใช้เมธอด Open ของออบเจ็ค Connection เพื่อเปิดการเชื่อมต่อ ดังตัวอย่าง

```
cn. Open()
```

หลังจากทำงานกับฐานข้อมูลเสร็จแล้ว ควรปิดการเชื่อมต่อกับฐานข้อมูลโดยเรียกเมธอด Close ของออบเจ็ค Connection ด้วยทุกครั้ง
การทำงานกับฐานข้อมูลด้วยออบเจค Command

```
cn. Close()
```

อันที่จริงออบเจ็ค Connection ยังมีพร็อพเพอร์ตี้ของเมธอด (รวมทั้งอีเวนต์) อื่นๆ อีกมาก แต่เท่าที่ได้กล่าวมานี้ก็เพียงพอสำหรับการเชื่อมต่อกับฐานช้อมูลแล้ว

2.5. การทำงานกับข้อมูลในฐานข้อมูลด้วยออบเจ็ค Command ออบเจ็ค Command ชื่อก็บอกอยู่แล้วว่าเป็นตัวแทนของคำสั่งที่ใช้ทำงานกับฐานข้อมูล ซึ่งคำสั่งทีว่านี้อาจเป็น query ของประโยคคำสั่ง SELECT, INSERT, UPDATE และ DELETE ในภาษา SQL หรือเป็นการเยกไปยัง stored procedure ในฐานข้อมูลก็ได้ (รวมทั้งอาจจะเป็นคำสั่งในกลุ่ม DDL ของภาษา SQL ที่ใช้สร้างเทเบิลใหม่. แก้ไขโครงสร้างของเทเบิล ฯลฯ ก็ได้เช่นกัน)

ออบเจ็ค Command อาจสร้างขึ้นมาจากคลาส OleDbCommand. SqlCommand. OracleCommand หรือ OdbcCommand ขึ้นอยู่กับชนิดของ Data Provider ที่คุณเลือกใช้ ในที่นี้จะกล่าวถึงคลาส OleDbCommand เนื่องจากฐานข้อมูลที่เราจะทำงานด้วยเป็นประเภท Microsoft Access

คอนสตรัคเตอร์ของคลาส OleDbCommand มีหลายรูปแบบ (เป็น overloaded constructor) แต่รูปแบบอย่างง่ายที่สุดนั้นไม่ต้องการอาร์กิวเมนต์ใดๆ เลย ตัวอย่างนี้แสดงการสร้างออบเจ็คของคลาส OleDbCommand โดยเรียกใช้คอนสตรัคเตอร์อย่างง่ายที่สุด

```
Dim cm As New OleDb.OleDbcommand()
```

หลังจากสร้างออบเจ็ค Command ขึ้นมาแล้ว คุณจะต้องกำหนดค่าให้กับพร็อพเพอร์ตี้ต่างๆ ของมัน ได้แก่พร็อพเพอร์ตี้ Connection, CommandText และ (ถ้าจำเป็น) CommandType ดังรายละเอียดต่อไปนี้

2.6. พร็อพเพอร์ตี้ Connection คำสั่งในการออบเจ็ค Command จะถูกส่งไปประมวลผลยังฐานข้อมูลผ่านทางการเชื่อมต่อที่ระบุไว้ที่พร็อพเพอร์ตี้ Connection นี้ ดังนั้นคุณจะต้องนำออบเจ็ค Connection มากำหนด โดยต้องเป็นชนิดที่เหมาะสมด้วย เช่น ถ้าออบเจ็ค Command เป็นชนิด OleDbCommand ก็จะต้องใช้ออบเจ็ค Connection ชนิด OleDbConnection เป็นต้น ดังตัวอย่าง

```
Dim cn As New OleDb.OleDbConnection()
Cn. ConnectoinString = “Provider=Microsoft.Jet.OLEDB.4.0;” & _
                     = “Data Source= C:\NWIND.MDB”
Dim cm As New OleDb.OleDbCommand()
cm. Connection = cn
```

ทั้งนี้การเชื่อมต่อกับฐานข้อมูลจะต้องเปิดอยู่ในระหว่างที่เรียกใช้เมธอดของออบเจ็ค Command เพื่อส่งคำสั่งไปประมวลผลยังฐานข้อมูล (จะกล่าวถึงเมธอดเหล่านี้ต่อไป) มิฉะนั้นจะต้องเกิดข้อผิดพลาดขึ้นในช่วง run time (exception)

2.7. พร็อพเพอร์ตี้ CommandText ใช้กำหนดคำสั่งที่จะส่งไปประมวลผลยังฐานข้อมูล เช่น ถ้าต้องการดึงข้อมูลจากฟิลด์ CustomerID และ CompanyName ในเทเบิล Customers จะต้องกำหนดค่าสตริงที่เป็นคำสั่ง SQL ให้กับพร็อพเพอร์ตี้ CommandText ของออบเจ็ค Command ดังนี้

```
cm. CommandTest = “SELECT CustomerID, CompanyName FROM Customers”
```

หรือถ้าต้องการเพิ่มเรคอร์ดใหม่เข้าไปในฐานข้อมูล ก็ให้ใช้ประโยคคำสั่ง INSERT แทน เช่น

```
cm. CommandText = “INSERT INTO Customer (CustomerID, CompanyNam) “ & _
                  “VALUES (‘ PROVIS’, ‘Provision’)”
```

คือการเพิ่มเรคอร์ดใหม่ลงในเทเบิล Customers โดยกำหนดข้อมูลเฉพาะฟิลด์ CustomerID และฟิลด์ CompanyName (อย่าลืมว่าคำสั่งสตริงในคำสั่ง SQL จะต้องครอบด้วย single quote)

2.8. พร็อพเพอร์ตี้ CommandType เป็นพร็อดเพอร์ตี้ที่ใช้กำหนดว่าค่าสตริงในพร็อพเพอร์ตี้ CommandText คือคำสั่งแบบใดโดยค่าที่กำหนดได้คือ (ค่าเหล่านี้ถูกรวบรวมไว้ในอีนิวเมอเรชั่น CommandType)

Text	กำหนดว่าค่าสตริงในพร็อพเพอร์ตี้ CommandText คือประโยคคำสั่งในภาษา SQL ซึ่งฐานข้อมูลของประมวลผลคำสั่ง SQL นั้น ค่า Text นี้เป็นค่าดีฟอลต์ของพร็อพเพอร์ตี้ CommandType ดังนั้นถ้าหากว่าสตริงที่ถูกกำหนดไว้ที่พร็อพเพอร์ตี้ CommandText คือคำสั่ง SQL ก็ไม่จำเป็นต้องกำหนดพร็อพเพอร์ตี้ CommandType
StoredProcedure	กำหนดว่าค่าสตริงในพร็อพเพอร์ตี้ CommandText คือชื่อ stored procedure ในฐานข้อมูล ซึ่งจะทำให้ stored procedure นั้นถูกประมวลผล
TableDirect	กำหนดว่าค่าสตริงในพร็อพเพอร์ตี้ CommandText คือชื่อเทเบิลในฐานข้อมูล ซึ่งออบเจ็ค Command จะให้ผลลัพธ์เป็น result set ของข้อมูลทุกฟิลด์และทุกเรคอร์ดในเทเบิลนั้น

2.9. ประมวลผลคำสั่งในออบเจ็ค Command หลังจากกำหนดพร็อพเพอร์ตี้ Connection, CommandText และ CommandType แล้วให้คุณเรียกใช้เมธอดใดเมธอดหนึ่งต่อไปนี้ของออบเจ็ค Command เพื่อส่งคำสั่งไปประมวลผลยังฐานข้อมูล

- ExecuteNonQuery
- ExecuteScalar
- ExecuteReader

ความแตกต่างของเมธอดทั้งสามสี่อยู่ที่ลักษณะของค่าที่ส่งคืนกลับมา (return value) โดยเมธอด ExecuteNonQuery จะส่งคำสั่งไปประมวลผลแต่ไม่คืนค่าใดๆ กลับมาให้ เราจึงมักใช้เมธอดนี้ประมวลผลคำสั่ง SQL หรือ stored procedure ที่เป็นการเพิ่ม ลบ และแก้ไขข้อมูล

สำหรับเมธอด ExecuteScalar จะให้ผลลัพธ์เป็นข้อมูลฟิลด์แรกของเรคอร์ดแรกใน result set ไม่ว่า result set ที่ได้จะมีกี่ฟิลด์และกี่เรคอร์ดก็ตาม จึงเหมาะสำหรับการประมวลผลคำสั่ง SQL หรือ stored procedure ที่ให้หาผลสรุปของข้อมูล (ซึ่งการหาผลสรุปของข้อมูลมักจะให้ผลลัพธ์เป็นค่าเดียวอยู่แล้ว เช่น การนับจำนวนเรคอร์ดทั้งหมดในเทเบิลหนึ่ง เป็นต้น)

สุดท้ายเมธอด ExecuteReader จะให้ผลลัพธ์เป็นออบเจ็ค DataReader ที่ใช้เข้าถึง result set ในลักษณะ forward-only และ read-only ดังที่ได้กล่าวไปแล้ว เมธอด ExecuteReader จึงเป็นเพียงเมธอดเดียวของออบเจ็ค Command ที่ช่วยให้คุณเข้าถึงข้อมูลทั้งหมดใน result set ได้

2.10. การใช้งาน DataReader การใช้งานออบเจ็ค Command เพื่อประมวลผลคำสั่งที่ไม่มีการคืนค่าหรือคืนค่าเพียงค่าเดียวนั้นไม่มีอะไรยุ่งยาก จึงไม่ขออธิบายรายละเอียด ต่ำสหรับการประมวลผลคำสั่งที่ให้ค่าเป็น result se คุณต้องเรียกใช้เมธอด ExecuteReader ของออบเจ็ค Command ซึ่งจะได้ผลลัพธ์เป็นออบเจ็ค DataReader ดังนั้นเราจะมาดูรายละเอียดการใช้งาน DataReader กันสักเล็กน้อย

การสร้าง DataReaderออบเจ็ค DataReader ก็คือออบเจ็คของคลาส OleDbDataReader, SqlDataReader, OracleDataReader หรือ OdbcDataReader จากคลาสทั้งสี่นี้ได้โดยตรง แต่ออบเจ็ค DataReader จะได้มาจากการเรียกเมธอด ExecuteReader ของออบเจ็ค Command เท่านั้น ดังตัวอย่าง (สมมติว่า cm คือ ออบเจ็ค Command ที่เป็นสนแทนซ์ของคลาส OleDbCommand)

```
Dim dr As OleDb. OleDbDataReader
```

```
dr = cm. ExecuteReader ()
```

ตัวแปรออบเจ็คที่ใช้รับค่าจากเมธอด ExecuteReader จะต้องถูกประกาศเป็นชนิดของคลาส DataReader ที่เหมาะสมด้วย ในที่นี้ตัวแปร dr ถูกประกาศเป็นชนิด OleDbDataReader เนื่องจาก cm คือ ออบเจ็คของคลาส OleDbCommand (ให้สังเกตว่าการประกาศตัวแปร dr ไม่ได้ระบุคำสั่ง new เพื่อสร้างออบเจ็คขึ้นมาด้วยแต่อย่างใด)

2.11. การใช้ DataReader เข้าถึงข้อมูลใน Result Set หลังจากนำออบเจ็ค DataReader ที่ได้จากเมธอด ExecuteReader มาเก็บไว้ในตัวแปรออบเจ็คแล้ว เมื่อต้องการเข้าถึงข้อมูลใน result set จะต้องเรียกเมธอด Read ของ DataReader เพื่ออ่านข้อมูลของเรคอร์ดแรกเข้ามาเก็บไว้ใน DataReader ซึ่งถ้าหากเมธอด Read สามารถอ่านเรคอร์ดมาได้ มันจะให้ผลลัพธ์เป็นค่า True พร้อมทั้งขยับตัวชี้ไปยังเรคอร์ดถัดไปเพื่อเตรียมอ่านเรคอร์ดอื่นๆ เมื่อมีการเยกเมธอด Read อีก แต่ถ้าไม่มีเรคอร์ดให้อ่านข้อมูลแล้วเมธอด Read จะให้ผลลัพธ์เป็นค่า False ดังนั้นคุณจึงสามารถใช้เมธอด Read ร่วมกับลูป Do While เพื่ออ่านเรคอร์ดทั้งหมดใน result set เข้ามาได้ ดังตัวอย่าง

```
Do, While dr.Read()
    'โค้ดที่ใช้ทำงานกับแต่ละเรคอร์ดใน result set ซึ่งถูกอ่านมาเก็บไวใน DataReader"
Loop
```

พร็อพเพอร์ตี้ Item ของ DataReaderในแต่ละครั้งที่ใช้เมธอด Read อ่านเรคอร์ดหนึ่งจาก result set เข้ามายัง DataReader เราสามารถเข้าถึงข้อมูลในฟิลด์ต่างๆ ของเรคอร์ดนั้น (เรคอร์ดที่เพิ่งอ่านเข้ามา) ได้จากพร็อพเพอร์ตี้ Item และ DataReader ตามรูปแบบดังนี้ ซึ่งพร็อพเพอร์ตี้ Item จะให้ค่าเป็นข้อมูลชนิด Object

```
datareder. Item(field
```

datareder	ออบเจ็ค DataReader
field	ชื่อหรือหมายเลขลำดับของฟิลด์ที่ต้องการเข้าถึงข้อมูล

ตัวอย่างเช่น

```
While dr. Read()
    Dim myObject As Object = dr. Item(3)
    Dim myOtherObject As Object = dr . Item (“CustomerID”)
End While
```

เนื่องจากข้อมูลที่ได้จะเป็น Object คุณจึงต้องแปลงไปเป็นชนิดข้อมูลที่ถูกต้องสำรหับฟิลด์ต่างๆ เอง เช่น ถ้าฟิลด์ CustomerID เป็นชนิดสตริง คุณอาจแสดงค่าของฟิลด์นี้ออกมาใน Nessage box ได้โดยเขียนโค้ดดังนี้

```
MsgBox(dr. Item(“CustomerID”). ToString())
```

