วิทยานิพนธ์

การออกแบบชุดควบคุม CNC 3 แกน

Design of Three-Axis CNC Controller

นายทรงชัย แซ่ตั้ง

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

พ.ศ. 2549

ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า)

ปริญญา

วิศวกรรมไฟฟ้า	วิศวกรรมไฟฟ้า
สาขา	**ภาควิชา**

เรื่อง การออกแบบชุดควบคุม CNC 3 แกน

Design of Three-Axis CNC Controller

นามผู้วิจัย นายทรงชัย แซ่ตั้ง

ได้พิจารณาเห็นชอบโดย

ประธานกรรมการ .....
(ผู้ช่วยศาสตราจารย์พีระยศ แสนโภชน์, D.Sc.)

กรรมการ .....
(อาจารย์จันทนี รุ่งเรืองพิทยากุล, D.Sc.)

กรรมการ .....
(อาจารย์ลัคน์ จันทนภาพ, วศ.ม.)

หัวหน้าภาควิชา .....
(อาจารย์ชูเกียรติ การะเกตุ, Ph.D.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

.....
(รองศาสตราจารย์วินัย อาจคงหาญ, M.A.)

คณบดีบัณฑิตวิทยาลัย

วันที่ 11 **เดือน** เมษายน **พ.ศ.** 2549

วิทยานิพนธ์

เรื่อง

การออกแบบชุดควบคุม CNC 3 แกน

Design of Three-Axis CNC Controller

โดย

นายทรงชัย แซ่ตั้ง

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์
เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า)
พ.ศ. 2549

ISBN 974-16-1518-3

ทรงชัย แซ่ตั้ง 2549: การออกแบบชุดควบคุม CNC 3 แกน ปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมไฟฟ้า) สาขาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า ประธานกรรมการที่ปรึกษา: ผู้ช่วยศาสตราจารย์พีระยศ แสนโภชน์, D.Sc. 129 หน้า
ISBN 974-16-1518-3

ในเครื่องจักรอุตสาหกรรมสมัยใหม่ ชุดควบคุมแบบดิจิทัลจัดได้ว่าเป็นส่วนที่สำคัญที่สุดและเป็นอุปกรณ์ที่ได้รับผลกระทบอย่างรุนแรงจากการก้าวหน้าอย่างรวดเร็วของนวัตกรรมทางคอมพิวเตอร์และอิเลคทรอนิคส์ ผลทำให้การลงทุนซื้อเครื่องจักรซีเอ็นซีและชุดควบคุมราคาแพงเป็นการตัดสินใจที่ไม่คุ้มค่า

วัตถุประสงค์หลักของงานวิจัยนี้คือ การพัฒนาต้นแบบชุดควบคุมการเคลื่อนที่ 3 แกนแบบต่อเนื่องโดยใช้งานคอนโทรลเลอร์ DSP ซึ่งมีความเร็วและสมรรถนะสูงร่วมกับเครื่องคอมพิวเตอร์ส่วนบุคคลที่มีสมรรถนะสูง ซึ่งมีการออกแบบชุดควบคุมที่มีประสิทธิภาพและมีความยืดหยุ่นสูงโดยใช้อัลกอริธึมการควบคุมแบบพีไอดี

ทรงชัย แซ่ตั้ง	พีระยศ แสนโภชน์	7 / 6 / 2549
ลายมือชื่อนิสิต	ลายมือชื่อประธานกรรมการ	

Songchai Sae-tung 2006: Design of Three-Axis CNC Controller.
Master of Engineering (Electrical Engineering), Major Field: Electrical Engineering,
Department of Electrical Engineering. Thesis Advisor: Assistant Professor Peerayot
Sanposh, D.Sc. 129 pages.
ISBN 974-16-1518-3

In modern industrial machines, digital controllers are virtually the most essential parts and every components are significantly affected by rapid improvement of computer and electronics technology. This renders investments in expensive CNC machines and controllers.

The main objective of this research is to develop a prototype of 3-axis CNC controller using a PC and a high-speed, high-performance of DSP Controller. The design prototype is high-efficient and flexible design PID control algorithm.

Songchai SAE-TUNG
Student's signature

Peerayot Sanposh 7 / 6 / 2006
Thesis Advisor's signature

กิตติกรรมประกาศ

ข้าพเจ้าขอกราบขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.พีระยศ แสนโภชน์ ประธานกรรมการที่ปรึกษา ที่ได้ให้คำปรึกษาแนะนำและตรวจแก้ไขข้อบกพร่องต่าง ๆ ที่เป็นประโยชน์ที่สามารถนำไปประยุกต์ใช้ในงานวิจัยและนำไปแก้ปัญหาที่ประสบขณะทำการวิจัย พร้อมทั้งให้ความสนับสนุนช่วยเหลือในด้านอุปกรณ์และเครื่องมือสำหรับการทำวิจัยในครั้งนี้ให้สำเร็จลุล่วงไปด้วยดี

ขอกราบขอบพระคุณ อาจารย์ลัคน์ จันทนภาพ และอาจารย์ ดร.วโรดม ตู้จินดา ที่ให้ความรู้และคำปรึกษาเกี่ยวกับ DSP และ FPGA และสนับสนุนช่วยเหลือในด้านอุปกรณ์และเครื่องมือสำหรับการทำวิจัยในครั้งนี้เป็นอย่างดี

ขอขอบพระคุณ ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติที่พิจารณาให้เงินทุนอุดหนุนและอุปกรณ์ที่ใช้ในการทำวิจัยในวิทยานิพนธ์ฉบับนี้

ขอขอบคุณ พี่สมหมายและน้องธรณินทร์ สำหรับการออกแบบ PCB เพื่อใช้ในการทำวิจัยครั้งนี้

ขอขอบคุณ พี่ ๆ น้อง ๆ และเพื่อน ๆ ที่คอยให้ความช่วยเหลือ ให้กำลังใจ ให้คำปรึกษาและสนับสนุนการทำวิทยานิพนธ์จนสำเร็จลุล่วงไปด้วยดี

ท้ายที่สุด ขอกราบขอบพระคุณ บิดา มารดา อย่างสูง ที่คอยให้การสนับสนุน และเป็นกำลังใจขณะที่กำลังศึกษาในระดับปริญญาโทจนประสบผลสำเร็จในการเล่าเรียน

ทรงชัย แซ่ตั้ง

เมษายน 2549

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

การออกแบบชุดควบคุม CNC 3 แกน

Design of Three-Axis CNC Controller

คำนำ

ปัจจุบันในสาขาต่าง ๆ ที่เกี่ยวข้องกับระบบควบคุมการเคลื่อนที่แบบเซอร์โว เช่น เครื่องกลึง/กัด/เจียรอัตโนมัติ หรือ หุ่นยนต์อุตสาหกรรม เราจะพบว่าลักษณะงานที่ผู้ใช้ต้องการจะมีความซับซ้อนมากขึ้น สาเหตุหนึ่งมาจากการ พัฒนาอย่างรวดเร็วของเทคโนโลยีคอมพิวเตอร์ที่มีประสิทธิภาพและราคาถูก โดยเฉพาะเครื่องคอมพิวเตอร์ส่วนบุคคลที่มีตัวประมวลผลกลาง และบัสความเร็วสูงสามารถที่จะสนับสนุนการทำงานในลักษณะที่ชุดควบคุมในสมัยก่อนไม่สามารถกระทำได้ ภาคอุตสาหกรรมในประเทศไทยจึงมีแนวโน้มที่จะใช้คอมพิวเตอร์ส่วนบุคคลในงานควบคุมการเคลื่อนที่ของแกนต่าง ๆ ของเครื่องจักรกล หรือหุ่นยนต์กันมากขึ้น เช่น เดียวกับสถานศึกษาที่เกี่ยวข้องกับเทคโนโลยีการผลิตหรือวิศวกรรมระบบควบคุมก็มีความต้องการที่จะสรรหาชุดควบคุมการเคลื่อนที่ที่สามารถใช้ประกอบการเรียนการสอน หรือใช้ในห้องปฏิบัติการ การออกแบบระบบควบคุมแบบป้อนกลับสำหรับเซอร์โวมอเตอร์ ชุดควบคุมที่ใช้งานร่วมกับคอมพิวเตอร์ส่วนบุคคลจึงเริ่มปรากฏ และถูกพัฒนาให้มีคุณภาพดีขึ้น อย่างไรก็ตาม ปัญหาหลักที่ยังประสบอยู่ในขณะนี้ เช่น ประเทศไทยยังขาดผู้ผลิตชุดควบคุมทั้งฮาร์ดแวร์และซอฟท์แวร์ที่ได้มาตรฐานสำหรับงานอุตสาหกรรมที่ต้องการความเที่ยงตรงสูง หรืองานออกแบบระบบควบคุมที่ซับซ้อน เราจึงได้ทำการพัฒนาต้นแบบชุดควบคุมการเคลื่อนที่ 3 แกนแบบต่อเนื่องโดยใช้งานคอนโทรลเลอร์ DSP ซึ่งมีความเร็ว และสมรรถนะสูงร่วมกับเครื่องคอมพิวเตอร์ส่วนบุคคลที่มีสมรรถนะสูง มีการออกแบบที่มีประสิทธิภาพ มีความยืดหยุ่น โดยใช้อัลกอริธึมการควบคุมแบบพีไอดี

วัตถุประสงค์

วัตถุประสงค์ของการทำวิจัยในครั้งนี้เพื่อสร้างและทดลองชุดควบคุมดิจิทัลสำหรับเครื่อง CNC 3 แกนสำหรับใช้ในการศึกษาและทดสอบทฤษฎี และเพื่อใช้ชุดทดลองที่สร้างขึ้นมาในการพัฒนาและเป็นต้นแบบในการสร้างเป็นผลิตภัณฑ์ที่ใช้ในโรงงานอุตสาหกรรม

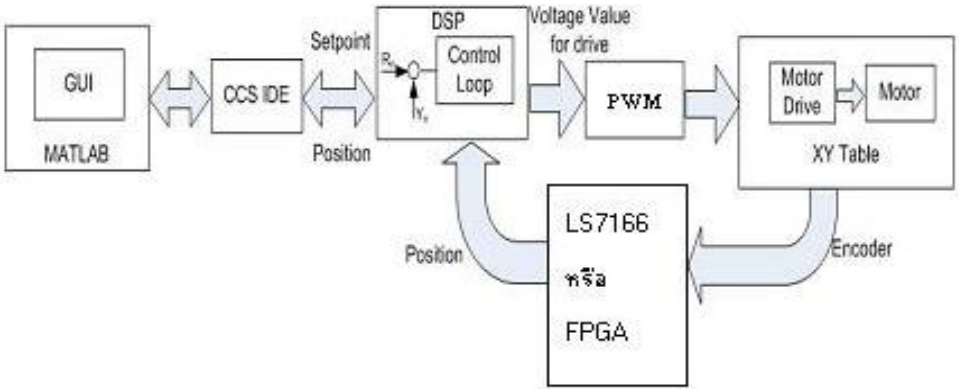
การตรวจเอกสาร

การออกแบบการควบคุมมอเตอร์

จากการค้นคว้าข้อมูลการควบคุมมอเตอร์แบบ 2 แกน และ 3 แกนด้วยวิธีต่างๆ พบว่า Craig Simal (1955) เขียนหนังสือเกี่ยวกับทฤษฎีควบคุมแกนหุ่นยนต์ ต่อมา Newman (1960) ได้เสนอการวิเคราะห์ผลตอบสนองและเสถียรภาพของระบบควบคุมป้อนกลับสำหรับมอเตอร์ 2 แกน ที่สัมพันธ์กัน ส่วน Lewis and Maciejewski (1990) ได้พิสูจน์ Trajectory Generation สำหรับแขนหุ่นยนต์ 2 link ต่อมา Decotignie (1991) ได้เสนอวิธีการสร้างเส้นทางและควบคุมความเร็วสำหรับการเคลื่อนที่ของเครื่องจักรทั่วไป Aoki, Iwazawa, Tsujisawa, Sakaguchi and Nakawatase (1994) ได้เสนอการควบคุมมอเตอร์ด้วยวิธี multi-axis coordinated control (MACC) เทียบกับวิธีประมาณค่าทั่วไปซึ่งจะเห็นว่าผลที่ได้จะมี มีค่าความผิดพลาดน้อยกว่าและความแม่นยำ,เที่ยงตรงมากกว่า ต่อมา Nobuyuki Matsui (1995) ได้เสนอการใช้ Digital Signal Processor (DSP) ในการควบคุมการเคลื่อนที่ของมอเตอร์กระแสสลับ ต่อมา Yamazaki, Jiancheng and Yokoyama (1998) ได้เสนอวิธีใหม่ซึ่งควบคุมมอเตอร์โดยใช้ Dynamic Gain Control วิธีนี้จะคำนวณความผิดพลาดของการเคลื่อนที่แต่ละแกนแล้วจึงมาปรับค่า Dynamic Gain Control เพื่อให้เหมาะสมกับการควบคุมมอเตอร์ ตามรายงานบอกว่าให้ผลที่ดีกว่าวิธีที่มีอยู่แล้ว ต่อมา Gon and Lee (1999) ได้เสนอวิธีการใช้ Neural network ในการเทรนหา function ปรับค่า PID Gain ในการควบคุมมอเตอร์ 2 แกน ต่อมา Syh-Shiuh Yeh and Pau-Lo Hsu. (2000, 2003) ได้เสนอการควบคุมมอเตอร์ 2 แกนด้วยการปรับปรุงเส้นทางการเดินของรากให้ดีขึ้น และได้ออกแบบ contouring error transfer function (CETF) ในชุดควบคุมด้วย ต่อมา Rouchon (2001) ให้เสนอการควบคุมสำหรับการเคลื่อนที่ N มิติ ต่อมา Frew and Rock ได้ใช้กล้องดิจิตอลในการติดตามการเคลื่อนที่ของมอเตอร์ 2 แกน และต่อมา Acosta, Alfaro, Gan and Hu (2004) ได้เสนอการควบคุมและติดตามการเคลื่อนที่ของแขนหุ่นยนต์ 5 DOF (Degree Of Freedom)

นอกจากรายงานการค้นคว้าที่กล่าวมาแล้วข้างต้นยังได้มีผู้เสนอแนวคิดในการนำวิธีการควบคุมแบบอื่น ๆ มาใช้ในการควบคุมมอเตอร์อีกเช่นกัน

ในการควบคุมการทำงานของเครื่อง CNC 3 แกนจะประกอบด้วยแกนการทำงาน 3 แกนที่พิจารณาโดยแยกอิสระออกจากกัน และในแต่ละแกนจะถูกขับเคลื่อนด้วยเซอร์โวมอเตอร์โดยการออกแบบชุดควบคุมจะแบ่งออกเป็น 2 ส่วนคือ ส่วนของชุดควบคุมโดยใช้ DSP board และส่วนที่ติดต่อกับผู้ใช้งานโดยจะมีระบบการทำงานแสดงดังภาพที่ 1



ภาพที่ 1 การเชื่อมต่อระหว่าง DSP board และส่วนติดต่อผู้ใช้งาน

การควบคุมมอเตอร์จะแบ่งออกเป็น 2 ประเภท คือ การควบคุมด้วย DC servo motor และการควบคุม AC servo motor โดยการควบคุม DC servo motor ต้องอาศัยการสร้างสัญญาณ PWM ซึ่งอาจสร้างจาก Software หรือ CPLD (Complex Programmable Logic Device) แล้วจึงส่งสัญญาณ PWM ให้แก่ DC servo motor drive ต่อไป ส่วนการควบคุม AC servo motor นั้นจะใช้ D/A (Digital to Analog) แทนการสร้างสัญญาณ PWM ซึ่งชิพ D/A จะส่งสัญญาณให้แก่ AC servo motor drive ต่อไป การอ่านค่าตำแหน่งของ DC servo motor drive และ AC servo motor drive จะทำได้ 2 วิธี คือ การใช้ชิพสำเร็จรูป LS7166 หรือชิพ FPGA (Field Programmable gate Array) ซึ่งจะส่งให้บอร์ด DSP ทำการหาค่า voltage สำหรับการขับมอเตอร์ต่อไป

ในส่วนติดต่อผู้ใช้เราจะติดต่อผ่านชุดคำสั่งใน library ของ RTDX ซึ่งเป็นชุดคำสั่งที่ใช้ติดต่อผ่านทาง USB port ของคอมพิวเตอร์ หรืออาจจะใช้ McBSP ของ DSP ติดต่อผ่านทาง serial port ของคอมพิวเตอร์ ซึ่งการติดต่อทั้ง 2 แบบเราได้เขียน GUI (graphical user interface) ไว้ติดต่อกับผู้ใช้ซึ่งสร้างโดยโปรแกรม MATLAB

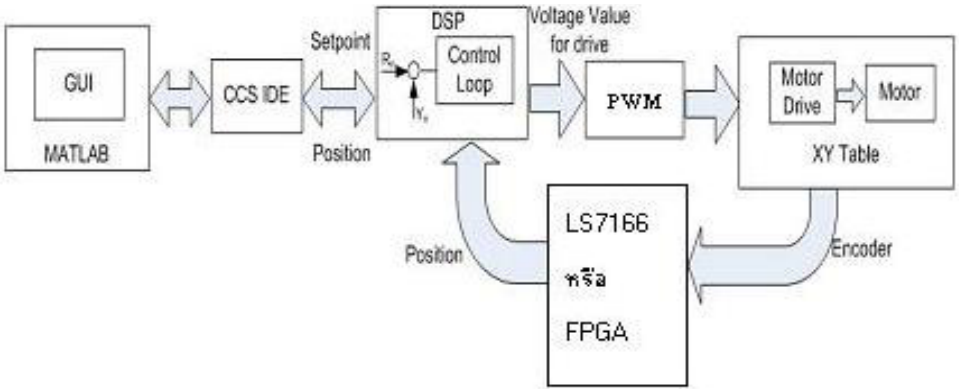
การตรวจเอกสาร

การออกแบบการควบคุมมอเตอร์

จากการค้นคว้าข้อมูลการควบคุมมอเตอร์แบบ 2 แกน และ 3 แกนด้วยวิธีต่างๆ พบว่า Craig Simal (1955) เขียนหนังสือเกี่ยวกับทฤษฎีควบคุมแกนหุ่นยนต์ ต่อมา Newman (1960) ได้เสนอการวิเคราะห์ผลตอบสนองและเสถียรภาพของระบบควบคุมป้อนกลับสำหรับมอเตอร์ 2 แกน ที่สัมพันธ์กัน ส่วน Lewis and Maciejewski (1990) ได้พิสูจน์ Trajectory Generation สำหรับแขนหุ่นยนต์ 2 link ต่อมา Decotignie (1991) ได้เสนอวิธีการสร้างเส้นทางและควบคุมความเร็วสำหรับการเคลื่อนที่ของเครื่องจักรทั่วไป Aoki, Iwazawa, Tsujisawa, Sakaguchi and Nakawatase (1994) ได้เสนอการควบคุมมอเตอร์ด้วยวิธี multi-axis coordinated control (MACC) เทียบกับวิธีประมาณค่าทั่วไปซึ่งจะเห็นว่าผลที่ได้จะมี มีค่าความผิดพลาดน้อยกว่าและความแม่นยำ,เที่ยงตรงมากกว่า ต่อมา Nobuyuki Matsui (1995) ได้เสนอการใช้ Digital Signal Processor (DSP) ในการควบคุมการเคลื่อนที่ของมอเตอร์กระแสสลับ ต่อมา Yamazaki, Jiancheng and Yokoyama (1998) ได้เสนอวิธีใหม่ซึ่งควบคุมมอเตอร์โดยใช้ Dynamic Gain Control วิธีนี้จะคำนวณความผิดพลาดของการเคลื่อนที่แต่ละแกนแล้วจึงมาปรับค่า Dynamic Gain Control เพื่อให้เหมาะสมกับการควบคุมมอเตอร์ ตามรายงานบอกว่าให้ผลที่ดีกว่าวิธีที่มีอยู่แล้ว ต่อมา Gon and Lee (1999) ได้เสนอวิธีการใช้ Neural network ในการเทรนหา function ปรับค่า PID Gain ในการควบคุมมอเตอร์ 2 แกน ต่อมา Syh-Shiuh Yeh and Pau-Lo Hsu. (2000, 2003) ได้เสนอการควบคุมมอเตอร์ 2 แกนด้วยการปรับปรุงเส้นทางการเดินของรากให้ดีขึ้น และได้ออกแบบ contouring error transfer function (CETF) ในชุดควบคุมด้วย ต่อมา Rouchon (2001) ให้เสนอการควบคุมสำหรับการเคลื่อนที่ N มิติ ต่อมา Frew and Rock ได้ใช้กล้องดิจิตอลในการติดตามการเคลื่อนที่ของมอเตอร์ 2 แกน และต่อมา Acosta, Alfaro, Gan and Hu (2004) ได้เสนอการควบคุมและติดตามการเคลื่อนที่ของแขนหุ่นยนต์ 5 DOF (Degree Of Freedom)

นอกจากรายงานการค้นคว้าที่กล่าวมาแล้วข้างต้นยังได้มีผู้เสนอแนวคิดในการนำวิธีการควบคุมแบบอื่น ๆ มาใช้ในการควบคุมมอเตอร์อีกเช่นกัน

ในการควบคุมการทำงานของเครื่อง CNC 3 แกนจะประกอบด้วยแกนการทำงาน 3 แกนที่พิจารณาโดยแยกอิสระออกจากกัน และในแต่ละแกนจะถูกขับเคลื่อนด้วยเซอร์โวมอเตอร์โดยการออกแบบชุดควบคุมจะแบ่งออกเป็น 2 ส่วนคือ ส่วนของชุดควบคุมโดยใช้ DSP board และส่วนที่ติดต่อกับผู้ใช้งานโดยจะมีระบบการทำงานแสดงดังภาพที่ 1



ภาพที่ 1 การเชื่อมต่อระหว่าง DSP board และส่วนติดต่อผู้ใช้งาน

การควบคุมมอเตอร์จะแบ่งออกเป็น 2 ประเภท คือ การควบคุมด้วย DC servo motor และการควบคุม AC servo motor โดยการควบคุม DC servo motor ต้องอาศัยการสร้างสัญญาณ PWM ซึ่งอาจสร้างจาก Software หรือ CPLD (Complex Programmable Logic Device) แล้วจึงส่งสัญญาณ PWM ให้แก่ DC servo motor drive ต่อไป ส่วนการควบคุม AC servo motor นั้นจะใช้ D/A (Digital to Analog) แทนการสร้างสัญญาณ PWM ซึ่งชิพ D/A จะส่งสัญญาณให้แก่ AC servo motor drive ต่อไป การอ่านค่าตำแหน่งของ DC servo motor drive และ AC servo motor drive จะทำได้ 2 วิธี คือ การใช้ชิพสำเร็จรูป LS7166 หรือชิพ FPGA (Field Programmable gate Array) ซึ่งจะส่งให้บอร์ด DSP ทำการหาค่า voltage สำหรับการขับมอเตอร์ต่อไป

ในส่วนติดต่อผู้ใช้เราจะติดต่อผ่านชุดคำสั่งใน library ของ RTDX ซึ่งเป็นชุดคำสั่งที่ใช้ติดต่อผ่านทาง USB port ของคอมพิวเตอร์ หรืออาจจะใช้ McBSP ของ DSP ติดต่อผ่านทาง serial port ของคอมพิวเตอร์ ซึ่งการติดต่อทั้ง 2 แบบเราได้เขียน GUI (graphical user interface) ไว้ติดต่อกับผู้ใช้ซึ่งสร้างโดยโปรแกรม MATLAB

Feedback Control Systems

1. ระบบควบคุม

ระบบควบคุม คือ การจัดการปริมาณที่สนใจเพื่อให้ได้ค่าที่ต้องการ หรือค่าที่ตั้งไว้โดย ทฤษฎีของระบบควบคุมสามารถแบ่งออกได้เป็น 2 ประเภทใหญ่ๆคือ

1.1 ระบบควบคุมแบบดั้งเดิม (Classical Control Systems)

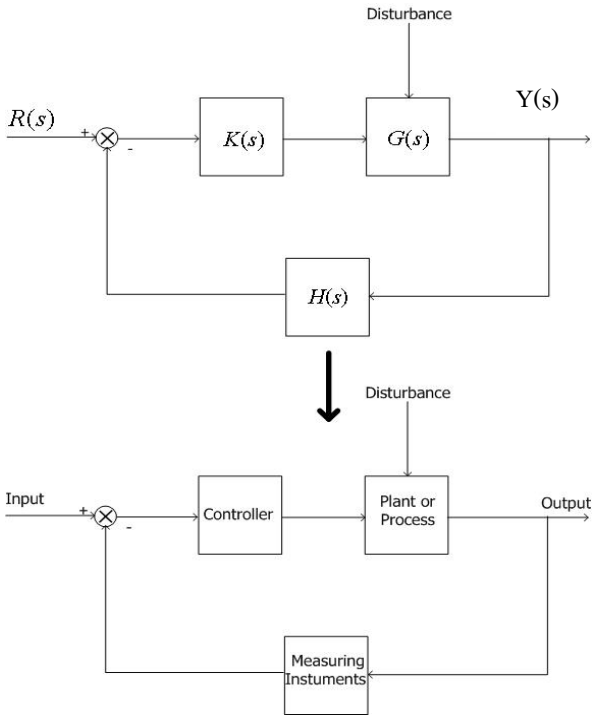
ระบบควบคุมแบบดั้งเดิมนั้นจะใช้กับระบบที่มี 1 สัญญาณด้านเข้า และ 1 สัญญาณด้านออกแต่อาจจะประยุกต์ใช้กับระบบควบคุมที่มีหลายสัญญาณด้านเข้า หลายสัญญาณด้านออกได้เช่นกันโดยจะใช้คณิตศาสตร์เรื่องสมการอนุพันธุ์ (Differential Equation) และการแปลงลาปลาซ (Laplace Transform) นำมาวิเคราะห์และออกแบบระบบโดยใช้ฟังก์ชันถ่ายโอน (Transfer function) เพื่อศึกษาเกี่ยวกับเสถียรภาพ การทำงานที่เป็นไปตามต้องการ และความเที่ยงตรงของระบบที่ถูกควบคุม

1.2 ระบบควบคุมสมัยใหม่ (Modern Control Systems)

ระบบควบคุมแบบสมัยใหม่นั้นใช้กับระบบที่มี 1 สัญญาณด้านเข้า 1 สัญญาณด้านออกได้แต่จะมีประสิทธิภาพในการใช้งานกับระบบที่มีหลายสัญญาณด้านเข้า หลายสัญญาณด้านออกมากกว่าโดยจะใช้คณิตศาสตร์เรื่องสมการอนุพันธุ์, เวคเตอร์ (Vector) และเมทริก (Matrix) มาวิเคราะห์ และออกแบบระบบโดยใช้ตัวแปรสถานะ (State Variables) เพื่อศึกษา และพิจารณาเกี่ยวกับความสามารถในการตรวจสอบ หรือสังเกตสถานะของระบบที่ถูกควบคุม และความสามารถในการควบคุมได้

2. โครงสร้างพื้นฐานของระบบควบคุม

ในการศึกษา วิเคราะห์ และออกแบบระบบควบคุม โดยใช้ทฤษฎีระบบควบคุมแบบดั้งเดิมนั้น จะจัดรูปแบบของระบบควบคุมได้ดังแสดงในภาพที่ 2



ภาพที่ 2 โครงสร้างของระบบควบคุม

โดยมีคำจำกัดความดังนี้

สัญญาณด้านขาเข้า (Input) ซึ่งนั้นบางครั้งเราอาจเรียกอีกชื่อหนึ่งว่า อินพุทอ้างอิง (Reference Input) หรือค่าที่ตั้ง (set point)

ตัวควบคุม (Controller) หมายถึง เครื่องมือหรืออุปกรณ์ที่ใช้ในการสร้างสัญญาณควบคุม เพื่อทำหน้าที่ควบคุมให้ระบบ หรือกระบวนการที่ต้องการควบคุมมีสัญญาณด้านออก หรือผลตอบสนองตามที่ต้องการ โดยตัวควบคุมจะมีหลายแบบ เช่น ตัวควบคุมแบบ ON-OFF ตัวควบคุมชนิดพี (Proportional, P) ตัวควบคุมชนิดไอ (Integral, I) ตัวควบคุมชนิดดี (Derivative, D) เป็นต้น

กระบวนการ (Plant or Process) หมายถึง ระบบหรือกระบวนการที่ถูกควบคุม หรืออาจจะเป็นวัตถุทางกายภาพที่ถูกควบคุมก็ได้ เช่น กระบวนการในการควบคุมอุณหภูมิเตาเผา กระบวนการควบคุมระบบแขนกลในโรงงานเป็นต้น

สัญญาณด้านออก (Output) หมายถึง ผลตอบสนองของระบบ หรือกระบวนการที่ถูกควบคุม ซึ่งโดยทั่วไปแล้วต้องการจะควบคุมให้สัญญาณด้านออกมีค่าตามสัญญาณด้านเข้าที่กำหนด (หรือตามค่าของสัญญาณด้านเข้าที่เปลี่ยนแปลงไป) หรือมีค่าคงเดิมได้เมื่อมีการรบกวนทั้งภายใน และภายนอกที่มากระทำต่อระบบควบคุม

การรบกวน (Disturbance) หมายถึง สัญญาณรบกวนที่อาจจะเกิดขึ้นในระบบที่ถูกควบคุม สัญญาณรบกวนนี้อาจเกิดขึ้นที่จุดใดๆในระบบก็ได้ เช่นเกิดขึ้นที่กระบวนการ เกิดขึ้นที่อุปกรณ์วัด เป็นต้น การเกิดขึ้นของสัญญาณรบกวนอาจเกิดขึ้นในเวลาใดๆทั้งที่คาดเดาได้ และคาดเดาไม่ได้ การรบกวนนี้แบ่งได้เป็น 2 ลักษณะคือ

- การรบกวนภายใน (Internal Disturbance) ซึ่งอาจเกิดจากการเปลี่ยนแปลงของพารามิเตอร์ต่างๆ ของอุปกรณ์ที่ใช้ในระบบ

- การรบกวนจากภายนอก (External Disturbance) เป็นการรบกวนที่เกิดขึ้นจากภายนอกระบบ แต่มีผลต่อระบบที่กำลังควบคุมอยู่ โดยทั่วไปจะถือว่าการรบกวนจากภายนอกเป็นสัญญาณด้านเข้าหนึ่งที่ไม่พึงประสงค์ ของระบบควบคุม

อุปกรณ์วัด (Measuring Instruments) หมายถึงอุปกรณ์ที่อาจจะได้แก่ เซ็นเซอร์ (Sensor) ทรานสดิวเซอร์ (Transducer) หรืออุปกรณ์แปลง หรือวัดสัญญาณอื่นๆที่ทำหน้าที่วัดค่าของสัญญาณด้านออกของระบบทีถูกควบคุม

ระบบ (System) หมายถึง การนำเอาอุปกรณ์ต่างๆที่สามารถทำงานร่วมกันได้ มารวบรวมเข้าด้วยกันเพื่อให้ทำงานอย่างใดอย่างหนึ่งที่ต้องการ เช่น ระบบเชิงกล ระบบทางกายภาพของวงจรไฟฟ้า เป็นต้น

3. <u>ระบบควบคุมป้อนกลับ (Feedback Control System)</u>

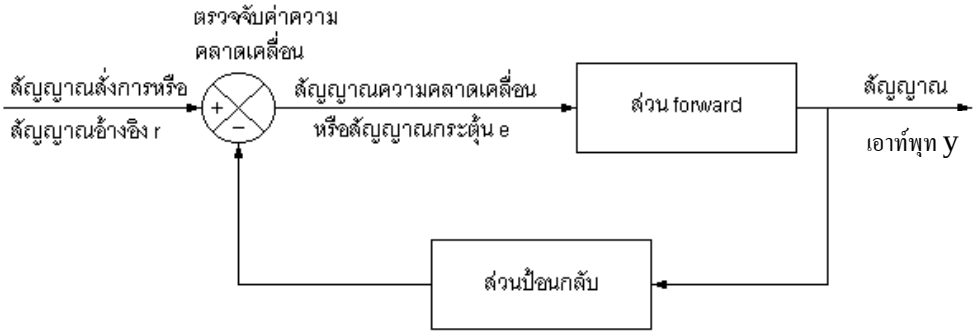
ระบบควบคุมป้อนกลับ หมายถึง ระบบควบคุมที่ต้องการควบคุมให้สัญญาณด้านออกหรือผลตอบสนองของระบบมีค่าตามที่ต้องการ โดยการนำเอาสัญญาณด้านออกป้อนกลับมาเปรียบเทียบกับสัญญาณด้านเข้า ซึ่งการป้อนกลับนี้จะต้องเป็นการป้อนกลับแบบลบ (Negative Feedback) จากนั้นนำค่าความแตกต่างระหว่างด้านเข้ากับด้านออก ส่งต่อไปยังส่วนสร้างสัญญาณของตัวควบคุม เพื่อสร้างสัญญาณควบคุมที่เหมาะสมที่จะทำให้สัญญาณด้านออกของระบบที่ถูกควบคุมมีค่าตามต้องการ ระบบควบคุมป้อนกลับนี้อาจเรียกอีกอย่างหนึ่งว่า ระบบควบคุมแบบลูปปิด หรือระบบควบคุมอัตโนมัติก็ได้

ชนิดของระบบควบคุม การจำแนกชนิดของระบบควบคุมทำได้หลายวิธี ขึ้นอยู่กับลักษณะของระบบ ลักษณะของสัญญาณ ลักษณะการวิเคราะห์และศึกษาระบบเป็นต้น โดยระบบควบคุมที่จะศึกษา จะตั้งอยู่บนพื้นฐานดังต่อไปนี้

- เป็นระบบเชิงเส้นหรือระบบไม่เชิงเส้น
- เป็นระบบที่ไม่แปรตามเวลาหรือระบบที่แปรตามเวลา
- เป็นระบบที่มีสัญญาณต่อเนื่องหรือระบบที่มีสัญญาณดีสครีต
- เป็นระบบ Causal หรือระบบ Non Causal
- เป็นระบบแบบไม่มีความจำ (Static) หรือระบบแบบมีความจำ (Dynamic)
- เป็นระบบควบคุมแบบลูปปิด (ระบบควบคุมป้อนกลับ) หรือระบบควบคุมแบบเปิด
- จัดให้อยู่ในรูปของฟังก์ชันถ่ายโอน และทำการศึกษาวิเคราะห์ และออกแบบโดยการใช้คณิตศาสตร์เรื่องสมการอนุพันธ์ และการแปลงลาปลาซ

4. <u>ส่วนประกอบพื้นฐานของระบบควบคุมป้อนกลับ</u>

จากหลักการพื้นฐานของระบบควบคุมป้อนกลับสามารถกล่าวได้ว่า ระบบควบคุมป้อนกลับประกอบด้วยเส้นทางหรือวงรอบของสัญญาณป้อนกลับซึ่งเป็นสัญญาณเอาท์พุท y ตั้งแต่หนึ่งวงรอบขึ้นไป แล้วนำสัญญาณป้อนกลับนี้มาเปรียบเทียบกับสัญญาณสั่งการหรือสัญญาณอ้างอิง r จะได้ผลต่างระหว่างสัญญาณทั้งสองเป็น $e = r - y$ เพื่อนำไปควบคุมสัญญาณเอาท์พุท y ให้มีค่าตามที่กำหนดโดยสัญญาณอ้างอิง r



ภาพที่ 3 แผนผังของระบบควบคุมป้อนกลับ

ดังภาพที่ 3 แสดงแผนผังของระบบควบคุมป้อนกลับ ระบบควบคุมนี้ประกอบด้วยส่วน forward (forward path), ส่วนป้อนกลับ (feedback path) และส่วนตรวจจับค่าความคลาดเคลื่อน (error-sensing device) ส่วนตรวจจับค่าความคลาดเคลื่อนนี้จะเปรียบเทียบค่าสัญญาณอินพุทอ้างอิงกับค่าสัญญาณเอาท์พุทจริงๆหรือค่าที่เป็นฟังก์ชันของสัญญาณเอาท์พุท แล้วส่งสัญญาณที่เกิดจากผลต่างของสัญญาณทั้งสองนี้ออกไป

5. ชนิดของระบบควบคุมป้อนกลับ

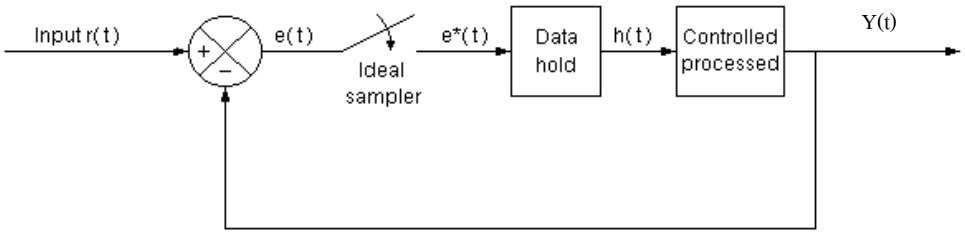
ถ้าพิจารณาในแง่วิธีการในการวิเคราะห์และออกแบบระบบแล้ว ระบบควบคุมแบบป้อนกลับสามารถแบ่งได้เป็นสองชนิด คือ ระบบควบคุมป้อนกลับแบบเชิงเส้น และระบบควบคุมป้อนกลับแบบไม่เป็นเชิงเส้น ในทางปฏิบัติแล้วระบบที่เป็นเชิงเส้นจะไม่มีอยู่จริงเนื่องจากลักษณะทางกายภาพของระบบทุกระบบจะเป็นเชิงเส้นในขอบเขตหนึ่งเท่านั้น ดังนั้นระบบแบบเชิงเส้นจึงเป็นเพียงระบบที่สมมุติขึ้นมาเพื่อให้ง่ายต่อการวิเคราะห์และออกแบบเท่านั้น ในระบบควบคุมแบบป้อนกลับจะมีขอบเขตการทำงานที่เป็นเชิงเส้นของระดับสัญญาณกระตุ้น e ซึ่งสามารถนำมาใช้ในแบบจำลองแบบเชิงเส้นได้ เมื่อระดับสัญญาณกระตุ้น e อยู่นอกขอบเขตที่เป็นเชิงเส้นนี้ระบบควบคุมแบบป้อนกลับก็จะเข้าสู่สภาวะที่ไม่เป็นเชิงเส้น แต่ถ้าพิจารณาในแง่ของลักษณะสัญญาณที่เกิดขึ้นภายในระบบ ก็จะสามารถแบ่งชนิดของระบบควบคุมแบบป้อนกลับได้เป็น ระบบที่มีสัญญาณต่อเนื่อง (continuous-data system) และระบบที่มีสัญญาณไม่ต่อเนื่อง (discrete-data system) หรือระบบที่มีการสุ่มตัวอย่างของสัญญาณ (sampled-data system)

5.1 ระบบควบคุมป้อนกลับชนิดที่มีสัญญาณต่อเนื่อง (Continuous-data Feedback Control System)

ระบบนี้เป็นระบบที่สัญญาณต่างๆภายในระบบเป็นสัญญาณที่เป็นฟังก์ชันแบบต่อเนื่องเมื่อตัวแปรของฟังก์ชันเป็นเวลา t ถ้าสัญญาณแบบต่อเนื่องในระบบนี้อยู่ในลักษณะที่ถูก modulated จะเรียกระบบนี้ว่า ระบบ a-c carrier system แต่ถ้าสัญญาณอยู่ในลักษณะที่เป็น unmodulated ก็จะเป็นระบบ d-c carrier system

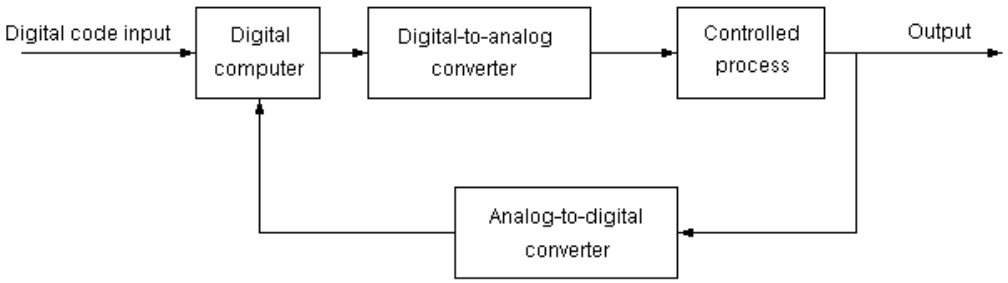
5.2 ระบบควบคุมป้อนกลับที่มีการสุ่มตัวอย่างของสัญญาณและระบบควบคุมที่มีสัญญาณไม่ต่อเนื่อง (Sampled-data and Discrete-data Control System)

ระบบควบคุมป้อนกลับที่มีการสุ่มตัวอย่างของสัญญาณ และระบบควบคุมแบบสัญญาณดิจิตอลจะมีสัญญาณที่อยู่ในลักษณะของขบวนของพัลส์หรือรหัสที่เป็นเชิงตัวเลข โดยปกติระบบที่มีการสุ่มตัวอย่างของสัญญาณจะหมายถึง ระบบที่มีสัญญาณในลักษณะของขบวนของพัลส์ ส่วนระบบควบคุมแบบสัญญาณดิจิตอลจะหมายถึงระบบที่มีคอมพิวเตอร์ หรืออุปกรณ์ตรวจจับสัญญาณแบบดิจิตอลเป็นส่วนประกอบ ในที่นี้ระบบควบคุมที่มีสัญญาณไม่ต่อเนื่องจะมีความหมายครอบคลุมถึงทั้งระบบแบบสุ่มตัวอย่างของสัญญาณ และระบบแบบดิจิตอล



ภาพที่ 4 ระบบควบคุมที่มีสัญญาณไม่ต่อเนื่อง

ระบบที่มีสัญญาณไม่ต่อเนื่องจะตรวจจับสัญญาณเข้ามาในลักษณะของพัลส์โดยแต่ละพัลส์จะใช้เวลาเพียงช่วงสั้นๆเท่านั้น ดังนั้นระบบจะไม่ได้รับข้อมูลของสัญญาณในช่วงเวลาที่อยู่ระหว่างพัลส์ที่อยู่ติดกันเลย ภาพที่ 4 แสดงถึงการทำงานของระบบควบคุมที่มีการสุ่มตัวอย่างของสัญญาณ สัญญาณอินพุท r(t) ที่เป็นสัญญาณแบบต่อเนื่องถูกส่งเข้าระบบและเปรียบเทียบกับสัญญาณป้อนกลับ y(t) ที่เป็นสัญญาณแบบต่อเนื่องเช่นกัน จะได้สัญญาณค่าความคลาดเคลื่อน e(t) ที่ต่อเนื่อง และส่งต่อไปเพื่อสุ่มตัวอย่างสัญญาณโดยอุปกรณ์สุ่มตัวอย่าง (sampler) ได้สัญญาณที่เป็นขบวนของพัลส์ โดยปกติอุปกรณ์สุ่มตัวอย่างจะสุ่มตัวอย่างสัญญาณในอัตราที่คงที่ แต่ในบางกรณีการสุ่มตัวอย่างอาจจะอยู่ในลักษณะที่เป็น periodic, cyclic, multirate, skip-rate, random และ pulse-width modulated



ภาพที่ 5 แผนผังของระบบควบคุมแบบดิจิตอล

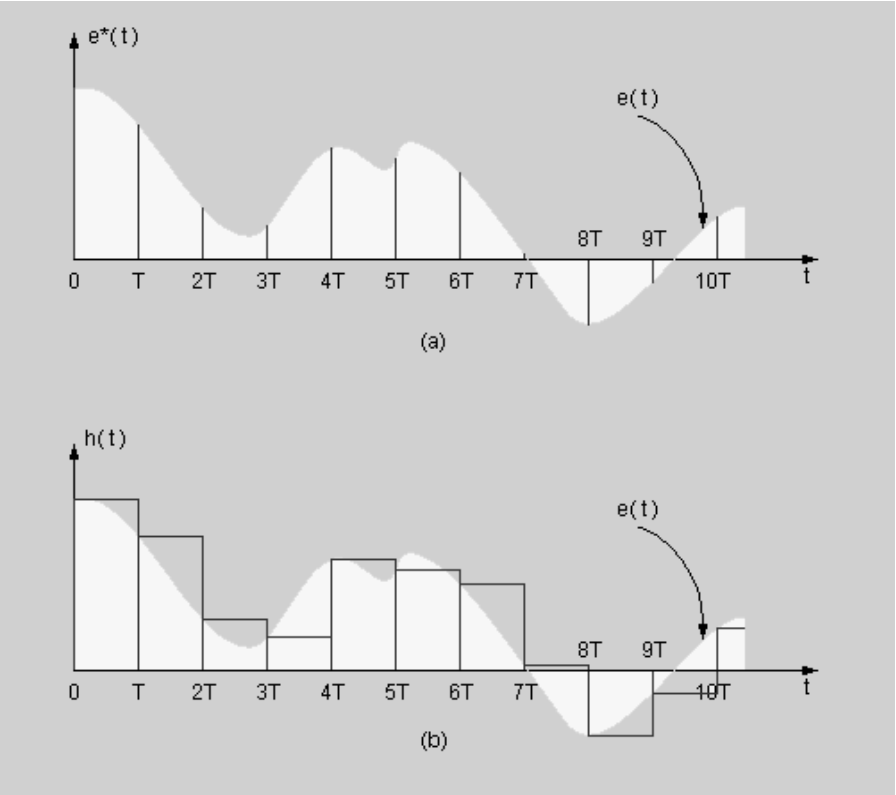
ในระบบนี้จำเป็นต้องมีตัวแปลงสัญญาณจากอนาล็อกไปเป็นดิจิตอล และตัวแปลงสัญญาณจากดิจิตอลกลับมาเป็นอนาล็อกเนื่องจากคอมพิวเตอร์สามารถประมวลผลข้อมูล และส่งผลลัพธ์ออกมาเป็นข้อมูลที่เป็นดิจิตอลเท่านั้น เอาท์พุทของอุปกรณ์สุ่มตัวอย่างจะเป็นขบวนของพัลส์โดยที่แอมพลิจูดของแต่ละพัลส์จะเท่ากับแอมพลิจูดของสัญญาณอินพุทที่ตรงกับช่วงเวลาที่อยู่ในช่วงความกว้างของพัลส์นั้น แต่การวิเคราะห์ระบบที่สุ่มตัวอย่างสัญญาณโดยมีความกว้างของพัลส์นั้นจะต้องอาศัยกระบวนการที่ซับซ้อน ดังนั้นถ้าเอาท์พุทของอุปกรณ์สุ่มตัวอย่างมีความกว้างของพัลส์น้อยมากเมื่อเทียบกับค่า time constant ของส่วนของระบบที่มีสัญญาณต่อเนื่อง และเมื่อเทียบกับช่วงเวลาระหว่างการสุ่มตัวอย่างแต่ละครั้ง ก็จะสามารถแทนที่ขบวนของพัลส์ด้วยขบวนของอิมพัลส์ และในการวิเคราะห์ระบบก็จะสามารถแทนที่อุปกรณ์สุ่มตัวอย่างด้วยอุปกรณ์สุ่มตัวอย่างในอุดมคติ (ideal sampler) ซึ่งมีเอาท์พุทเป็นขบวนของอิมพัลส์ได้

ถ้าอินพุทของอุปกรณ์สุ่มตัวอย่างในอุดมคติเป็นฟังก์ชันที่ต่อเนื่อง e(t) ก็จะได้เอาท์พุทของอุปกรณ์สุ่มตัวอย่างเป็น e*(t) ดังสมการต่อไปนี้

$$e^*(t) = \sum_{n=0}^{\infty} e(nT)\delta(t - nT) \qquad (1)$$

โดยที่ T เป็นช่วงเวลาระหว่างการสุ่มตัวอย่างแต่ละครั้ง

จากสมการจะเห็นว่าอิมพัลส์แต่ละอิมพัลส์จะมีพื้นที่เท่ากับค่าของฟังก์ชัน e(t) ณ ขณะเวลาที่ t = nT ของอิมพัลส์นั้น



ภาพที่ 6 การ Sampling และ Zero-Order hold

ในระบบควบคุมที่มีการสุ่มตัวอย่างสัญญาณมักจะมีอุปกรณ์ที่เรียกว่า data hold device ดังแสดงในภาพที่ 4 อุปกรณ์นี้จะสร้างสัญญาณที่ต่อเนื่องขึ้นมาใหม่โดยแปลงกลับจากสัญญาณที่ถูกสุ่มตัวอย่าง hold device ที่ใช้โดยทั่วไปจะอยู่ในลักษณะของ zero-order hold ซึ่งมีความสัมพันธ์ระหว่างอินพุทกับเอาท์พุทดังแสดงในภาพที่ 6 และเขียนเป็นสมการทางคณิตศาสตร์ได้ดังนี้

$$h(t) = e(nT) \quad for \quad nT < t < (n+1)T \tag{2}$$

โดยที่ T เป็นช่วงเวลาระหว่างการสุ่มตัวอย่างแต่ละครั้ง

6. <u>Characteristic Equation ของระบบควบคุมป้อนกลับ</u>

จากภาพที่ 3 เราได้ closed-loop transfer function เป็นดังนี้

$$M(s) = \frac{C(s)}{R(s)} = \frac{G(s)}{[1 + G(s)\,H(s)]} \tag{3}$$

characteristic equation ของระบบควบคุมแบบป้อนกลับจะได้มาจากการทำตัวหารของ closed-loop transfer function ให้เป็นศูนย์ดังนี้

$$1 + G(s)\,H(s) = 0 \tag{4}$$

พิจารณา closed-loop transfer function ดังต่อไปนี้

$$M(s) = \frac{Y(s)}{R(s)} = \frac{K[(s + z_1) \ldots (s + z_m)]}{[(s + p_1) \ldots (s + p_n)]} \tag{5}$$

จะได้ว่า $-z_1, \ldots, -z_m$ เป็น zero ของ transfer function และ $-p_1, \ldots, -p_n$ เป็น pole ของ transfer function อีกทั้งยังเป็นรากของ characteristic equation อีกด้วย ถ้า M(s) เป็นผลหารที่ประกอบด้วยเศษและส่วนที่เป็น polynomial จะได้ว่า pole และ zero ของ M(s) ประกอบด้วยจำนวนจริงหรือคู่ conjugate ของจำนวนเชิงซ้อนเท่านั้น

ตัวควบคุมแบบพีไอดี

ตัวควบคุมแบบพีไอดี (PID = Proportional Integral Derivative) เป็นอัลกอริธึมการควบคุมที่ใช้ในงานอุตสาหกรรมมาเป็นเวลานานและยังได้รับความนิยมเป็นอย่างมาก อัลกอริธึมแบบพีไอดีนั้นสามารถเข้าใจได้ง่ายและเพียงพอสำหรับการควบคุมระบบพลวัตที่ไม่ซับซ้อนมากและไม่ต้องการสมรรถนะที่สูงขั้นสุดยอด จุดเด่นที่สำคัญคือการที่ตัวควบคุมแบบนี้ไม่ต้องการโมเดลที่สมบูรณ์ของระบบ ผู้ใช้งานเพียงแต่ปรับค่าของพารามิเตอร์พื้นฐาน 3 ค่าเพื่อให้ได้ผลตอบสนองที่เหมาะสมกับงาน (วโรดม, 2547)

1. อัลกอริธึมพีไอดี

โครงสร้างของตัวควบคุมพีไอดีจะมีลักษณะดังนี้

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + T_d\,\frac{de(t)}{dt} \right) \tag{6}$$

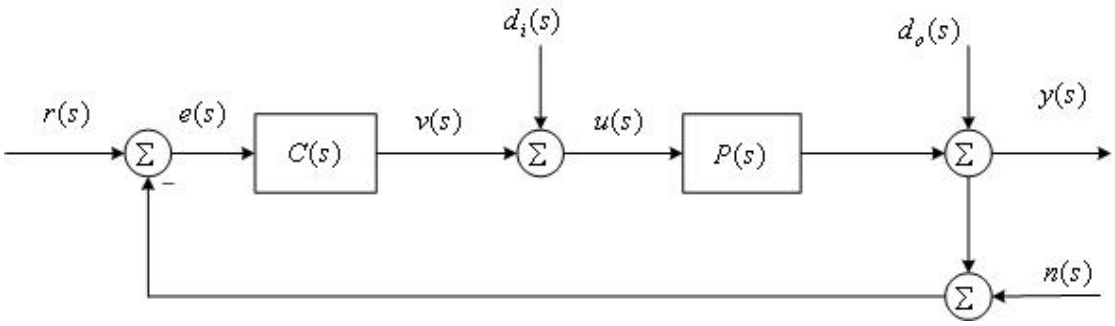
โดยที่ u คือ ตัวแปรควบคุม (control variable) และ e เป็นค่าแตกต่าง $(e = r - y)$ จะเห็นได้ว่าตัวแปรควบคุมเป็นผลรวมของ 3 พจน์ คือ พจน์ P (ซึ่งเป็นสัดส่วนโดยตรงกับค่าแตกต่าง) พจน์ I (เป็นสัดส่วนกับปริพันธ์ของค่าแตกต่าง) และพจน์ D (เป็นสัดส่วนกับอนุพันธ์ของค่าแตกต่าง) โดยพารามิเตอร์ของตัวควบคุม คือ อัตราขยายสัดส่วน K เวลาปริพันธ์ T_i และเวลาอนุพันธ์ T_d

1.1 ผลจากพจน์สัดส่วน (Proportional)

สมมุติว่าเราตั้งค่าของ T_i และ T_d ที่ทำให้สมการ (6) เหลือเพียงพจน์สัดส่วน

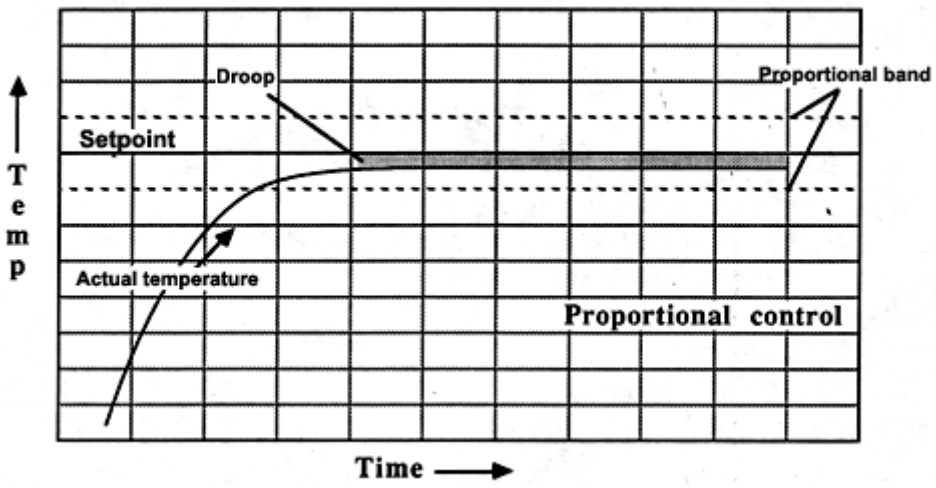
$$u(t) = Ke(t) \tag{7}$$

กรณีที่ใช้ตัวควบคุมในระบบป้อนกลับดังภาพที่ 7



ภาพที่ 7 ระบบป้อนกลับหนึ่งอินพุตหนึ่งเอาต์พุต

จะพบว่าฟังก์ชันถ่ายโอนจาก r ไป y จะเท่ากับฟังก์ชันประกอบความไว (complementary sensitivity function) $T(s) = L(s)\left(1+L(s)\right)^{-1}$ โดยที่ $L(s) = P(s)C(s)$ ผลก็ คือ การที่จะให้เอาต์พุตตามรอยคำสั่งหรือให้ $e(t)$ เข้าสู่ศูนย์ทำได้วิธีเดียวคือเพิ่มอัตราขยาย K ซึ่งในระบบส่วนมากแล้วเราไม่สามารถเพิ่มอัตราขยายสูงมาก ๆ ได้โดยไม่เสียเสถียรภาพ นอกจากนั้นในทางปฏิบัติทั้งตัวควบคุมและตัวขับเร้า (actuator) จะถูกจำกัดด้วยค่าสูงสุดของแหล่งจ่ายกำลัง ดังนั้นตัวควบคุมที่มีเพียงพจน์สัดส่วนจะมีข้อดี คือ พารามิเตอร์ที่ต้องการปรับนั้นจะมีเพียงตัวเดียว คือ อัตราขยาย K แต่อย่างไรก็ตามตัวควบคุมแบบนี้จะมีสมรรถนะที่ไม่ดี โดยไม่สามารถกำจัดค่าแตกต่างในสถานะนิ่ง (steady-state error) ผลจากพจน์สัดส่วนจะเป็นดังภาพที่ 8



ภาพที่ 8 แผนภาพแสดงผลจากพจน์สัดส่วน

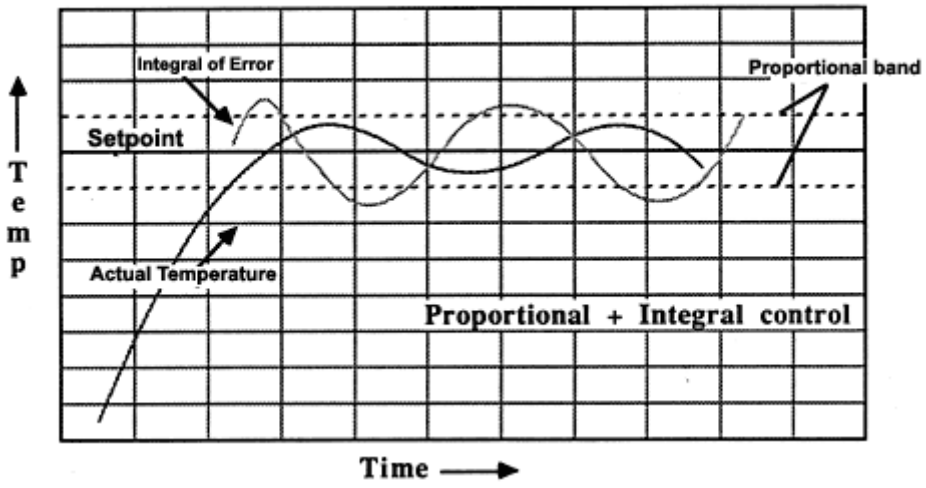
1.2 ผลจากพจน์ปริพันธ์ (Integral)

หน้าที่หลักของพจน์ปริพันธ์ คือ การพยายามทำให้เอาต์พุตมีค่าเท่ากับค่าตั้ง (setpoint) ในสถานะนิ่ง ซึ่งจากการใช้พจน์สัดส่วนอย่างเดียวจะเกิดค่าแตกต่างในสถานะนิ่งที่ไม่สามารถขจัดได้ ทั้งนี้เนื่องจากสัญญาณควบคุมจะคงตัวเมื่อถึงจุด ๆ หนึ่งถึงแม้ว่าจะยังมีค่าแตกต่างจากการตามรอยอยู่ก็ตาม แต่เมื่อใช้พจน์ปริพันธ์ ค่าแตกต่างในทางบวกจะทำให้สัญญาณควบคุมเพิ่มค่าและค่าแตกต่างในทางลบจะทำให้สัญญาณควบคุมลดค่าเสมอ ไม่ว่าค่าแตกต่างนั้นจะเล็กมากเพียงใดก็ตาม

ซึ่งสามารถอธิบายได้ว่าทำไมค่าแตกต่างในสถานะนิ่งถึงเป็นศูนย์โดยพจน์ปริพันธ์ สมมุติว่าระบบอยู่ในสถานะนิ่ง โดยสัญญาณควบคุม u_0 และค่าแตกต่าง e_0 มีค่าคงที่ จากสมการ (6) จะได้ว่าสัญญาณควบคุมมีค่าเท่ากับ

$$u_0 = \left(Ke_0 + \frac{e_0}{T_i} t \right) \tag{8}$$

ตราบเท่าที่ e_0 ไม่เท่ากับศูนย์ สมการนี้จะขัดแย้งกับสมมุติฐานที่ว่า สัญญาณควบคุม u_0 มีค่าคงที่ จะเห็นได้ว่า u_0 จะพยายามปรับตัวไปในทิศทางของ e_0 ผลก็คือตัวควบคุมที่มีพจน์ปริพันธ์จะให้ค่าแตกต่างในสถานะนิ่งเป็นศูนย์ ผลจากการรวมพจน์สัดส่วนและพจน์ปริพันธ์จะเป็นดังภาพที่ 9



ภาพที่ 9 แผนภาพแสดงผลจากพจน์สัดส่วนและพจน์ปริพันธ์

กล่าวโดยสรุปของการควบคุมแบบพีไอคือ

1. การควบคุมแบบปริพันธ์ (Integral) จะถูกนำมาใช้ควบคู่กับการควบคุมแบบสัดส่วน (Proportional) เพื่อขจัดออฟเซ็ต ในกระบวนการที่ไม่สามารถให้มีออฟเซ็ตเกิดขึ้นได้

2. สัญญาณควบคุมจากเทอมปริพันธ์ จะเปลี่ยนแปลงอย่างต่อเนื่องยาวนานเท่ากับที่ค่าความผิดพลาดยังคงปรากฏอยู่

3. ขนาดของสัญญาณควบคุมนอกจากจะขึ้นอยู่กับขนาดของค่าความผิดพลาดแล้วยังขึ้นอยู่กับเวลาที่ค่าความผิดพลาดนั้นสะสมอยู่ด้วย

4. พารามิเตอร์ของตัวควบคุม คือ เกณฑ์การขยาย (gain) หรือ proportional band และเวลาปริพันธ์ (integral time T_i) หรือ reset rate ($T_i^R = 1/T_i$ ครั้งต่อนาที)

5. เวลาปริพันธ์ คือ เวลาที่ใช้ในการเปลี่ยนสัญญาณควบคุมของการควบคุมแบบปริพันธ์ให้มีค่าเท่ากับสัญญาณควบคุมที่ได้จากการควบคุมแบบสัดส่วนเพียงอย่างเดียวหนึ่งครั้ง (ในกรณีที่ค่าความผิดพลาดมีค่าคงที่)

6. เทอมปริพันธ์คือ ไบอัส (bias) ที่เปลี่ยนค่าได้ เพื่อจำกัดออฟเซ็ตและจะหยุดการเปลี่ยนแปลงเมื่อความผิดพลาดหมดไป

7. มีหลายครั้งเมื่อเกิดความผิดพลาดขึ้นแล้วไม่สามารถกำจัดได้อย่างรวดเร็ว ดังนั้นเมื่อเวลาผ่านไปสัญญาณควบคุมที่ได้จากเทอมปริพันธ์จึงมีค่ามากขึ้นเรื่อย ๆ จนกระทั่งถึงจุดอิ่มตัวหรือ (saturated) ซึ่งในสภาวะนี้เรียกว่ากำลังเกิดปรากฏการณ์ปริพันธ์ หรือ reset windup ขึ้น และเกิดขึ้นในระหว่างการดำเนินการควบคุมแบบ manual คล้ายกับการหยุด (shut-down) หรือเกิดการเปลี่ยนแปลงมากเกินไป (change-over) เป็นต้น เมื่อกระบวนการถูกนำกลับเข้ามาสู่การควบคุมแบบอัตโนมัติ การควบคุมก็ยังคงอยู่ในสภาวะอิ่มตัวอยู่ซึ่งจะนำไปสู่สภาพที่จะทำให้เกิด overshoot ขนาดใหญ่

8. เมื่อ K เพิ่มขึ้นจะทำให้ผลตอบสนองของระบบควบคุมเร็วขึ้น (ซึ่งเป็นผลมาจากการควบ-คุมแบบพีไอ) และเกิดการแกว่งไปมามากขึ้นเมื่อค่าเป้าหมายเปลี่ยนไป คือ overshoot และ decay ratio เพิ่มขึ้น (เป็นผลมาจากการควบคุมแบบปริพันธ์) ค่า K ที่มากขึ้นจะทำให้ผลตอบสนองของระบบไวมากขึ้นรวมทั้งยังอาจทำให้การทำงานของระบบขาดเสถียรภาพ

9. ถ้าลดค่า T_i ลงในขณะที่ให้ K มีค่าคงที่แล้ว ก็จะทำให้ผลตอบสนองของระบบเร็วขึ้นแต่ก็แกว่งมากขึ้นด้วยค่าของ overshoot และ decay ratio ที่สูงขึ้น (เป็นผลมาจากการควบคุมแบบปริพันธ์)

ข้อดีของการควบคุมแบบพีไอ

1. ออฟเซ็ตถูกจำกัด (เป็นผลมาจากการควบคุมแบบปริพันธ์)
2. การทำงานมีเสถียรภาพดีขึ้น (เป็นผลมาจากการควบคุมแบบสัดส่วน)เหมาะที่จะนำมาใช้กับกระบวนการที่โหลดมีการเปลี่ยนแปลง

ข้อเสียของการควบคุมแบบพีไอ

1. จะสร้าง Lag ขึ้นมาในเครื่องควบคุม นั่นคือ คาบเวลา (period) ของการออสซิลเลตจะยาวขึ้นประมาณ 50 เปอร์เซ็นต์
2. ถ้าช่วงของเวลา T_i นั้นสั้นเกินไป จะทำให้สัญญาณควบคุมที่ได้จากเทอมปริพันธ์นั้นขับส่วนขับเคลื่อนจนถึงเขตจำกัด (limit) ก่อนที่กระบวนการจะตอบสนองได้ทัน
3. ถ้าความผิดพลาดนั้นเกิดขึ้นยาวนาน จะเกิดปรากฏการณ์ที่เรียกว่า windup ขึ้นซึ่งเป็นผลมาจากการควบคุมแบบปริพันธ์ ซึ่งเรียกกันว่า integral windup หรือ reset windup

1.3 ผลจากพจน์อนุพันธ์ (Derivative)

วัตถุประสงค์ของการเพิ่มพจน์อนุพันธ์ก็เพื่อปรับปรุงเสถียรภาพของระบบวงปิด กลไกที่ทำให้เสียเสถียรภาพสามารถกล่าวได้อย่างง่ายคือ ผลจากพลวัตของระบบทำให้ต้องใช้เวลาช่วงหนึ่งก่อนที่การเปลี่ยนแปลงของตัวแปรควบคุมจะเกิดผลกับเอาต์พุต ดังนั้นระบบควบคุมจะทำการแก้ไขค่าแตกต่างสายเกินไปเสมอ ผลจากตัวควบคุมแบบที่มีพจน์สัดส่วนและพจน์อนุพันธ์สามารถพิจารณาได้เสมือนว่าสัญญาณควบคุมถูกสร้างขึ้นเป็นสัดส่วนกับเอาต์พุตที่ถูกคาดเดา โดยการคาดเดากระทำโดยการประมาณค่านอกช่วง (extrapolation) ของค่าแตกต่างที่แท้จริงด้วยเส้นสัมผัส โครงสร้างพื้นฐานของตัวควบคุมแบบพีดี (PD) คือ

$$u(t) = K\left(e(t) + T_d \frac{de(t)}{dt} \right) \tag{9}$$

จากการกระจายอนุกรมเทย์เลอร์ของ $e(t+T_d)$ จะได้ว่า

$$e(t+T_d) = e(t) + T_d \frac{de(t)}{dt} \tag{10}$$

ดังนั้น สัญญาณควบคุมจะเป็นสัดส่วนกับค่าแตกต่างที่ประมาณล่วงหน้าเป็นเวลาเท่ากับ T_d โดยใช้วิธีที่เรียกว่าวิธีการประมาณค่านอกช่วง การควบคุมแบบพีดี นี้เหมาะที่จะนำมาใช้ในระบบการควบคุมที่มี dead time และส่วนสะสมพลังงาน (storage element) อยู่เป็นจำนวนมาก กล่าวโดยสรุปของการควบคุมแบบพีดี คือ

1. การควบคุมแบบอนุพันธ์ (Derivative) จะถูกนำมาใช้ควบคู่กับการควบคุมแบบสัดส่วน (Proportional) เมื่อลักษณะของการควบคุมนั้นต้องการ “การควบคุมล่วงหน้า (anticipation)”
2. จะไม่มีสัญญาณควบคุมจากเทอมอนุพันธ์ ถ้าความผิดพลาดที่เกิดขึ้นมีค่าคงที่หรือเป็นศูนย์
3. พารามิเตอร์ของเครื่องควบคุมคือ เกณฑ์การขยายหรือ proportional band และเวลาอนุพันธ์ (derivative time) หรือ rate time
4. ในการควบคุมจะเกิดออฟเซ็ตขึ้นจำนวนหนึ่ง แต่จะเล็กกว่าออฟเซ็ตที่เกิดขึ้นจากการควบคุมแบบสัดส่วนเพียงอย่างเดียว เนื่องจากค่าเกณฑ์การขยายที่เพิ่มขึ้น
5. ขนาดของสัญญาณเอาท์พุตจากตัวควบคุมแบบอนุพันธ์ จะขึ้นอยู่กับอัตราการเปลี่ยนแปลงของค่าความผิดพลาด แม้ว่าขนาดของค่าความผิดพลาดจะมีค่าน้อยมากก็ตาม
6. การควบคุมแบบอนุพันธ์ จะทำให้ผลตอบสนองของการควบคุมแบบสัดส่วนเกิดขึ้นก่อนล่วงหน้า
7. เวลาล่วงหน้านี้จะขึ้นอยู่กับค่าของเวลาอนุพันธ์ (T_d)
8. จะปรับปรุงการควบคุมของกระบวนการที่มี lag หรือ process time มาก
9. สามารถนำมาใช้ได้ดีกับลูปการควบคุมที่มีการเปลี่ยนแปลงค่อนข้างช้า เช่น ลูปของการควบคุมอุณหภูมิ

ข้อดีของตัวควบคุมแบบพีดี

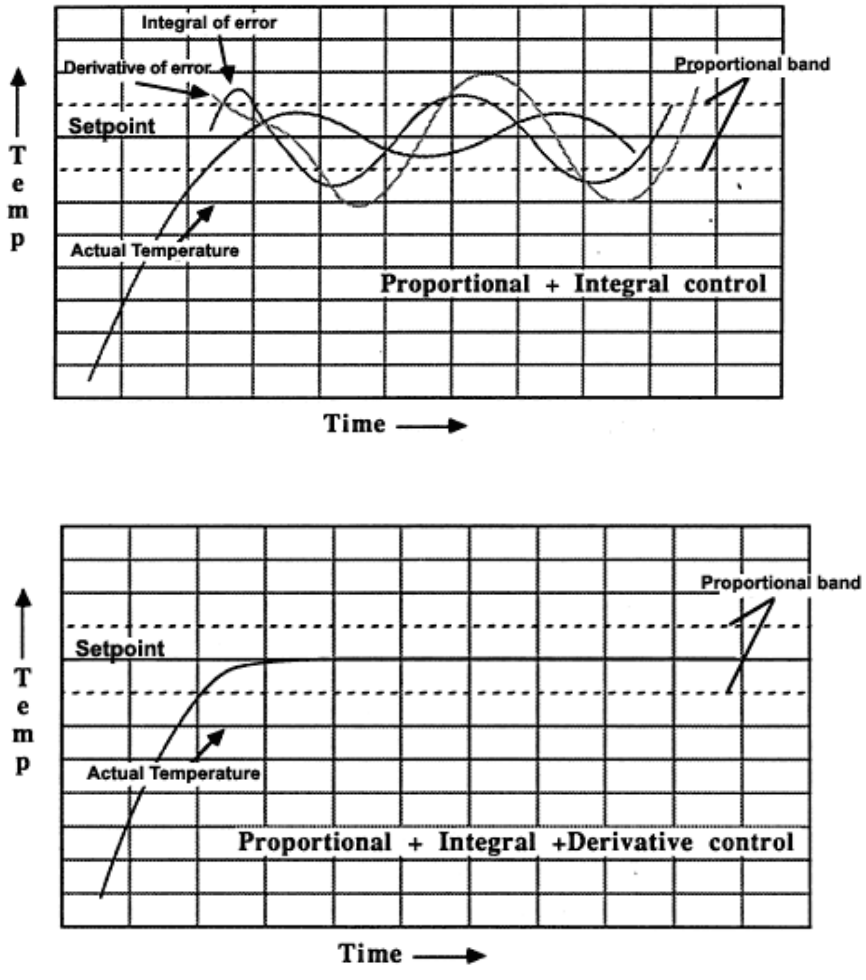
1. ทำให้เกณฑ์การขยายของตัวควบคุมเพิ่มขึ้น
2. จะหักล้าง lag ที่มีอยู่ในลูปของการควบคุมซึ่งจะมีผลทำให้คาบเวลาของการแกว่งสั้นลง
3. ทำให้ผลตอบสนองของการควบคุมไวขึ้น โดยไม่ทำให้เกิดการแกว่งมากเกินไป

ข้อเสียของตัวควบคุมแบบพีดี

1. ในการทำงานจะทำให้เกิดออฟเซ็ตขึ้นจำนวนหนึ่งกับตัวแปรควบคุม

2. ผลของการควบคุมจะทำให้กระบวนการนั้นไวต่อสิ่งรบกวน จึงไม่เหมาะกับลูปหรือกระบวนการที่ไวต่อการเปลี่ยนแปลง

ซึ่งผลที่ได้จากการรวมพจน์สัดส่วน, พจน์ปริพันธ์และพจน์อนุพันธ์ ตามสมการ (6) จะเป็นดังภาพที่ 10