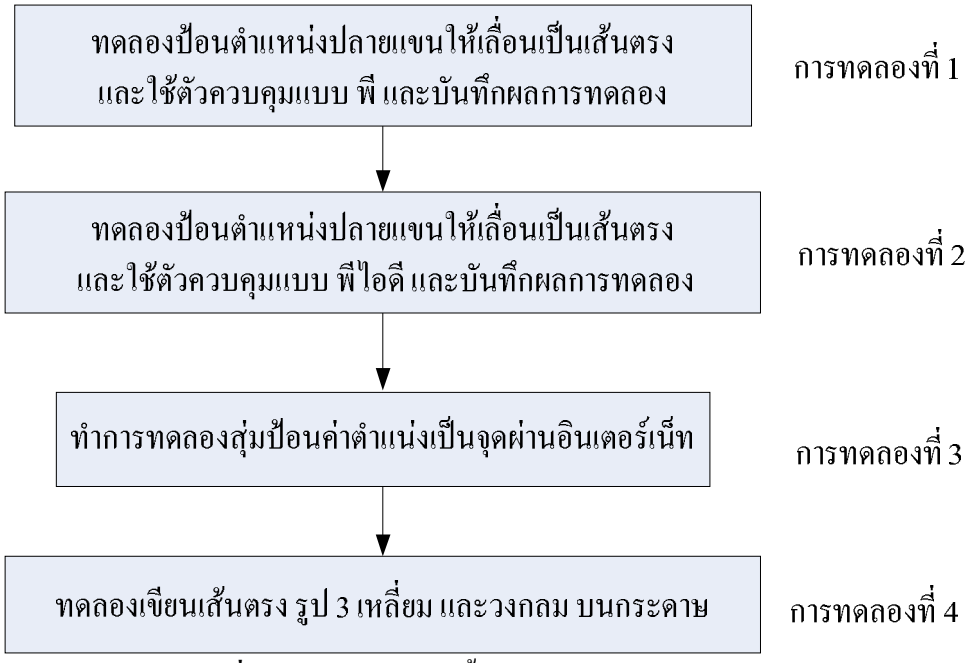
นั่นคือจะต้องเรียกเมธอดในข้อมูลชนิด Object ที่ได้เพื่อแปลงไปเป็นข้อมุลชนิดสตรองก่อน แล้วจึงส่งเป็นอาร์กิวเมนต์ให้กับฟังก์ชั่น MsgBox

2.12. ปิด DataReader หลังจากใช้งาน DataReader เสร็จแล้ว ให้ปิด DataReader โดยเรียกเมธอด Close ของมันด้วย มิฉะนั้น DataReader จะยังคงติดต่อกับฐานข้อมูลและส่งผลให้ออบเจ็คอื่นๆ ไม่สามารถติดต่อกับฐานข้อมูลผ่านทางออบเจ็ค Connection ที่ DataReader ใช้อยู่ได้ ตัวอย่างนี้แสดงการเรียกเมธอด Close เพื่อปิด DataReader ในตัวแปร dr

```
dr. Cloese()
```

ผลและวิจารณ์

ผล

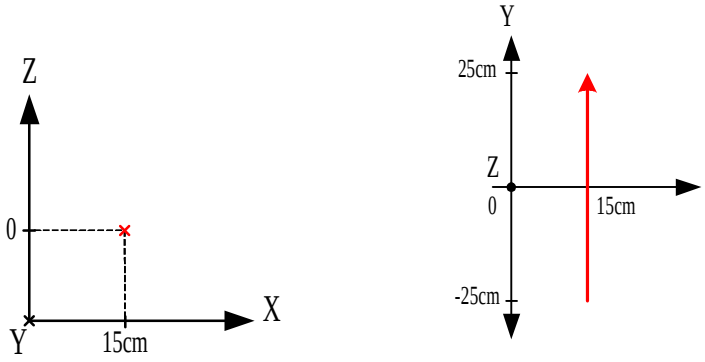


ภาพที่ 86โฟลชาร์จแสดงขั้นตอนการทดลอง

งานวิจัยนี้แบ่งการทดลองออกเป็น 3 ตามภาพที่ 86โฟลชาร์จแสดงขั้นตอนการทดลอง

การทดลองที่ 1 **ใช้ระบบควบคุมแบบพี** (P controller)

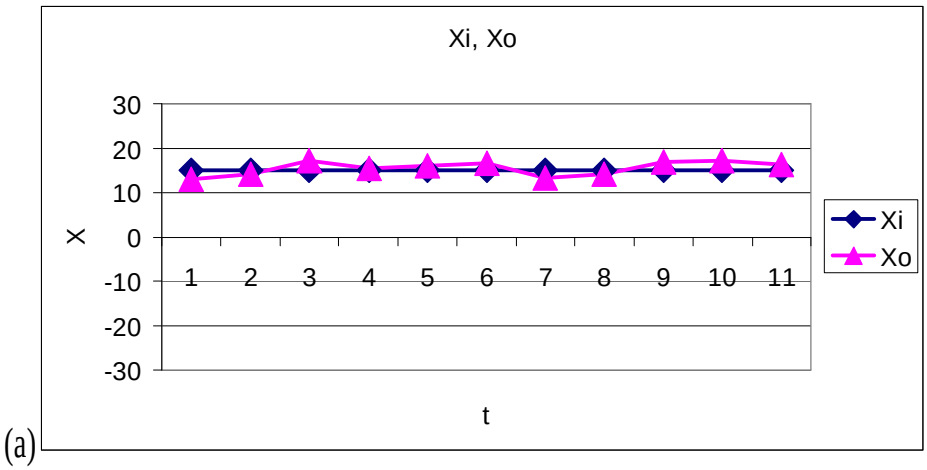
การทดลองที่ 1 เป็นการใช้ตัวควบคุมจาก พี เพื่อทดลองการทำงานของหุ่ยยนต์และหาความผิดพลาดของปลายแขนโดยเริ่มจากการป้อนตำแหน่งให้แขนหุ่นยนต์เคลื่อนที่เป็นเส้นตรงตามภาพที่ 87 เส้นทางการเคลื่อนที่ input คือตำแหน่องที่ต้องการ Output คือตำแหน่งที่วัดได้จริงแล้วบันทึกผลลงในลงตารางที่ 4 และนำผลที่ได้ไปวาดกราฟตาม ภาพที่ 88 กราฟผลการทดลองใช้ระบบควบคุมแบบพี (P controller) โดยที่ (a) คือแกน x, (b) คือแกน y และ (c) คือแกน z



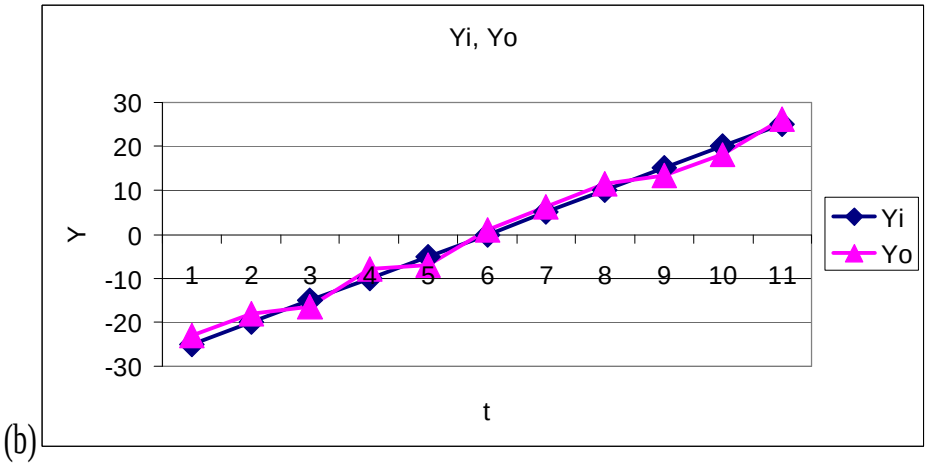
ภาพที่ 87 เส้นทางการเคลื่อนที่ในการทดลอง

Input					Output				
Xi	Yi	Zi	Roll_i	Pitch_i	Xo	Yo	Zo	Roll_o	Pitch_o
15	-25	0	0	0	13	-23.1	2	0	3
15	-20	0	0	0	14.1	-18	0.7	0	1
15	-15	0	0	0	17	-16.5	-1	0	2
15	-10	0	0	0	15.5	-8	-1.8	0	3
15	-5	0	0	0	16	-7	-0.5	0	2
15	0	0	0	0	16.5	1	-1.5	0	3
15	5	0	0	0	13.2	6.3	1	0	2
15	10	0	0	0	14	11.6	1.5	0	1
15	15	0	0	0	16.8	13.5	2	0	2
15	20	0	0	0	17	18.2	1	0	2
15	25	0	0	0	16.3	26	2	0	1

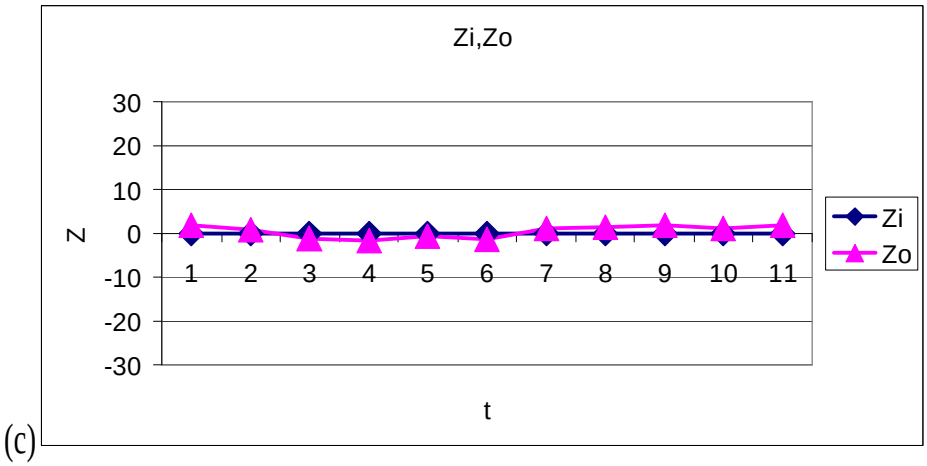
ตารางที่ 4 ผลการทดลองที่ 1 ระบบควบคุมแบบพี (P controller)



(a)



(b)



(c)

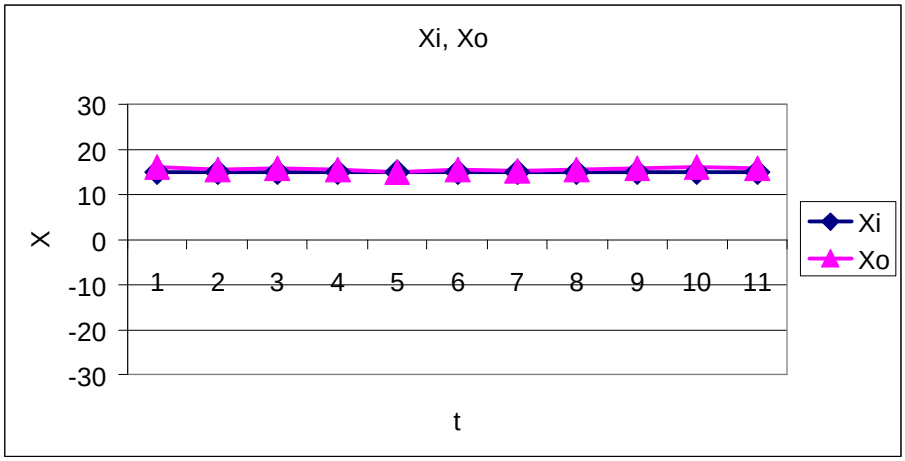
ภาพที่ 88 กราฟผลการทดลองใช้ระบบควบคุมแบบพี (P controller)

การทดลอง 2 ใช้ระบบควบคุมแบบพีไอดี (PID Controller)

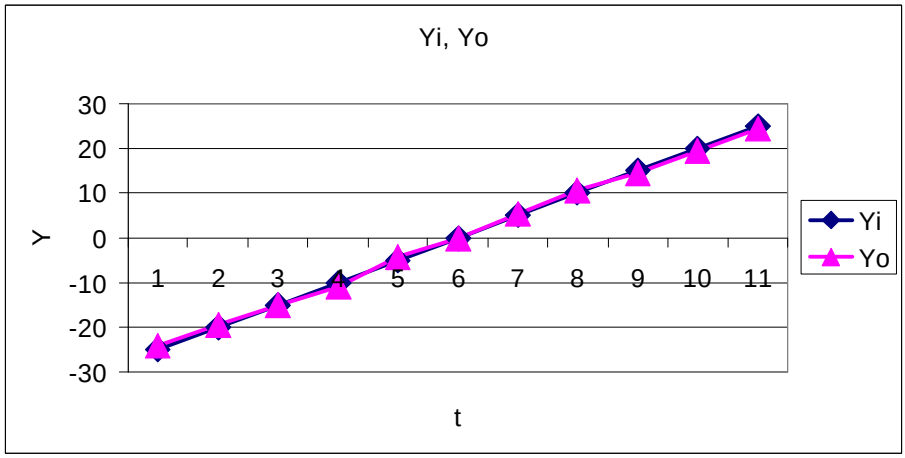
การทดลองที่ 2 เป็นการแก้ปัญหาเรื่องความผิดพลาดของตำแหน่งปลายแขนจากการทดลองที่ 1 ด้วยการเปลี่ยนระบบตัวควบคุมจาก พี เป็นแบบ พีไอดี ให้ผิดพลาดน้อยลง ขั้นตอนการทดลองคือ ป้อนตำแหน่งให้แขนหุ่นยนต์เคลื่อนที่เป็นเส้นตรงตามภาพที่ 84 เส้นทางการเคลื่อนที่ 3 โดนที่input คือตำแหน่องที่ต้องการ Output คือตำแหน่งที่วัดได้จริงแล้วบันทึกผลลงในลงตารางที่ 5 และนำผลที่ได้ไปวาดกราฟตาม ภาพที่ 89 กราฟผลการทดลองใช้ระบบควบคุมแบบพีไอดี (PID Controller) โดยที่ (a) คือแกน x, (b) คือแกน y และ (c) คือแกน z

Input					Output				
Xi	Yi	Zi	Roll_i	Pitch_i	Yo	Yo	Zo	Roll_o	Pitch_o
15	-25	0	0	0	15.9	-24.2	0.9	0	3
15	-20	0	0	0	15.5	-19.5	0.7	0	3
15	-15	0	0	0	15.8	-15	0.5	0	2
15	-10	0	0	0	15.5	-10.8	0.8	0	1
15	-5	0	0	0	15	-4.4	0.6	0	2
15	0	0	0	0	15.5	0	0.5	0	1
15	5	0	0	0	15.3	5.5	0.4	0	2
15	10	0	0	0	15.6	10.6	0.5	0	2
15	15	0	0	0	15.8	14.5	0.8	0	3
15	20	0	0	0	15.9	19.4	0.8	0	3
15	25	0	0	0	15.8	24.5	0.9	0	2