ภาพที่ 10 แผนภาพแสดงผลจากพจน์สัดส่วน,พจน์ปริพันธ์และพจน์อนุพันธ์

โดยสรุปแล้วตัวควบคุมที่ใช้การควบคุมแบบพีไอดี (PID) จะมีพารามิเตอร์ที่ต้องทำการปรับอยู่สามตัวคือ เกณฑ์การขยาย หรือ proportional band, reset time หรือ reset rate และ rate time ซึ่งการควบคุมแบบพีไอดีเหมาะที่จะนำมาใช้กับการควบคุมกระบวนการหรือลูปการควบคุมที่มีค่าคงตัวเวลา (time constant) ยาวนานและปราศจากสัญญาณรบกวนใด ๆ

ข้อดีของของตัวควบคุมแบบพีไอดี

1. ให้ความสามารถในการดูแลขณะที่กระบวนการกำลังดำเนินหรือเปลี่ยนแปลงอยู่
2. กำจัดออฟเซ็ต ที่เกิดจากการควบคุมแบบสัดส่วน
3. ลดหรือกด (depresses) สัญญาณควบคุมเมื่อผลตอบสนองของการควบคุมมีแนวโน้มที่จะเกิดการออสซิเลต

2. <u>รูปแบบของอัลกอริธึมพีไอดี</u>

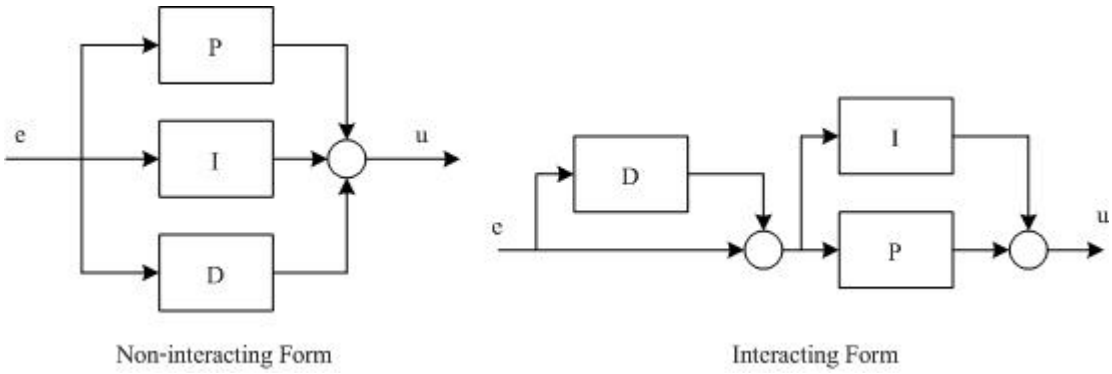
อัลกอริธึมพีไอดีตามสมการ (6) จะเหมาะสมกับการศึกษาหลักการ แต่มักไม่พบบ่อยในการใช้งานจริง เพราะในทางปฏิบัติอัลกอริธึมมักถูกดัดแปลงเพื่อให้มีสมรรถนะดีขึ้น อัลกอริธึมตามสมการ (6) เมื่อแปลงเป็นฟังก์ชันถ่ายโอนในโดเมนความถี่จะได้เป็น

$$G(s) = K\left(1 + \frac{1}{sT_i} + sT_d\right) \tag{11}$$

ในขณะที่อีกรูปแบบหนึ่งที่นิยมใช้มากในผลิตภัณฑ์ชุดควบคุมอุตสาหกรรมจะเขียนได้เป็น

$$G'(s) = K'\left(1 + \frac{1}{sT_i'}\right)\left(1 + sT_d'\right) \tag{12}$$

ซึ่งโครงสร้างของตัวควบคุมทั้งสองแบบสามารถเขียนเป็นบล็อกไดอะแกรมได้ดังภาพที่ 11



ภาพที่ 11 แผนภาพบล็อกไดอะแกรมของตัวควบคุมทั้ง 2 แบบ

ตัวควบคุมตามสมการ (11) เรียกว่าแบบไม่มีอันตรกิริยาต่อกัน (non-interacting) ส่วนตัวควบคุมตามสมการ (12) เรียกว่าแบบมีอันตรกิริยาต่อกัน (interacting) เหตุผลในการเรียกชื่อแบบนี้ก็เนื่องมาจากในสมการ (11) เวลาปริพันธ์ T_i ไม่มีผลกระทบกับเวลาอนุพันธ์ T_d ขณะที่สมการ (12) ค่าทั้งสองนี้มีผลกระทบต่อกัน

ข้อสังเกตคือ ตัวควบคุมแบบอันตรกิริยาสมการ (12) สามารถแทนที่โดยตัวควบคุมแบบไม่มีอันตรกิริยาสมการ (11) ได้โดยการแทนค่าพารามิเตอร์ดังนี้

$$\begin{aligned} K &= K' \frac{T_i' + T_d'}{T_i'} \\ T_i &= T_i' + T_d' \\ T_d &= \frac{T_i' T_d'}{T_i' + T_d'} \end{aligned} \tag{13}$$

ในทางกลับกันเราสามารถหาตัวควบคุมแบบอันตรกิริยาเพื่อแทนตัวควบคุมแบบไม่มีอันตรกิริยาได้ก็ต่อเมื่อ

$$T_i \geq 4T_d$$

ซึ่งจะได้ว่า

$$K' = \frac{K}{2}\left(1+\sqrt{1-4\frac{T_d}{T_i}}\right)$$
$$T_i' = \frac{T_i}{2}\left(1+\sqrt{1-4\frac{T_d}{T_i}}\right) \quad (14)$$
$$T_d' = \frac{T_i}{2}\left(1+\sqrt{1-4\frac{T_d}{T_i}}\right)$$

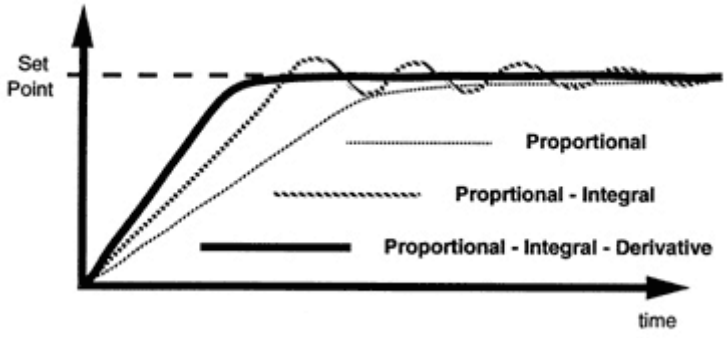
อัลกอริธึมพีไอดีอีกแบบหนึ่งที่สามารถพบได้คือ

$$G''(s) = k + \frac{k_i}{s} + sk_d \quad (15)$$

ซึ่งเรียกว่าแบบขนาน (parallel form) ตัวอย่างที่พบเห็นก็คือในแบบจำลองซอฟท์แวร์ Simulink ความสัมพันธ์ของพารามิเตอร์ของตัวควบคุมในแบบนี้กับในแบบมาตรฐานตามสมการ (11) จะเป็น

$$k = K$$
$$k_i = \frac{K}{T_i} \quad (16)$$
$$k_d = KT_d$$

ซึ่งผลตอบสนองของการควบคุมแต่ละรูปแบบจะเป็นดังภาพที่ 12



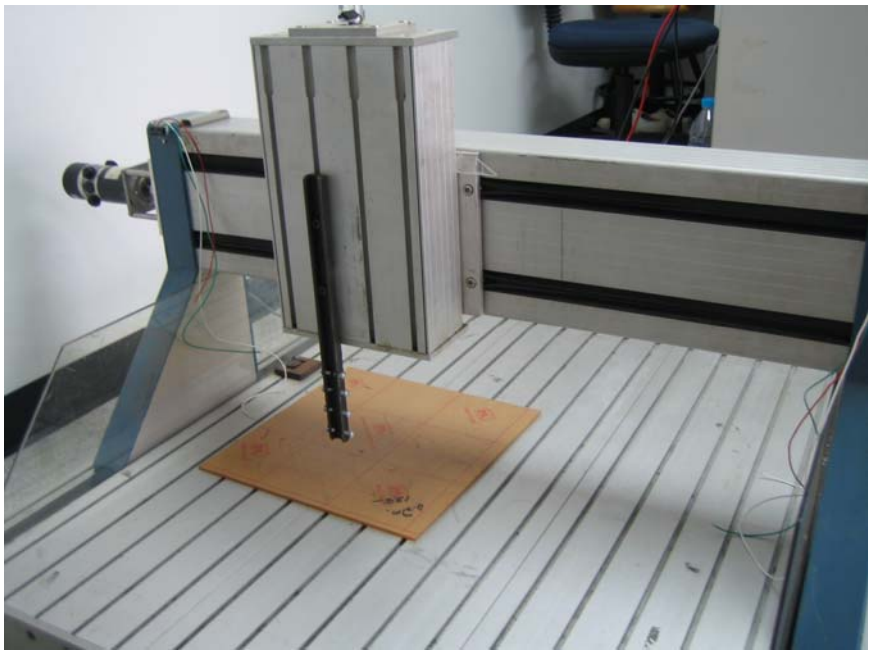
ภาพที่ 12 แผนภาพแสดงผลตอบสนองของการควบคุม

รายละเอียดส่วน Hardware

ในหัวข้อนี้เราจะกล่าวถึงส่วนประกอบหลักๆ ของการออกแบบ Controller สำหรับ CNC 3 แกน

1. แท่น XYZ

แท่น XYZ เป็นแท่นทดลองสร้างการเคลื่อนที่ 3 มิติ ทั้ง 3 แกนเป็นแบบ Ball Screw เลื่อนได้โดยระยะหมุน 1 รอบจะเคลื่อนที่ได้ 0.5 ซ.ม. รายละเอียดของแต่ละแกนมี แกน X, แกน Y = 56 x 56 ซ.ม. และแกน Z = 5 ซ.ม. แสดงดังภาพที่ 13



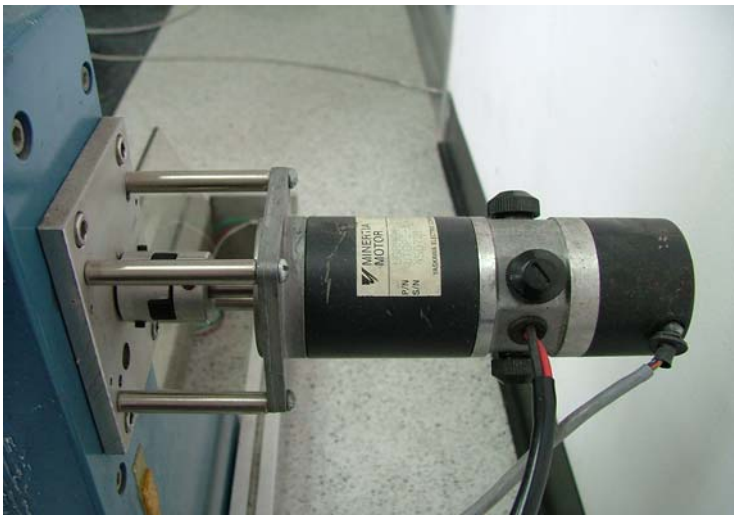
ภาพที่ 13 แท่น XYZ Table

2. เซอร์โวมอเตอร์

เซอร์โวมอเตอร์เป็นส่วนประกอบสำคัญที่ทำหน้าที่สร้างแรงขับให้เกิดการเคลื่อนที่ของแกนต่างๆ ในเครื่องจักร ไม่ว่าจะเป็นการทำงานในลักษณะหมุน หรือผ่านตัวแปลงเปลี่ยนให้เป็นการเคลื่อนที่เชิงเส้นก็ตาม เซอร์โวมอเตอร์ในปัจจุบันถูกผลิตให้มีขนาดเล็กและน้ำหนักเบา มีประสิทธิภาพสูง ง่ายต่อการประกอบเข้าเครื่อง และมีความคงทนไม่ต้องการบำรุงรักษามาก โดยเฉพาะเซอร์โวมอเตอร์แบบไม่มีแปรงถ่าน (brushless servo motors) อย่างไรก็ตาม มอเตอร์กระแสตรง (DC servo motors) แบบที่ใช้แปรงถ่านก็ยังมีใช้อยู่บ้าง เพราะมีคุณสมบัติที่ดี เช่น มีระบบพลวัตและวงจรขับที่ง่าย

2.1 มอเตอร์ไฟฟ้ากระแสตรง (DC servo motors)

หลักการพื้นฐานของมอเตอร์ไฟฟ้ากระแสตรงคือ กระแสไฟจากแหล่งจ่ายไฟตรงจะไหลเข้าสู่ตัวนำในโรเตอร์ (rotor) โดยผ่านแปรงถ่านและคอมมิวเตเตอร์ (commutator) ทำให้เกิดอำนาจแม่เหล็กไฟฟ้าตัดกับสนามแม่เหล็กที่เกิดจากแม่เหล็กถาวร (permanent magnet) ขั้ว N และ S ทำให้เกิดแรงบิดที่ทำให้โรเตอร์หมุนไป เมื่อโรเตอร์หมุนได้ 90 องศา คอมมิวเตเตอร์จะทำให้ทิศทางของกระแสย้อนกลับเป็นตรงข้ามเป็นผลให้โรเตอร์ยังหมุนต่อไปได้ ตัวอย่างมอเตอร์ไฟฟ้ากระแสตรงแสดงดังภาพที่ 14



ภาพที่ 14 เซอร์โวมอเตอร์กระแสตรง

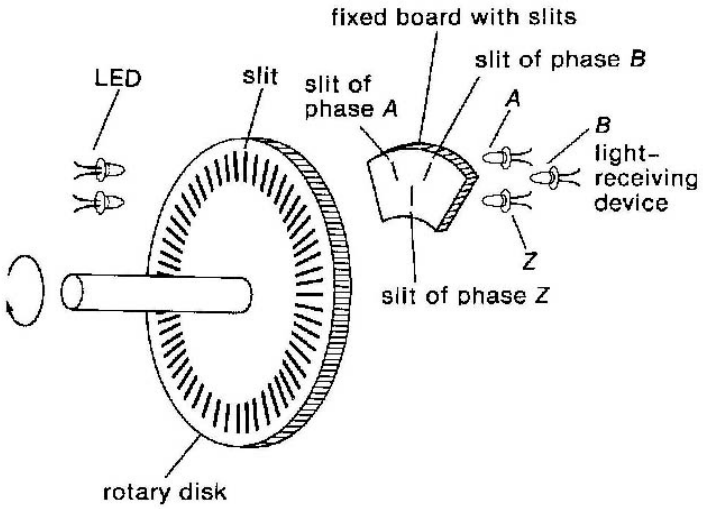
2.2 มอเตอร์ไฟฟ้ากระแสสลับ (AC servo motors)

ในการทำงานของมอเตอร์ไฟฟ้ากระแสสลับ คอมมิวเตเตอร์จะถูกแทนที่ด้วยแหวนสลิป (slip ring) ที่จะได้รับแรงดันไฟฟ้ากระแสสลับ สมมุติว่าเริ่มต้นด้วยการที่แปรงถ่าน A มีศักดาเป็นบวกเมื่อเทียบกับ B ก็จะทำให้เกิดแรงบิดขึ้นเช่นเดียวกับในกรณีมอเตอร์ไฟฟ้ากระแสตรง แต่หลังจากช่วงหนึ่งมอเตอร์ก็จะเริ่มที่จะหยุดหมุนเนื่องจากไม่มีการคอมมิวเตชัน ดังนั้น จะต้องมีการกลับขั้วของแรงดันในช่วงเวลาที่เหมาะสม นั้นคือ การใช้ไฟฟ้ากระแสสลับเป็นตัวจ่ายกำลังจะทำให้มอเตอร์หมุนอย่างต่อเนื่องได้โดยอัตราความเร็วของการหมุนจะขึ้นกับความถี่ของแรงดันกระแสสลับ ตัวอย่างมอเตอร์ไฟฟ้ากระแสสลับแสดงดังภาพที่ 15

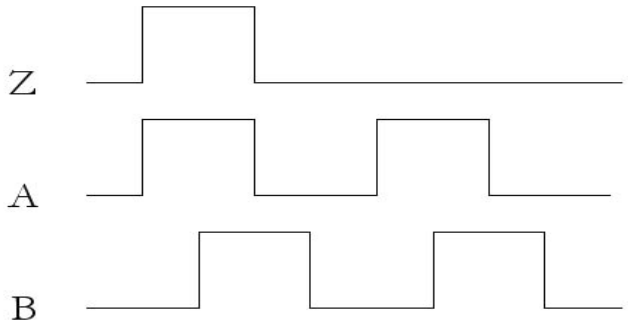
ภาพที่ 15 เซอร์โวมอเตอร์กระแสสลับพร้อมชุดขับ

3. <u>วงจรต่อประสานกับเอนโคเดอร์</u>

ส่วนสำคัญที่จะทำให้ทราบตำแหน่งของการเคลื่อนที่ก็คือวงจรรับสัญญาณจากเอนโคเดอร์และแปลงเป็นค่าตำแหน่ง ในงานวิจัยนี้เราใช้เอนโคเดอร์แบบส่วนเพิ่ม (incremental encoder) ดังในรูปที่ 16 ที่จะให้สัญญาณเป็นพัลซ์ A,B,Z ดังแสดงในรูปที่ 17 แต่เอนโคเดอร์ที่ดีจะให้สัญญาณออก 6 สัญญาณ คือ $A, \overline{A}, B, \overline{B}, Z, \overline{Z}$ ซึ่งทั้ง 6 สัญญาณก็คือสามสัญญาณหลักในรูปที่ 17 และสัญญาณกลับเฟส 180^{o} เนื่องจากเป็นช่วยกำจัดสัญญาณรบกวนดังจะได้กล่าวถึงต่อไป



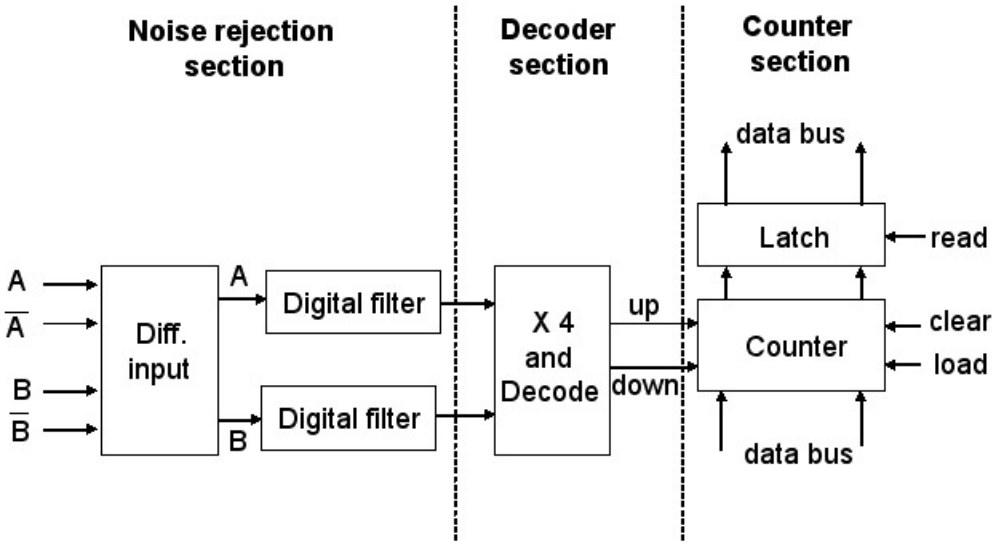
<u>ภาพที่ 16</u> โครงสร้างของเอนโคเดอร์แบบส่วนเพิ่ม (incremental encoder)



<u>ภาพที่ 17</u> พัลซ์ที่เกิดจากเอนโคเดอร์แบบส่วนเพิ่ม

ภาพที่ 18 คือ บล็อกไดอะแกรมที่แสดงส่วนต่าง ๆ ของวงจรต่อประสานกับเอนโคเดอร์ โดยเราสามารถแบ่งได้เป็น 3 ภาค คือ ภาคกำจัดสัญญาณรบกวน ภาคถอดรหัส (decoder) และภาควงจรนับ ซึ่งจะได้กล่าวอธิบายแต่ละภาคตามลำดับ (จะเห็นว่าเราไม่ได้ใช้สัญญาณอินเด็กซ์พัลซ์ Z ในงานวิจัยนี้ สัญญาณ Z จะมีประโยชน์ในการสร้างชุดควบคุมสำหรับเครื่องจักรกลอุตสาหกรรมที่จำเป็นต้องมีจุดอ้างอิงในการทำงาน ตัวอย่างเช่น การทำ homing ในเครื่องซีเอ็นซี)

วงจรภาคต่าง ๆ ทั้งหมดยกเว้นภาคกำจัดสัญญาณรบกวนในภาพที่ 18 จะถูกอิมพลิเม้นท์ลงบน FPGA ซึ่งซอฟท์แวร์ MAXPlus II ของบริษัท Altera จะมีไลบรารี่ของไอซี TTL ให้ใช้งาน



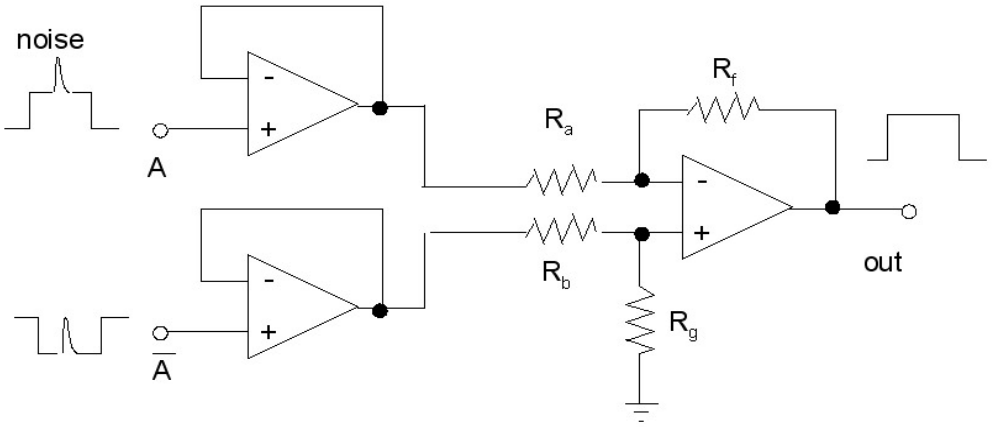
ภาพที่ 18 บล็อกไดอะแกรมของวงจรต่อประสานกับเอนโคเดอร์

4. ภาคกำจัดสัญญาณรบกวน

ภาคกำจัดสัญญาณรบกวนเป็นภาคแรกที่รับสัญญาณอินพุตจากเอนโคเดอร์ ซึ่งถึงแม้ว่าเราจะชีลด์สายสัญญาณอย่างดีก็ยังมีโอกาสที่สัญญาณรบกวนจะแทรกเข้ามาได้ เพราะในระบบควบคุมการเคลื่อนที่ประกอบด้วยอุปกรณ์หลายชนิดที่กำเนิดคลื่นสนามไฟฟ้าแม่เหล็ก เช่นเซอร์โวมอเตอร์ หากไม่มีวงจรกำจัดสัญญาณรบกวนจะทำให้การอ่านสัญญาณเกิดความผิดพลาด และในเอนโค-

เดอร์แบบส่วนเพิ่มนี้ค่าผิดพลาดจะสะสมอยู่ในวงจรนับที่แสดงตำแหน่งจนกว่าจะมีการตั้งค่าอ้างอิงใหม่ ดังนั้นการจำกัดสัญญาณรบกวนจึงเป็นสิ่งที่จำเป็นอย่างมาก

ภาพที่ 19 แสดงวงจรขยายผลต่าง (differential amplifier) อันเป็นวงจรแรกที่รับสัญญาณจากเอนโคเดอร์ ออปแอมป์สองตัวแรกที่อินพุตเป็นวงจรตามแรงดัน (voltage follower) ที่มี

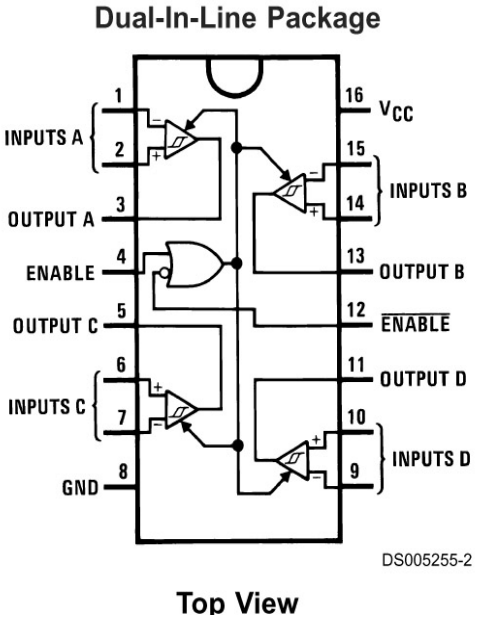


ภาพที่ 19 วงจรขยายสัญญาณผลต่าง

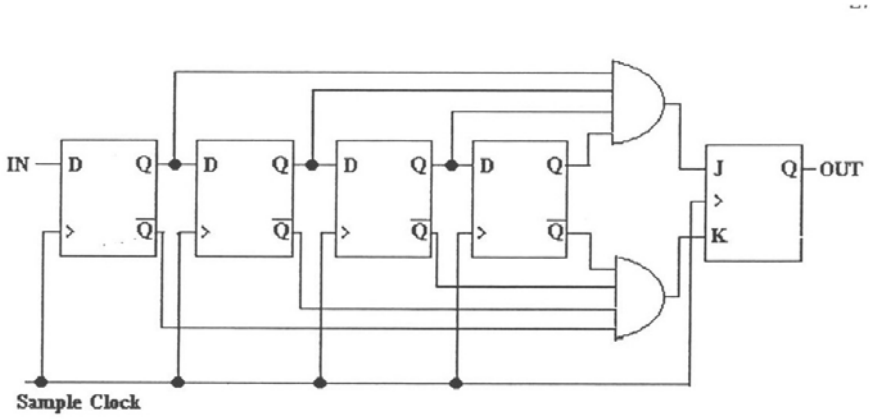
อัตราขยายเท่ากับหนึ่งโดยทำหน้าที่เป็นเสมือนบัฟเฟอร์ (buffer) ที่มีความต้านทานทางอินพุตสูง ส่วนออปแอมป์ทางด้านเอาต์พุตจะเป็นตัวขยายผลต่าง โดยสัญญาณที่เหมือนกันจะถูกขจัดทิ้งไป หลักการนี้เรียกว่า การขจัดโหมดร่วม (common mode rejection) จากรูปเราจะเห็นว่าสัญญาณที่ได้รับจากเอนโคเดอร์จะต่างเฟสกัน 180^{o} แต่ถ้าหากเราเดินสายสัญญาณอยู่ในเคเบิลเดียวกัน สัญญาณรบกวนที่เข้ามาจะมีรูปร่างและขนาดใกล้เคียงกัน เหมือนเช่นสัญญาณรบกวนที่เป็นยอดแหลมในรูป ดังนั้นวงจรขยายผลต่างจะสามารถขจัดสัญญาณนี้ออกไปได้ ในวงจรตามรูปที่ถ้าหากเราเลือก $R_a = R_b$ และ $R_f = R_g$ จะได้อัตราขยายวงจรเท่ากับ R_f/R_a

เรายกตัวอย่างวงจรแสดงดังภาพที่ 19 เพียงเพื่ออธิบายหลักการ ในการสร้างวงจรจริงเราจะเลือกใช้ชิพสำเร็จรูป DS26LS32 ดังภาพที่ 20 ที่ในไอซีตัวหนึ่งจะประกอบด้วยภาคขยายสัญญาณผลต่างจำนวน 4 ชุด

หลังจากที่ผ่านวงจรขยายผลต่างแล้ว เราจะใช้วงจรที่เรียกว่า วงจรกรองแบบดิจิทัล (digital filter) ดังภาพที่ 21 ทำการขจัดสัญญาณรบกวนอีกครั้งหนึ่ง ซึ่งหลักการทำงานคือ สัญญาณอินพุตหรือเอนโคเดอร์พัลซ์จะมีความถี่ต่ำเมื่อเทียบกับสัญญาณรบกวน ดังนั้นหากเราใช้สัญญาณนาฬิกาในรูปที่ ที่มีความถี่สูง สัญญาณเอนโคเดอร์จะถูกส่งผ่านฟลิปฟลอปทั้งสี่ตัว โดยถ้าเอาต์พุตของทั้งสี่ตัวมีค่าเหมือนกันจะถือว่าเป็นสัญญาณเอนโคเดอร์ที่ถูกต้อง แต่ถ้าหากว่ามีสัญญาณรบกวนที่ทำให้เอนโคเดอร์ พัลซ์มีค่าเปลี่ยนไปในช่วงสั้น ๆ สัญญาณรบกวนนั้นจะไม่ปรากฏที่เอาต์พุตของวงจรกรองนี้ นอกเสียจากว่าสัญญาณนาฬิกามีความถี่สูงกว่าสัญญาณ รบกวนมาก ดังนั้น เราอาจต้องทำการเลือกความถี่สัญญาณนาฬิกาให้เหมาะสมกับความถี่ของสัญญาณรบกวนในระบบ



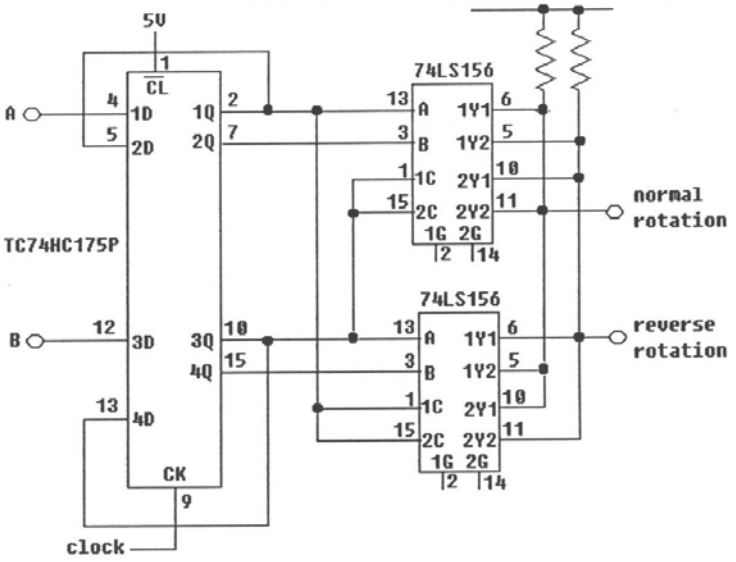
ภาพที่ 20 ชิพ DS26LS32 Differential Line Receiver



ภาพที่ 21 วงจรกรองแบบดิจิทัล

5. ภาคถอดรหัส

ภาพที่ 22 แสดงวงจรถอดรหัสแบบง่าย ที่ใช้ไอซีแบบ TTL โดยวงจรนี้มีความสามารถในการเพิ่มความละเอียดของเอนโคเดอร์ พัลซ์เป็น 4 เท่า ทำให้แสดงค่าตำแหน่งการเคลื่อนที่ได้ละเอียดมากขึ้น โดยเอาต์พุตจะเป็นพัลซ์ออกที่ขา normal หรือ reverse ขึ้นอยู่กับว่ามอเตอร์หมุนในทิศทางใด



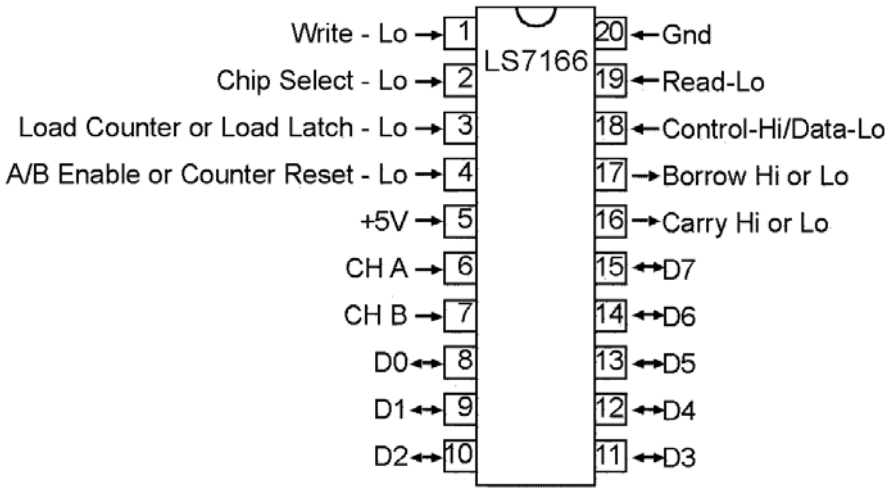
ภาพที่ 22 วงจรถอดรหัสเอนโคเดอร์

6. ภาควงจรนับและแลทช์

พัลซ์ที่ได้จากวงจรถอดรหัสเช่นในภาพที่ 22 จะถูกนำไปเข้าวงจรนับ ซึ่งในงานวิจัยนี้จะใช้ขนาด 32 บิตเพื่อที่จะสามารถเก็บตำแหน่งของแกนได้ระยะทางมากและละเอียด นอกจากนั้นยังมีความเหมาะสมเนื่องจากบัสของ TMS320C6713 มีขนาดเท่ากับ 32 บิตเช่นกัน ทำให้สามารถอ่านค่าจากวงจรนับได้ทั้งหมดในครั้งเดียว ด้วยเหตุนี้เองเราจึงไม่จำเป็นต้องแลทช์ข้อมูลก่อนอ่าน ทำให้ลดจำนวนเกตที่ต้องใช้ลง

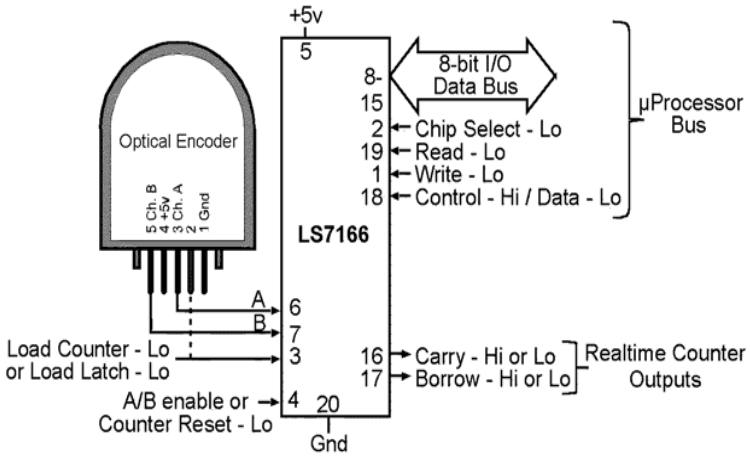
7. วงจร LS7166

LS7166 เป็น IC 24 bit Up/Down Counter หรือ Encoder to Microprocessor Interface Chip โดย IC จะแสดงดังภาพที่ 23 ซึ่งสามารถใช้แทนวงจรในหัวข้อที่ 5 และ 6 ในภาพที่ 22 ได้



ภาพที่ 23 ชิพสำเร็จรูป LS7166

การใช้งาน LS7166 จะทำโดยติดต่อกับ Microcontroller ผ่านทาง Data bus 8 bits และติดต่อกับ Encoder ทางอินพุต A, B ขา 6,7 ภาพที่ 24



ภาพที่ 24 แผนภาพตัวอย่างการต่อ LS7166

LS7166 มีโหมดการทำงานที่สามารถโปรแกรมได้คือ

- โหมดการนับขึ้นหรือนับลงแบบ ไบนารี, บีซีดี, นาฬิกา (24 ชั่วโมง), การนับแบบมีค่าของการ หาร (Divide-by-N),และมีโหมดการนับแบบคูณ 1,2 หรือ 4 ต่อ 1 สัญญาณการนับได้
- รับความถี่ในการนับเมื่อสัญญาณเป็นแบบ DC ได้สูงสุด 20 MHz
- มีอินพุต/เอาพุตขนาด 8 บิตเพื่อเชื่อมต่อกับไมโครโพรเซสเซอร์ในการสื่อสารและการควบคุม
- มีความละเอียดในการตั้งค่าเริ่มต้นของการนับขนาด 24 บิตในการนับแบบเปรียบเทียบ
- มีรีจิสเตอร์บอกสถานะที่สามารถอ่านค่าได้
- อินพุต/เอาพุต เป็นแบบ TTL หรือ CMOS ขนาด 0-5 โวลต์

7.1 โครงสร้างภายในของ LS7166

โครงสร้างภายในของ LS7166 นั้นประกอบไปด้วย I/O Buffer ที่ทำหน้าที่ในการรับส่งข้อมูลแบบขนานขนาด 8 บิต และ Input Buffer And Decode Logic ที่ทำหน้าที่ในการควบคุมการอ่านหรือเขียนข้อมูลกับ LS7166 ซึ่งในการอ่านหรือเขียนข้อมูลลงใน LS7166 นั้น Input Buffer And Decode Logic กับ I/O Buffer จะทำหน้าที่ร่วมกันเพื่อที่จะทำการกำหนดตำแหน่งที่อยู่ผ่าน PR/OL Address เพื่อทำการเขียนหรืออ่านข้อมูลไปยังรีจิสเตอร์ QR, OCCR, ICR, MCR, หรือ PR รีจิสเตอร์ซึ่งในการเขียนหรืออ่านข้อมูลจาก PR รีจิสเตอร์นั้นจำเป็นต้องมีการกำหนดค่าที่อยู่เริ่มต้น

ให้กับ PR/OL Address ก่อนส่วนการเขียนหรืออ่านข้อมูลกับรีจิสเตอร์ทั้งสี่ตัวนั้นไม่จำเป็นต้องทำการกำหนดตำแหน่งเริ่มต้น หลังจากที่มีการกำหนดตำแหน่งในการอ่านหรือเขียนข้อมูลแล้ว ข้อมูลจะถูกส่งผ่าน Data Bus ขนาด 8 บิตไปยังรีจิสเตอร์ต่าง ๆ ตามที่ I/O Buffer เป็นตัวกำหนดโดยใช้ข้อมูลในบิตที่ 6 และบิตที่ 7 เป็นตัวกำหนด ข้อมูลนั้นจะถูกนำไปกำหนดสถานะของการนับ หรือเป็นค่าเริ่มต้นของการนับขึ้นอยู่กับว่ามีการกำหนดค่าให้กับรีจิสเตอร์แต่ละตัวอย่างไร ดังแสดงในภาพที่ 25

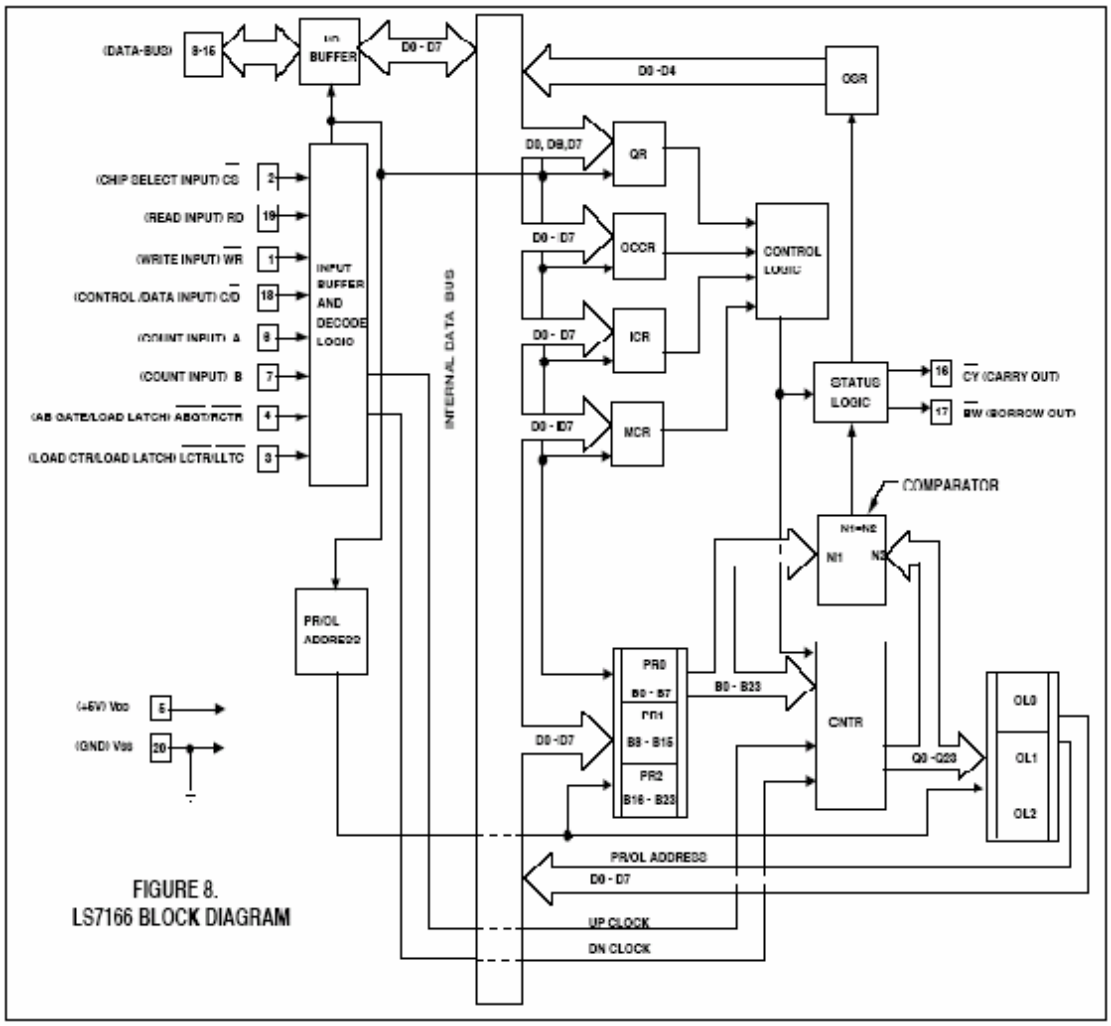


FIGURE 8.
LS7166 BLOCK DIAGRAM

ภาพที่ 25 โครงสร้างภายในของ LS7166

7.2 การเข้าถึงรีจิสเตอร์และหน้าที่ของรีจิสเตอร์ภายใน LS7166

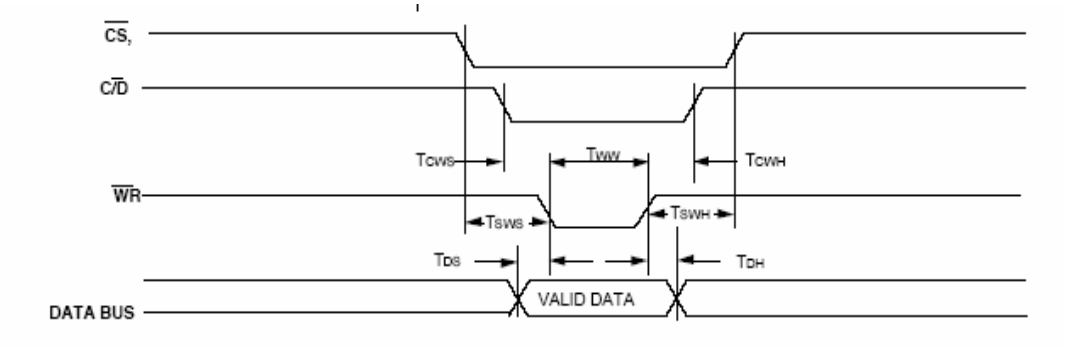
รีจิสเตอร์ภายในของ LS7166 ประกอบไปด้วย QR, OCCR, ICR และ MCR รีจิสเตอร์ซึ่งเป็นรีจิสเตอร์ที่ทำหน้าที่ในการกำหนดการทำงานของ LS7166 ว่าจะให้มีการนับขึ้น หรือนับลง เป็นต้น ส่วนรีจิสเตอร์ OSR เป็นรีจิสเตอร์ที่บอกสถานการณ์ทำงานของ LS7166 ว่ามีการทำงานเป็นแบบใดเช่น การนับเป็นแบบขึ้นหรือลง มีการเกิดการนับเกิน 24 บิตหรือมีการนับลงน้อยกว่าศูนย์หรือไม่เป็นต้น นอกจากนี้แล้ว LS7166 ยังมีรีจิสเตอร์ขนาด 24 บิตอีกสามตัวคือ PR, CNTR และ OL รีจิสเตอร์ที่ทำหน้าที่กำหนดค่าเริ่มต้นของการนับหรือเป็นตัวนับเมื่อมีสัญญาณของการนับเข้ามาทางขาสัญญาณ A,B ซึ่งในการโหลดค่าเริ่มต้นของการนับนั้นค่าเริ่มต้นของการนับจะถูกโหลดผ่านรีจิสเตอร์ PR และจะผ่านไปยังรีจิสเตอร์ CNTR ต่อไป ซึ่งในการโหลดค่าเริ่มต้นในแต่ละครั้งนั้นต้องทำการีเซ็ตค่าของที่อยู่ของ PR ให้เป็นตำแหน่งที่ศูนย์ก่อนทุกครั้ง ส่วนการอ่านค่าของการนับนั้น ค่าของการนับจะถูกโหลดจากรีจิสเตอร์ CNTR ผ่านไปยังรีจิสเตอร์ OL และค่าจากรีจิสเตอร์ OL จะถูกส่งผ่าน Data Bus ออกไปยัง I/O Buffer และถูกส่งออกไปภายนอกต่อไป

การเข้าถึงรีจิสเตอร์ของ LS7166 นั้นสามารถกระทำได้โดยการกำหนดค่าของบิตข้อมูลบิตที่ 6 และบิตที่ 7 ร่วมกับสัญญาณควบคุมซึ่งประกอบไปด้วย C/D\ , RD\, WR\ และ CS\ ซึ่งเป็นไปดังภาพที่ 26

D7	D6	C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Function
X	X	X	X	X	1	Disable Chip for Read or Write
0	0	1	1	⌐	0	Write to Master Control Register
0	1	1	1	⌐	0	Write to Input Control Register
1	0	1	1	⌐	0	Write to Output Control Register
1	1	1	1	⌐	0	Write to Quadrature Control Register
X	X	0	1	⌐	0	Write to Preset Register, then increment Address Counter
X	X	0	⌐	1	0	Read Output Latch, then increment Address Counter
X	X	1	⌐	1	0	Read Output Status Register

ภาพที่ 26 การกำหนดค่าของสัญญาณเพื่อการเข้าถึงรีจิสเตอร์ของ LS7166

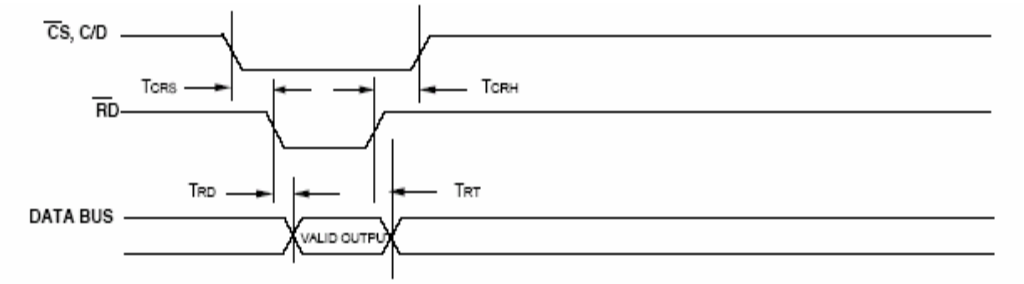
การเขียนข้อมูลลงใน LS7166 นั้นสามารถกระทำได้โดยการกำหนดขาสัญญาณควบคุมของ LS7166 ให้เป็นไปดังภาพที่ 27 ซึ่งขาสัญญาณนั้นประกอบไปด้วย CS\, C/D, WR\, และ Data Bus ขนาด 8 บิต



ภาพที่ 27 การกำหนดสัญญาณเพื่อเขียนข้อมูลลงใน LS166

เมื่อต้องการเขียนข้อมูลลงใน LS7166 จะต้องเริ่มทำการกำหนดค่าของสัญญาณที่ขา Chip Select ให้มีค่าเป็นลอจิกศูนย์ และตามด้วยกำหนดสัญญาณ C/D\ และ WR\ ให้เป็นลอจิกศูนย์ตามลำดับ แล้วจึงทำการส่งข้อมูลออกไปที่ Data Bus ของ LS7166 โดยข้อมูลในบิตที่ 6 และบิตที่ 7 จะเป็นตัวบอกว่าจะทำการเขียนข้อมูลไปยังรีจิสเตอร์ตัวใดในกรณีที่เขียนข้อมูลไปยังรีจิสเตอร์ QR, OCCR, ICR และ MCRส่วนข้อมูลที่เหลือจะเป็นข้อมูลที่นำไปกำหนดค่าภายในรีจิสเตอร์นั้น ๆ ในกรณีที่เขียนข้อมูลลงไปยัง PR รีจิสเตอร์ ค่าของข้อมูลในบิตที่ 6 และบิตที่ 7 จะทำหน้าที่เป็นเพียงบิตข้อมูลธรรมดาเท่านั้น

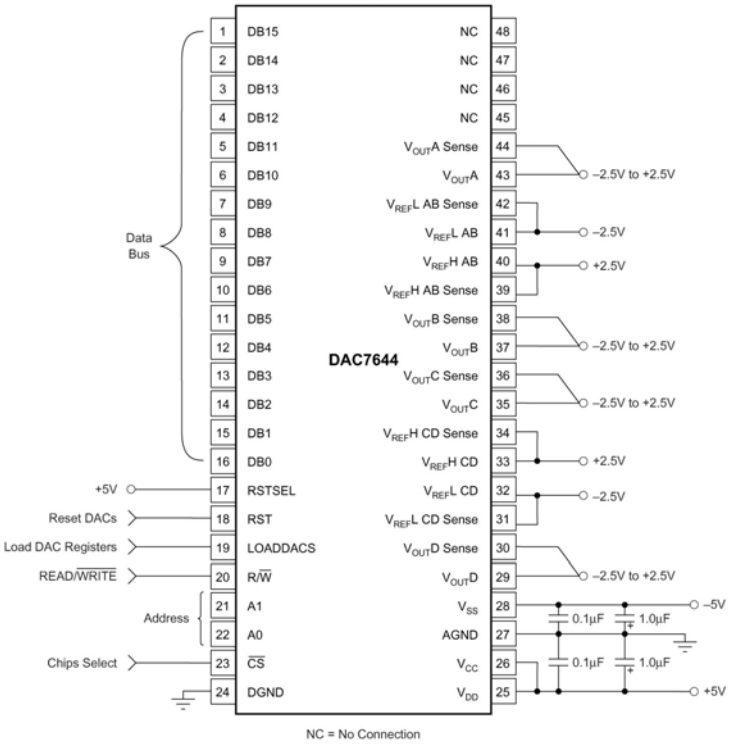
การอ่านข้อมูลจาก LS7166 นั้นสามารถกระทำได้โดยการกำหนดขาสัญญาณควบคุมเช่นเดียวกันกับการเขียนข้อมูลซึ่งขาสัญญาณนั้นจะประกอบไปด้วย CS\, C/D, RD\, และ Data Bus ขนาด 8 บิต ซึ่งแสดงในภาพที่ 28



ภาพที่ 28 แสดงการกำหนดขาสัญญาณของ LS7166 เพื่ออ่านข้อมูล

8. วงจร DAC และวงจรขยายสัญญาณแอนะล็อก

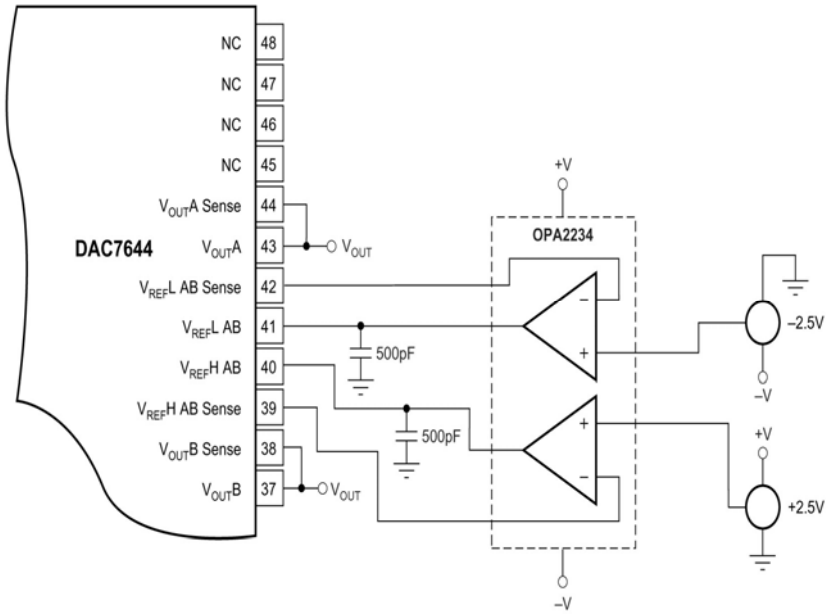
สำหรับวงจร DAC เราจะอาศัยไอซีเบอร์ DAC7644 ของบ. Texas Instruments ที่จะมี 4 แชนเนลอยู่ในตัว ดังภาพที่ 29 ในการออกแบบเราจะให้ DAC7644 ทำงานในแบบ dual-supply operation ดังนั้นจะต้องการไฟเลี้ยง Vcc (และ Vdd) 5 โวลต์ และไฟเลี้ยง Vss ที่ -5 โวลต์



ภาพที่ 29 ชิพ DAC7644 Dual-Supply Operation

สิ่งสำคัญที่ควรสังเกตในภาพที่ 29 คือถึงแม้ว่าขา V_{out} กับ V_{out} Sense ของแต่ละแชนเนลจะต่อกัน แต่จะลากสายไปต่อกันที่จุดใช้งานจริงหรือจุดที่ต่อกับวงจรถัดไป มิใช่ต่อกันที่บริเวณขาของ DAC7644 การทำเช่นนี้จะช่วยลดผลกระทบจากการสูญเสียแรงดันที่ตกคร่อมความต้านทานของลายทองแดง ที่ค่อนข้างมีผลมากสำหรับ DAC 16 บิต ขอให้ดูคำอธิบายจาก datasheet ของ DAC7644 ซึ่งในการออกแบบลายวงจรพิมพ์ เส้นลายทองแดงที่จะต่อจากขา V_{out} กับ V_{out} Sense ไปยังจุดใช้งานจะต้องมีความยาวและขนาดใกล้เคียงกันมากที่สุด เพื่อให้มีค่าความต้านทานใกล้เคียงกัน

นอกจากนั้น เราจะเห็นได้ว่า DAC7644 ต้องการแรงดันอ้างอิง $\pm$ 2.5 โวลต์ ถ้าหากเราใช้ความต้านทานมาต่อกับแบบ voltage divider ค่าความต้านทานทั้งสองตัวไม่ควรมีค่าเกิน 1 กิโลโอห์ม และใช้แบบ 1% เพื่อให้ได้ค่าแรงดันอ้างอิงที่เที่ยงตรง วิธีการที่ดีกว่าคือใช้ออปแอมป์ดังภาพที่ 30

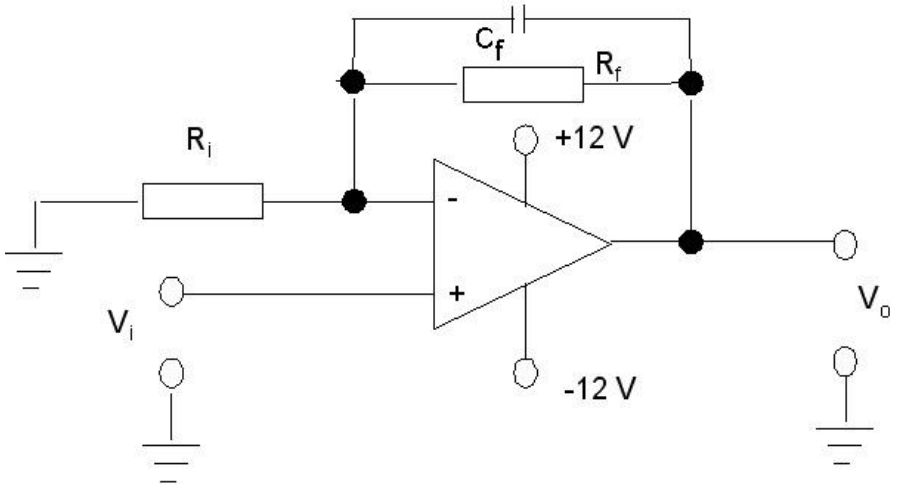


ภาพที่ 30 แสดงการใช้ออปแอมป์สร้างสัญญาณแรงดันอ้างอิงที่เที่ยงตรง

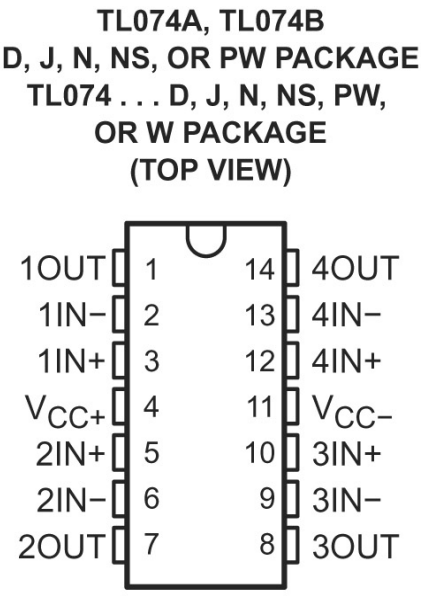
เอาต์พุตที่ได้จาก DAC7644 จะอยู่ในช่วง $\pm$ 2.5 โวลต์ ส่วนภาคอินพุตของชุดขับเซอร์โว โดยทั่วไปจะรับสัญญาณในช่วง $\pm$ 10 โวลต์ ดังนั้นเราจึงต้องการวงจรขยายแรงดัน 4 เท่า ซึ่งสามารถสร้างจากออปแอมป์ได้ดังภาพที่ 31 โดยอัตราขยายเท่ากับ

$$A = 1 + \frac{R_f}{R_i} \tag{17}$$

เราสามารถเลือก R_f = 30 KΩ และ R_i = 10 KΩ ส่วน C_f ใช้ในการฟิลเตอร์สัญญาณรบกวนความถี่สูง เราจะใช้ไอซี ออปแอมป์ เบอร์ TL074 จาก บ.Texas Instruments ที่จะมีออปแอมป์ 4 ตัว ดังแสดงดังภาพที่ 32



ภาพที่ 31 วงจรขยายแรงดันโดยใช้ออปแอมป์

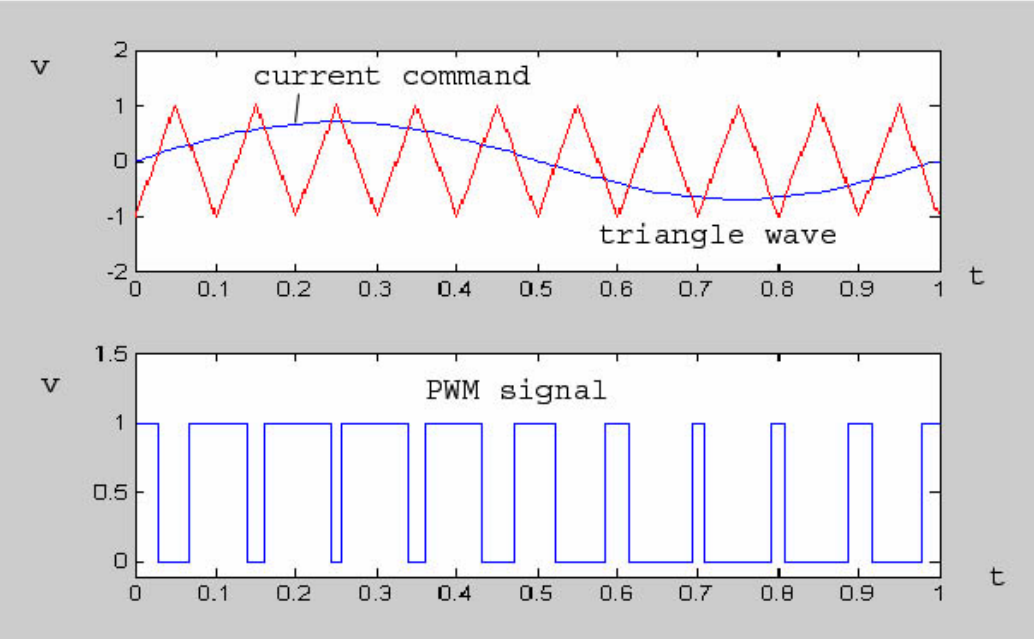


ภาพที่ 32 ชิพ TL074 Quad Bi-FET Op-amp

9. วงจรกำเนิดสัญญาณ PWM

ชุดขับเซอร์โวมอเตอร์ในปัจจุบันจะมีทั้งแบบรับสัญญาณอินพุตแอนะล็อก และแบบรับสัญญาณ PWM โดยทั่วไปแล้วชุดขับแบบ PWM จะมีขนาดเล็ก วงจรไม่ซับซ้อนมากและชุดขับสำหรับมอเตอร์ไฟฟ้ากระแสตรง มีราคาไม่สูงมาก ทำให้เหมาะสมกับการใช้งานด้านการเรียนการสอนระบบควบคุม ดังนั้นเพื่อให้ฮาร์ดแวร์ของเราสามารถรองรับชุดขับแบบนี้ได้ด้วย จึงได้เพิ่มส่วนวงจรกำเนิดสัญญาณ PWM เข้ามา

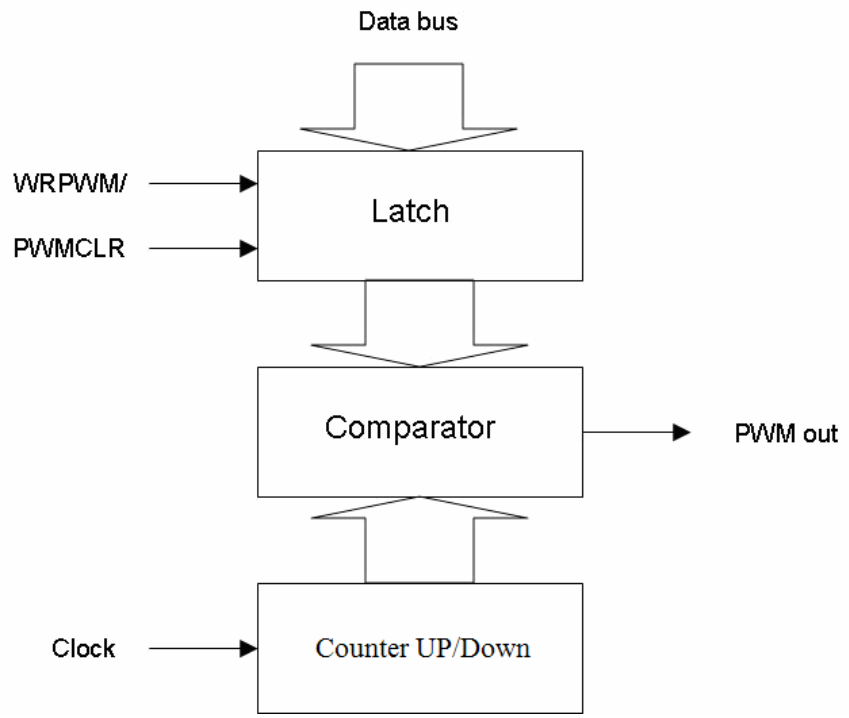
หลักการการสร้างสัญญาณ PWM คือเราจะแปลงรูปคลื่นสัญญาณต่อเนื่องให้เป็นพัลซ์ที่มีความกว้างเป็นสัดส่วนกับระดับของสัญญาณ วิธีการที่นิยมใช้กันแสดงได้ดังภาพที่ 33 คือ เราจะทำการเปรียบเทียบสัญญาณคำสั่งควบคุมกระแสมอเตอร์กับรูปคลื่นสามเหลี่ยมที่สร้างขึ้น และให้แรงดันของสัญญาณ PWM เป็นบวกหรือศูนย์ตามผลการเปรียบเทียบ ดังนั้นจะเห็นว่าเมื่อแรงดันของสัญญาณคำสั่งมีระดับสูง สัญญาณ PWM จะมีความกว้างของพัลซ์บวกมาก ซึ่งเมื่อส่งไปให้สวิทชิ่งทรานซิสเตอร์ก็จะจ่ายกระแสให้มอเตอร์มีค่าเฉลี่ยตามที่เราต้องการ การที่ทรานซิสเตอร์ทำงานในแบบสวิทชิ่งทำให้กำลังสูญเสียในรูปความร้อนมีน้อยกว่าชุดขับแบบเชิงเส้น



ภาพที่ 33 หลักการสร้างสัญญาณ PWM

ในทางปฏิบัติเราสามารถสร้างวงจรกำเนิดสัญญาณ PWM ได้ง่าย ๆ ตามบล็อกไดอะแกรมดังภาพที่ 34 โดยส่งค่าที่ตัวควบคุมคำนวนได้เข้าไปเก็บไว้ในวงจรแลทช์ ในขณะเดียวกันวงจรนับก็จะรับอินพุตสัญญาณนาฬิกาและทำการนับค่าอยู่ตลอดเวลา วงจรเปรียบเทียบจะทำการเปรียบเทียบค่าของคำสั่ง PWM กับค่าในวงจรนับ หากค่าคำสั่ง PWM มากกว่า เอาต์พุตของตัวเปรียบเทียบจะเป็น 1 แต่ถ้าหากค่าคำสั่ง PWM น้อยกว่า เอาต์พุตของตัวเปรียบเทียบจะเป็น 0 เมื่อ counter นับจนเกิดการ overflow ค่า counter จะถูกนับลงแทนจนเป็นศูนย์อีกแล้วจึงจะนับขึ้นอีกครั้ง

สิ่งที่ต้องคำนึงอย่างหนึ่งถึงคือความสัมพันธ์ของจำนวนบิตข้อมูล ความถี่สัญญาณนาฬิกากับ ความถี่ของสัญญาณ PWM ที่ได้ หากเราต้องการความละเอียดที่สูงขึ้นสามารถทำได้โดยการเพิ่มจำนวนบิตของข้อมูล แต่ผลเสียคือวงจรนับจะใช้เวลามากขึ้นกว่าจะครบหนึ่งรอบ นั่นคือความถี่ของเอาต์พุต PWM ที่ได้จะลดลง หากความถี่ PWM อยู่ในช่วงที่หูของมนุษย์สามารถได้ยินก็อาจก่อให้เกิดเสียงดัง หรือหากความถี่ต่ำเกินไปอาจเกิดการสั่นของมอเตอร์ วิธีทางแก้คือเพิ่มความถี่ของสัญญาณนาฬิกา ในวงจรต้นแบบจะใช้ค่าคำสั่ง PWM ขนาด 12 บิต (ไม่รวมบิตแสดงทิศทางการหมุน) และสัญญาณนาฬิกาความถี่ 20 MHz ซึ่งจะทำให้ได้ความถี่ของสัญญาณ PWM เท่ากับ 4883 Hz



ภาพที่ 34 บล็อกไดอะแกรมของวงจรกำเนิดสัญญาณ PWM

10. บอร์ดประมวลผลสัญญาณดิจิทัล (DSP board)

ในการทดลองเราใช้ DSP board รุ่น TMS320C6713 DSK เป็นผลิตภัณฑ์ของบริษัท Spectrum Digital มีลักษณะดังภาพที่ 35 ตัวประมวลผลเป็นแบบตัวประมวลสัญญาณดิจิทัล รุ่น TMS320C6713 ที่เป็นแบบ floating point โดยที่ DSK ย่อมาจาก DSP Starter Kit ชุด C6713 DSK นอกจากตัวประมวลผลแล้วยังประกอบด้วยหน่วยความจำ และมีคอนเนคเตอร์ต่อเชื่อมอุปกรณ์ภายนอก ในชุดยังประกอบด้วยซอฟท์แวร์คอมไพเลอร์ภาษา C ที่มีชื่อเรียกว่า Code Composer Studio การเขียนโปรแกรมจะต้องอาศัยเครื่องคอมพิวเตอร์ โดยจะติดต่อกับ C6713 DSK ผ่านทางพอร์ท USB

ข้อได้เปรียบสำหรับการใช้ C6713 DSK เป็นชุดควบคุม

- ต้นทุนการผลิตต่ำ ราคาของ C6713 DSK ประมาณ US$395 ต่อชุด
- DSP เหมาะกับงานที่ต้องมีการคำนวณทางคณิตศาสตร์มาก ๆ เช่น งานควบคุมการเคลื่อนที่ของมอเตอร์

ข้อเสียเปรียบสำหรับการใช้ C6713 DSK เป็นชุดควบคุม

- การพัฒนาระบบปฏิบัติการเวลาจริง บน DSP ยากกว่าระบบที่ใช้ตัวประมวลผลที่ใช้ในคอมพิวเตอร์ทั่วไป เช่น ของบริษัทอินเทล ที่มีเครื่องมือสนับสนุนมากกว่า
- ไม่มีฮาร์ดแวร์สนับสนุนการแสดงผลบน LCD panel

ภาพที่ 35 ชุดทดลอง TMC320C6713 DSK

อุปกรณ์และวิธีการ

อุปกรณ์

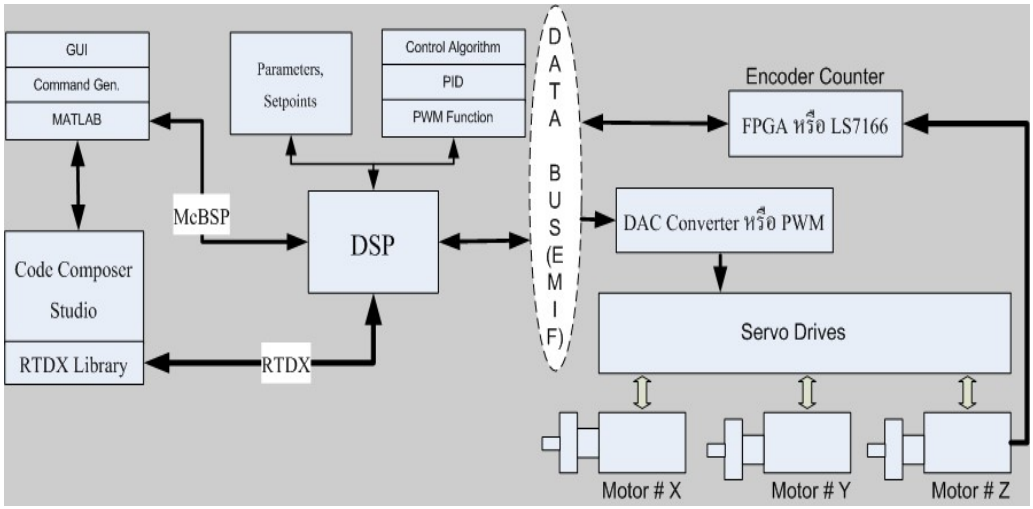
1. คอมพิวเตอร์ 1 ชุด
2. โปรแกรม Matlab และ GUI
3. แท่น XYZ Table (ใช้แทนเครื่อง CNC 3 แกน) 1 ชุด
4. AC servo motor พร้อมชุดขับ 1 ชุด
5. DC servo motor 24 volts พร้อม Encoder 3 ชุด
6. PWM motor drive board 3 ชุด
7. ชุดทดลอง TMC320C6713 DSK
8. Extensive board สำหรับ DC servo motor 1 ชุด
9. Extensive board สำหรับ AC servo motor 1 ชุด
10. ชิพสำหรับอ่าน Encoder LS7166 7 ตัว
11. ชิพสำหรับการแปลง digital to analog DAC7644 1 ตัว
12. Power supply

วิธีการ

1. การทำงานของระบบโดยรวม

เริ่มแรกเราจะมาอธิบายถึงภาพรวมของระบบกันก่อนว่ามีการทำงานอย่างไรบ้าง ในการทำการทดลองนี้เราจะสร้างเส้นทางการเคลื่อนที่โดยผู้ใช้ผ่านทาง GUI ของโปรแกรม Matlab โดยผู้ใช้จะสร้างเส้นทางโดยการป้อนรหัสจี (G-code) เมื่อผู้ใช้ป้อนรหัสจีเรียบร้อยแล้ว Matlab จะส่งข้อมูลเข้าไปที่ส่วนของ trajectory generation เพื่อคำนวณค่าตำแหน่งและเวลาซึ่งใช้สำหรับส่งเป็นค่า set point เมื่อกระบวนการในการคำนวณเรียบร้อยแล้วเราจะส่งค่าตำแหน่งและเวลาทั้งหมดลงไปเก็บที่ buffer ของ DSP ผ่านทาง RTDX จากนั้น DSP ก็จะเริ่มทำงานโดยจะอ่านค่าจาก encoder ของ motor ซึ่งผ่านทางชิพ LS7166 เอามาเปรียบเทียบกับค่าที่ได้คำนวณไว้แล้วทำงานตาม PID Control Algorithm ส่วนของ Control loop นี้ก็จะส่งค่า PWM สำหรับการควบคุมที่ใช้ DC servo motor หรือ

ค่า Voltage ผ่านทางชิพ DAC7644 สำหรับการควบคุมที่ใช้ AC servo motor เพื่อทำงานตามที่ต้องการ การทำงานทั้งหมดนี้แสดงดังภาพที่ 36