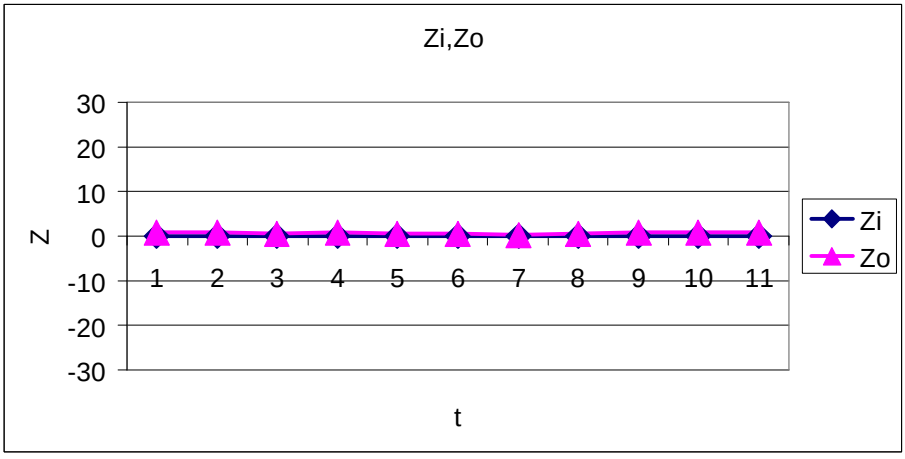
ตารางที่ 5 ผลการทดลองที่ 2 ระบบควบคุมแบบพีไอดี (PID Controller)



(a)



(b)



(c)

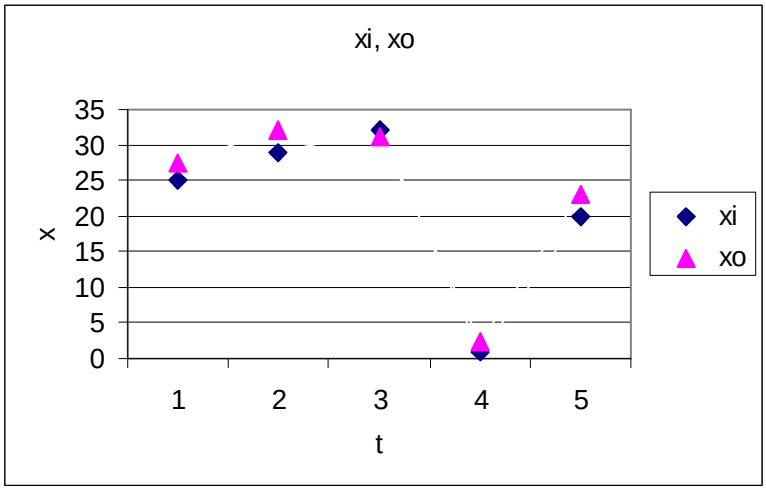
ภาพที่ 89 กราฟผลการทดลองใช้ระบบควบคุมแบบพีไอดี (PID Controller)

การทดลอง 3 การทดลองสุ่มป้อนค่าตำแน่งเป็นจุดผ่านอินเตอร์เน็ท

ผลการทดลองสุ่มป้อนค่าตำแน่งเป็นจุดผ่านอินเตอร์เน็ทคือการให้ผู้ใช้ (user) สุ่มป้อนตำแน่ง (x,y,z) ที่ผู้ใช้ต้องการให้มือจับไปผ่านโปแกรม Internet Explorer ในช่อง Adress ว่า http://158.108.45.58/remode เพื่อเข้าหน้าเวป จากนั้นกดปุ่ม OK เพื่อส่งค่าตำแหน่งไปยังแขนหุ่นยนต์ที่อยู่ที่เครื่อง Server ควบคุมหุ่นยนต์แล้วบันทึกผลลงในลงตารางที่ 6 การทดลองสุ่มป้อนค่าตำแน่งเป็นจุดผ่านอินเตอร์เน็ท และนำผลที่ได้ไปวาดกราฟตาม ภาพที่ 90 กราฟผลการทดลองสุ่มป้อนค่าตำแน่งเป็นจุดผ่านอินเตอร์เน็ท โดยที่ (a) คือแกน x, (b) คือแกน y และ (c) คือแกน z

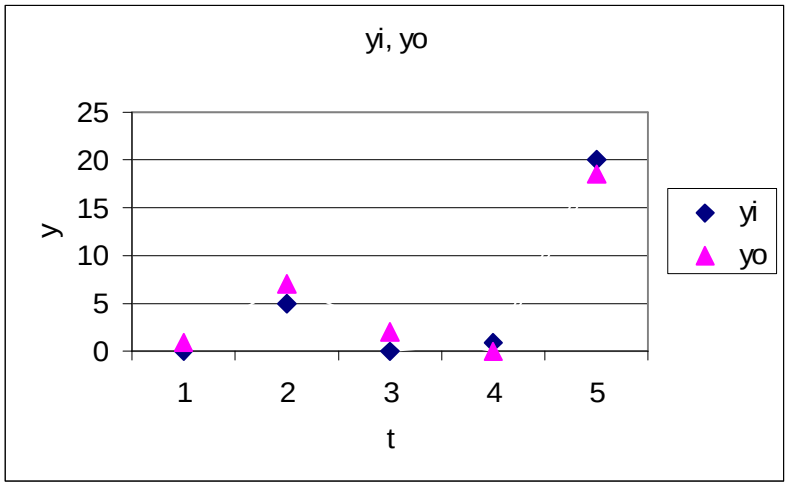
xi	yi	zi	xo	yo	zo
25	0	0	27.5	1	-1.5
29	5	10	32	7	9.5
32	0	0	31	2	2.5
1	1	25	2.5	0	27.5
20	20	0	23	18.5	2

ตารางที่ 6 ผลการทดลองที่ 3 ทดลองสุ่มป้อนค่าตำแน่งเป็นจุดผ่านอินเตอร์เน็ท

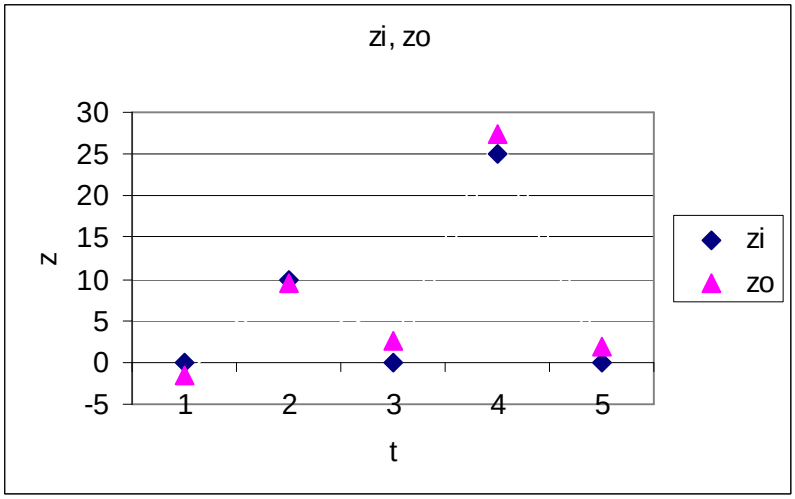


(a)

(b)



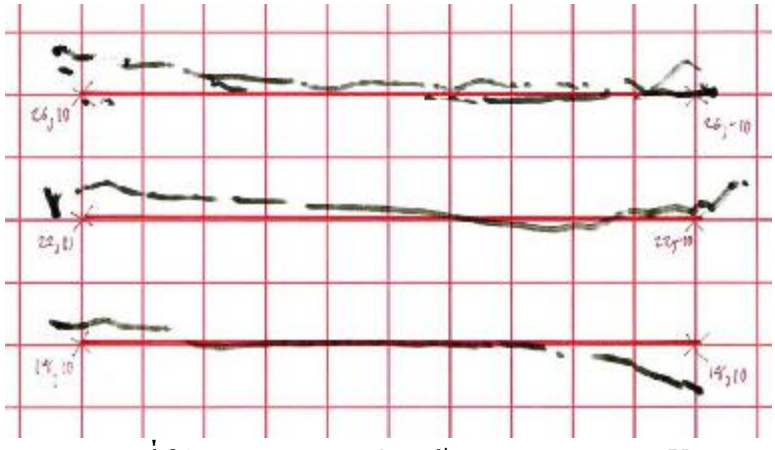
(c)



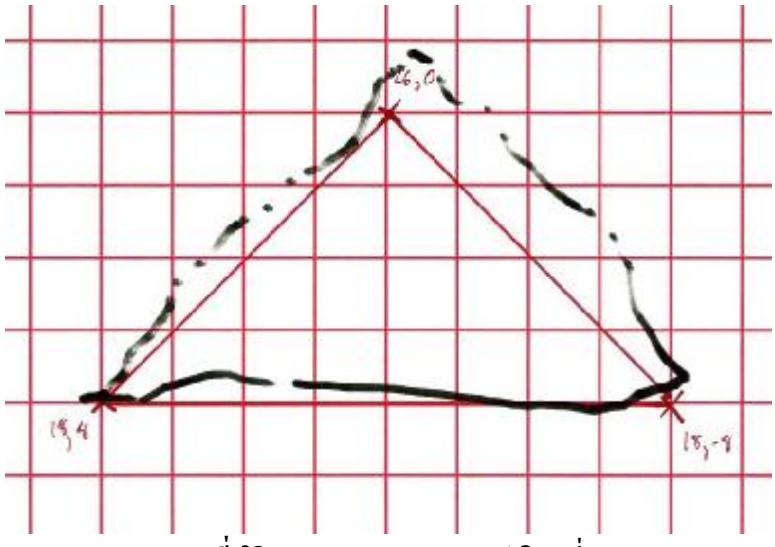
ภาพที่ 90 กราฟผลการทดลองสุ่มป้อนค่าตำแน่งเป็นจุดผ่านอินเตอร์เน็ท

การทดลอง 4 การทดลองเขียนเส้นตรง รูป 3 เหลี่ยม และวงกลม บนกระดาษ

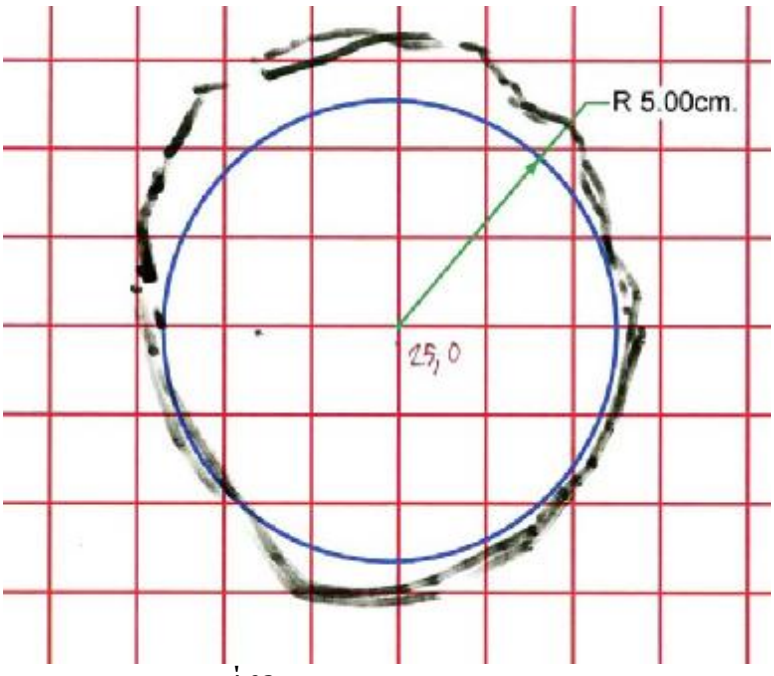
กาทดลอง 4 การทดลองเขียนเส้นตรง รูปสามเหลี่ยม และวงกลม บนกระดาษเพื่อการทดสอบ การสร้างเส้นทางเดินของแขนหุ่นยนต์ โดยใช้ตัวควบคุมเหมือนการทดลองที่ 3 เริ่มจากนำปากกาไปติดไว้ที่ปลายแขนแล้วทดลองสั่งให้เขียนเส้นตรง รูป 3 เหลี่ยม และวงกลม บนกระดาษที่ขีดเส้นตารางขนาด 2x2cm.ไว้ ตามภาพที่ 91 ผลการทดลองเขียนเส้นตรงตามแนวแกน Y ภาพที่ 92 ผลการทดลองวาดรูป 3 เหลี่ยม และ ภาพที่ 93 ผลการทดลองวาดรูปวงกลม เพื่อหาค่าความผิดที่เกิดขึ้น



ภาพที่ 91 ผลการทดลองเขียนเส้นตรงตามแนวแกน Y



ภาพที่ 92 ผลการทดลองวาดรูป 3 เหลี่ยม



ภาพที่ 93 ผลการทดลองวาดรูปวงกลม

วิจารณ์

การหาความผิดพลาดในการเคลื่อนของหุ่นยนต์จากแบบจำลองทางคณิตศาสตร์

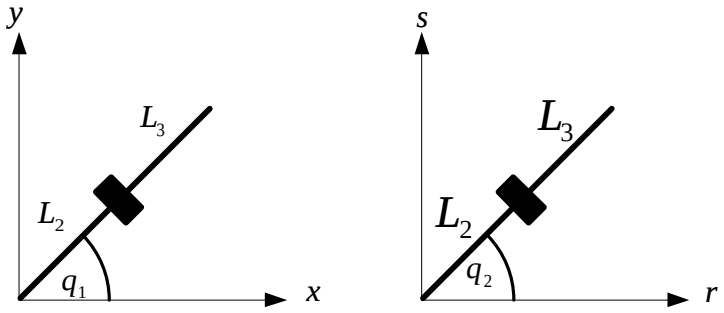
การความผิดพลาดในการเคลื่อนที่คือระยะทางที่น้อยที่สุดที่สามารถเคลื่อนที่ได้ถูกต้องซึ่งจากแบบจำลองทางคณิตศาสตร์จะทำให้ผลที่ได้ไม่เท่ากันซึ่งสัมพันธ์กับตำแหน่งปลายแขนที่ต้องการแต่การใช้งานจริงต้องเลือกเอาค่าความผิดพลาดที่เกิดขึ้นมากที่สุด โดยเริ่มจากหาความผิดของมอเตอร์แต่ละตัวก่อน โดยเริ่มจากค่ามุมที่น้อนที่สุดที่มอเตอร์จะสามารถหมุนไปได้ ถ้าพิจารณาจากความสามารถของโมดูล ADC ขนาด 10 Bits จะได้ค่าสูงสุดของ ADC ตามสมการที่ 64

$$ADC_{\max} = 2^{10} = 1024 \tag{64}$$

เมื่อมุมของแขนหุ่นยนต์ในแบบจำลองทางคณิตสาสตร์ จะเท่ากับ $0 - 2p \quad rad$

$\therefore$ มุมที่น้อนที่สุดที่มอเตอร์จะสามารถเคลื่อนที่ได้คือ

$$\Delta q = \frac{2p}{1024} = 0.0061 \quad rad \tag{65}$$



ภาพที่ 94 แสดงตำแหน่งของแขนหุ่นยนต์โดยกำหนดให้ค่า$q_3 = 0$

หลังจากที่ได้ค่า Δq มาแล้วก็จะนำมาหาค่าผิดพลาดของปลายแขนหุ่นยนต์ จากภาพที่ 94 แสดงตำแหน่งของแขนหุ่นยนต์โดยกำหนดให้ค่า$q_3 = 0$ เพราะว่าต้องการให้การเปลี่ยนแปลงของมุมมีผลต่อตำแหน่งของปลายแขนให้ค่ามากที่สุด แขนหุ่นยนต์ยืดยาวที่สุดแล้ว จะได้ $y = s$