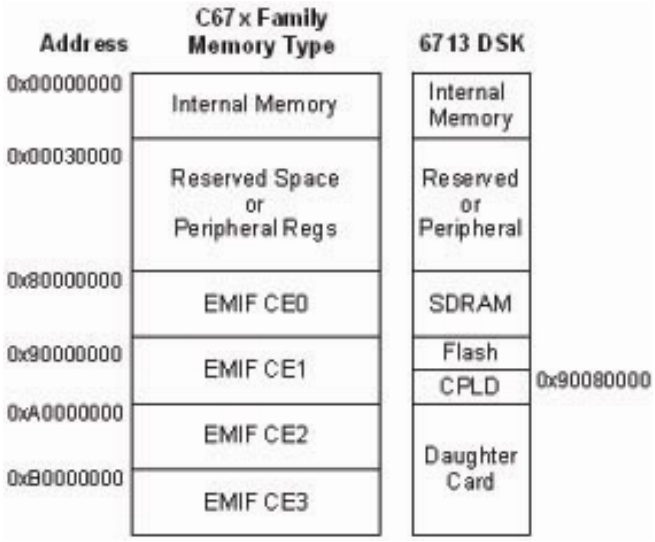


ภาพที่ 36 การทำงานของระบบโดยรวม

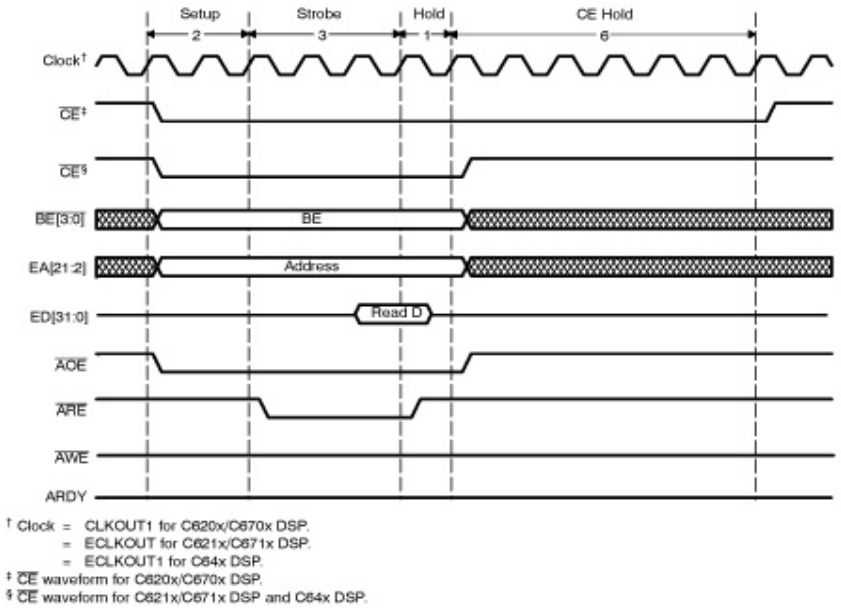
ส่วนของการติดต่อกับอุปกรณ์ภายนอกจาก DSP board มีคอนเนคเตอร์ที่ใช้ต่อประสานหลายรูปแบบ แต่ที่เหมาะสมที่สุดสำหรับงานวิจัยนี้คือ สัญญาณต่อประสานกับหน่วยความจำภายนอก (External memory interface) กล่าวคือซีพียูจะมองเห็นอุปกรณ์ต่าง ๆ เหล่านี้เป็นเสมือนแอดเดรสหนึ่งในหน่วยความจำที่จะสามารถทำการอ่านและเขียนค่าไปได้โดยตรง ดังนั้นเราจำเป็นจะต้องทำการกำหนดตำแหน่งของแอดเดรสให้กับฮาร์ดแวร์ทั้งสามภาคให้เหมาะสม เพื่อมิให้เกิดการชนกันของข้อมูล นอกจากนั้นต้องคำนึงด้วยว่าควรกำหนดแอดเดรสอย่างไรจึงจะง่ายต่อการออกแบบวงจรถอดรหัสของแอดเดรส (address decoder)

ภาพที่ 37 แสดงการจัดตำแหน่งแอดเดรสของซีพียูตระกูล C67x เทียบกับที่กำหนดบนบอร์ด DSK จะเห็นว่าฮาร์ดแวร์ส่วนที่สร้างขึ้นเองจะถูกกำหนดให้ใช้หน่วยความจำในย่านที่ระบุว่า Daughter Card คือตั้งแต่แอดเดรส 0xA0000000 เป็นต้นไป ซึ่งในย่านนี้ยังถูกแบ่งแยกเป็นพื้นที่ย่อย 2 ส่วน คือ CE2 และ CE3 โดยเมื่อ DSP ทำการอ่านเขียนกับย่านหน่วยความจำทั้งสองนี้ก็จะดึงให้สัญญาณ ACE2# และ ACE3# ลงเป็นลอจิกต่ำตามลำดับ ตามไดอะแกรมเวลาในภาพที่ 38 และ 39 เนื่องจากฮาร์ดแวร์ของเรามิได้มีการใช้แอดเดรสของหน่วยความจำสำหรับอุปกรณ์อื่น ๆ นอกเหนือจากการอ่านค่า Encoder และส่งค่า PWM หรือ Voltage ออกไปควบคุมมอเตอร์ ดังนั้นจึง

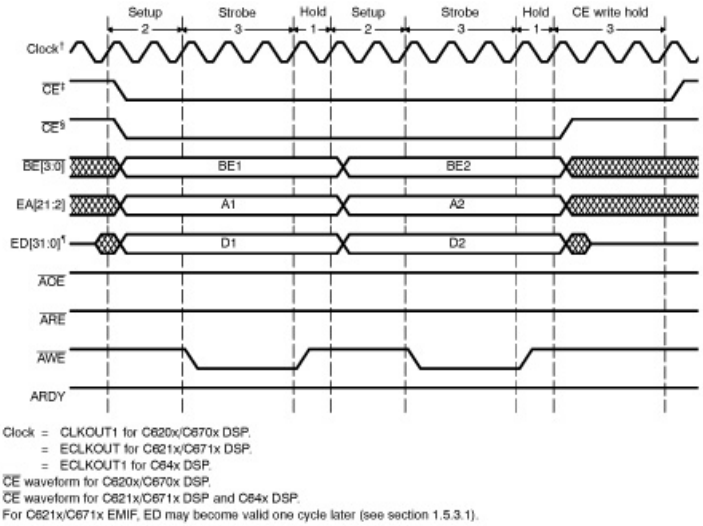
ไม่ต้องการวงจรถอดรหัสที่มีความซับซ้อนมาก เราได้ตกลงเลือกกำหนดย่านหน่วยความจำ CE2 ใช้ในกำหนดควบการส่งข้อมูลในทั้ง 2 ส่วน



ภาพที่ 37 การจัดสรรหน่วยความจำของบอร์ด 6713 DSK



ภาพที่ 38 ไดอะแกรมเวลาในการอ่านข้อมูลจากหน่วยความจำของ C6x DSP



ภาพที่ 39 ไดอะแกรมเวลาในการเขียนข้อมูลลงในหน่วยความจำของ C6x DSP

2. การออกแบบ Extensive Board

Extensive board คือ บอร์ดที่ออกแบบมาเพื่อติดต่อกับ EMIF (External Memory Interface) ซึ่งเป็น data bus ของ DSP (Digital Signal Processor) board

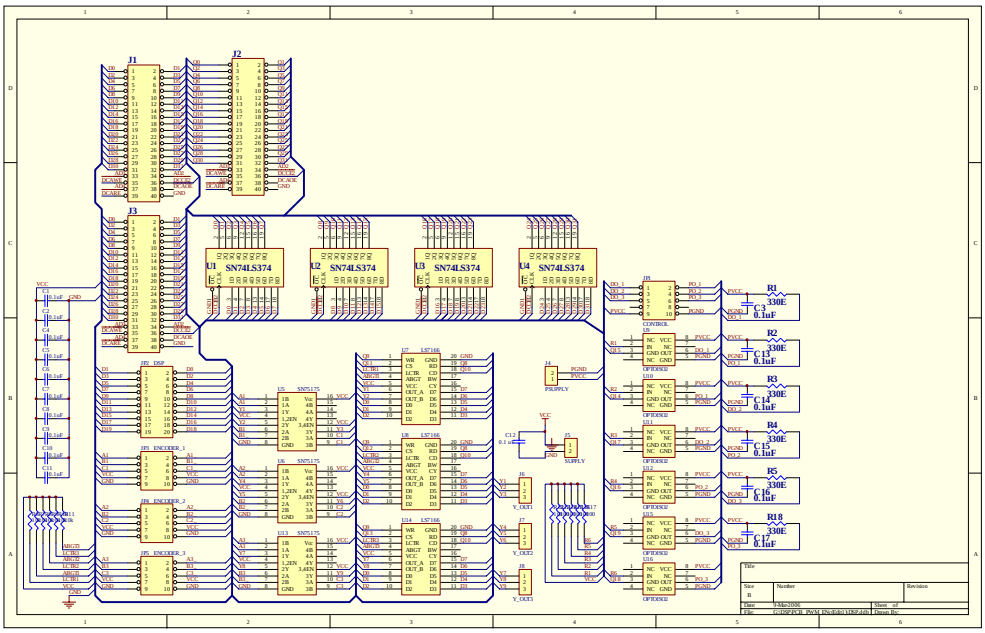
Extensive board จะประกอบไปด้วยส่วนที่ติดต่ออ่านค่าจาก Encoder motor และส่วนที่จะส่งค่าออกไปควบคุม motor โดยเราจะแบ่ง Extensive board ได้เป็น 2 แบบตามที่ได้ออกแบบไว้คือ

2.1 Extensive board สำหรับควบคุมมอเตอร์ไฟฟ้ากระแสตรง

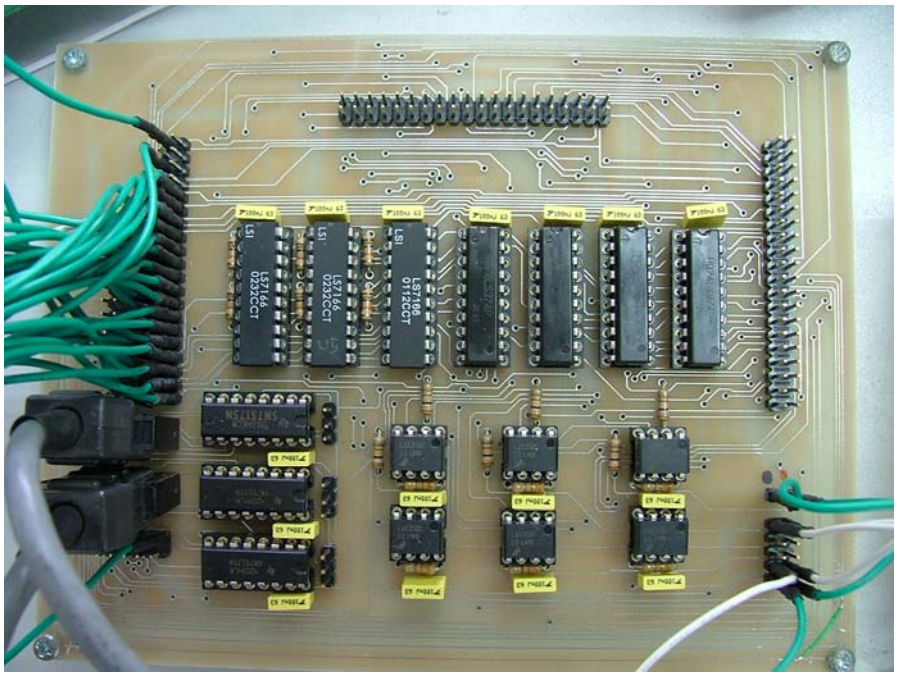
ในบอร์ดจะประกอบไปด้วย

- ชิพสำหรับอ่าน Encoder เบอร์ LS7166 3 ตัว
- ชิพสำหรับการค้างค่าของข้อมูล 74LS374 4 ตัว
- ชิพ Isolator สำหรับส่งค่า PWM เบอร์ 6N137 จำนวน 6 ตัว
- ชิพ Differential Line Receiver เบอร์ SN75175 หรือ AM26LS32 4 ตัว

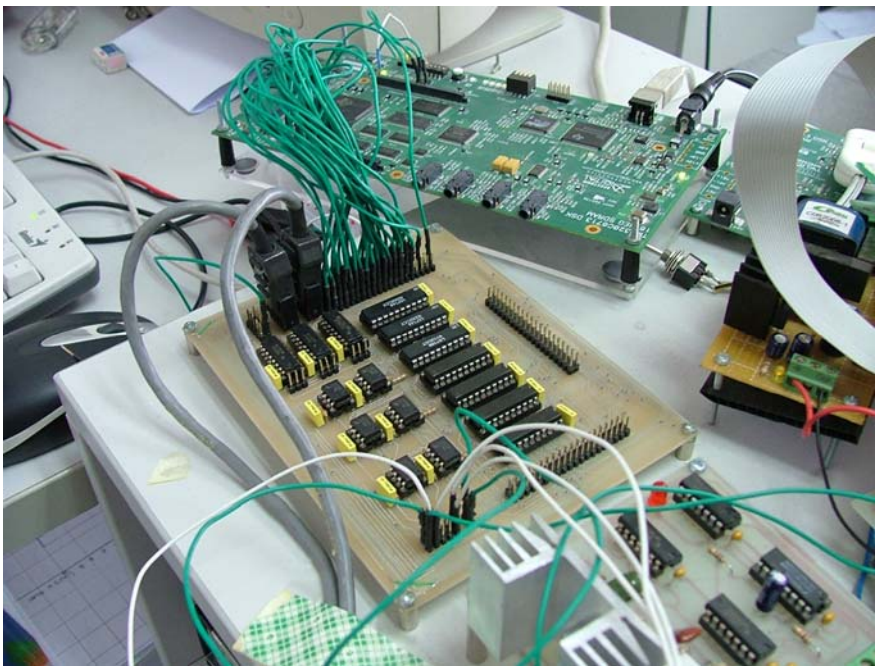
ซึ่งแสดงดังภาพที่ 40 และ 41 ส่วนภาพที่ 42 จะแสดงการเชื่อมต่อระหว่าง DSP board และ Extensive board สำหรับ DC servo motor



ภาพที่ 40 Schematic ของ Extensive board สำหรับ DC servo motor



ภาพที่ 41 Extensive board สำหรับ DC servo motor



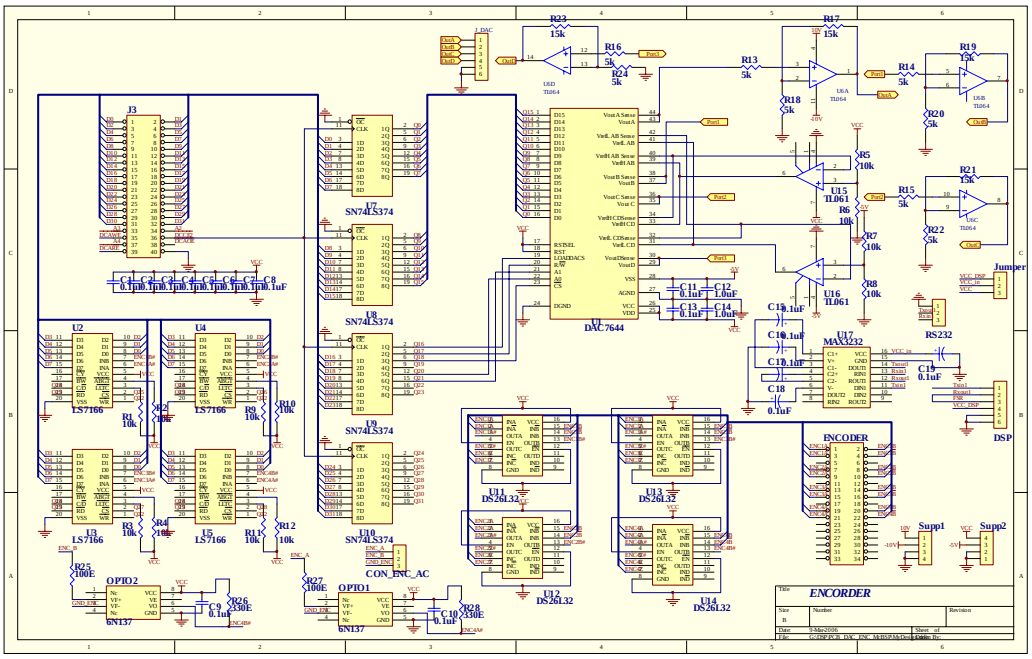
ภาพที่ 42 การเชื่อมต่อระหว่าง DSP board และ Extensive board สำหรับ DC servo motor

2.2 Extensive board สำหรับควบคุมมอเตอร์ไฟฟ้ากระแสสลับ

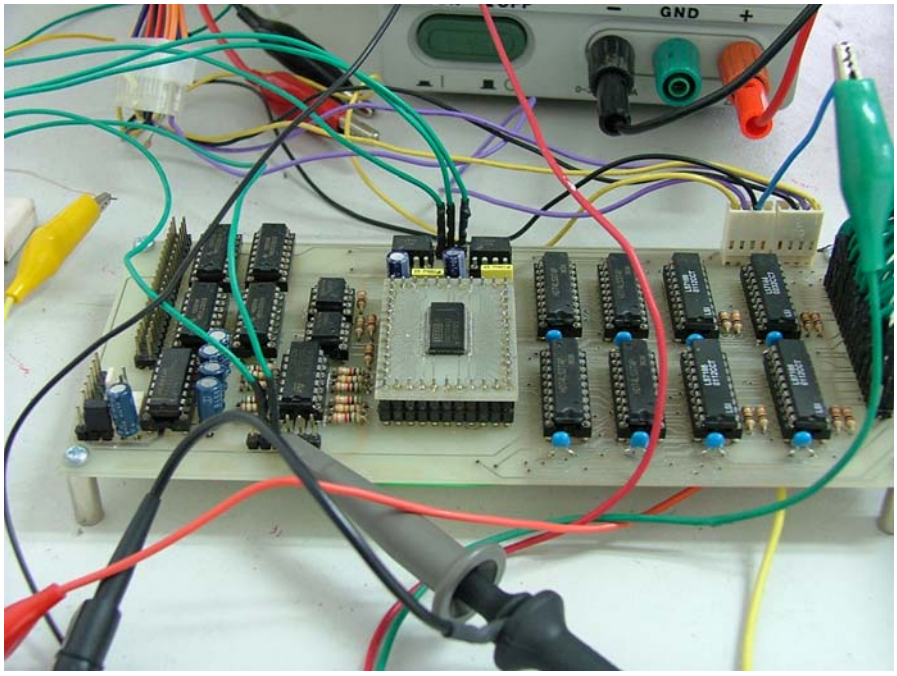
ในบอร์ดจะประกอบไปด้วย

- ชิพสำหรับอ่าน Encoder เบอร์ LS7166 4 ตัว
- ชิพสำหรับการค้างค่าของข้อมูล 74LS374 4 ตัว
- ชิพ Isolator สำหรับรับสัญญาณ Encoder AC servo motor เบอร์ 6N137 จำนวน 2 ตัว
- ชิพ Differential Line Receiver เบอร์ SN75175 หรือ AM26LS32 4 ตัว
- ชิพแปลง Digital to Analog เบอร์ DAC7644 1ตัว
- ชิพสำหรับการติดต่อสื่อสารแบบอนุกรม MAX3232 1 ตัว

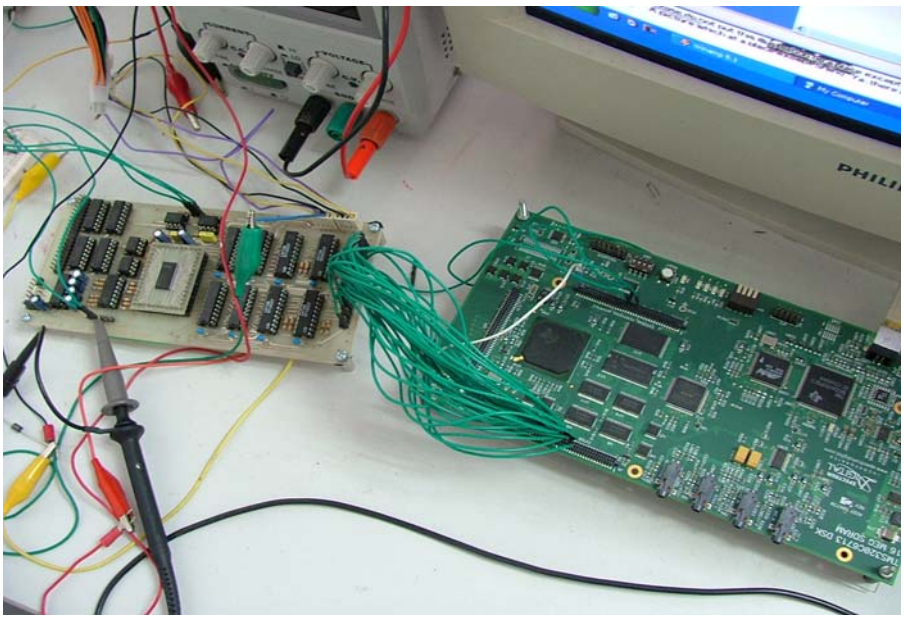
ซึ่งแสดงดังภาพที่ 43 และ 44 ส่วนภาพที่ 45 จะแสดงการเชื่อมต่อระหว่าง DSP board และ Extensive board สำหรับ AC servo motor



ภาพที่ 43 Schematic ของ Extensive board สำหรับ AC servo motor



ภาพที่ 44 Extensive board สำหรับ AC servo motor



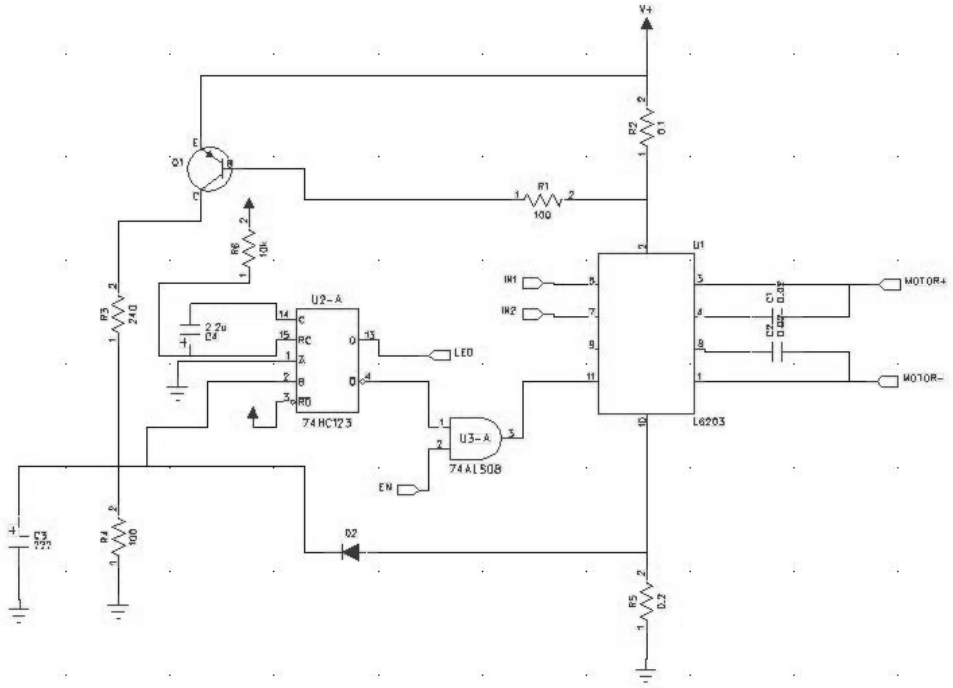
ภาพที่ 45 การเชื่อมต่อระหว่าง DSP board และ Extensive board สำหรับ AC servo motor

3. Motor Drive

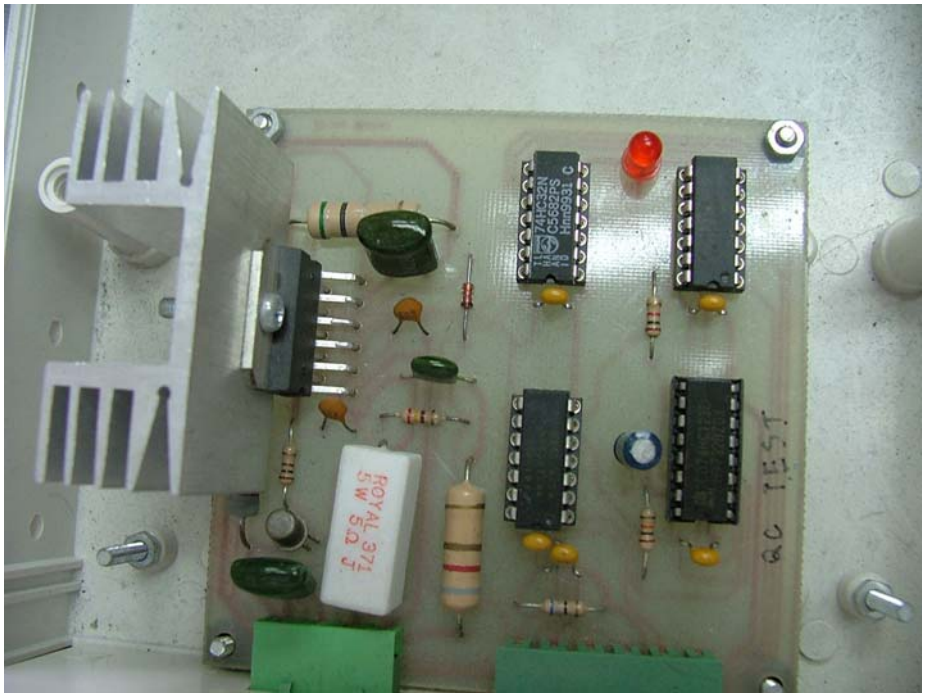
ในส่วนนี้เราจะกล่าวถึงชุดขับ DC servo motor และ AC servo motor โดยชุดขับ DC servo motor จะเป็นการออกแบบโดยใช้ชิพสำเร็จรูป L6203 และชุดขับ AC servo motor จะเป็ยชุดขับสำเร็จรูปที่มาพร้อมกับ AC servo motor ของบริษัท Panasonic

3.1 DC motor drive

ชุดขับ DC servo motor จะทำงานโดยรับค่า PWM ในการควบคุมโดยผ่านชิพสำเร็จรูป L6203 ซึ่งจะสามารถขับมอเตอร์ด้วยแรงดันสูงสุดที่ 48 volt และให้กระแสสูงสุดได้ที่ 4 A โดยในรูปที่ 46 จะแสดงให้เห็น schematic ของ DC servo motor drive และในรูปที่ 47 เป็น DC servo motor drive ของจริง



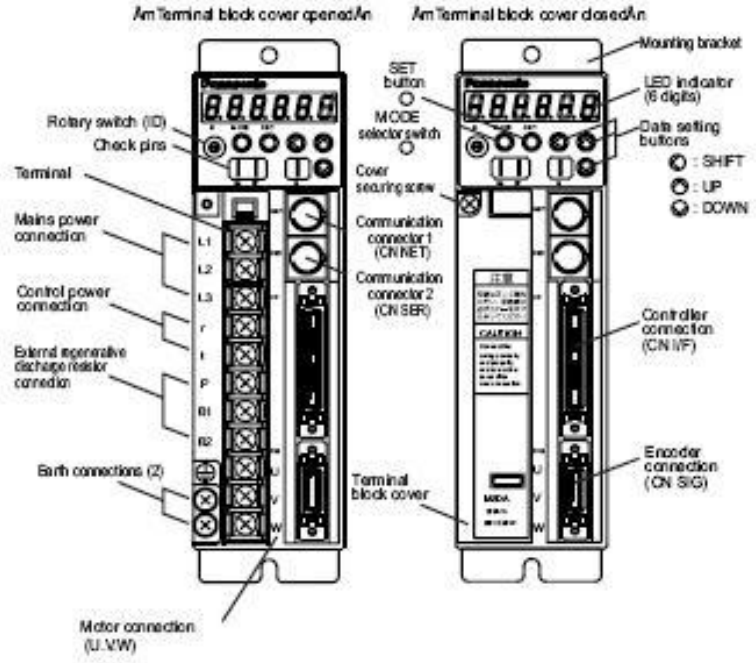
ภาพที่ 46 Schematic ของ DC servo motor drive



ภาพที่ 47 บอร์ดขับเคลื่อนมอเตอร์โดยใช้ PWM

3.2 AC motor drive

ชุดขับ AC servo motor จะทำงานโดยจะรับค่าแรงดันระหว่าง -10 ถึง +10 volt เข้ามาในชุดขับที่มาพร้อมกับ AC servo motor ของบริษัท Panasonic เพื่อมาแปลงเป็นแรงดันที่จะไปขับ AC servo motor โดยรูปที่ 15 เป็น AC servo motor พร้อมกับ drive ของจริง และรูปที่ 48 จะแสดงให้เห็น รายละเอียดหน้าตาของ AC servo motor drive



ภาพที่ 48 รายละเอียดของ AC servo motor drive

4. Control Algorithm

ในส่วนนี้เราจะกล่าวถึงการออกแบบ Control Algorithm ซึ่งจะประกอบด้วยส่วนของ PID Controller รวมถึงส่วนที่ใช้ในการสร้างเส้นทาง (Path Generation) และคำสั่งรหัสจี (G-code)

4.1 การออกแบบ PID บน DSP board

ตามทฤษฎีของระบบควบคุมดิจิทัล อัลกอริธึมที่ใช้ในการควบคุมมอเตอร์จะต้องถูกเรียกทำงานด้วยอัตราการสุ่มที่แน่นอน จึงจำเป็นต้องมีระบบปฏิบัติการที่จะสามารถสนับสนุนการทำงานในลักษณะนี้ หรืออย่างน้อยต้องอาศัยซอฟท์แวร์ที่เรียกว่า แก่นเวลาจริง (real-time kernel) ในระบบขนาดเล็กเช่นงานวิจัยนี้เราสามารถทำการพัฒนาแก่นเวลาจริงขึ้นเองโดยใช้ภาษาที่อยู่ในระดับต่ำ เช่น ภาษาซีหรือแอสเซมบลี อย่างไรก็ตามบริษัทผู้ผลิตตัวประมวลสัญญาณดิจิทัลชั้นนำส่วนใหญ่จะเล็งเห็นถึงความสำคัญของการทำงานแบบเวลาจริงและจัดทำเครื่องมือช่วยพัฒนาให้กับผู้ใช้งาน ในกรณีของบริษัทเทกซัส อินสตรูเม้นท์ เราสามารถใช้เครื่องมือที่เรียกว่า ดีเอสพีไบออส (DSP-bios) ที่เป็นส่วนหนึ่งของชุดพัฒนา CCS (Code Composer Studio) อยู่แล้ว โดยการใช้เครื่องมือดังกล่าว การกำหนดอัตราการสุ่มของเซอร์โวและการจัดลำดับความสำคัญของงานสามารถกระทำได้โดยผ่านส่วนติดต่อผู้ใช้งานแบบกราฟฟิค

ในการออกแบบ PID Controller นั้นเราจะออกแบบให้ทำงานด้วยวิธี Interrupt โดยจะตั้งค่าที่ Software Interrupt ใน DSP-bios ให้ทำงานทุกๆ 3 มิลลิวินาที โดยจะการทำงานที่การอ่านค่า encoder, c(n) จริงมาเปรียบเทียบกับค่าที่ต้องการเคลื่อนที่, r(n) ซึ่งได้จากการคำนวณโดย path generation บนโปรแกรม Matlab และถูกส่งมาเก็บที่ buffer ของ DSP board ตามสมการที่ (18)

$$e(n) = r(n) - c(n) \tag{18}$$

โดย e(n) เป็นค่าความผิดพลาดจากผลต่างของ r(n) และ c(n) จากนั้นจะนำค่า e(n) สมการที่ (19) เพื่อคำนวณหาค่า m(n)

$$m(n) = m(n-1) + K_0 e(n) + K_1 e(n-1) + K_2 e(n-2) \tag{19}$$

ขั้นตอนต่อมาเราจะส่งค่า m(n) ให้กับวงจรสร้าง PWM หรือวงจร DAC เพื่อที่จะควบคุม motor จากนั้นเราก็จะ update ค่าจาก

$$e(n-2) = e(n-1)$$
$$e(n-1) = e(n)$$
$$m(n-1) = m(n)$$

เมื่อทำการ update ค่าแล้วเราก็จะวนกลับไปอ่านค่า encoder, c(n) เพื่อมาคำนวณค่าความผิดพลาด, e(n) ในสมการที่ (18) อีกครั้ง จะวนลูปทำไปเรื่อยๆจนกว่าจะจบโปรแกรม

ค่าของ K_0, K_1, K_2 หาได้จากสมการที่ (20)

$$K_0 = K_p + K_i T + \frac{K_d}{T}$$
$$K_1 = -K_p - \frac{2K_d}{T} \qquad (20)$$
$$K_2 = \frac{K_d}{T}$$

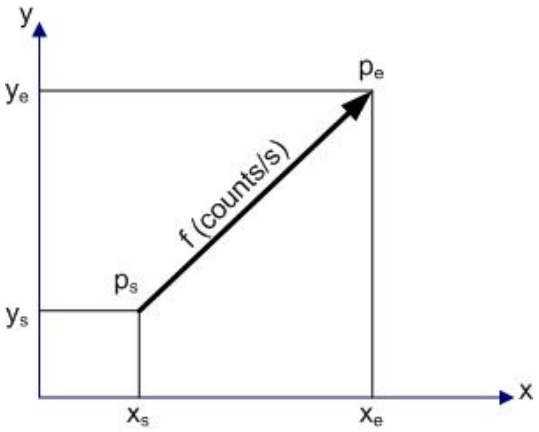
โดยค่า T คือค่าที่เราตั้งค่าในการเข้าทำ Interrupt 3 มิลลิวินาที เราเลือกค่า K_p เป็น 60000, ค่า K_i เป็น 57000 และค่า K_d เป็น 10

5. Path Generation

การกำเนิดคำสั่งการเคลื่อนที่ไปยังจุดที่กำหนดโดยการแปลงคำสั่งจากผู้ใช้เช่น คำสั่งรหัสจี เป็นสัญญาณคำสั่งที่จะทำการสั่งให้ตัวควบคุมซึ่งมีลักษณะเป็นแถวของจุดข้อมูลตำแหน่งที่เรียกว่า จุดคำสั่ง (setpoint) โดยโปรแกรมที่ทำการกำเนิดจุดตั้งค่าเรียกว่า ส่วนกำเนิดคำสั่ง (command generator) โดยปกติแล้วส่วนใหญ่ของรูปทรงต่างๆจะประกอบไปด้วยส่วนของเส้นตรง และเส้นโค้ง ดังนั้นเราจะกล่าวถึงวิธีการสร้างจุดค่าตั้งเป็นเส้นตรงแบบ 2 มิติ และสร้างจุดค่าตั้งเป็นเส้นโค้ง

5.1 การสร้างจุดค่าตั้งเป็นเส้นตรง (Linear interpolate algorithm)

สมมุติเราต้องการเคลื่อนที่เป็นเส้นตรงจากจุด $p_s(x_s,y_s)$ ไปยังจุด $p_e(x_e,y_e)$ แสดงดังภาพที่ 49 ที่ 1 ณ เวลา t



ภาพที่ 49 เส้นทางการเคลื่อนที่จาก Ps ไป Pe

ตำแหน่งของแกนทั้ง 2 จะเป็น

$$x(t) = x(k * \Delta t) = x_s + \int_0^t \mathrm{f_x}(t)dt \tag{21}$$

$$y(t) = y(k * \Delta t) = y_s + \int_0^t \mathrm{f_y}(t)dt \tag{22}$$

โดย $\mathrm{f_x}$, $\mathrm{f_y}$ คือความเร็ว ณ ช่วงเวลาที่ $\mathrm{T_i}$

ปกติแล้วเราจะแบ่งจุดค่าตั้งบนเส้นตรงที่ต้องการเคลื่อนที่ไปเป็นจำนวน N ค่า โดยเราสามารถเขียนสมการที่ (21) และ (22) ในรูปแบบสมการ discrete คือ

$$x(k) = x_s + \sum_{j=1}^{k} f_x(j)T_i(j) = x_s + \sum_{j=1}^{k-1} f_x(j)T_i(j) + f_x(k)T_i(k) \tag{23}$$

$$y(k) = y_s + \sum_{j=1}^{k} f_y(j)T_i(j) = y_s + \sum_{j=1}^{k-1} f_y(j)T_i(j) + f_y(k)T_i(k) \tag{24}$$

หรือ

$$x(k) = x(k-1) + f_x(k)T_i(k) \tag{25}$$

$$y(k) = y(k-1) + f_y(k)T_i(k) \tag{26}$$

โดยความเร็วแต่ละแกน ณ ช่วงเวลาที่ k คือ

$$f_x(k) = \frac{\Delta x}{T_i(k)}, \quad f_y(k) = \frac{\Delta y}{T_i(k)} \tag{27}$$

ระยะระหว่างจุดค่าตั้งของแต่ละแกนจะเป็นค่าคงที่และมีค่าเป็น

$$\Delta x = \frac{x_e - x_s}{N} \ , \ \Delta y = \frac{y_e - y_s}{N} \tag{28}$$

ถ้าเรารวมสมการที่ (23), (24), (25), (26), (27) และ (28) จะได้เป็น

$$\begin{aligned} x(k * \Delta t) &= x_s + k * \Delta x = x(k-1) + \Delta x \\ y(k * \Delta t) &= y_s + k * \Delta y = y(k-1) + \Delta y \end{aligned} \tag{29}$$

ในการคำนวณจุดค่าตั้งเราสามารถออกแบบมาในรูปของโปรแกรมคำนวณโดยมีตัวแปรดังนี้

T_i(k)	คือ เวลาระหว่างจุดค่าตั้งที่ k และ k-1
f	คือ ความเร็วสูงสุดที่ต้องการ
x_s , y_s	คือ ตำแหน่งเริ่มต้น
x_e , y_e	คือ ตำแหน่งสุดท้าย
δx	คือ ระยะที่ต้องการเคลื่อนที่ในแกน x
δy	คือ ระยะที่ต้องการเคลื่อนที่ในแกน y
$sign\_x$	คือ ทิศทางเคลื่อนที่ในแกน x
$sign\_y$	คือ ทิศทางเคลื่อนที่ในแกน y
N	คือ จำนวนจุดค่าตั้งในระยะการเคลื่อนที่
dx , dy	คือ ระยะห่างระหว่างจุดค่าตั้ง
x_{rem}	คือ ค่าที่เหลือที่เกิดจากการแบ่งจุดค่าตั้งในแกน x
y_{rem}	คือ ค่าที่เหลือที่เกิดจากการแบ่งจุดค่าตั้งในแกน y
Δu	คือ ระยะห่างระหว่างจุดที่ถูกแบ่ง (set point)

เราจะคำนวณค่าเริ่มต้นของตัวแปรได้ดังนี้

δx = abs(x_e-x_s)

δy = abs(y_e-y_s)

sign_x = sign(x_e-x_s)

sign_y = sign(y_e-y_s)

Δu = f*T_i

$N = \sqrt{\delta x^2 + \delta y^2} \Big/ \Delta u$

dx = fix(δx /N)

dy = fix(δy /N)

x_{rem} = δx - (dx*N)

y_{rem} = δy - (dy*N)

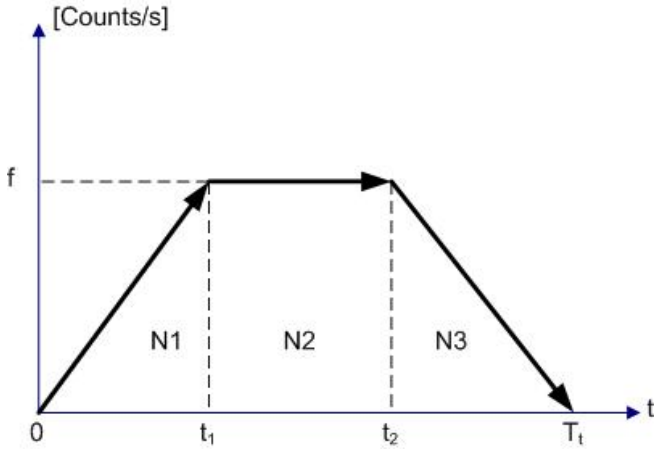
line(x_s,dx,x_{rem},sign_x,N) สร้างจุดค่าตั้งของแกน x

line(y_s,dy,y_{rem},sign_y,N) สร้างจุดค่าตั้งของแกน y

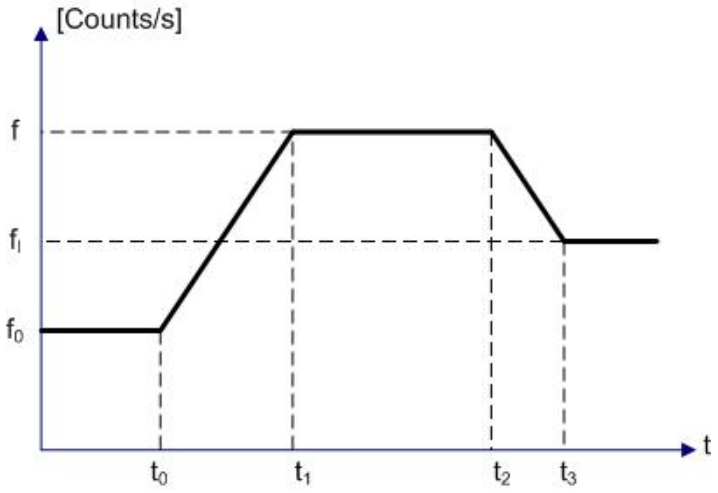
หลังจากคำนวณค่าเริ่มต้นของตัวแปรแล้วเราจะสามารถคำนวณจุดค่าตั้งในแต่ละช่วงเวลา T_i ได้ดังนี้

```
function line(x_s,dx,x_rem,sign_x,N)
x(1) = x_s
x_error = 0
for i=0:N+1
        x(i)   = x(i-1) + sign_x*dx
        x_error = x_error + x_rem
        if (x_error >= N)
                x(i) = x(i) + sign_x
                x_error = x_error –N
        end
end
```

ในการหาเวลา T_i ระหว่างแต่ละจุดค่าตั้งหาได้โดยเราจะแปลงจุดค่าตั้งเป็นความเร็วโดย ความเร็วที่เราได้รับนั้นจะเรียกว่า โพรไฟล์ความเร็ว (velocity profile) มีลักษณะดังภาพที่ 50, 51



ภาพที่ 50 โพรไฟล์ความเร็วแบบความเร็วเริ่มต้นเป็นศูนย์



ภาพที่ 51 โพรไฟล์ความเร็วแบบความเร็วเริ่มต้นไม่เป็นศูนย์

โดยจากรูปจะเห็นว่าเราจะแบ่งการเคลื่อนที่ได้เป็น 3 ช่วงคือช่วง N1 ที่มีความเร่ง A, ช่วง N2 ที่มีความเร็วคงที่และช่วง N3 ที่มีความหน่วง D โดยค่าผลรวมของ N1, N2 และ N3 จะเท่ากับ N เราจะเขียนเป็นสมการหาค่า T_i คือ

$$T_i(k) = \frac{2\Delta u}{f(k) + f(k-1)} \tag{30}$$

ดังนั้นเราสามารถเขียนเป็นรูปแบบโปรแกรมคำนวณหาค่า T_i ได้ดังนี้

ช่วง N1

for k=1:N1

$$f(k) = \sqrt{f_0^{\,2} + 2kA\Delta u}$$

$$T_i(k) = \frac{2\Delta u}{f(k) + f(k-1)}$$

end

ช่วง N2

for k=1:N2

$$T_i(k) = \frac{\Delta u}{f_{\max}}$$

end

ช่วง N3

for k=1:N3

$$f(k) = \sqrt{f_0^{\,2} - 2kD\Delta u}$$

$$T_i(k) = \frac{2\Delta u}{f(k) + f(k-1)}$$

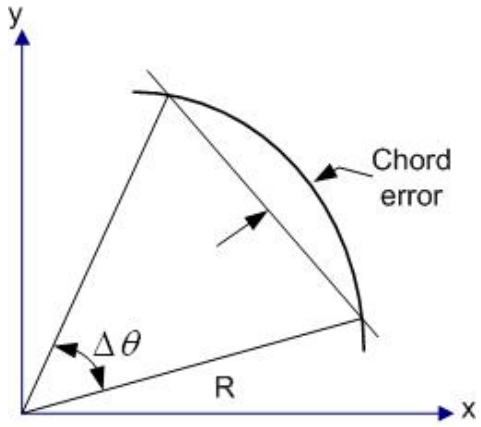
end

ดังนั้นเราจึงสามารถหาจุดค่าตั้งและเวลาแต่ละช่วงจุดค่าตั้งของแต่ละแกนเพื่อส่งไปควบคุมตำแหน่งของแกน x,y ในเครื่องจักร CNC ได้

5.2 การสร้างจุดค่าตั้งเป็นเส้นโค้ง (Circular interpolate algorithm)

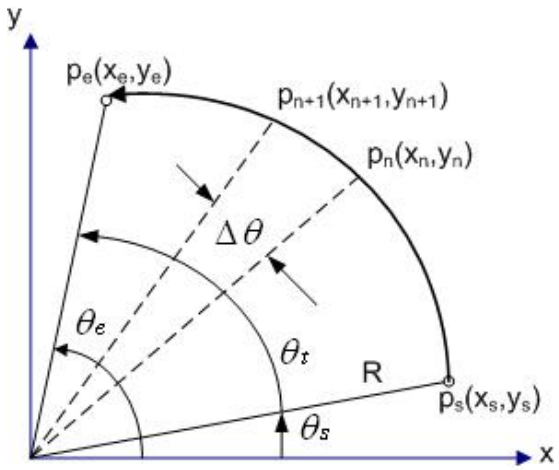
เราจะพิจารณาส่วนของเส้นโค้งที่มีจุดศูนย์กลางอยู่ที่จุดกำเนิดของระบบ CNC แสดงดังภาพที่ 52 โดยความยาวของเส้นโค้งคือ

$$L = R(\theta_e - \theta_s) = R\Delta\theta \tag{31}$$



ภาพที่ 52 ส่วนโค้งที่มีมุม $\triangle\theta$

เราต้องแบ่งเส้นโค้งออกเป็นส่วนย่อย N ส่วนโดยมีความยาวแต่ละส่วนเป็น $\triangle u$ และทำมุมเท่ากับ $\triangle\theta$ แสดงดังภาพที่ 53



ภาพที่ 53 เส้นโค้งที่ถูกแบ่งย่อยเป็นมุม $\triangle\theta$

โดยการแบ่งเส้นโค้งออกเป็น N ส่วนนั้นจะต้องทำให้ค่า chord error ไม่มากกว่าค่าที่ยอมรับได้คือ ไม่มากกว่า 1 เราสามารถเขียนเป็นสมการได้คือ

$$chord\_error = R(1-\cos\frac{\triangle\theta}{2}) \le 1 \tag{32}$$

เราสามารถเขียนให้อยู่ในรูป $\triangle\theta$ ได้คือ

$$\triangle\theta \le 2\cos^{-1}(\frac{R-1}{R}) \tag{33}$$

เราจะเลือกค่า $\triangle\theta$ ให้มีค่าเท่ากับ

$$\triangle\theta = \cos^{-1}(\frac{R-1}{R}),\ \cos\triangle\theta = \frac{R-1}{R} \tag{34}$$

ความสัมพันธ์ของ $\triangle u$ และ $\triangle\theta$ คือ

$$\triangle u = R\triangle\theta \tag{35}$$

โดย $\triangle u$ คือ ระยะห่างระหว่างจุดที่ถูกแบ่ง ช่วงเวลาในระหว่างจุดค่าตั้งเราสามารถหาได้จากสมการที่ (30) และการคำนวณความเร็วจะถูกแบ่งในรูปแบบโพรไฟล์ความเร็วเป็น 3 ช่วง คือ N1, N2 และ N3 แสดงดังภาพที่ 50 โดยคำนวณจาก

$$N_1 = \frac{f_{\max}^{\ 2}}{2A\triangle u},\ N_3 = \frac{f_{\max}^{\ 2}}{2D\triangle u},\ N_2 = N - N_1 - N_3 \tag{36}$$

โดยความเร็วในการเคลื่อนที่หาได้จาก

$$f(k) = \sqrt{f_0^{\ 2} + 2kA\triangle u} \qquad \text{, สำหรับช่วง N1} \tag{37}$$

$$f(k) = f_{\max} \qquad \text{, สำหรับช่วง N2} \tag{38}$$

$$f(k) = \sqrt{f_0^{\ 2} \text{-} 2kD\triangle u} \qquad \text{, สำหรับช่วง N3} \tag{39}$$

เมื่อเราสามารถหาความเร็วในแต่ละช่วงเวลาได้แล้วเราจะสามารถหาความเร็วเชิงมุม, $\omega(t)$ และ ตำแหน่งเชิงมุม ,$\theta(t)$ ณ เวลา t ได้

$$\omega(t) = \frac{f(t)}{R},\ \theta(t) = \omega(t) * t = \frac{f(t)}{R}t \tag{40}$$

เราสามารถหาค่าตำแหน่งบนแกน x, y เมื่อเคลื่อนที่ทิศทางทวนเข็มนาฬิกาคือ

$$\begin{aligned} x(t) &= R\cos\theta(t) = R\cos(\frac{f(t)}{R}t) \\ y(t) &= R\sin\theta(t) = R\sin(\frac{f(t)}{R}t) \end{aligned} \tag{41}$$

และความเร็วแต่ละแกนคือ

$$f_x = \frac{dx}{dt} = -\frac{f(t)}{R} R\sin(\frac{f(t)}{R}t) = -\frac{f(t)}{R} y(t)$$
$$f_y = \frac{dy}{dt} = \frac{f(t)}{R} R\cos(\frac{f(t)}{R}t) = \frac{f(t)}{R} x(t) \quad (42)$$

ในกรณีเคลื่อนที่ตามเข็มนาฬิกาจะได้เป็น

$$x(t) = R\cos(-\theta(t)) = R\cos(\frac{f(t)}{R}t)$$
$$y(t) = R\sin(-\theta(t)) = -R\sin(\frac{f(t)}{R}t) \quad (43)$$

และความเร็วแต่ละแกนคือ

$$f_x = \frac{dx}{dt} = -\frac{f(t)}{R} R\sin(\frac{f(t)}{R}t) = \frac{f(t)}{R} y(t)$$
$$f_y = \frac{dy}{dt} = -\frac{f(t)}{R} R\cos(\frac{f(t)}{R}t) = -\frac{f(t)}{R} x(t) \quad (44)$$

เราพิจารณาระหว่างของจุดค่าตั้งใดๆบนเส้นโค้งโดยทำมุม $\triangle\theta$ แสดงดังภาพที่ 53

$$x(n) = R\cos(\theta_s + n\triangle\theta)$$
$$y(n) = R\sin(\theta_s + n\triangle\theta) \quad (45)$$

$$x(n+1) = R\cos(\theta_s + (n+1)\triangle\theta)$$
$$y(n+1) = R\sin(\theta_s + (n+1)\triangle\theta) \quad (46)$$

เราพิจารณาค่าของ x(n+1) สามารถเขียนใหม่ได้เป็น

$$x(n+1) = R\cos(\theta s + n\triangle\theta + \triangle\theta)$$
$$y(n+1) = R\sin(\theta s + n\triangle\theta + \triangle\theta) \quad (47)$$

หรือ

$$x(n+1) = R\cos(\theta s + n\triangle\theta)\cos\triangle\theta - R\sin(\theta s + n\triangle\theta)\sin\triangle\theta$$
$$y(n+1) = R\sin(\theta s + n\triangle\theta)\cos\triangle\theta + R\cos(\theta s + n\triangle\theta)\sin\triangle\theta \quad (48)$$

เราสามารถแตกพจน์การคูณกันของ cos กับ sin และ sin กับ sin ออกมาเป็นผลบวกได้เป็น

$$R\sin(\theta s + n\triangle\theta)\sin\triangle\theta = \frac{1}{2}\{\cos[\theta s + (n-1)\triangle\theta] - \cos[\theta s + (n+1)\triangle\theta]\}$$
$$R\cos(\theta s + n\triangle\theta)\sin\triangle\theta = \frac{1}{2}\{-\sin[\theta s + (n-1)\triangle\theta] + \sin[\theta s + (n+1)\triangle\theta]\} \quad (49)$$

เมื่อรวมสมการกลับไปจะได้เป็น

$$x(n+1) = R\cos(\theta s + n\triangle\theta)\cos\triangle\theta - \frac{R}{2}\cos[\theta s + (n-1)\triangle\theta] + \frac{R}{2}\cos[\theta s + (n+1)\triangle\theta]$$
$$y(n+1) = R\sin(\theta s + n\triangle\theta)\cos\triangle\theta - \frac{R}{2}\sin[\theta s + (n-1)\triangle\theta] + \frac{R}{2}\sin[\theta s + (n+1)\triangle\theta] \quad (50)$$

เรานำค่า x_n ในสมการที่ (45), (46) แทนค่าไปจะได้

$$x(n+1) = x(n)\cos\triangle\theta - \frac{1}{2}x(n-1) + \frac{1}{2}x(n+1)$$
$$y(n+1) = y(n)\cos\triangle\theta - \frac{1}{2}y(n-1) + \frac{1}{2}y(n+1) \quad (51)$$

เมื่อจัดรูปสมการใหม่จะได้

$$x(n+1) = 2x(n)\cos\Delta\theta - x(n-1)$$
$$y(n+1) = 2\mathrm{y}(n)\cos\Delta\theta - y(n-1) \quad (52)$$

เราจะสามารถหาจุดค่าตั้งจากสมการที่ (52) เพื่อส่งไปควบคุมตำแหน่งเครื่องจักร CNC ได้

6. ภาษาโปรแกรมสำหรับระบบควบคุมซีเอ็นซี

ในการทดลองนี้เราได้ใช้ภาษาโปรแกรมสำหรับระบบควบคุมซีเอ็นซีคือ รหัสจี (G-code) ในการติดต่อกับผู้ใช้เพื่อสั่งการทำงานของแท่น XYZ Table โดยมีรายละเอียดของแต่ละคำสั่งดังตารางที่ 1

ตารางที่ 1 ฟังก์ชัน G-code พื้นฐาน

คำสั่งรหัส	ความหมาย/การทำงาน
G00	การเคลื่อนที่เร็ว
G01	การเคลื่อนที่แนวเส้นตรงโดยสามารถควบคุมอัตราการป้อน
G02	การเคลื่อนที่แนวส่วนโค้งตามเข็มนาฬิกา
G03	การเคลื่อนที่แนวส่วนโค้งทวนเข็มนาฬิกา
G04	เวลาหยุด (dwell)
G13-G16	การเลือกแกน
G17-G19	การเลือกระนาบ
G40	ยกเลิกการชดเชยขนาดรัศมีมีดกัด
G41	เรียกใช้การชดเชยขนาดรัศมีมีดกัดด้านซ้ายของเส้นขอบรูป
G42	เรียกใช้การชดเชยขนาดรัศมีมีดกัดด้านขวาของเส้นขอบรูป
G43	เรียกใช้การชดเชยขนาดรัศมีมีดกัดมุมใน
G44	เรียกใช้การชดเชยขนาดรัศมีมีดกัดมุมนอก
G90	การกำหนดขนาดแบบสัมบูรณ์ (absolute)
G91	การกำหนดขนาดเชิงส่วนเพิ่ม (incremental)
G94	อัตราป้อนเป็น มม.ต่อนาที
G95	อัตราป้อนเป็น มม. ต่อรอบ
M00	หยุดโปรแกรม
M03	สปินเดิลหมุนตามเข็มนาฬิกา
M04	สปินเดิลหมุนทวนเข็มนาฬิกา
M05	หยุดสปินเดิล
M06	เปลี่ยนเครื่องมือ
M07	เปิดปั๊มสารหล่อเย็นหมายเลข 2
M08	เปิดปั๊มสารหล่อเย็นหมายเลข 1
M09	ปิดปั๊มสารหล่อเย็น

เราจะอธิบายเพิ่มเติมในส่วนของคำสั่งสำคัญๆที่ใช้เป็นมาตรฐานในภาษาโปรแกรมสำหรับระบบควบคุมซีเอ็นซีได้แก่

G00 การเคลื่อนที่เร็ว
G01 การเคลื่อนที่แนวเส้นตรงตามค่าอัตราป้อน
G02 การเคลื่อนที่แนวส่วนโค้งตามเข็มนาฬิกา
G03 การเคลื่อนที่แนวส่วนโค้งทวนเข็มนาฬิกา

1. คำสั่งการเคลื่อนที่เร็ว (G00)

คำสั่งที่ใช้ในการทำงานแบบเคลื่อนที่เร็ว จะกำหนดในโปรแกรมด้วยคำสั่ง G00 คำสั่งนี้จะใช้ในการเคลื่อนที่ของเครื่องมือ เช่น มีดกัด มีดกลึง เป็นต้น ไปยังจุดเป้าหมายอัตราการเคลื่อนที่เร็วของเครื่องเมื่อใช้คำสั่งนี้จะต้องมีเงื่อนไขเสริมคือ ค่าพิกัดแกน XYZ ของจุดเป้าหมายที่ต้องการเลื่อนเครื่องมือไป ตัวอย่างการใช้คำสั่ง G00 คือ

G00 ตำแหน่งค่าแกน X ตำแหน่งค่าแกน Y ตำแหน่งค่าแกน Z

ตัวอย่าง เช่น G00 X5 Y-4 Z3 หมายความว่า วิ่งไปที่ค่าแกน X เป็น 5 ซม. ค่าแกน Y เป็น -4 ซม. และค่าแกน Z เป็น 3 ซม. ด้วยอัตราเร็วสูงที่สุด

2. การเคลื่อนที่แนวเส้นตรงตามค่าอัตราป้อน (G01)

เป็นคำสั่งที่ใช้สำหรับการเคลื่อนที่แนวเส้นตรงตามค่าอัตราป้อนที่ใช้ซึ่งจะมีเงื่อนไขในการทำงานคือ ค่าพิกัดแกน XYZ ของจุดเป้าหมายและอัตราป้อน ตัวอย่างการใช้คำสั่ง G01 คือ

G00 ตำแหน่งจุดเป้าหมาย อัตราป้อน

ตัวอย่าง เช่น G01 X10 Y-3 Z1 F10 หมายความว่า วิ่งไปที่ค่าแกน X เป็น 10 ซม. ค่าแกน Y เป็น -3 ซม. และค่าแกน Z เป็น 1 ซม. ด้วยอัตราเร็วที่เรากำหนดคือ 10 โดยหน่วยเป็นมิลลิเมตรต่อวินาที (mm/s)

3. การเคลื่อนที่แนวส่วนโค้งตามเข็มและทวนเข็มนาฬิกา (G02/G03)

คำสั่งสำหรับการเคลื่อนที่แนวเส้นโค้งที่กำหนดเป็นมาตรฐาน จะมีลักษณะการเคลื่อนที่ที่แตกต่างกันระหว่างคำสั่ง G02 และ G03 ซึ่งขึ้นอยู่กับทิศทางการหมุน การใช้คำสั่งเคลื่อนที่แนวส่วนโค้งตามเข็มนาฬิกา (G02) และการใช้คำสั่งเคลื่อนที่แนวส่วนโค้งทวนเข็มนาฬิกา (G03) จะมีเงื่อนไขในการทำงานคือ ค่าพิกัดแกน XYZ ของจุดเป้าหมาย ข้อมูลของขนาดรัศมีหรือจุดศูนย์กลางส่วนโค้งและอัตราป้อน ตัวอย่างการใช้คำสั่ง G02/G03 คือ

G02/G03 ตำแหน่งจุดเป้าหมาย ข้อมูลรัศมีส่วนโค้ง อัตราป้อน

ตัวอย่าง เช่น G02 X0 Y0 Z1 I-4 J-4 F20 หมายความว่า วิ่งเป็นวงกลมในทิศทางตามเข็มนาฬิกาไปที่ค่าแกน X เป็น 0 ซม. ค่าแกน Y เป็น 0 ซม. และค่าแกน Z เป็น 1 ซม. โดยจุดเริ่มต้นห่างจากจุดศูนย์กลางในทิศทางแกน X เท่ากับ -4 ซม. (ค่า I เป็น -4) และทิศทางแกน Y เท่ากับ -4 ซม. (ค่า J เป็น -4) ด้วยอัตราเร็วที่เรากำหนดคือ 20 มิลลิเมตรต่อวินาที (mm/s)

7. ฟังก์ชันการติดต่อสื่อสารระหว่างตัวควบคุมกับซอฟท์แวร์ MATLAB

ดังที่ได้กล่าวมาแล้วว่าฟังก์ชันการติดต่อสื่อสารระหว่างตัวควบคุมกับซอฟท์แวร์ MATLAB กระทำโดยผ่านเครื่องมือของ MATLAB ที่เรียกว่า RTDX (Real-Time Data Exchange) มีรายละเอียดดังนี้

7.1 Link for Code Composer Studio โดยการเชื่อมต่อสำหรับ RTDX

Link for Code Composer Studio เป็นเครื่องมือที่ใช้ MATLAB function ในการติดต่อสื่อสารกับ Code Composer Studio integrated development environment (CCS IDE) โดยเราสามารถติดต่อสื่อสารส่งผ่านข้อมูลจาก CCS IDE ด้วย Embedded Objects เราสามารถดึงข้อมูลของสัญญาณ DSP ที่ถูกเก็บไว้ใน memory หรือ register มาดูได้

Link for Code Composer Studio จะทำงานโดย CCS IDE และ Real-Time Data Exchange (RTDX) ในการใช้ RTDX ในการติดต่อกับ target processor (DSP) เราสามารถ

- ส่งและรับข้อมูลจาก memory บน target processor ได้
- สามารถเปลี่ยนขั้นตอนการทำงานโดยไม่ต้องหยุดการทำงานของ target processor

เราสามารถใช้ RTDX ติดต่อกับ target processor โดยการเปิดและอนุญาตให้สามารถใช้ Channels จากนั้นจึงส่งหรือรับข้อมูลจาก target เมื่อสิ้นสุดการติดต่อก็ทำการปิดการใช้ Channels และลบ Channels ออกไปโดยการใช้ function ต่อไปนี้

ส่วนที่ติดต่อกับ CCS IDE

- ccsdsp สร้าง link to CCS IDE และ RTDX
- cd เปลี่ยน directory ที่ทำงานของ CCS IDE จาก MATLAB
- open เปิด project file ใน CCS IDE
- load โหลด program ลงบน targer processor
- run สั่งให้ target processor ทำงาน
- rtdx คำสั่งเรียกใช้ RTDX

ส่วนของชุดคำสั่งที่ติดต่อผ่าน RTDX

- close ปิดการติดต่อ RTDX ระหว่าง MATLAB และ target
- configure กำหนดจำนวน channel buffers ที่ใช้และขนาดของแต่ละ buffer
- disable ไม่อนุญาตให้สามารถใช้ RTDX ก่อนที่เราจะปิดการติดต่อ
- display and disp จะแสดงค่าที่เกิดจากการใช้ function get และ set
- enable อนุญาตให้สามารถใช้ RTDX ในการรับและส่งข้อมูลกับ target
- isenabled บอกสถานะอนุญาตให้ใช้ RTDX
- isreadable บอกสถานะว่า MATLAB สามารถอ่านค่าจาก memory ได้หรือไม่
- iswritable บอกสถานะว่า MATLAB สามารถส่งค่าไปที่ target ได้หรือไม่
- open เปิดการติดต่อของ RTDX
- readmat อ่านข้อมูลขนาด n x m จาก target สู่ MATLAB
- readmsg อ่านข้อมูลขนาด 1 x m จาก channel
- writemsg เขียนข้อมูลไปที่ target โดยผ่าน channel

ตัวอย่างการใช้คำสั่ง

1) เริ่มต้นติดต่อกับ target processor

```
cc = ccsdsp;
```

2) เปิด project file ที่เราต้องการติดต่อด้วย

```
open (cc , 'project file');
```

3) เปลี่ยน directory ที่ทำงานของ CCS IDE ไปที่เราเก็บ project file

```
cd (cc , 'project path');
```

4) โหลด program ที่เราต้องการให้ทำงานไปที่ target

```
load (cc , 'project file.out');
```

5) สั่งให้ target ทำงาน

```
run (cc);
```

6) กำหนด buffer ใน RTDX ที่จะเก็บข้อมูลจนกระทั่ง MATLAB สามารถอ่านค่าได้ (MATLAB ไม่สามารถอ่านข้อมูลได้เร็วกว่าที่ target สามารถส่งข้อมูลได้)

```
cc.rtdx.configure (1024,4);          % define 4 channels of 1024 byte each
```

7) สร้าง 1 channel สำหรับ channel ของการส่ง โดยกำหนดชื่อเป็น 'ichan' ในแบบ 'w'

```
cc.rtdx.open ('ichan' , 'w');
```

8) อนุญาตให้สามารถใช้ channel ที่เปิดได้

```
cc.rtdx.enable ('ichan');
```

9) ทำตามข้อ 8 และ 9 ในกรณีกำหนด channel ของการรับ

```
cc.rtdx.open (‘ochan’ , ‘r’);
cc.rtdx.enable (‘ochan’);
```

10) ส่งข้อมูลขนาด 1x10 ผ่าน channel ของการส่ง

```
indata = 1:10;
cc.rtdx.writemsg (‘ichan’ , int16(indata));
```

11) การรับข้อมูลแบบ int16 มาจาก target

```
outdata = cc.rtdx.readmsg (‘ichan’ , ‘int16’);
```

หรือ

```
outdata = cc.rtdx.readmat (‘ichan’ , ‘int16’ , ‘ขนาด matrix’);
```

12) ในกรณีที่เราไม่ต้องการรับข้อมูลที่เหลืออยู่ในใน buffer ของ channel แล้วจะใช้คำสั่ง

```
cc.rtdx.flush(‘ochan’ , number);
```

number คือ จำนวนข้อมูลที่เราไม่ต้องการรับ หรือ ‘All’ คือ ไม่ต้องการข้อมูลที่เหลืออยู่ทั้งหมด

13) เมื่อเราต้องการหยุดการทำงานของ target processor จะใช้คำสั่ง

```
cc.halt;
```

14) เมื่อการทำงานของ target processor หยุดเราจะต้องไม่อนุญาตให้สามารถใช้ channel ได้

```
cc.rtdx.disable ('ichan');
cc.rtdx.disable ('ochan');

หรือ

cc.rtdx.disable ('all');
```

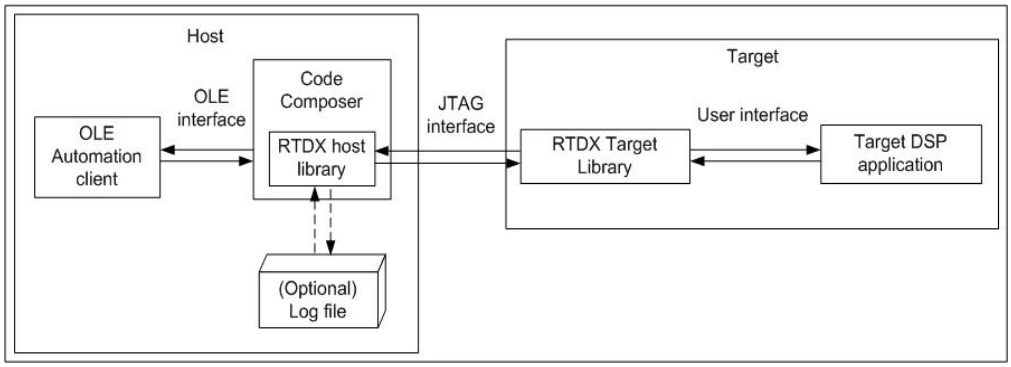
15) เราจะปิด channel การติดต่อผ่าน RTDX ระหว่าง MATLAB กับ target

```
cc.rtdx.close ('ichan');
cc.rtdx.close ('ochan');

หรือ

cc.rtdx.close ('all');
```

7.2 Real-Time Data Exchange บน DSP Board (RTDX)

RTDX สามารถทำงานได้โดยไม่ใช้ DSP/BIOS โดยตัวอย่างโปรแกรมที่ใช้ DSP/BIOS อยู่ที่ C:\ti\tutorial ข้อมูลชื่อ Volume4, hostio1, and hostio2 และตัวอย่างโปรแกรมที่ไม่ใช้ DSP/BIOS อยู่ที่ C:\ti\examples\target\rtdx

การติดต่อสื่อสารระหว่าง host (PC) และ target (Ti processor) โดยใช้ RTDX ผ่านทาง CCS IDE แสดงดังภาพที่ 54



ภาพที่ 54 การทำงานของ RTDX ระหว่าง Host และ Target

จากตัว target การส่งผ่านข้อมูลให้ host จะผ่านทาง JTAG interface จากนั้น RTDX host library จะรับข้อมูลจาก JTAG interface และเก็บเอาไว้ โดยข้อมูลที่ถูกส่งมาจะถูกเก็บไว้ใน memory buffer หรือที่ RTDX log file ข้อมูลที่ถูกเก็บเอาไว้สามารถถูกเรียกได้โดย Object Linking and Embedding (OLE) automation client เช่น

- Visual Basic applications
- Visual C++ applications
- LabVIEW
- MATLAB
- Microsoft Excel

RTDX มีการรับข้อมูลจาก target application อยู่ 2 แบบคือ

1) Non-Continuous คือ ข้อมูลจะถูกเขียนไว้ที่ log file บน host ก่อน วิธีนี้จะใช้เมื่อเราต้องการรู้จำนวนข้อมูลและเก็บไว้ใน log file

2) Continuous คือ ข้อมูลจะถูกเก็บไว้โดย RTDX host library แต่จะไม่ถูกเขียนที่ log file วิธีนี้จะใช้เมื่อเราต้องการรับข้อมูลอย่างเนื่องและแสดงข้อมูลจาก DSP application โดยเราไม่ต้องการให้เก็บข้อมูลที่ log file

function ในการรับและส่งข้อมูลบน DSP แบ่งตามชนิดข้อมูลและ function ที่ถูกกำหนดไว้ที่ header file rtdx.h

Declaration Macros

- RTDX_CreateInputChannel
- RTDX_CreateOutputChannel

Functions

- RTDX_channelBusy
- RTDX_disableInput
- RTDX_disableOutput
- RTDX_enableOutput
- RTDX_enableInput
- RTDX_read
- RTDX_sizeofInput
- RTDX_write

Macros

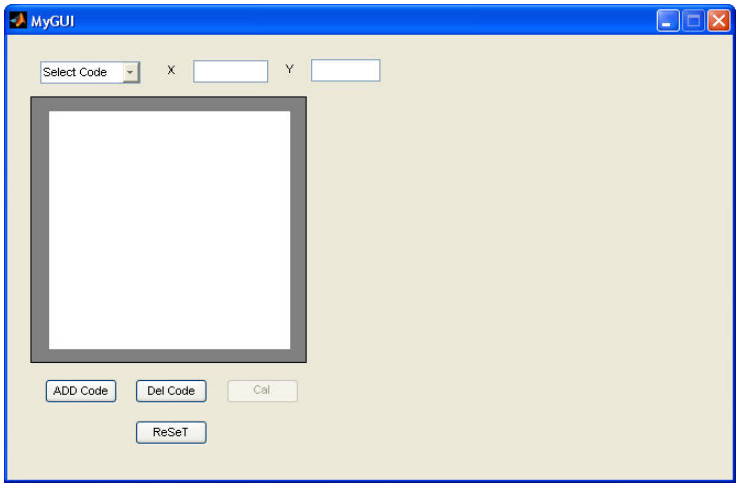
- RTDX_isInputEnabled
- RTDX_isOutputEnabled

7.3 การติดต่อสื่อสารระหว่าง MATLAB กับ DSP Board

การติดต่อสื่อสารนี้จะเป็นการที่เราจำลองการสร้างเส้นทางที่เราต้องการเคลื่อนที่แท่น XYZ Table บน MATLAB จากนั้นเราแปลงเส้นทางเป็นค่าตำแหน่งและเวลาในการเคลื่อนที่แล้วจึงส่งค่าผ่าน RTDX ไปที่ DSP Board แล้วเข้าสู่ Control loop เพื่อจะส่งค่าออกไปควบคุมมอเตอร์บนแท่น XYZ Table จากนั้นตัวมอเตอร์จะส่งค่า Encode มาเข้า FPGA เพื่อทำการแปลงค่า Pulse เป็นค่าตำแหน่งแล้วกลับไปเข้า control loop บน DSP Board เพื่อทำงานเปรียบเทียบ พร้อมกับส่งค่าตำแหน่งกลับไปแสดงผลที่ GUI บน host (PC) ดังภาพที่ 36