และ $x = r$ ดังนั้นการหาความผิดพลาดนั้นสามารถใช้ q_1 หรือ q_2 ตัวใดตัวหนึ่งก็ได้ ในที่นี้จะใช้ค่า q_1 เท่านั้นโดยกำหนดให้

$$q_1 = 0 : \Delta q : \frac{p}{2} \tag{66}$$

$$q_2 = q_3 = 0 \tag{67}$$

ในสมการที่ 66 หมายถึงให้เพิ่มค่าq_1 ที่ละ Δq โดยเริ่มตังแต่ 0 ถึง $\frac{p}{2}$ หลังจากนั้นนำค่ามุมที่ได้ไปแทนค่าลงในแบบจำลองของแขนหุ่นยนต์

$$x = (\text{c1c2c3} - \text{c1s2s3})\text{L3} + \text{c1c2L2} \tag{68}$$

$$y = ((\text{c2c3} - \text{s2s3})\text{L3} + \text{c2L2})\text{s1} \tag{69}$$

$$z = (-\text{s2c3} - \text{c2s3})\text{L3} - \text{s2L2} \tag{70}$$

หลังจากนั้นใช้โปรแกรม MatLab หาค่าตำแหน่งของ x, y และ z ออกมา เมื่อได้ตำแหน่งของปลายแขนออกมาแล้วก็นำผลที่ได้ไปแทนในสมการที่ 71, 72 และ 73 เพื่อหาค่าความผิดพลาดในการเคลื่อนที่

$$dx = abs(x_{i-1} - x_i) \tag{71}$$

$$dy = abs(y_{i-1} - y_i) \tag{72}$$

$$dz = abs(z_{i-1} - z_i) \tag{73}$$

$$ds = \sqrt{dx^2 + dy^2 + dz^2} \tag{74}$$

$$ds = \sqrt{1.963^2 + 1.963^2 + 1.963^2} = 0.34 \tag{75}$$

จากที่ใช้โปรแกรม MatLab หาค่า dx, dy และ dz ตามสมการที่ 71, 72 และ 73 ก็สรุปได้ว่าค่าค่าความผิดพลาดแต่ละแกนที่น้อยที่สุดจากแบบจำลองทางคณิตศาสตร์คือ 0.018mm และมากที่สุดคือ 1.963 mm หรือ ซึ่งหมายความว่า ความแม่นยำของหุ่นยนต์ คือ ± 1.963mm ถ้านำผลที่ได้ไปหาเป็นเวคเตอร์ ds ตามสมการที่ 75 จะได้ ความแม่นยำของหุ่นยนต์เป็น ± 0.34 mm

การทดลอง 1

การทดลองนี้ใช้คัวควบคุมแบบพี (P controller) กำหนดการเคลื่อนที่ของมือจับเคลื่อนที่เป็นตรงตามภาพที่ 87 โดย ให้กำหนดให้จุดของตำแหน่งปลายแขนหลายๆจุดตามตารางที่ 4 ผลที่ได้คือแขนหุ่นยนต์สามารถเคลื่อนที่ไปยังตำแหน่งที่ต้องการได้ในระหว่างการเคลื่อนจะมีการแกว่งไปมาตลอดและมีค่าผิดพลาดของตำแหน่งปลายแขนประมาณไม่เกิน ± 2 cm.

การทดลอง 2

การทดลองนี้ใช้คัวควบคุมแบบพีไอดี (PID Controller) จากผลการทดลองในการทดลองที่ 1 ใช้คัวควบคุมแบบพี (P controller) มีค่าความผิดพลาดมากแต่ถ้าลองไปพิดจารณาการหาความผิดพลาดในการเคลื่อนของหุ่นยนต์จากแบบจำลองทางคณิตศาสตร์พบว่าความแม่นยำของหุ่นยนต์คือ ± 1.963mm แต่ในการทดลองที่ผ่านมาได้ ความผิดพลาด ± 2 cm. เพราะว่ามันเกิดได้จากหลายๆส่วนรวมกันทำให้มีค่ามาก เช่น โครสร้างทางฮาดแวร์ ความผิดพลาดของตัวเซอร์โวมอเตอร์ การวัดค่าความยาวของแขน และ การแปลงค่าของมุมให้เป็นค่ามุมหน่วยเป็น rad เป็น set point ของมอเตอร์ ในการทดลองที่ 2 นี้จะมุ่งไปที่การลอค่าความผิดพลาดลงให้มากที่สุด ซึ่งจะทดลองปรับปรุงในส่วนของเซอร์โวมอเตอร์โดยเปลี่ยนตัวควบคุมเป็นแบบพีไอดี (PID Controller) เพื่อที่จะลดค่า Steady State Error ลงโดย กำหนดการเคลื่อนที่ของมือจับเคลื่อนที่เป็นตรงตาม ภาพที่ 87 โดยให้กำหนดให้จุดของตำแหน่งปลายแขนหลายๆจุดตามตารางที่ 5 ผลที่ได้คือแขนหุ่นยนต์สามารถเคลื่อนที่ไปยังตำแหน่งที่ต้องการได้และในระหว่างการเคลื่อนยังมีการแกว่งไปมาลดลงเล็กและมีค่าผิดพลาดของตำแหน่งปลายแขนก็ลดลงเหลือประมาณไม่เกิน ± 1 cm. แต่กลับมีปัญหาใหม่เกิดขึ้นมาคือมอเตอร์มีการสั่นมาก

การทดลอง 3

การทดลองที่3 ทดลองสุ่มป้อนค่าตำแน่งเป็นจุดผ่านอินเตอร์เน็ท ในการทดลองนี้จะใช้ตัวควบคุมแบบพี(P controller) กับ เซอร์โวมอเตอร์เพราะว่าต้องการให้มอเตอร์สั่นน้อยที่สุดเพื่อยืดอายุการใช้งานถึงแม้ว่าวิธีการแบบนี้จะทำให้เกิดความผิดพลาดของตำแหน่งปลายแขนหุ่นยนต์ มากก็ตามแต่ถ้ามอเตอร์เสียหายมันก็ไม่คุ้มค่า หลังจากให้ผู้ใช้สั่งงานผ่านอินเตอร์เน็ทเป็นจุดๆแล้ว ผลที่ได้ในการทดลองจะคล้ายกับการทดลองที่ 1 คือมีค่าผิดพลาดของตำแหน่งปลายแขนประมาณไม่

เกิน ± 2 cm. และปัญหาในส่วนของอินเตอร์เน็ทคือ เมื่อกดปุ่ม OK แล้วจะเกิดการหน่วงเวลาเล็กน้อยก่อนที่แขนหุ่นยนต์จะเริ่มทำงาน เพราะว่าการส่งข็อมูลผ่านอินเตอร์เน็ทจะต้องเกิดการหน่วงเวลาที่ไม่เท่ากันเสมอเนื่องจากระยะและจำนวนผู้ใช้ในเครื่อข่าย

การทดลอง 4

การทดลองเขียนเส้นตรงและวงกลมลงบนกระดาษโดยใช้ตัวควบคุมเหมือนการทดลองที่ 3 ผลที่ได้นั้นคือเกิดค่าผิดพลาดของเส้นที่เขี่ยนประมาณ ± 2 cm. เหมือนกับการทดลองที่ 1 เพราะว่านำตัวควบคุมแบบพี ในการทดลองที่ 1 มาใช้ และเมื่อพิจารณาจากเส้นของปากกาที่พยายามเกาะเส้นอ้างอิงไม่เกิน 1 ช่องเสมอคือผิดพลาดประมาณ ± 2 cm

สรุปและข้อเสนอแนะ

สรุป

จากผลการทดลองใช้ตัวควบคุมแบบพี (P Controller) เปรียบเทียบกับแบบคัวควบคุมแบบพีไอดี (PID Controller) ก็พอจะสรุปได้ว่า การควบคุมแบบ พี นั้นมีค่าความผิดพลาดมากกว่าแบบพีไอดี เพราะว่า พีไอดีนั้นจะไม่มีความผิดพลาดในช่วง ภาวะสถานะคงตัว เป็นสาเหตุหนึ่งที่ทำให้ค่าความผิดพลาดนั้นลดลงจากเดิม ไม่เกิด ± 2 cm. ลดลงเป็น ± 1cm. แต่ในเรื่องของการใช้งานจริงนั้นก็ไม่ได้หมายความว่าแบบ พีไอดี นั้นจะดีกว่า แบบพี เสมอเนื่องจากแขนหุงยนต์ที่ใช้เป็นของเก่า ความแน่นของมอเตอร์กับจุดเชื่อมต่อของแขนมีช่องว่างมากพอสมควร การใช้ตัวควบคุมแบบพีไอดีนั้นจะดึงให้มอเตอร์วิ่งเข้าสู่ set poit ได้อย่างรวดเร็วและแม่นยำก็จริงแต่ก็ทำให้เกิดการสั่นของแขนหุ่นยนต์อย่างมากเช่นกัน ซื่อจะมีอายุการใช้งานของมอเตอร์และเฟืองนั้นสั้นลง ซึ่งการใช้งานจริงจะไม่คุ้มค่าถ้ามอเตอร์เสียหาย จึงใช้คัวควบคุมแบบพีอย่างเดียวถึงจะเหมาะสมที่สุดกับแขนหุ่นยนต์ตัวนี้ เพื่อเป็นการยืดอายุการทำงานของอุปกรณ์ให้ใช้งานได้นานๆ

ในส่วนของการควบคุมการทำงานผ่านอิเตอร์เน็ทข้อดีคือสามารถที่จะใช้เครื่องคอมพิวเตอร์เครื่องไหนก็ได้ที่ต่ออินเตอร์เน็นอยู่ควบคุมโดยไม่ต้องติดตังโปรแกรมอื่นแต่อย่างใด ส่วนข้อเสียของการควบคุมแบบนี้คือ ผู้ใช้สามารถที่จะเรียกใช้เวปได้พร้อมๆกันหลายคนก็จะเกิดปัญหาตามมาคือแขนหุ่นยนต์จะทำตามผู้ใช้คนสุดท้ายที่กดปุ่ม OK

ข้อเสนอแนะ

ข้อเสนอแนะในงานวิจัยนี้ต้องพัฒนาเรื่องความผิดพลาดที่เกิดขึ้นในการทดลองให้มีค่าลดลงน้อยกว่า ± 1cm. โดยที่มอเตอร์จะต้องไม่ทำงานหนักจนเกินไปด้วยและความปลอดภัยในการควบคุมระยะไกล

1.ปรับปรุงในส่วนของโครสร้างทางฮาดแวร์ปัญหาในข้อนี้น่าจะเป็นส่วนที่ทำให้มีค่าผิดพลาดมากที่สุดเพราะข้อต่อของแขนขยับได้เนื่องจากจุดเชื่อมต่อชุดเกียร์ที่ใช้หลวม

2.ต้องลอความผิดพลาดของเซ็นเซอร์โดยเพิ่มจำนวนบิตในการแปลงอะนาลอกเป็นดิจิตอล

3.การควบคุมระยะไกลต้องเพิ่มส่วน login ที่หน้าแรกเพื่อทำให้มีผู้ใช้คนเดียวที่สามารถควบคุมหุ่นยนต์ได้

เอกสารและสิ่งอ้างอิง

นคร ภักดีชาติ และ ชัยวัฒน์ ลิ้มพรจิตรวิไล. 2549 dsPIC microcontroller Basic experiment in C programming Witch MPLAB C30, สำนักพิมพ์ อินโนเวตีฟเอ็กเพอริเมนต์ จำกัด 3133/53 ซ.สุขุมวิท 102/2 ถ.สุขุมวิท บางนา กทม.10260

สัจจะ จรัสรุ่งรวีวรม. 2537. **คู่มือการเขียนโปรแกรมและใช้งาน** Visual basic.NET **ฉบับสมบูรณ์** 200 หมู่ 4 ห้อง 508 อาคารจัสมินอินเตอร์เนชั่นทาวเวอร์ ชั้น 5 ถ.แจ้งวัฒนะ อ.ปากเกร็ด จ. นนทบุรี 11120

สิทธิกร ลาภาพงศ์, จตุรนต์ พลวิชัย, วรพจน์ สันติพฤทธิ์ และ สยาม เจริญเสียง. 2545 ระบบการผลิตผ่านเว็บ, ใน **การประชุมวิชาการเครือข่ายวิศวกรรมเครื่องกลแห่งประเทศไทย ครั้งที่** 16 มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี 91 ถ.ประชาอุทิศ แขวงบางมด เขตทุ่งครุ กรุงเทพ 10140

สุรสิทธิ์ คิวประสพศักดิ์. 2546. **อินไซท์** Visual Basic .NET **ฉบับสมบูรณ์** สำนักพิมพ์ โปรวิชั่น กัด 408/75 ชั้น17 อาคารพหลโยธินเพลส ถ.พหลโยธิน สามเสนใน พญาไทย กทม.10400

Erkorkmaz, K and Altintas, Y. 2001. High speed CNC system design. Part I: jerk limited trajectory generation and quintic spline interpolation, pp. 1323-1345 In International Journal of Machine Tools & Manufacture 41 The University of British Columbia, Department of Mechanical Engineering, 2324 Main Mall, Vancouver, BC, Canada V6T 1Z4.

J. Craig, J. 2005. Introduction to Robotics Mechanics and Control 3th Edition Person Education, Inc. Pearson Prentice Hall Upper Saddle River, NJ07458.

Luo, X, Fan, X, Zhang, Z and Chen, T. 2004. Integrated Optimization of Trajectory Planning for Robot Manipulators Based on Intensified Evolutionary Programming, pp. 546-551 In IEEE International Conference on Robotics and Biomimetics College of I nfor mation Science and Engineering Central So uth University Changsha 410075, China.