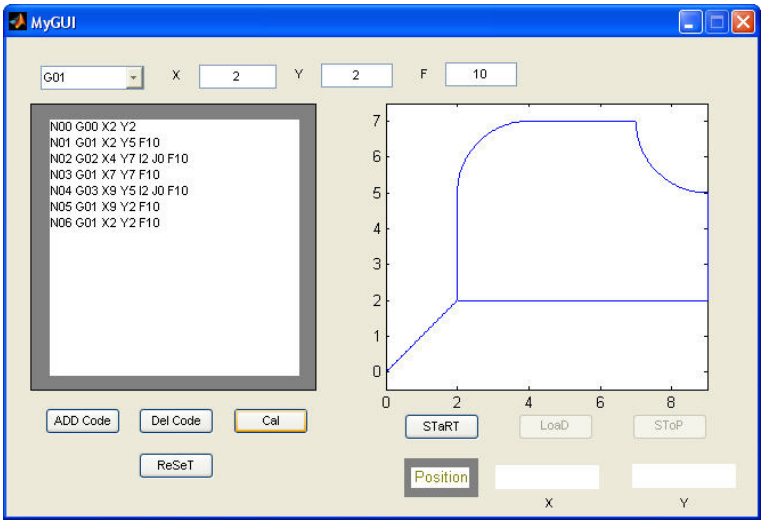
ลำดับขั้นตอนการทำงาน

1) เรียก GUI ขึ้นมาเพื่อเริ่มต้นการทำงานดังภาพที่ 55



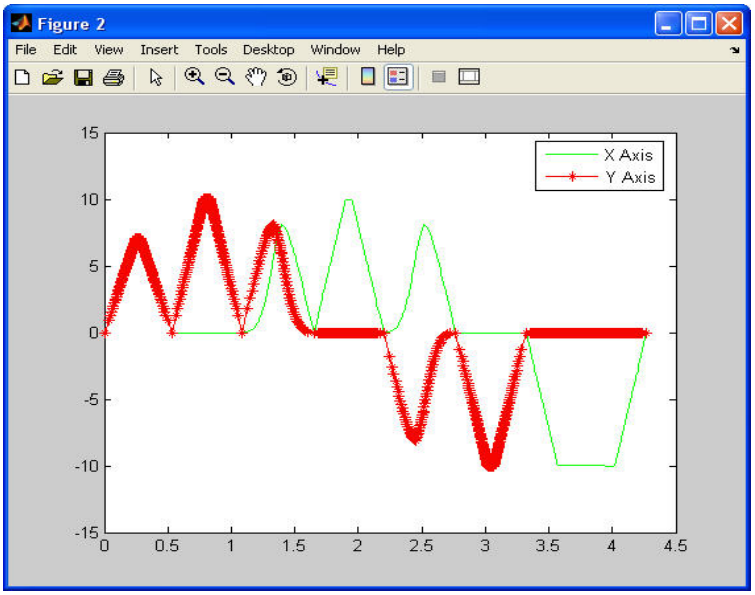
ภาพที่ 55 GUI for แท่น XYZ Table Control

2) พิมพ์ค่า G Code เพื่อสร้างเส้นทางการเคลื่อนที่ของแท่น XYZ Table แล้วกด Cal เพื่อแสดงภาพการจำลองการเคลื่อนที่ดังภาพที่ 56



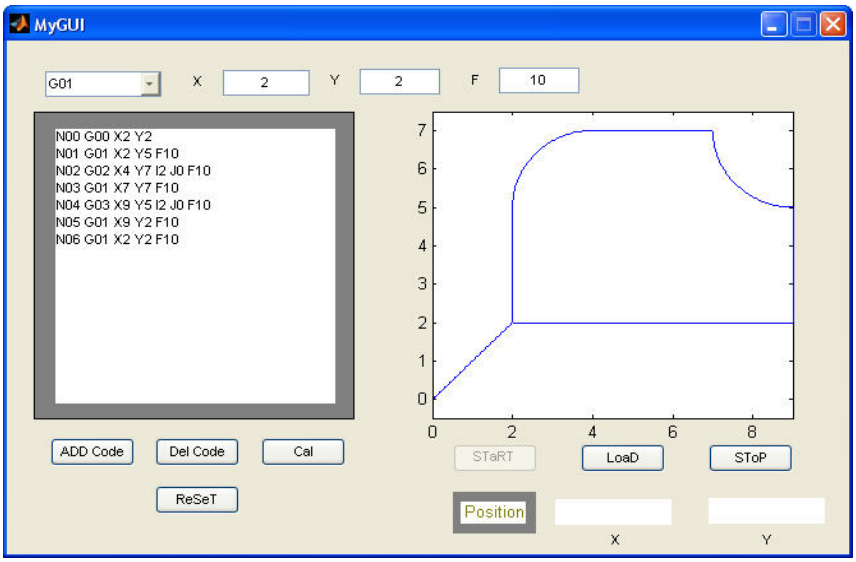
ภาพที่ 56 จำลองเส้นทางการเคลื่อนที่ของแท่น XYZ Table

3) ความเร็วในการเคลื่อนที่ของแกน X และY บนแท่น XYZ Table ดังภาพที่ 57



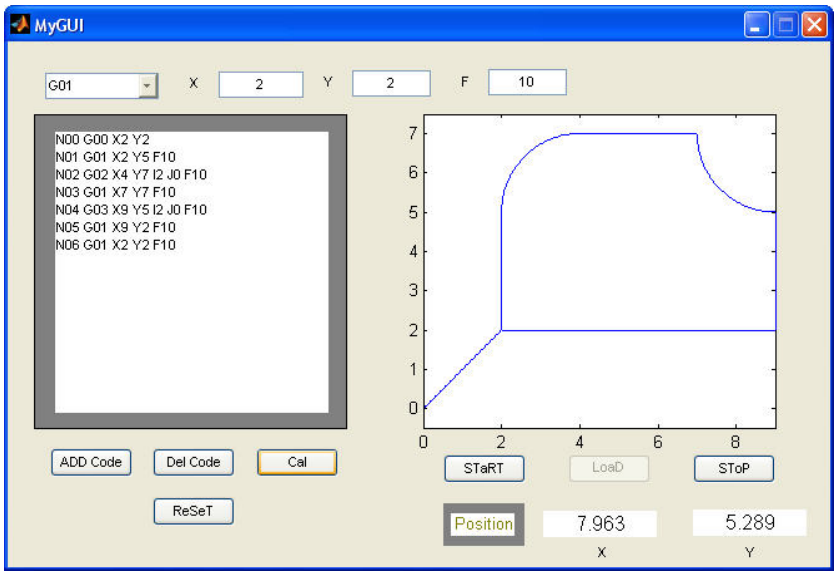
ภาพที่ 57 ความเร็วในการเคลื่อนที่ของแท่น XYZ Table

4) กด Start เพื่อติดต่อกับ CCS IDE และกด Load เพื่อสั่งให้ DSP Board ทำงานพร้อมส่งค่าตำแหน่งดังภาพที่ 58



ภาพที่ 58 แสดงการกด Start และ Load

5) DSP Board จะส่งค่าตำแหน่งเพื่อนำมาแสดงค่าบน GUI ดังภาพที่ 59

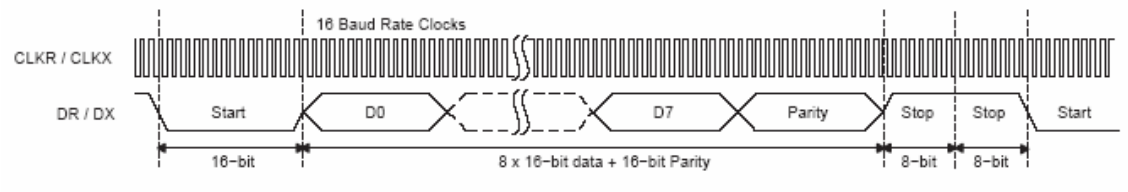


ภาพที่ 59 GUI แสดงค่าตำแหน่งขณะแท่น XYZ Table เคลื่อนที่

7.4 การติดต่อสื่อสารแบบ serial ระหว่าง DSP กับ Computer โดยผ่าน McBSP

ในส่วนนี้เราจะกล่าวถึงการใช้ multichannel buffered serial port (McBSP) ซึ่งอยู่ใน TMS320C6000 (C6000) digital signal processors (DSP) ในการติดต่อสื่อสารแบบ Universal Asynchronous Receiver/Transmitter (UART) ปกติ McBSP จะไม่สนับสนุนมาตรฐาน UART อย่างไรก็ตามเราใช้การเปลี่ยนแปลง serial control registers ทำให้ McBSP สามารถติดต่อสื่อสารในแบบมาตรฐาน UART ได้

The Universal Asynchronous Receiver/Transmitter (UART) เป็นมาตรฐานซึ่งสร้างขึ้นมาเป็นข้อกำหนดรูปแบบของข้อมูลที่ใช้ในการติดต่อสื่อสารแบบ serial เนื่องจาก UART เป็นการติดต่อสื่อสารแบบ asynchronous สายติดต่อสื่อสารจึงไม่จำเป็นต้องมีสัญญาณนาฬิการับ-ส่ง การรับ-ส่งข้อมูลอาศัยสัญญาณนาฬิกา serial ที่ทำงานอยู่บนตัวรับ-ส่งเอง รูปแบบการติดต่อสื่อสารของ UART จะมี start bit และ stop bit ช่วยให้ข้อมูลที่ได้รับสอดคล้องกัน UART timing จะแสดงดังภาพที่ 60



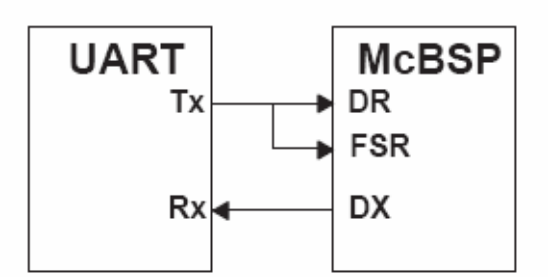
ภาพที่ 60 UART timing

การส่งเริ่มที่ start bit คือการเปลี่ยนจาก logic 1 เป็น logic 0 จากนั้นจึงเริ่มส่งข้อมูลโดยส่ง LSB คือ bit ที่มีความสำคัญน้อยสุดก่อน เมื่อส่งข้อมูลจนครบ 8 bit จึงส่ง parity bit ซึ่งจะมีหรือไม่มีก็ได้ขึ้นกับรูปแบบ UART สุดท้ายจึงส่ง stop bit (logic 1)

การติดต่อสื่อสารแบบ UART โดยผ่านทาง RS-232 ของคอมพิวเตอร์ ระดับสัญญาณของข้อมูลต้องผ่าน RS-232 level converter เพื่อเปลี่ยนระดับสัญญาณไปเป็น RS-232 logic level โดย RS-232 ใช้ +3 ถึง +25 volts ในการบอก logic 0 และ -3 ถึง -25 volts ในการบอก logic 1 ส่วนช่วง -3 ถึง +3 volts จะไม่สามารถระบุ logic ได้ ขาสัญญาณส่งข้อมูลของ UART จะต่อกับขา input ของข้อมูล และ frame synchronization input บน McBSP เพราะว่าขาสัญญาณ serial จะประกอบด้วย

framing และข้อมูล ส่วนขาสัญญาณรับข้อมูลของ UART จะต่อกับขา output ของข้อมูลของ McBSP ตามภาพที่ 61

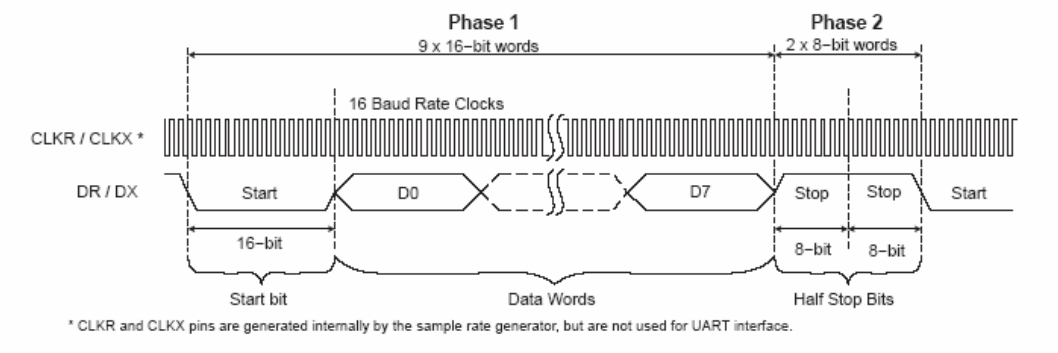
เนื่องจาก McBSP ใช้สัญญาณนาฬิกาภายในตัวเอง ทำให้การรับ-ส่งข้อมูลของแต่ละ UART bit ต้องเป็น 16 bit ของ DSP โปรแกรมจะต้องเพิ่มแต่ละ bit เป็น 16 bit ก่อนส่ง และลดแต่ละ 16 bit เหลือเพียง bit เดียวหลังจากรับข้อมูลแล้ว



ภาพที่ 61 การติดต่อสื่อสารระหว่าง UART กับ McBSP

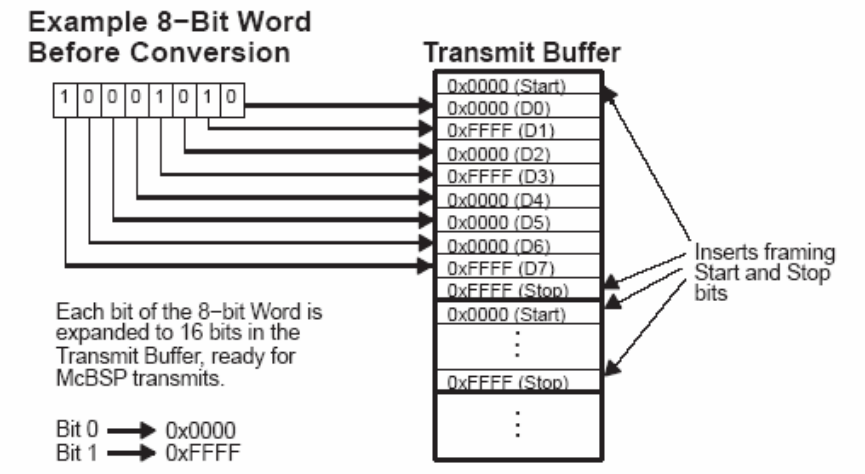
ใน start bit ของ UART จะเป็นการเปลี่ยน logic 1 เป็น logic 0 (ขอบขาลง) ซึ่งจะนำมาใช้กับขา frame sync input ของ McBSP ที่ทำงานที่ logic 0 เช่นกัน เป็นที่มาว่าทำไมขาข้อมูลและ frame sync input ถึงต่อกับขา output ของ UART ทั้งคู่

เราจะส่งข้อมูลแบบ UART ในแบบ 8N1 (ข้อมูล 8 bit, ไม่มี parity bit และ 1 stop bit) โดยจะแบ่งการส่งข้อมูลออกเป็น 2 ช่วง ดังภาพที่ 62 โดยช่วงแรกจะประกอบด้วย start bit และข้อมูล 8 bits และช่วงที่ 2 จะเป็น stop bit เราจะแทนการส่ง logic 1 ของ UART ด้วย 0xFFFF และ logic 0 ของ UART ด้วย 0x0000



ภาพที่ 62 McBSP ติดต่อ UART ในแบบ 8N1

รูปแบบการส่งข้อมูล เราจะทำการขยายข้อมูลแต่ละ bit เป็น 16 bit และส่งเข้า transmit buffer โดยเริ่มที่ start bit (0x0000) และสิ้นสุดที่ stop bit (0xFFFF) ในทุกๆ buffer ดังภาพที่ 63 จากนั้น EDMA จะเป็นตัวจัดการส่งข้อมูลจาก transmit buffer เข้าสู่ McBSP เนื่องจากข้อมูลใน transmit buffer อยู่ในรูปแบบของ UART แล้ว ทำให้ McBSP frame sync generator สามารถส่งข้อมูลออกได้ทันที



ภาพที่ 63 การจัดเรียงข้อมูลใน transmit buffer

สำหรับรูปแบบการรับข้อมูล EDMA จะอ่านข้อมูลจาก McBSP และจัดเก็บข้อมูลลง receive buffer โปรแกรมจะรอจนกว่า EDMA จะอ่านค่าและจัดเก็บข้อมูลรวมทั้ง start bit และ stop bit ลง receive buffer จนเสร็จ จากนั้นจึงนำ buffer ไปจัดการลดจำนวน bit จาก 16 bit เป็น 1 bit เพื่ออ่านข้อมูลที่แท้จริงต่อไป

สถานที่ทำการวิจัย

ห้อง 2308 ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์และห้อง 5/14 ชั้น 5 ตึกวิศวกรรมศาสตร์ 60 ปี คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์

ระยะเวลาทำการวิจัย

การวิจัยเริ่มตั้งแต่ เดือนตุลาคม 2547 สิ้นสุดเดือนมีนาคม 2549

ผลและการวิจารณ์

เราจะแบ่งการทดลองสร้างการเคลื่อนที่แท่น XYZ Table 2 การทดลอง คือ ใช้ DC servo motor และ AC servo motor ในส่วนของ DC servo motor จะทดลองการเคลื่อนที่ 3 แกนโดยจะเคลื่อนที่เป็นรูปเส้นตรง, รูปวงกลม และรูปประยุกต์อื่นๆ ในส่วนของ AC servo motor จะทดลองการเคลื่อนที่ 1 แกนเนื่องจากมี AC servo motor เพียงตัวเดียว ในส่วนของการอธิบายคำสั่ง G-Code จะอยู่ในส่วนหัวข้อ ภาษาโปรแกรมสำหรับระบบควบคุมซีเอ็นซี หน้า 67

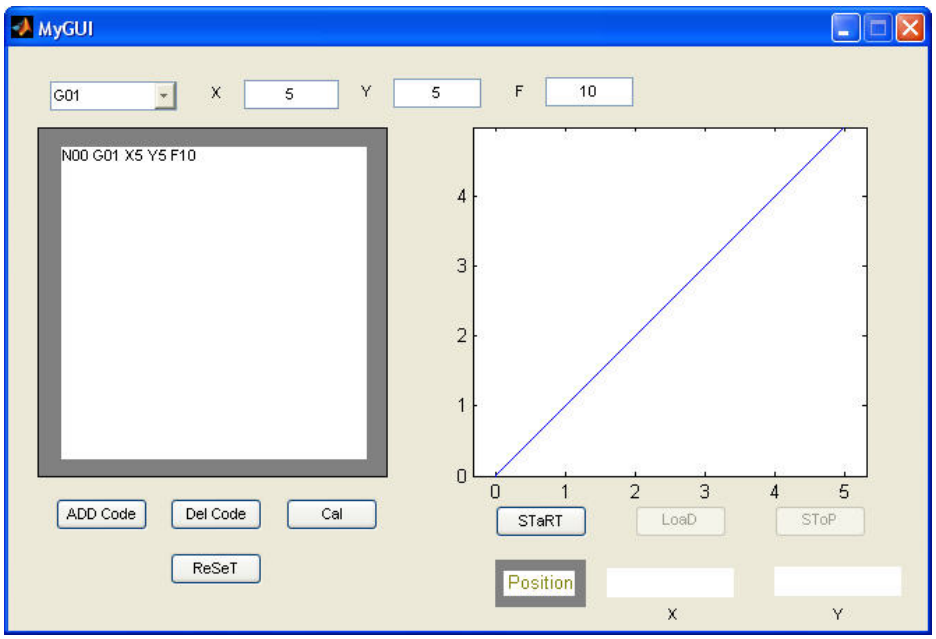
1. <u>การทดลองสร้างการเคลื่อนที่โดยใช้ DC servo motor</u>

1.1 รูปเส้นตรงแกน X 5 ซม. แกน Y 5 ซม.

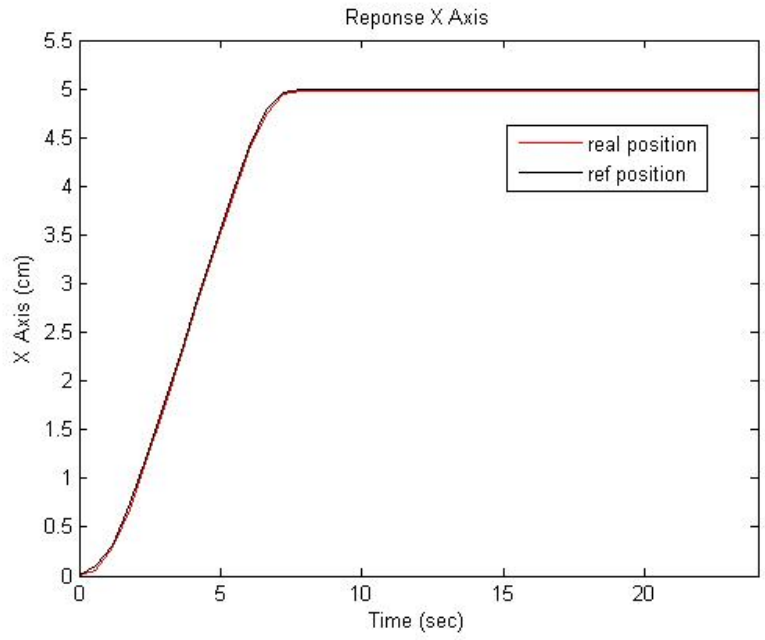
เราจะใส่รหัสจีใน GUI ของ Matlab คือ

G01 X5 Y5 F10 เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (5, 5)

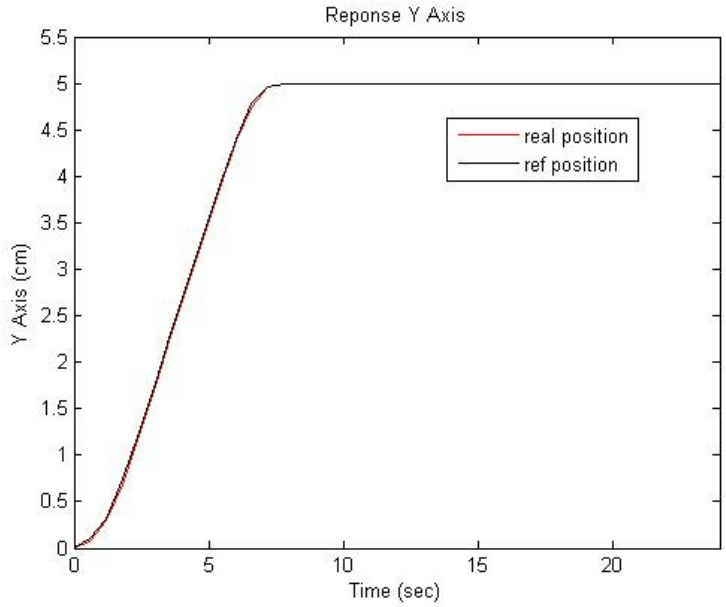
ซึ่งแสดงดังภาพที่ 64 และได้ผลตอบสนองการเคลื่อนของแกน X,Y ดังภาพที่ 65, 66 ตามลำดับ ส่วนในภาพที่ 67, 68 แสดงให้เห็นค่าความผิดพลาดของการเคลื่อนที่ของ X, Y ตามลำดับ ภาพที่ 69, 70 แสดงการเปรียบเทียบระยะที่เคลื่อนที่ได้จริงเทียบกับระยะที่ต้องการให้เคลื่อนที่พร้อมกับแสดงค่าความผิดพลาดตามลำดับ ภาพที่ 71 แสดงการเคลื่อนที่จริงบนแท่น XYZ Table



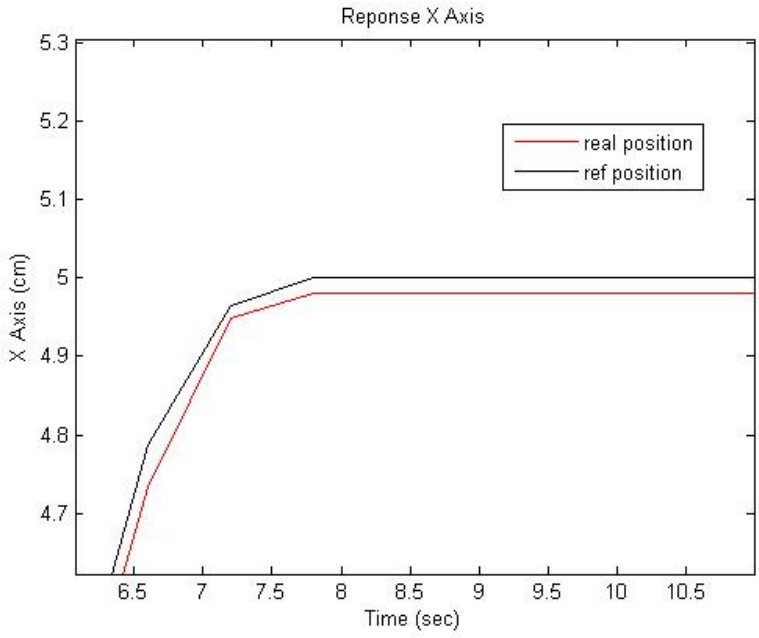
ภาพที่ 64 GUI รูปเส้นตรงแกน X 5 ซม. แกน Y 5 ซม.



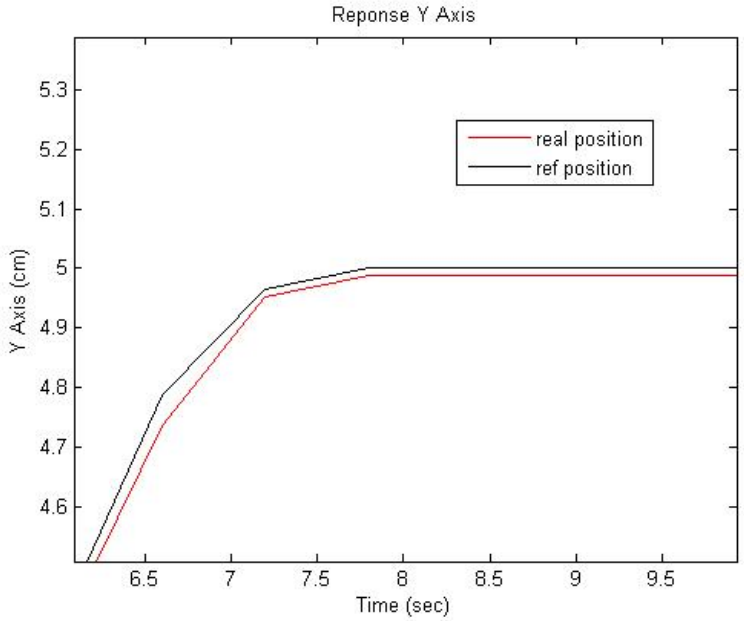
ภาพที่ 65 ผลตอบสนองของแกน X เคลื่อนที่ 5 ซม.



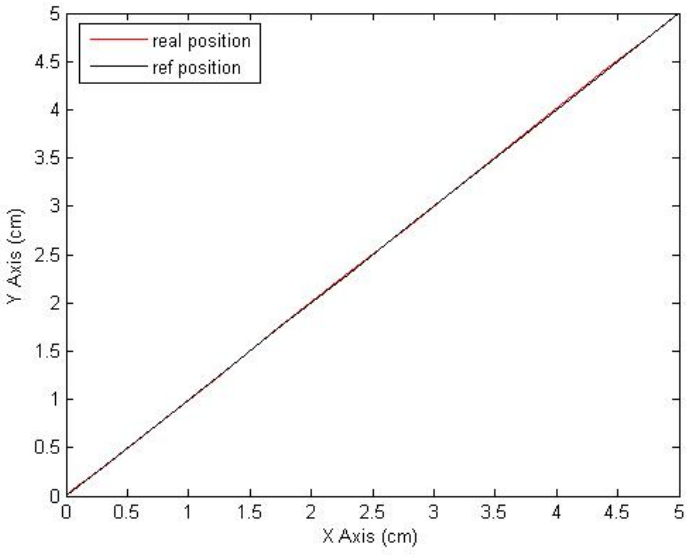
ภาพที่ 66 ผลตอบสนองของแกน Y เคลื่อนที่ 5 ซม.



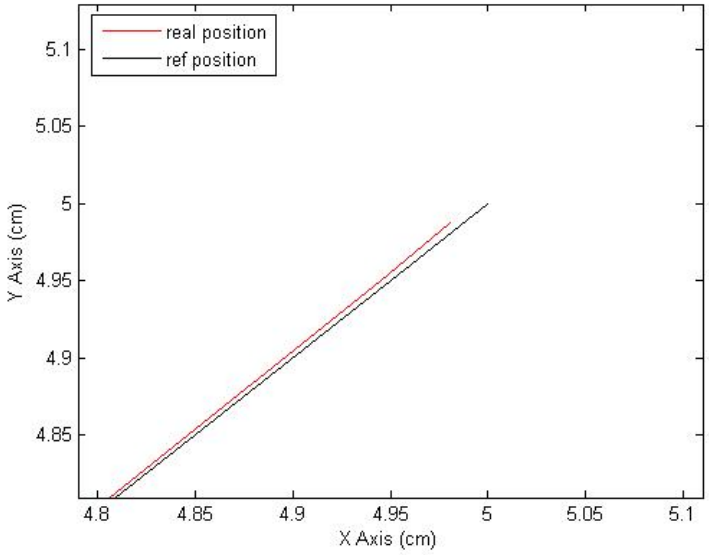
ภาพที่ 67 ค่าความผิดพลาดของการเคลื่อนที่ 5 ซม. ของ X



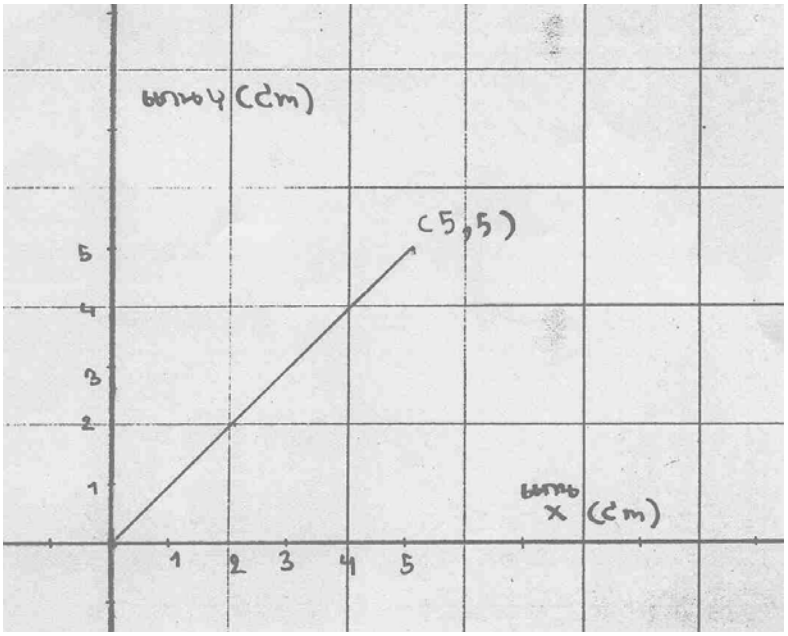
ภาพที่ 68 ค่าความผิดพลาดของการเคลื่อนที่ 5 ซม. ของ Y



ภาพที่ 69 เปรียบเทียบระยะที่เคลื่อนที่ได้จริงเทียบกับที่ต้องการให้เคลื่อนที่ (5, 5) ซม.



ภาพที่ 70 ค่าความผิดพลาดของการเคลื่อนที่ได้จริงเทียบกับที่ต้องการให้เคลื่อนที่ (5, 5) ซม.



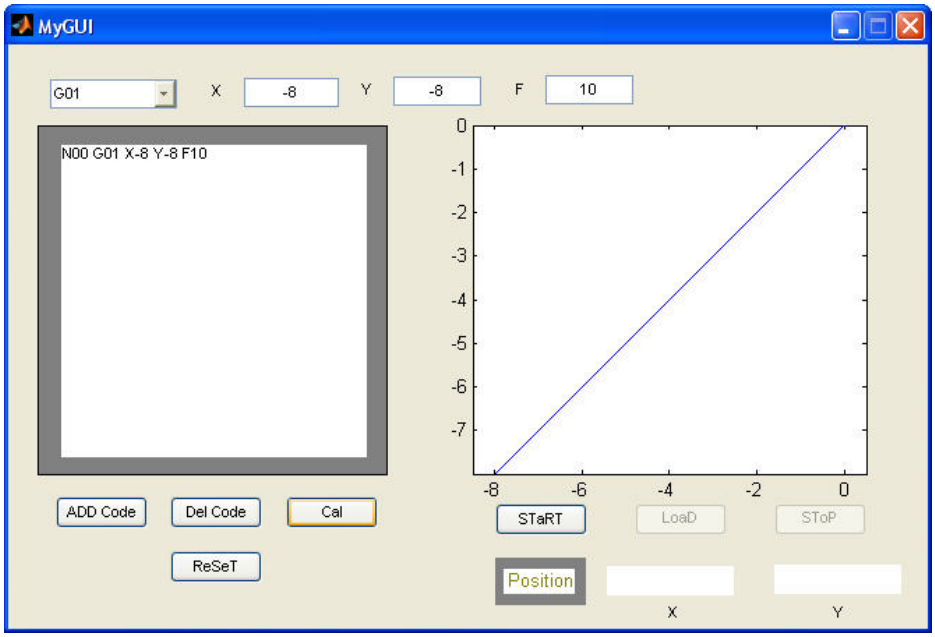
ภาพที่ 71 การเคลื่อนที่จริงรูปเส้นตรงแกน X 5 ซม. แกน Y 5 ซม.

1.2 รูปเส้นตรงแกน X -8 ซม. แกน Y -8 ซม.

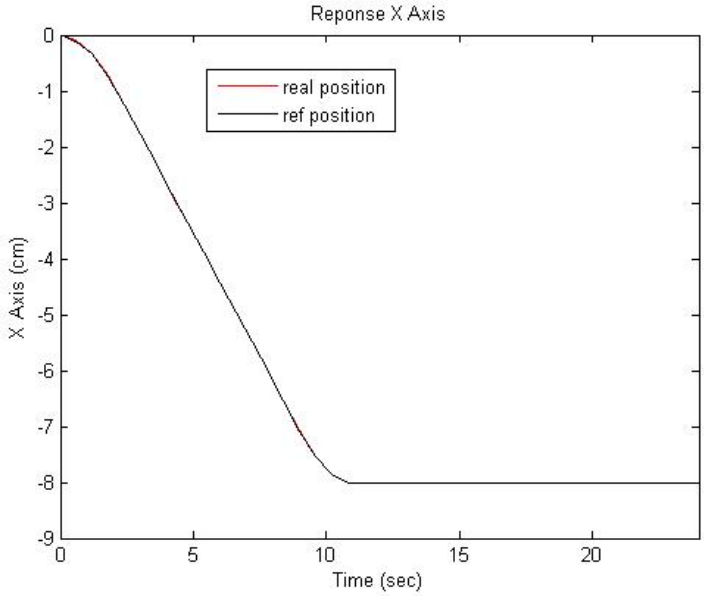
เราจะใส่รหัสจีใน GUI ของ Matlab

G01 X-8 Y-8 F10 เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (-8, -8)

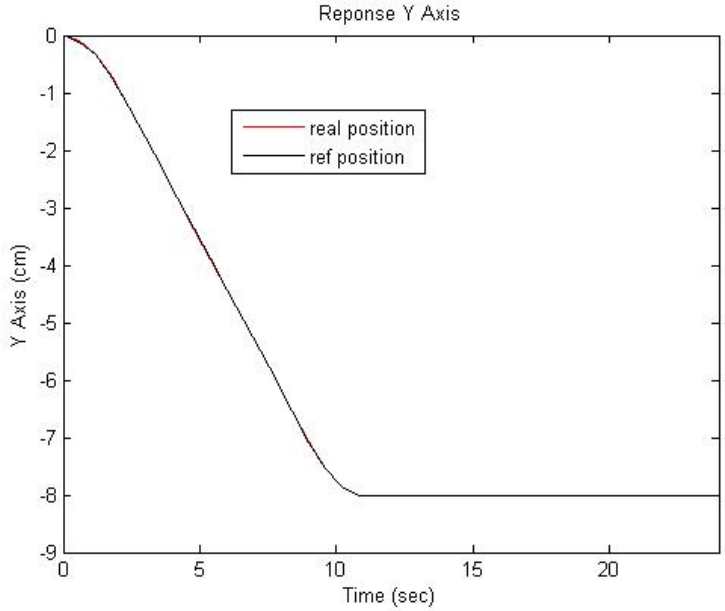
ซึ่งแสดงดังภาพที่ 72 และได้ผลตอบสนองการเคลื่อนของแกน X,Y ดังภาพที่ 73, 74 ตามลำดับ ส่วนในภาพที่ 75, 76 แสดงให้เห็นค่าความผิดพลาดของการเคลื่อนที่ของ X, Y ตามลำดับ ภาพที่ 77, 78 แสดงการเปรียบเทียบระยะที่เคลื่อนที่ได้จริงเทียบกับระยะที่ต้องการให้เคลื่อนที่พร้อมกับแสดงค่าความผิดพลาดตามลำดับ ภาพที่ 79 แสดงการเคลื่อนที่จริงบนแท่น XYZ Table



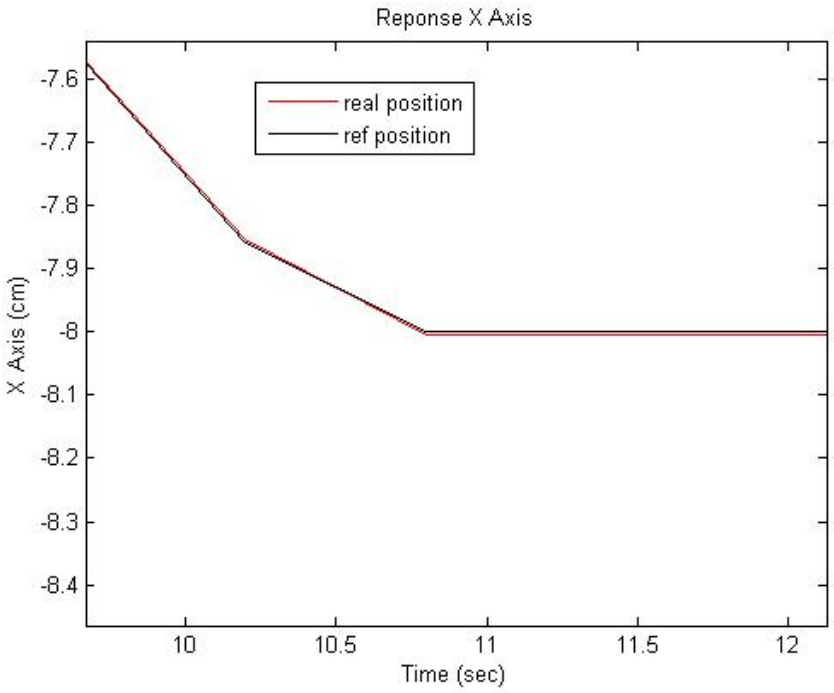
ภาพที่ 72 GUI รูปเส้นตรงแกน X -8 ซม. แกน Y -8 ซม.