M. Wronka, C and W. Dunnigan, M. 2006. Internet remote control interface for a Multipurpose robotic arm, pp. 179-183 In International Journal of Advanced Robotic Systems, Vol. 3, No. 1 Electrical, Electronic & Computer Engineering School of Engineering and Physical Sciences Heriot Watt University EH14 4AS Edinburgh, Scotland.

Roy, J and L. Whitcomb, L. 2002. Adaptive Force Control of Position/Velocity Controlled Robots: Theory and Experiment, pp. 121-137 In IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, VOL. 18, NO. 2 The authors are with the Department of Mechanical Engineering, JohnsHopkins University, Baltimore, MD 21218 USA.

W.Spong, M. 2006. Robot Modeling and Control John Wiley & Sons, Inc., Printed in the United States of America.

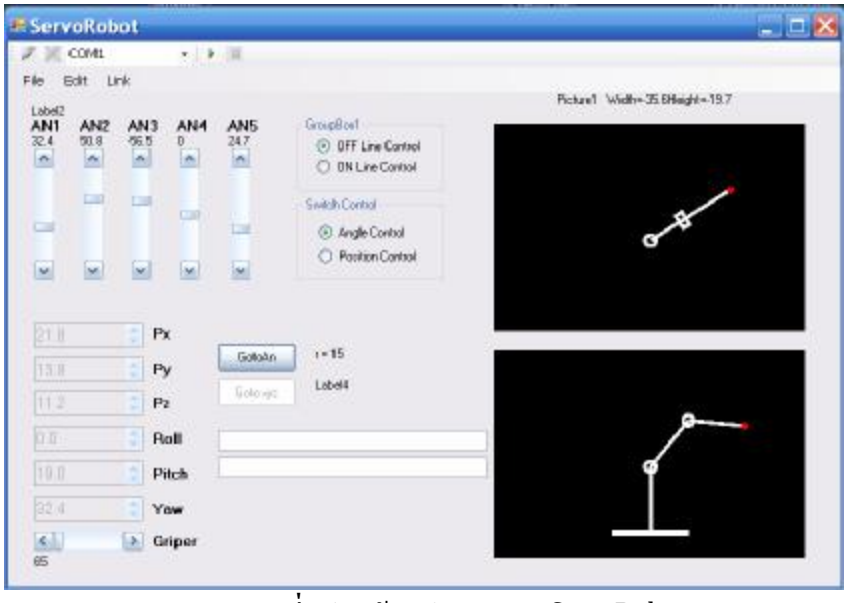
ภาคผนวก

ภาคผนวก ภาคผนวก ก

วิธีการใช้งานโปรแกรม ServoRobot

วิธีการใช้งานโปรแกรม ServoRobot

ก1.ขั้นแรกก็ดับเบิลคลิกเข้าโปรแกรมมาแล้วก็จะปรากฏโปรแกรม ServoRobot หน้าตาดังภาพ



ภาพผนวกที่ ก1 หน้าตาโปรแกรม ServoRobot

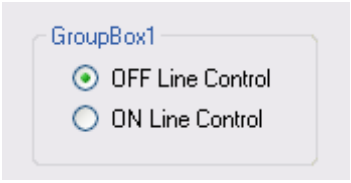
ก2.เลือกพล๊อตอนุกรมที่ต่ออยู่กับบอร์ด แล้วก็กด เพื่อ เชื่อมต่อหลังจากนั้นก็ลองสั่งงานแขนหุ่นยนต์โดยกดที่ปุ่ม GotoXYZ หุ่นยนต์ก็จะขยับตามคำสั่ง

ภาพผนวกที่ ก2 เลือกพล๊อตอนุกรมที่ใช้

ก3. เลือกว่าจะควบคุมมุมโดยตรงหรือจะคำนวณจากตำแหน่ง ถ้าเลือก Angle Control ในส่วนที่รับค่าตำแหน่งก็จะไมสามารถรับค่าได้แต่จะแสดงตำแหน่งปัจจุบันให้ดูตามจริง แต่ถ้าเลือก Position Control ก็จะไม่ลามารถเปลี่ยนค่าของแถบเลื่อนมุมได้ ปรับมุมได้

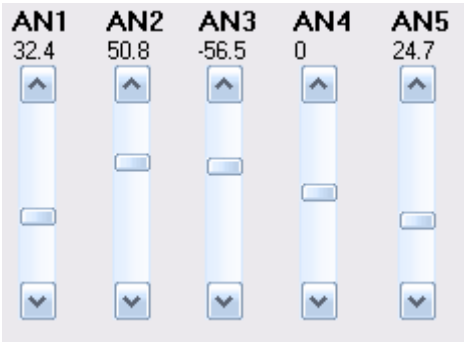
ภาพผนวกที่ ก3 เลือกโหมดการควบคุม

ก4.เลือกว่าจะควบคุมผ่านอินเตอร์เน็ทหรือแบบโดยตรง ถ้าเลือกแบบ on line หน้าจอในส่วนรับค่าทั้งหมดจะเป็นแบบดูได้อย่างเดียว



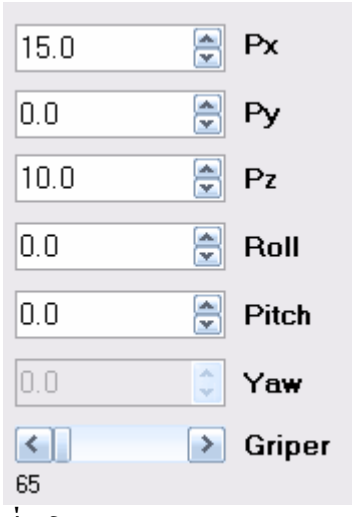
ภาพผนวกที่ ก4 ควบคุมผ่านอินเตอร์เน็ทหรือแบบโดยตรง

ก5.แถบเลื่อนมุม มีหน้าที่แสดงมุมจริงของมอเตอร์ และรับค่าด้วยในโหมด Angle control



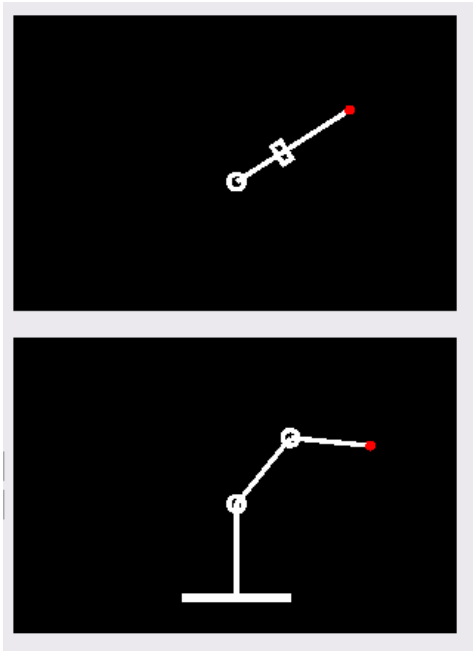
ภาพผนวกที่ ก5 แถบเลื่อนมุม

ก6.ช่องรับค่าตำแหน่งและมุมของมือจับ มีหน้าที่รับค่าตำแหน่งที่ต้องการให้เคลื่อนที่ไปและแสดงตำแหน่งปัจจุบันของมือจับในโหมดของ Position control



ภาพผนวกที่ ก6 ช่องรับค่าตำแหน่งและมุมของมือจับ

ก7.ส่วนแสดงผลแบบกราฟิกมีหน้าที่แสดงภาพรวมของแขนหุ่นยนต์ตามเวลาจริง



ภาพผนวกที่ ก7 ส่วนแสดงผลกราฟิก

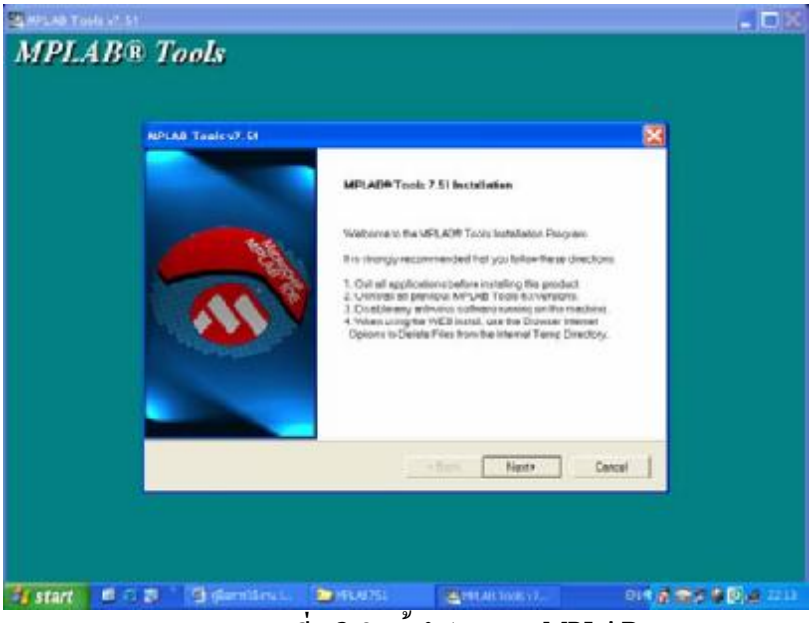
ภาคผนวก ภาคผนวก ข

ขั้นตอนการติดตั้งและใช้งาน ICD2

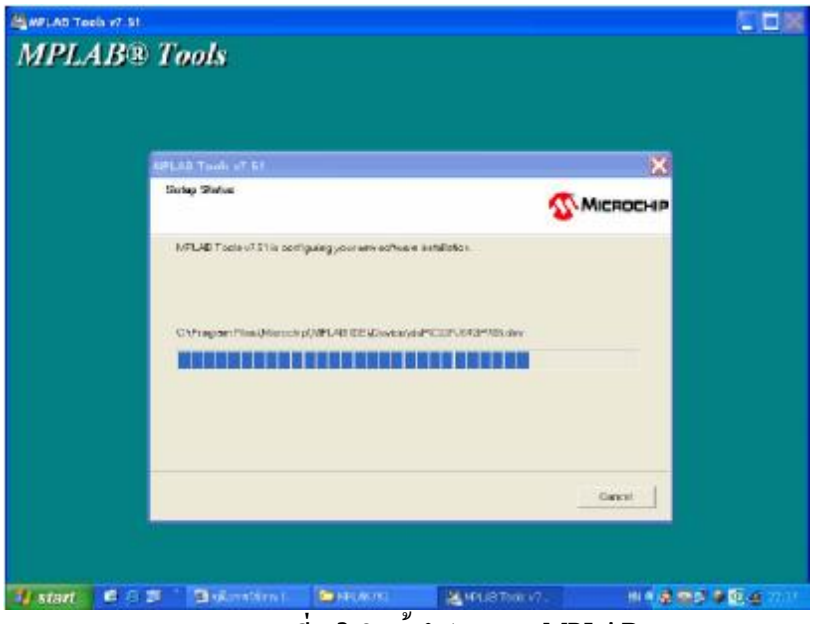
ขั้นตอนการติดตั้งและใช้งาน ICD2

ข1 ทำการติดตั้งโปรแกรม MPLAB ดังภาพผนวกที่ ข1-4

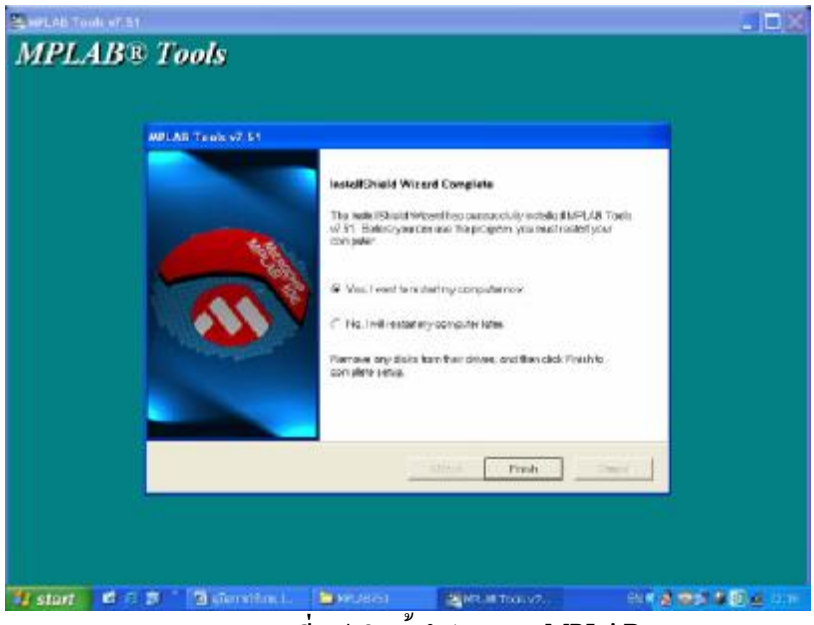
ภาพผนวกที่ ข1 ติดตั้งโปรแกรม MPLAB



ภาพผนวกที่ ข2 ติดตั้งโปรแกรม MPLAB

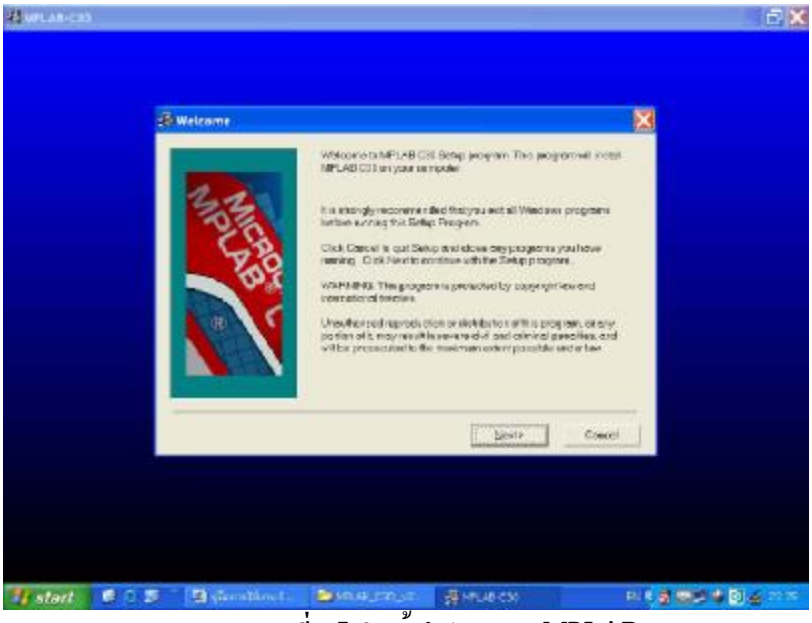


ภาพผนวกที่ ข3 ติดตั้งโปรแกรม MPLAB



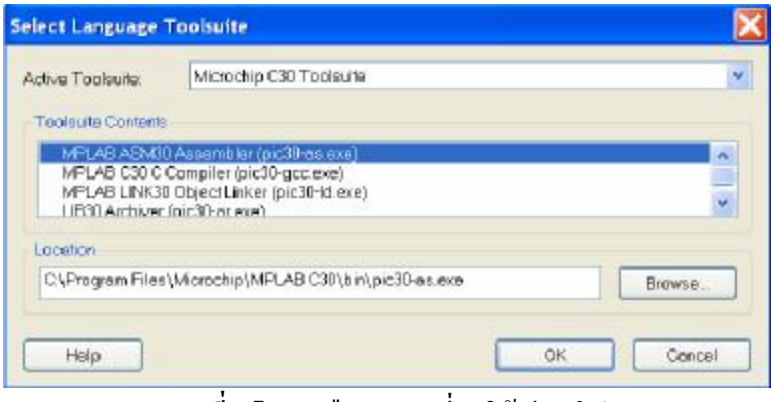
ภาพผนวกที่ ข4 ติดตั้งโปรแกรม MPLAB

หลังจากติดตั้งเสร็จแล้วจะมี  ขึ้นมาที่ Desktop ด้วย จากนั้นให้ทำการลงโปรแกรม MPLAB-C30 เข้าไปด้วยหากต้องการพัฒนาด้วยภาษา C ดังภาพผนวกที่ ข5



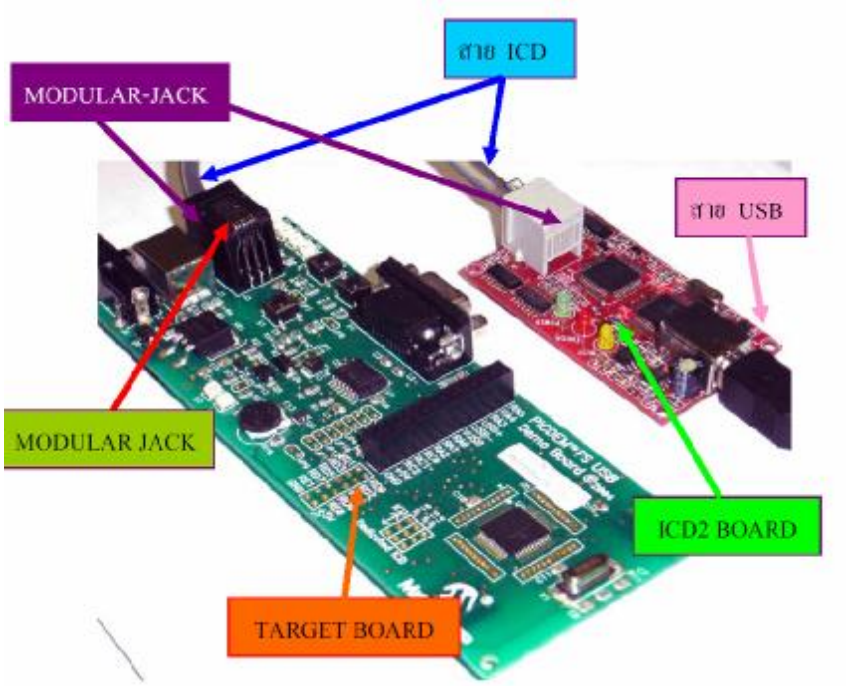
ภาพผนวกที่ ข5 ติดตั้งโปรแกรม MPLAB

Note: ถ้าหากลง MPLAB-C30 แล้วให้ Select Language Tool suite
โดยทำการเปิดโปรแกรม MPLAB แล้วจากนั้นไปที่ Project-> Select Language Tool suite ครับ



ภาพผนวกที่ ข5 การเลือกภาษาที่จะใช้เขียนโปรแกรม

ข1.1 จากนั้นทำการต่อสาย USB เข้ากับ Computer ทางช่อง USB และเสียบอีกด้านเข้ากับบอร์ด ICD2 จากนั้นเสียบ สาย ICD ระหว่าง MODULAR-JACK ของบอร์ด ICD2 เข้ากับ TARGET BOARD ตามภาพผนวกที่ ข6



ภาพผนวกที่ ข6 ส่วนประกอบของ ICD2

NOTE: การใช้งาน ICD2-USB เพื่อ Debug หรือ Program Target Board จะต้องจ่ายไฟเลี้ยงให้ Target Board ด้วย เนื่องจาก ICD-USB จะรับไฟเลี้ยง (VPP) จาก Target Board ในขณะทำงาน หลังจากนั้นให้สังเกตที่มุมขวาล่างของ Computer จะพบว่า

ภาพผนวกที่ ข6 การติดตั้งDriver ICD2

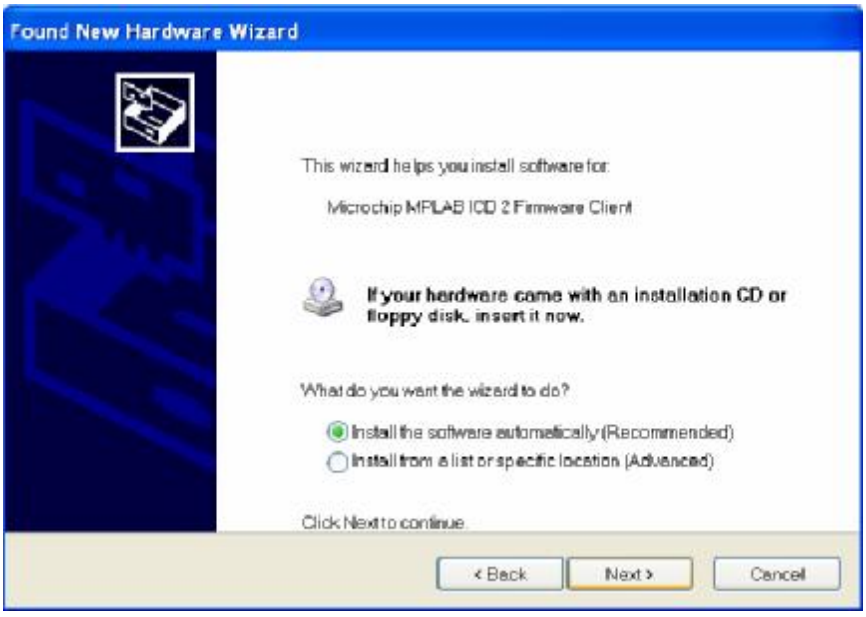
รอสักพัก Computer จะให้เลือก Driver เพื่อทำการติดตั้ง USB Component

ภาพผนวกที่ ข7 การติดตั้งDriver ICD2

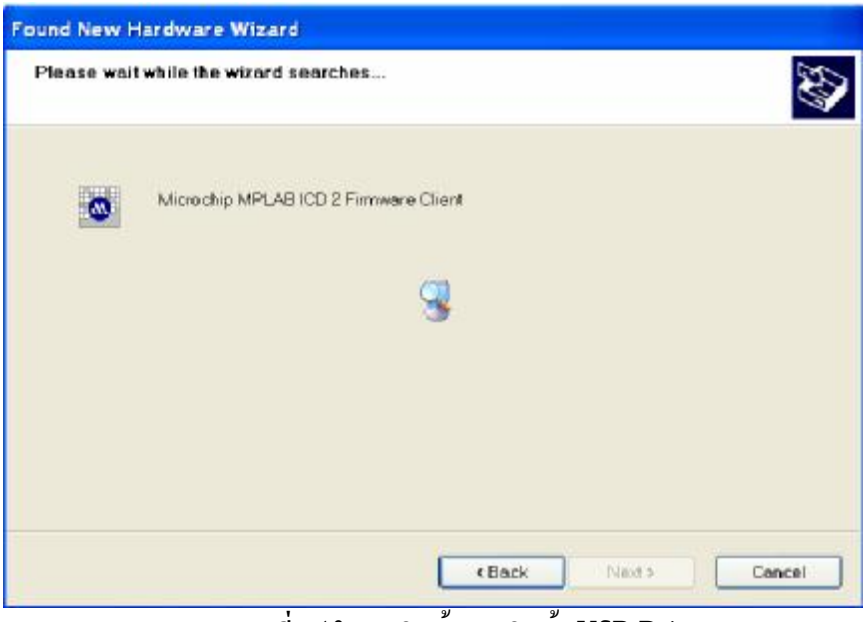
Note: หากเราไม่ทำการติดตั้ง USB Driver จะพบว่า

ภาพผนวกที่ ข8 การติดตั้งDriver ICD2

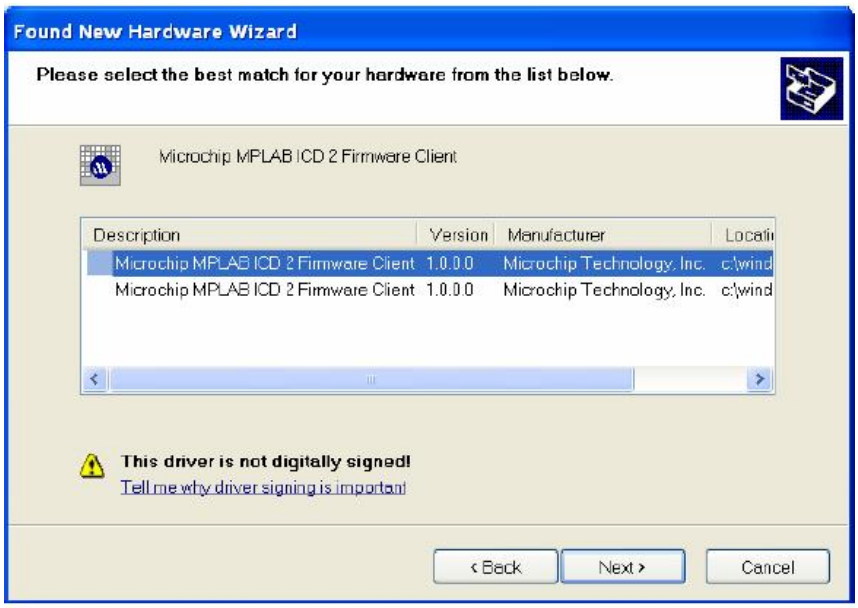
หลังจากนั้นให้เราคลิก NEXT เพื่อทำการติดตั้ง USB Driver ดังภาพผนวกที่ ข9



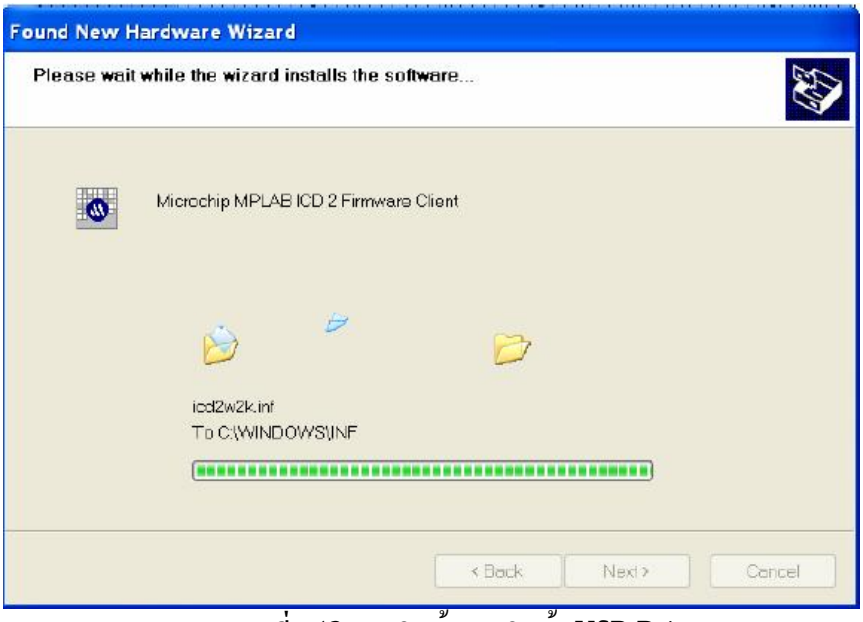
ภาพผนวกที่ ข9 การติดตั้งการติดตั้ง USB Driver



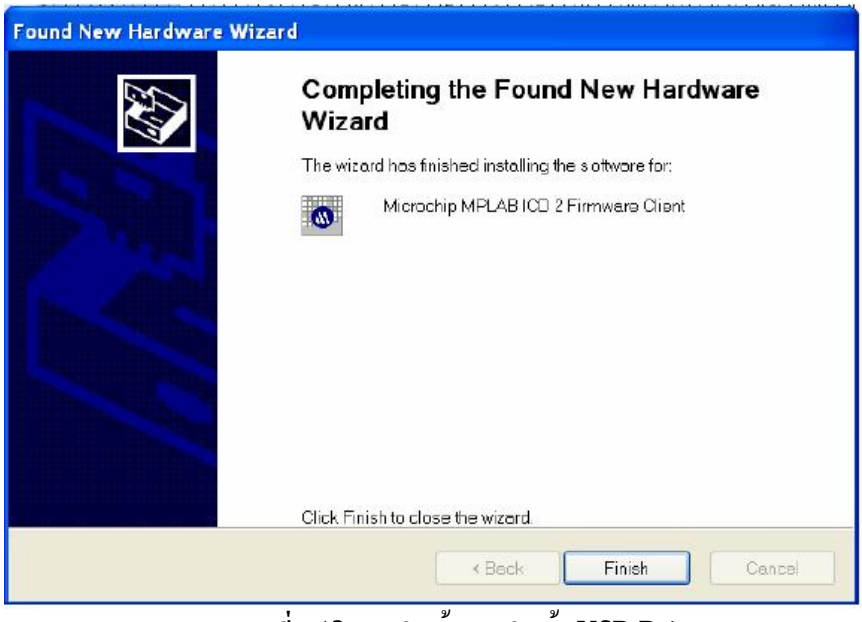
ภาพผนวกที่ ข10 การติดตั้งการติดตั้ง USB Driver



ภาพผนวกที่ ข11 การติดตั้งการติดตั้ง USB Driver

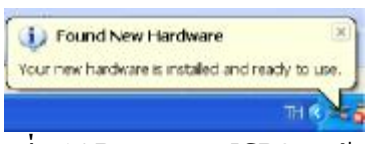


ภาพผนวกที่ ข12 การติดตั้งการติดตั้ง USB Driver



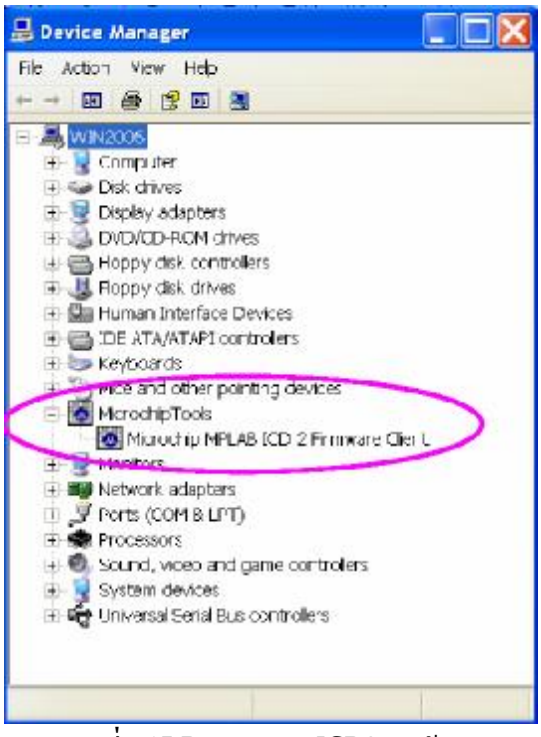
ภาพผนวกที่ ข13 การติดตั้งการติดตั้ง USB Driver

หากการติดตั้ง Driver ของ ICD2 ถูกต้องสมบูรณ์จะปรากฏข้อความดังภาพผนวกที่ ข14



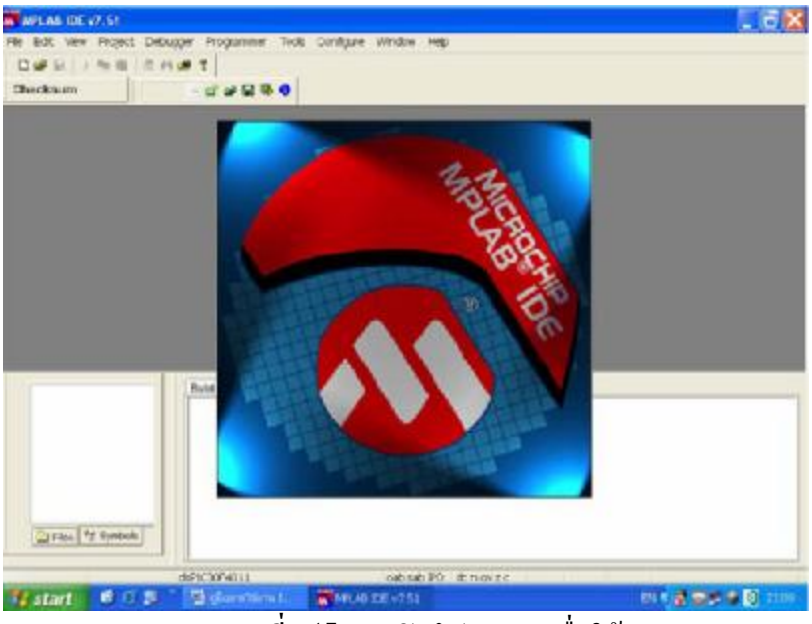
ภาพผนวกที่ ข14 Driver ของ ICD2 ถูกต้องสมบูรณ

Note: สามารถเข้าไปดูรายการ Devices ที่ติดตั้งใหม่ได้โดยไปที่ My computer->properties->Hardware->device manager ดัง ภาพผนวกที่ ข15



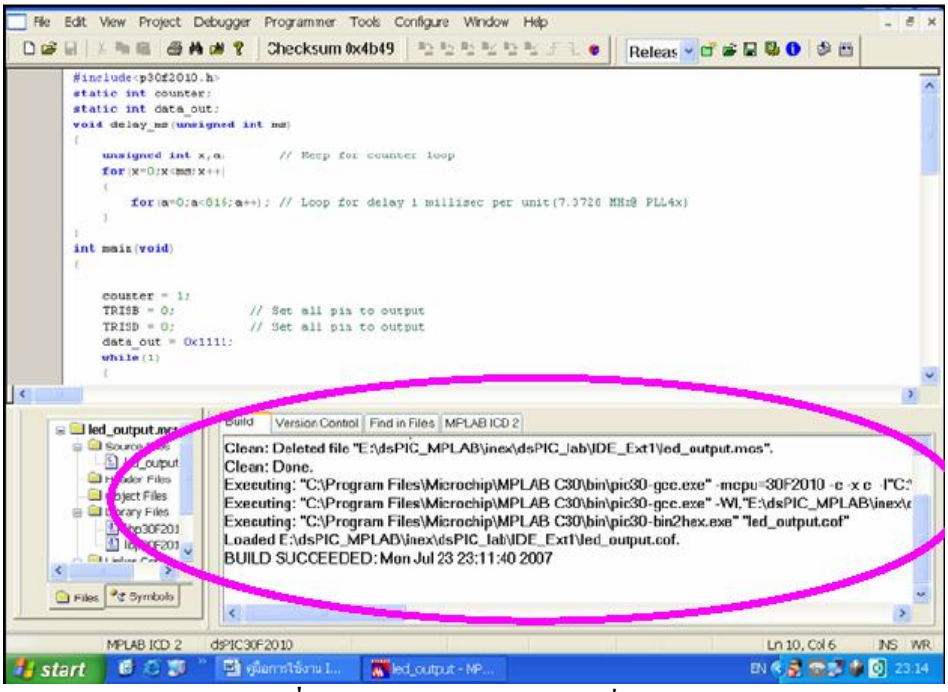
ภาพผนวกที่ ข15 Driver ของ ICD2 ถูกต้องสมบูรณ

ข1.2 เมื่อติดตั้งเรียบร้อยแล้วให้ทำการเปิดโปรแกรมโดยดับเบิ้ลคลิกที่ไอคอน 



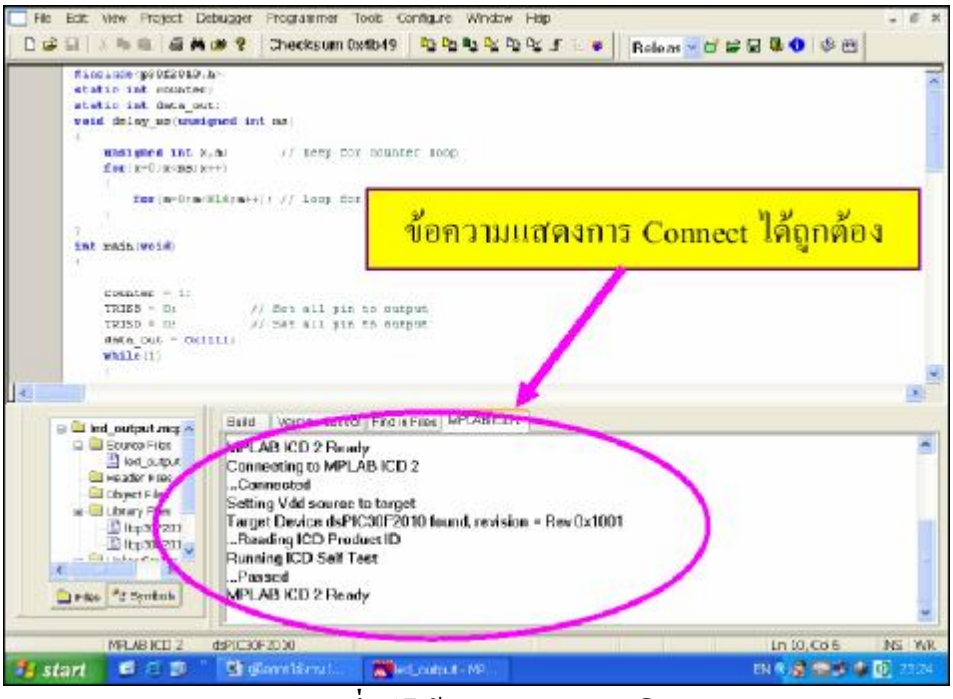
ภาพผนวกที่ ข15 การเปิดโปรแกรมเพื่อใช้งาน

ข1.3 เปิด Workspace หรือไฟล์ที่ต้องการพัฒนา แล้วทำการ complier ดังภาพผนวกที่ ข16



ภาพผนวกที่ ข16 Workspace หรือไฟล์ที่ต้องการพัฒนา

ข1.4 หากไม่มีข้อผิดพลาดใด ๆ MPLAB จะทำการสร้างไฟล์ .hex ที่ สามารถนำไป Program ได้ซึ่งเป็นโหมด Programmer แต่ก่อนที่เราจะเริ่มทำการ Program ชิปไอซีให้ทำการ connect กับบอร์ด ICD2 โดยไปที่เมนู Programmer->select programmer->MPLAB ICD2 จากนั้นไปคลิกที่ หรือเมนู Programmer->connect หากไม่มีข้อผิดพลาดใดๆ จะปรากฏข้อความดังภาพผนวกที่ ข17

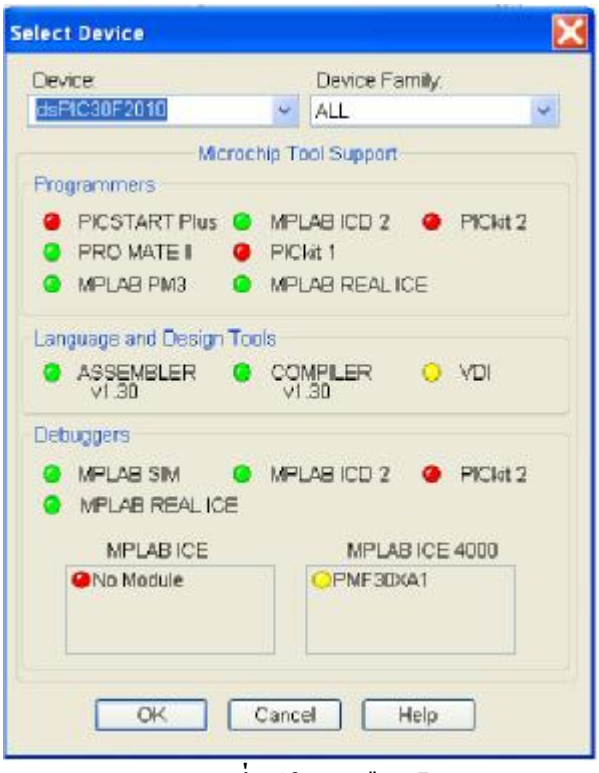


ภาพผนวกที่ ข17 ข้อความแสดงการ Connect

Note: สังเกตว่าขณะที่ Computer กำลังติดต่อกับ ICD2 นั้น จะมีไฟกระพริบที่ LED BUSY เป็นจังหวะ ๆ แต่หาก ICD2 ไม่สามารถ Connect กับ Target Board ได้จะปรากฏข้อความดังภาพผนวกที่ ข18

ภาพผนวกที่ ข18 ICD2ไม่สามารถ Connect กับ Target Board

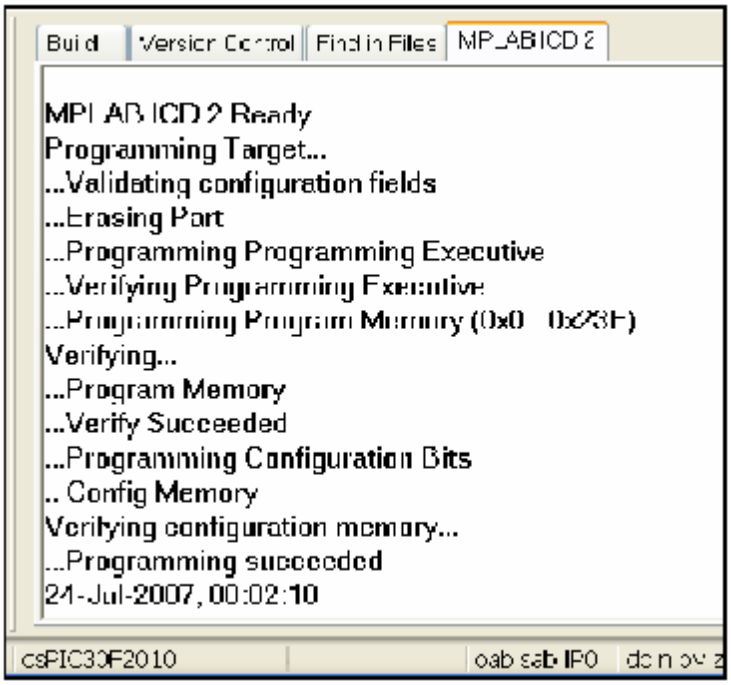
ให้ไปที่เมนู Configure->Select Device แล้วทำการเลือก Device ให้ตรงกับ Target Board ที่เรากำลังจะทำการ Program เช่น ภาพผนวกที่ ข19 จากนั้นทำการ Connect อีกครั้ง



ภาพผนวกที่ ข19 การเลือก Driver

ข1.5 หลังจากที่เราทำการ Connect ICD2 เรียบร้อย ตอนนี้ก็พร้อมที่Program ไฟล์ที่เราพัฒนาขึ้นได้แล้ว โดยไปที่เมนูProgrammer->program และถ้าการโหลดไฟล์เพื่อทำการโปรแกรมถูกต้องสมบูรณ์จะปรากฏข้อความดังภาพผนวกที่ ข20

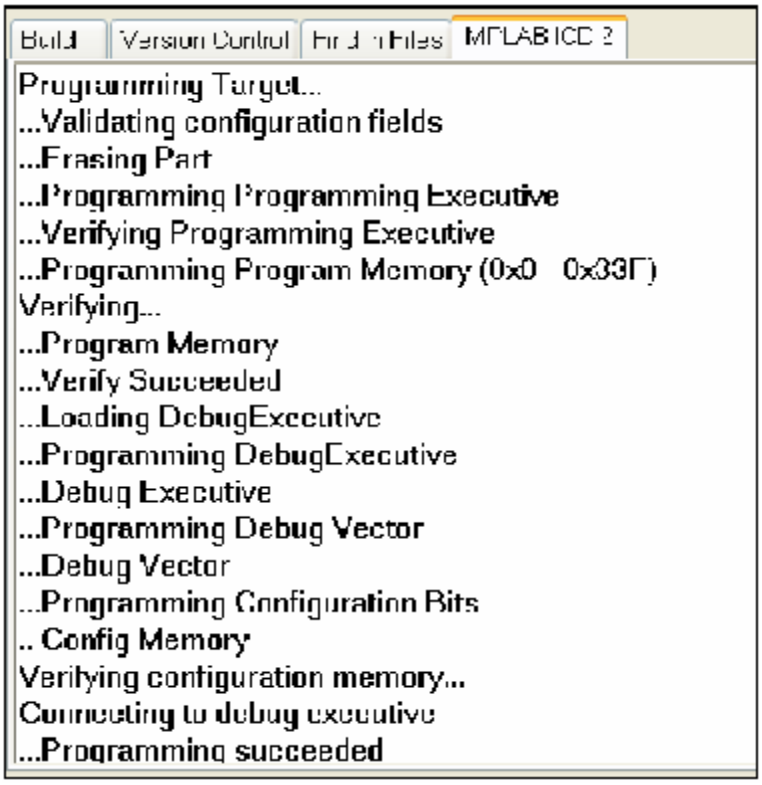
ภาพผนวกที่ ข20 การProgram



ภาพผนวกที่ ข21 การProgram

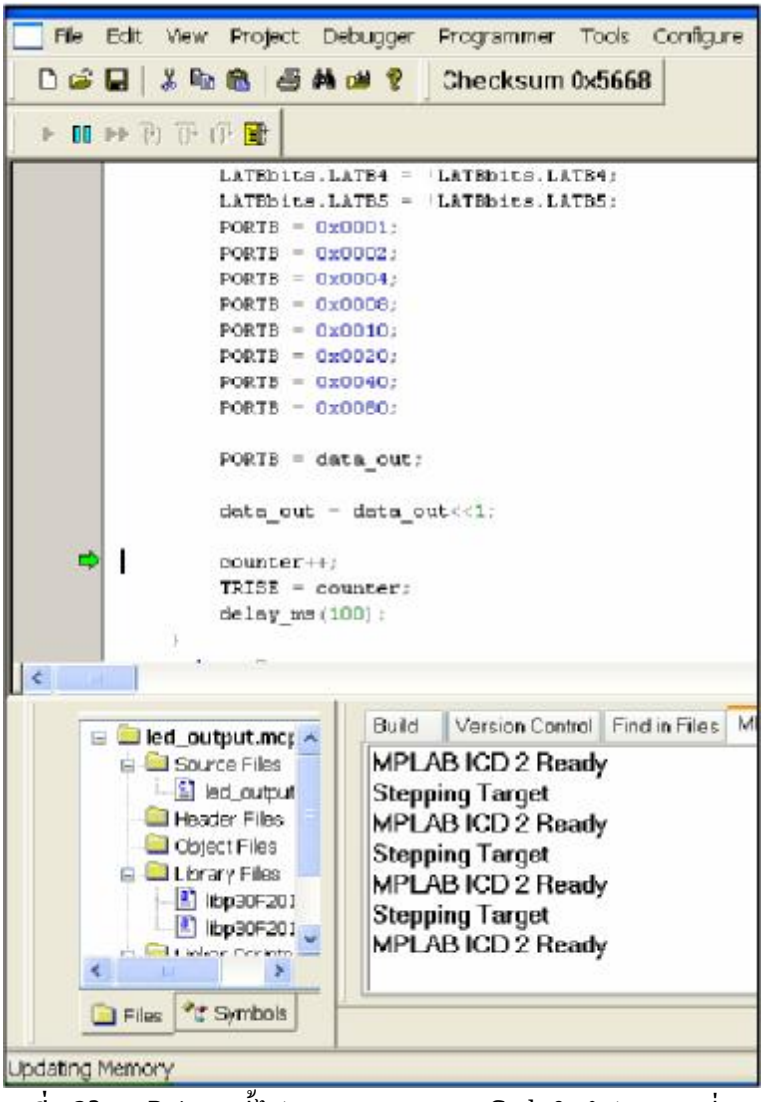
ข1.6 เมื่อการ Program เรียบร้อยแล้วให้ทำการสั่ง RUN โดยการสั่งฟังก์ชัน Release from Reset ก็เป็นอันเสร็จสิ้น ใน ทำนองเดียวกันเราสามารถใช้ฟังก์ชันอื่น ๆ ได้เช่น Verify, Erase Part, Blank Check, Read EEPROM

ข1.7 การใช้งานในโหมด Debugger เริ่มด้วยการไปที่เมนูDebugger->Select Tool->MPLAB ICD2 แล้วไปที่ Debugger->connect เพื่อทำการติดต่อกับ ICD2 หากการ เชื่อมต่อไม่มีปัญหา ให้ไปที่คำสั่ง Debugger->program



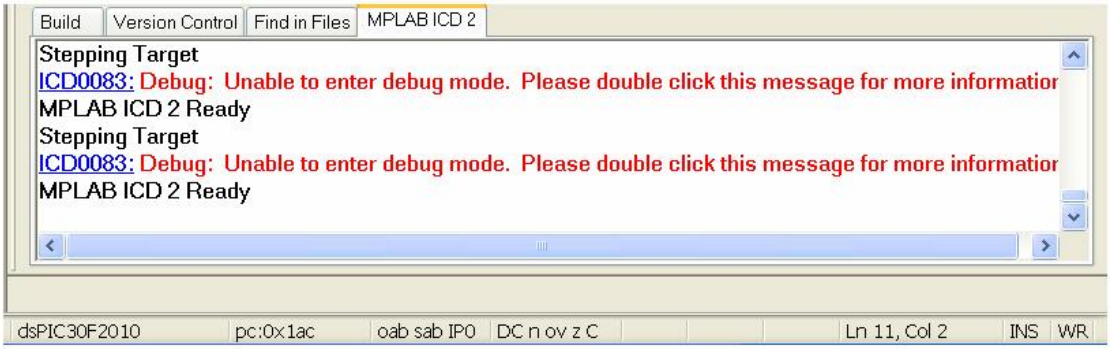
ภาพผนวกที่ ข22 การProgram

ข1.8 ในขณะนี้บอร์ด ICD2 ได้เข้าสู่โหมด Debug แล้ว เราสามารถเลือกคำสั่งต่างๆ จากเมนูต่างที่มีเช่น คำสั่ง RUN ให้ไปที่ Debugger->Run หรือคำสั่ง Animate ซึ่งเป็น คำสั่งที่มีการทำงานที่ละ Step ดังภาพผนวกที่ ข23 จะแสดงตัว Pointer ชี้ไปยังบรรทัดของตัว Code ในโปรแกรมที่กำลังทำงาน



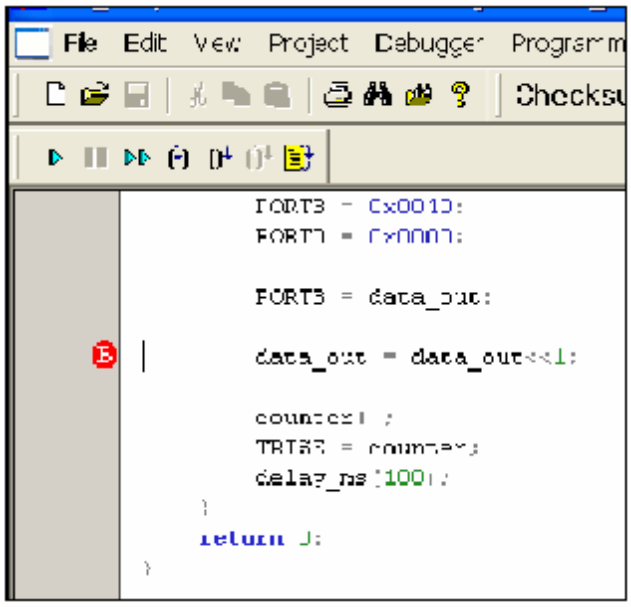
ภาพผนวกที่ ข23 ตัว Pointer ชี้ไปยังบรรทัดของตัว Code ในโปรแกรมที่กำลังทำงาน

Note: หากต้องการสั่งให้หยุดการ Debug ให้ไปที่ Debugger->Halt และถ้าเกิดปัญหาขึ้นระหว่างการ Debug จะมีไฟกระพริบที่ LED ERROR เป็นจังหวะ ๆ และอาจเกิด massage ดังภาพผนวกที่ ข24 ให้เราทำ การออกจากโหมด Debug โดยการกดสวิตช์ Reset ที่ Target Board หรือให้ไปที่ Debugger->Halt เพื่อหยุดการทำงาน แล้วกระจึงค่อย ทำการ Connect และ Program ใหม่อีกครั้ง



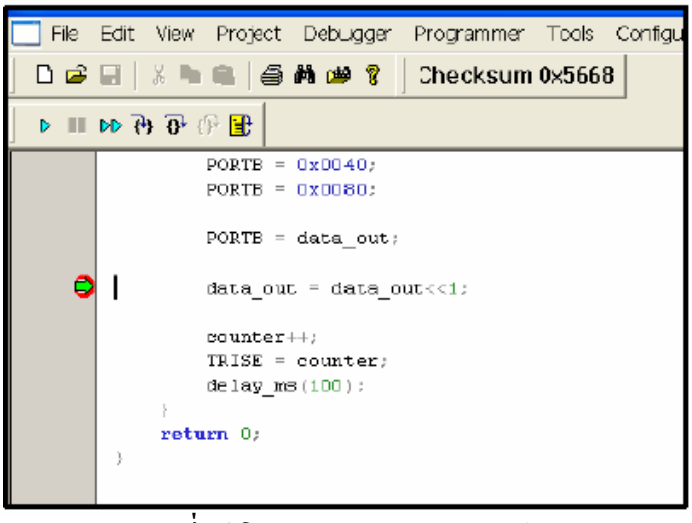
ภาพผนวกที่ ข24 การออกจากโหมด Debug

ข1.9 ในบางครั้งหากเราต้องการหยุดการทำงานของโปรแกรม ที่เราเขียนขึ้นอย่างทันทีทันใดนั้น เพื่อตรวจสอบค่าใน Register หรือตรวจสอบการทำงานของ Hardware สามารถ ทำได้ด้วยการกำหนด Break Point ในตำแหน่งที่เราต้องการ ซึ่งสามารถทำได้โดยการดับเบิ้ลคลิกยังบรรทัดที่เราต้องการ Break ดังภาพผนวกที่ ข25



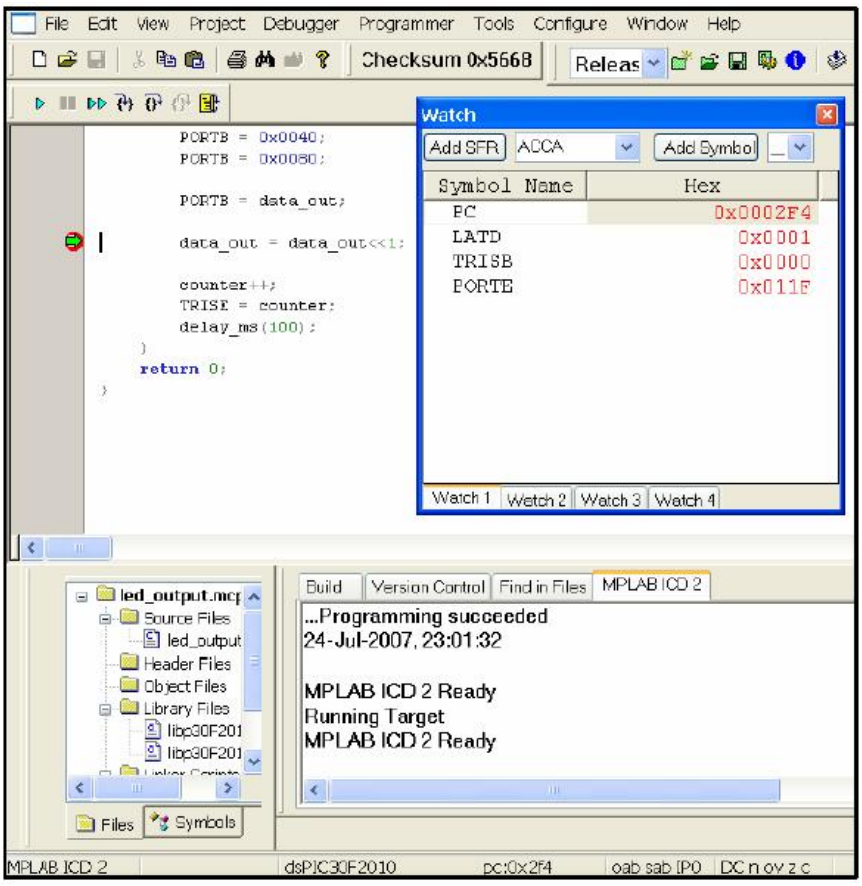
ภาพผนวกที่ ข25 การหยุดการทำงานของโปรแกรม

จากนั้นสั่ง Run โดยไปที่ Debugger->Run โปรแกรมจะทำงานจนกระทั่งมาหยุดที่ตำแหน่งที่เรากำหนด Break Point ดังภาพผนวกที่ ข26



ภาพผนวกที่ ข26 การหยุดการทำงานของโปรแกรม

Note: การดูค่า Register สามารถไปที่เมนู View->Watch จากนั้นเลือก Register ที่เราต้องการดูค่าได้เลย ดังภาพผนวกที่ ข27



ภาพผนวกที่ ข27 การดูค่า Register

ประวัติการศึกษา และการทำงาน

ชื่อ –นามสกุล	นายสกล กงแก้ว
วัน เดือน ปี ที่เกิด	วันที่ 6 มิถุนายน 2526
สถานที่เกิด	นครนายก
ประวัติการศึกษา	ปริญญาตรี สถาบันเทคโนโยยีราชมงคลคลองหก
ตำแหน่งหน้าที่การงานปัจจุบัน	นักศึกษา
สถานที่ทำงานปัจจุบัน	-
ผลงานดีเด่นและรางวัลทางวิชาการ	-การประชุมวิชาการครั้งที่ 2 มหาวิทยาลัยขอนแก่น เรื่อง การออกแบบระบบควบคุมแขนหุ่นยนต์ผ่านอินเตอร์เน็ต น. 216. การประชุมวิชาการเทคโนโลยีและนวัตกรรมสำหรับการพัฒนาอย่างยั่งยืน คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น ครั้งที่ 2. -การประชุมวิชาการครั้งที่ 46 มหาวิทยาลัยเกษตรศาสตร์ เรื่อง การออกแบบและสร้างต้นแบบตัวควบคุมแขนหุ่นยนต์ น. 241 - 248. การประชุมทางวิชาการ ครั้งที่ 46. มหาวิทยาลัยเกษตรศาสตร์
ทุนการศึกษาที่ได้รับ	- ปี พ.ศ. 2549 ได้รับทุนผู้ช่วยสอน วิชา ไมโครโปรเซสเซอร์ ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์