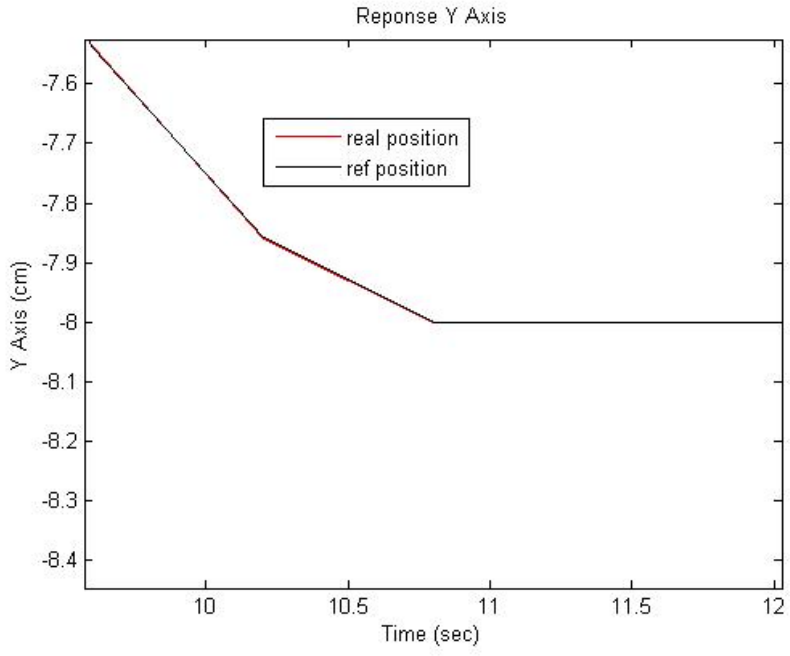
ภาพที่ 73 ผลตอบสนองของแกน X เคลื่อนที่ -8 ซม.



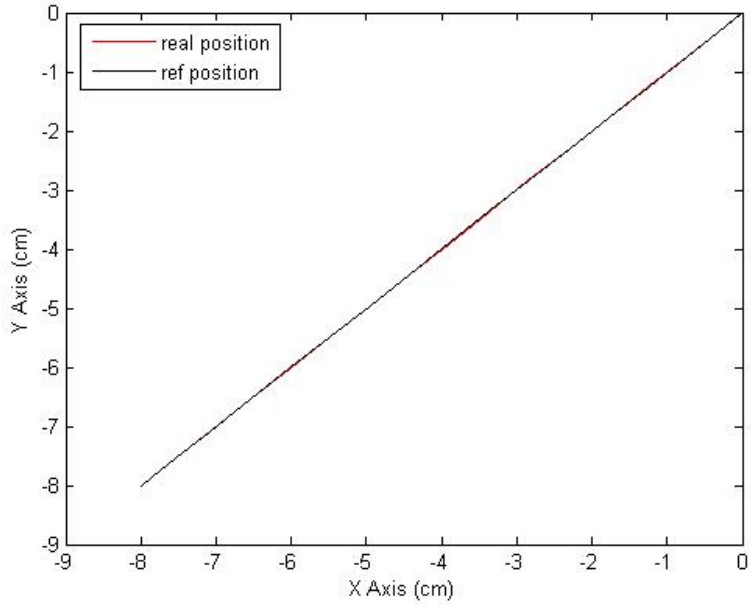
ภาพที่ 74 ผลตอบสนองของแกน Y เคลื่อนที่ -8 ซม.



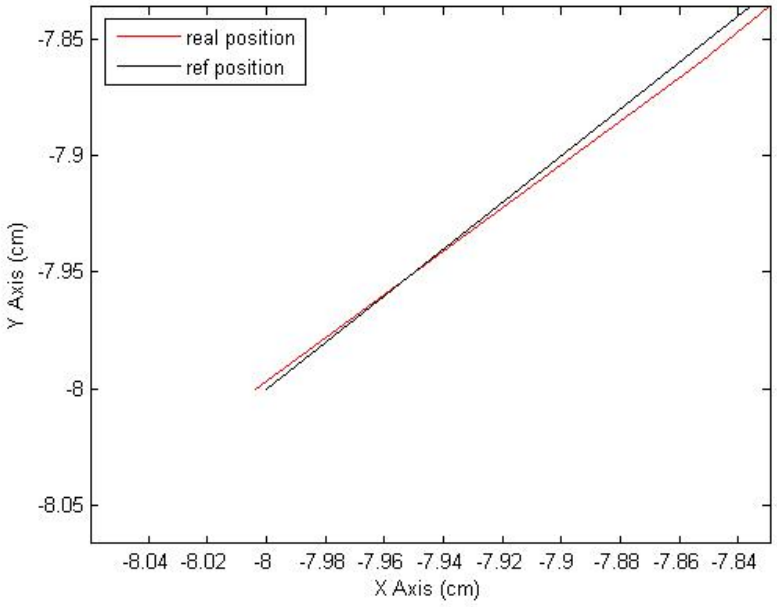
ภาพที่ 75 ค่าความผิดพลาดของการเคลื่อนที่ -8 ซม.ของ X



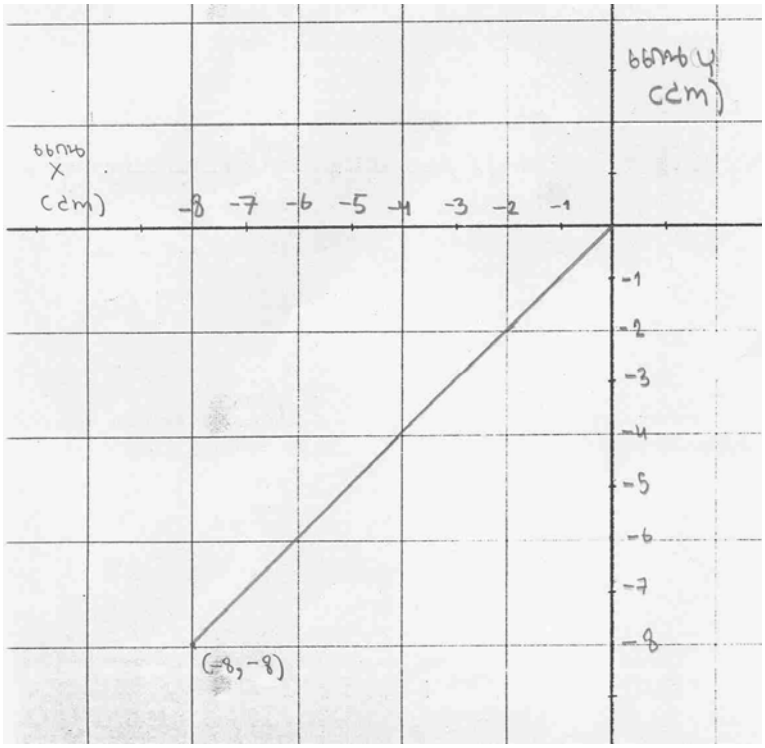
ภาพที่ 76 ค่าความผิดพลาดของการเคลื่อนที่ -8 ซม.ของ Y



ภาพที่ 77 เปรียบเทียบระยะที่เคลื่อนที่ได้จริงเทียบกับที่ต้องการให้เคลื่อนที่ (-8, -8) ซม.



ภาพที่ 78 ค่าความผิดพลาดของการเคลื่อนที่ได้จริงเทียบกับที่ต้องการให้เคลื่อนที่ (-8, -8) ซม.



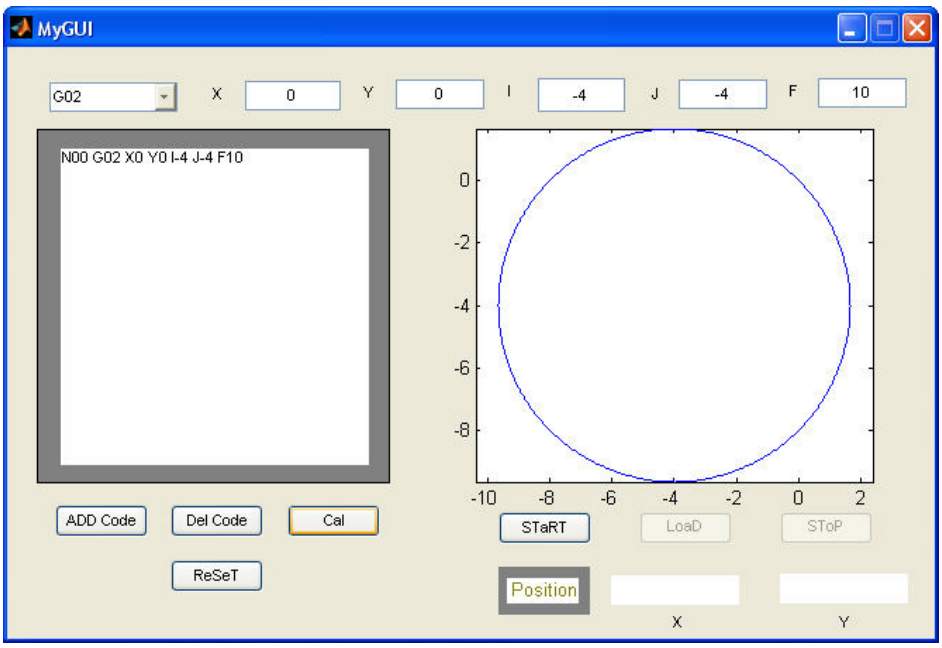
ภาพที่ 79 การเคลื่อนที่จริงรูปเส้นตรงแกน X -8 ซม. แกน Y -8 ซม.

1.3 รูปวงกลมรัศมี 5.6569 ซม.

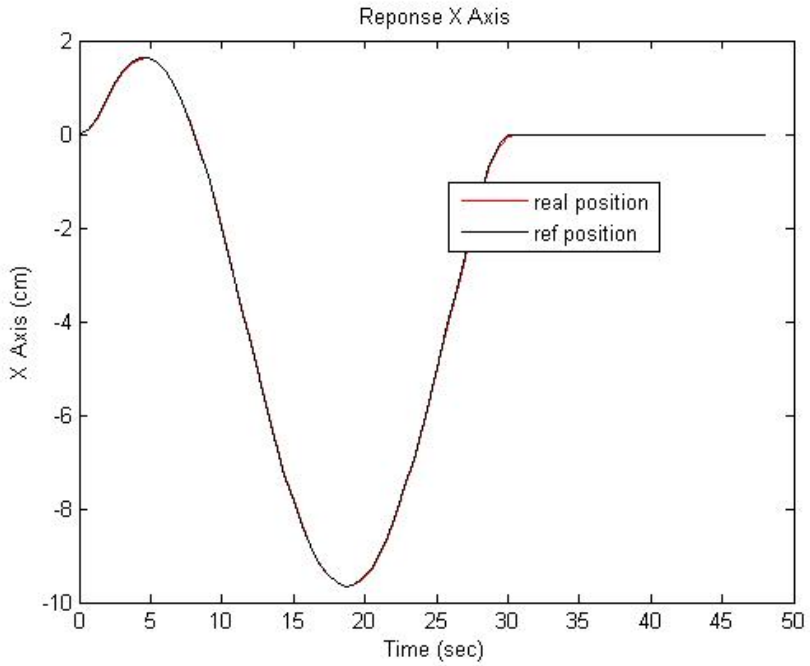
เราจะใส่รหัสจีใน GUI ของ Matlab โดยใช้

G02 X0 Y0 I-4 J-4 F10 ในการวาดวงกลม

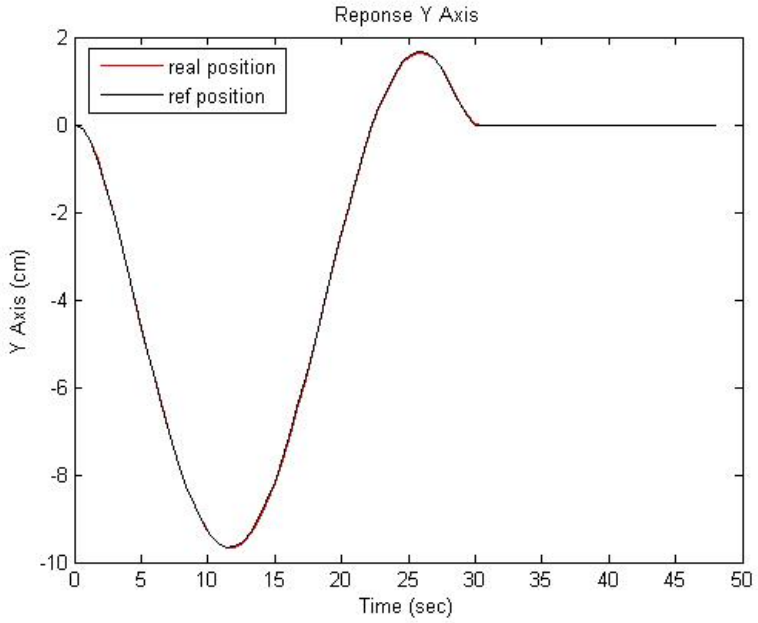
ซึ่งแสดงดังภาพที่ 80 และได้ผลตอบสนองการเคลื่อนของแกน X,Y ดังภาพที่ 81, 82 ตามลำดับ ส่วนในภาพที่ 83, 84 แสดงให้เห็นค่าความผิดพลาดของการเคลื่อนที่ของ X, Y ตามลำดับ ภาพที่ 85, 86 แสดงการเปรียบเทียบระยะที่เคลื่อนที่ได้จริงเทียบกับระยะที่ต้องการให้เคลื่อนที่พร้อมกับแสดงค่าความผิดพลาดตามลำดับ ภาพที่ 87 แสดงการเคลื่อนที่จริงบนแท่น XYZ Table



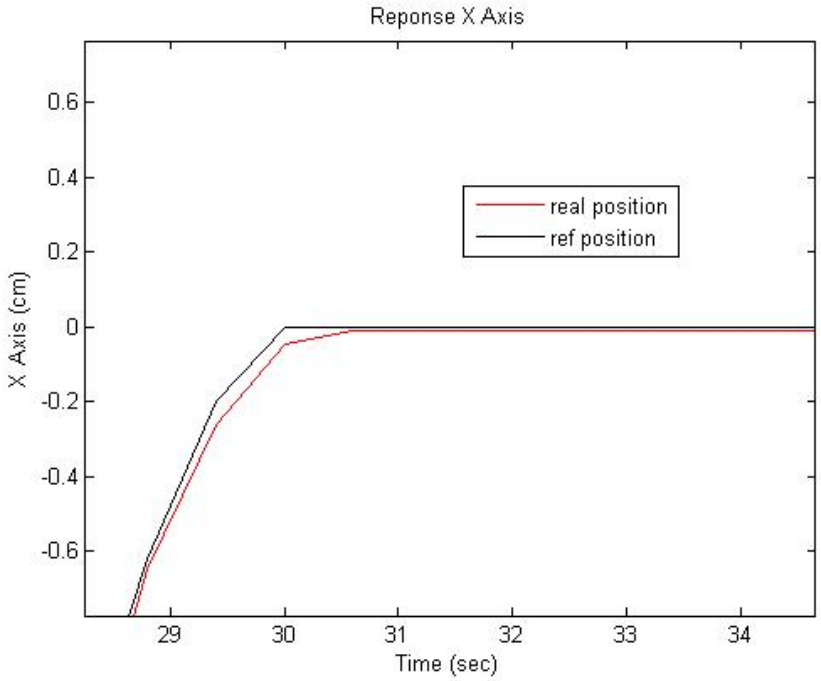
ภาพที่ 80 GUI รูปวงกลมรัศมี 5.6569 ซม.



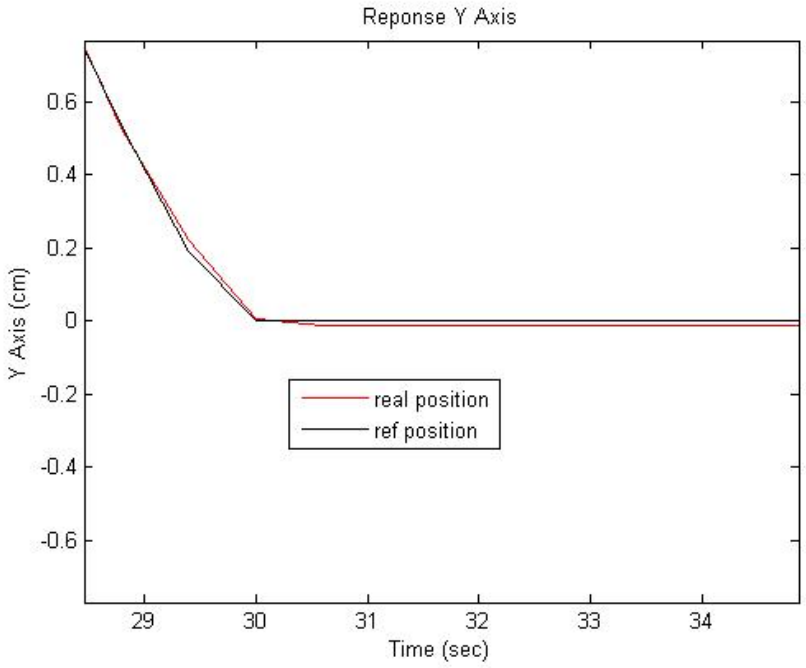
ภาพที่ 81 ผลตอบสนองของแกน X เมื่อเคลื่อนที่เป็นรูปวงกลมรัศมี 5.6569 ซม.



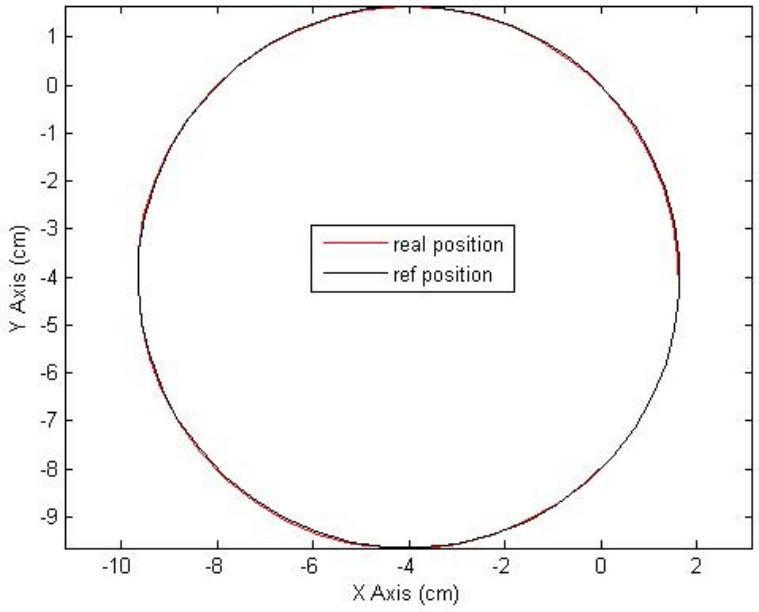
ภาพที่ 82 ผลตอบสนองของแกน Y เมื่อเคลื่อนที่เป็นรูปวงกลมรัศมี 5.6569 ซม.



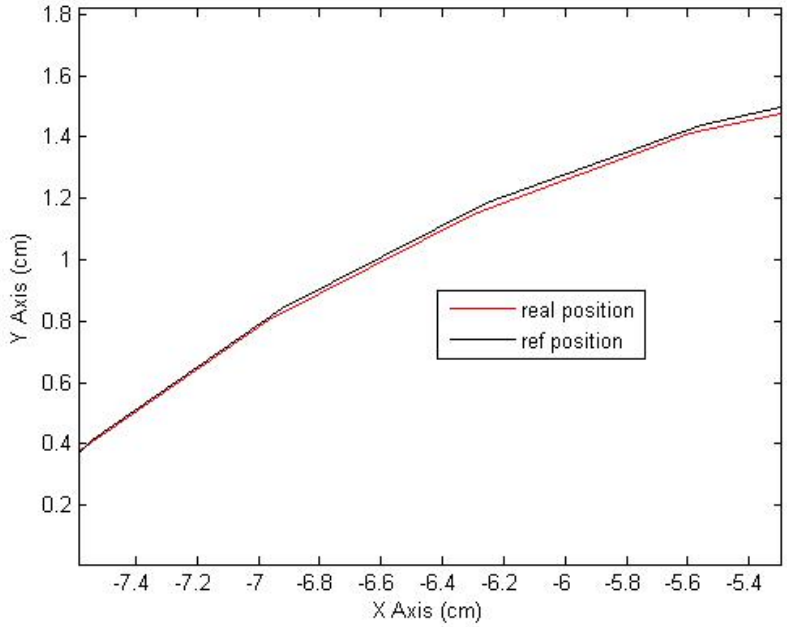
ภาพที่ 83 ค่าความผิดพลาดของแกน X เมื่อเคลื่อนที่เป็นรูปวงกลมรัศมี 5.6569 ซม.



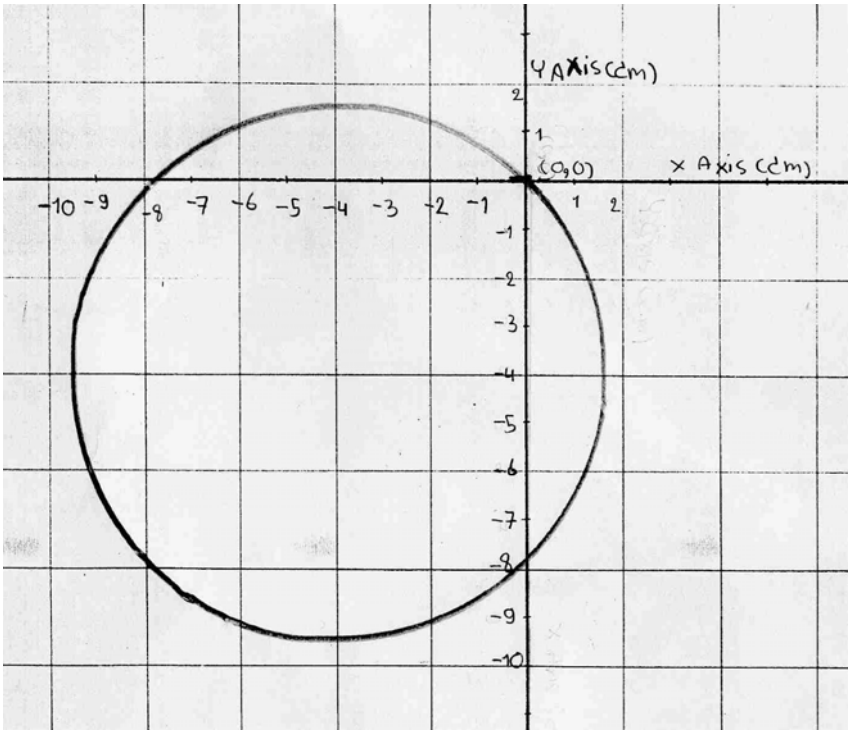
ภาพที่ 84 ค่าความผิดพลาดของแกน Y เมื่อเคลื่อนที่เป็นรูปวงกลมรัศมี 5.6569 ซม.



ภาพที่ 85 เปรียบเทียบระยะที่เคลื่อนที่เป็นวงกลมรัศมี 5.6569 ซม.



ภาพที่ 86 ค่าความผิดพลาดของการเคลื่อนที่เป็นวงกลมรัศมี 5.6569 ซม.



ภาพที่ 87 การเคลื่อนที่จริงเป็นวงกลมรัศมี 5.6569 ซม.

1.4 รูปวงกลมรัศมี 7.0711 ซม.

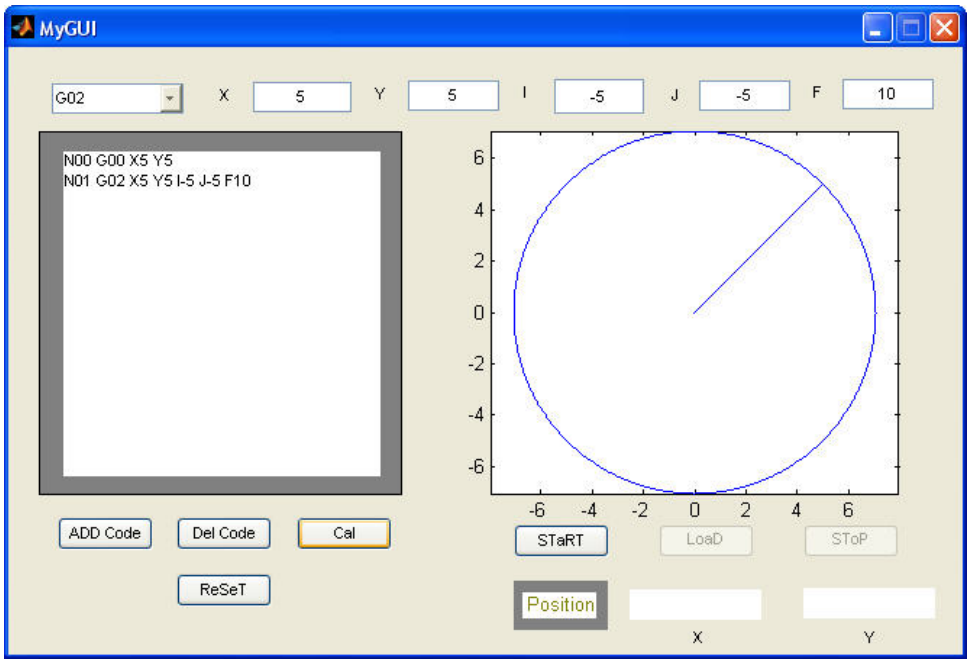
เราจะใส่รหัสจีใน GUI ของ Matlab โดยเราจะลากเส้นตรงออกไปเป็นระยะ X 5 ซม. แกน Y 5 ซม. ด้วยคำสั่ง

G01 X5 Y5 F10 เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (5, 5)

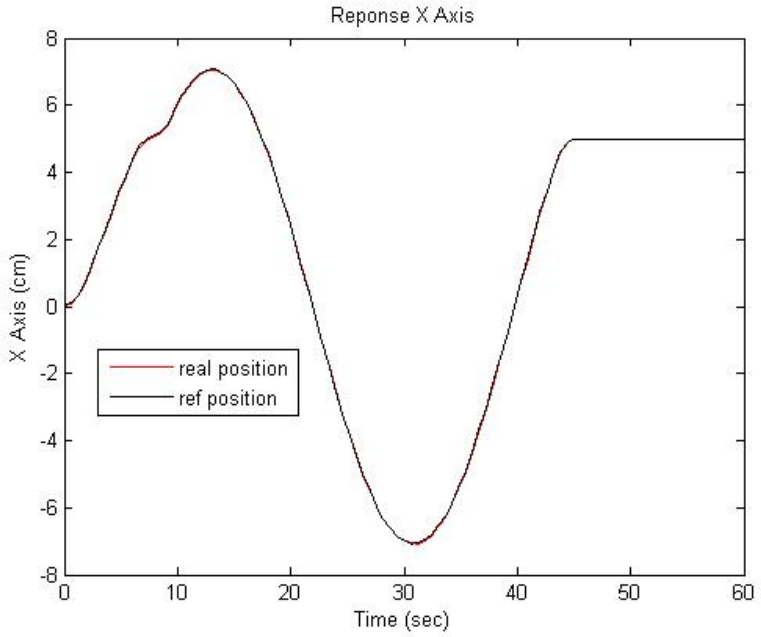
จากนั้นจึงเคลื่อนที่เป็นวงกลมโดยใช้รหัสจี

G02 X5 Y5 I-5 J-5 F10 ในการวาดวงกลม

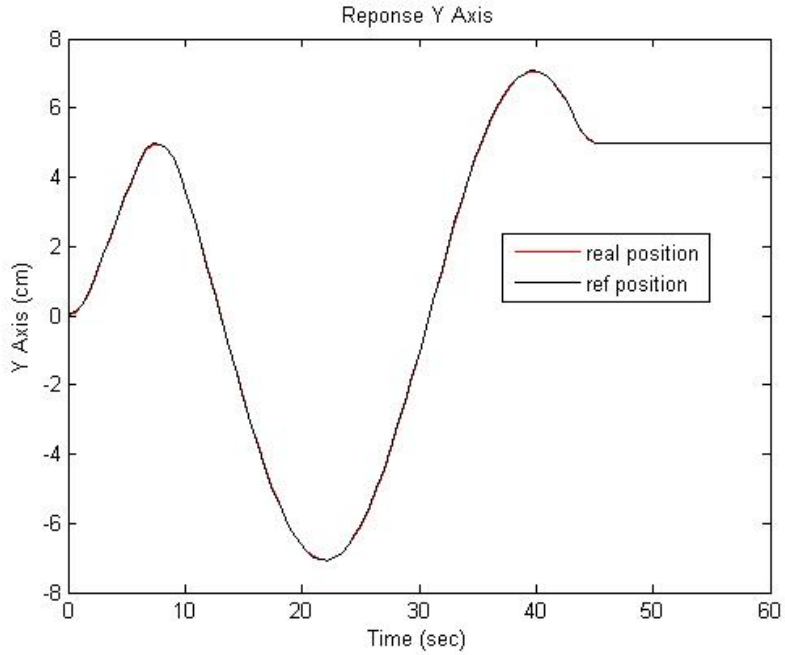
ซึ่งแสดงดังภาพที่ 88 และได้ผลตอบสนองการเคลื่อนของแกน X,Y ดังภาพที่ 89, 90 ตามลำดับ ส่วนในภาพที่ 91, 92 แสดงให้เห็นค่าความผิดพลาดของการเคลื่อนที่ของ X, Y ตามลำดับ ภาพที่ 93, 94 แสดงการเปรียบเทียบระยะที่เคลื่อนที่ได้จริงเทียบกับระยะที่ต้องการให้เคลื่อนที่พร้อมกับแสดงค่าความผิดพลาดตามลำดับ ภาพที่ 95 แสดงการเคลื่อนที่จริงบนแท่น XYZ Table



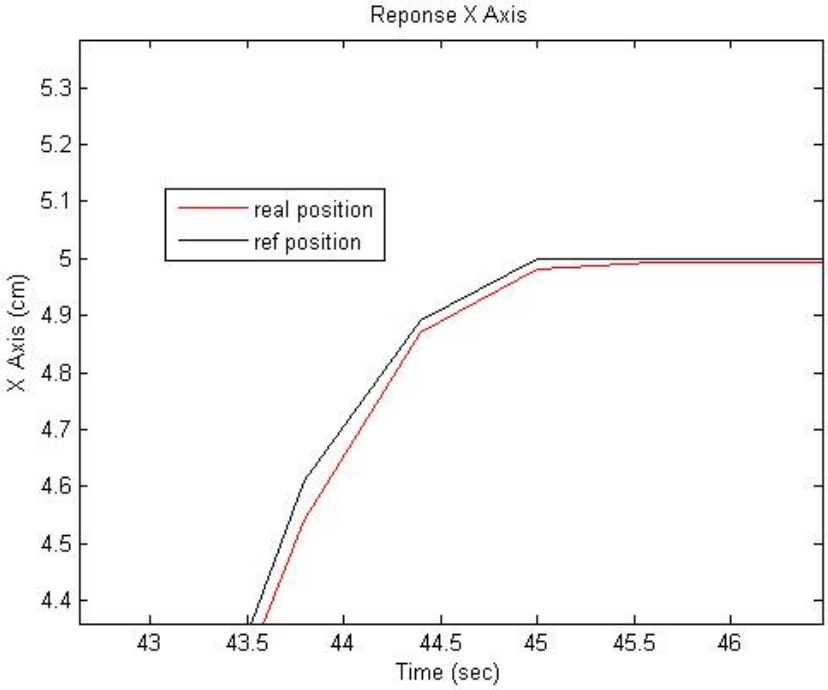
ภาพที่ 88 GUI รูปวงกลมรัศมี 7.0711 ซม.



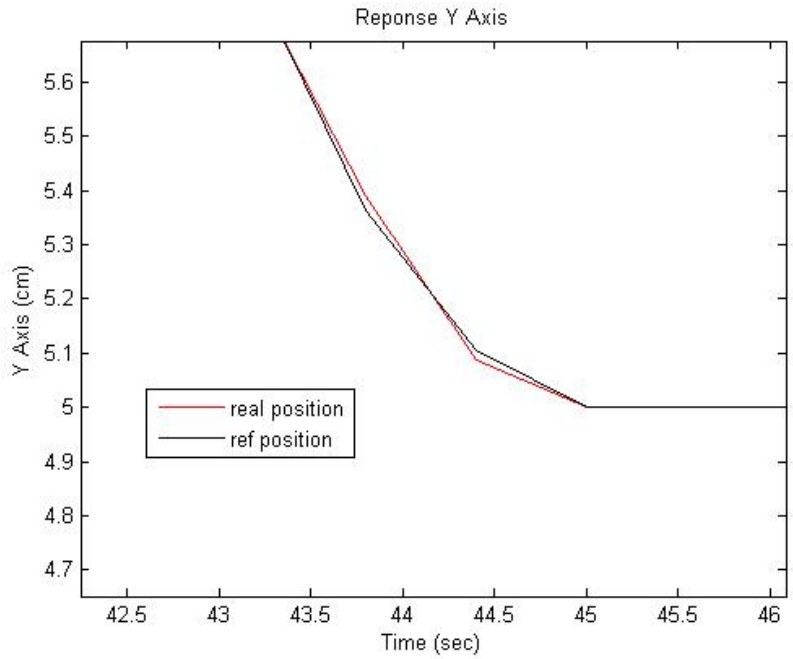
ภาพที่ 89 ผลตอบสนองของแกน X เมื่อเคลื่อนที่เป็นรูปวงกลมรัศมี 7.0711 ซม.



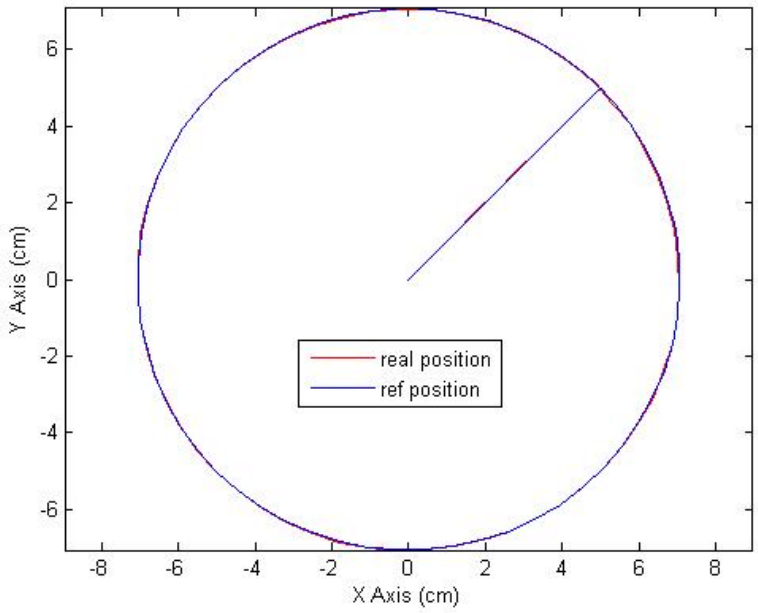
ภาพที่ 90 ผลตอบสนองของแกน Y เมื่อเคลื่อนที่เป็นรูปวงกลมรัศมี 7.0711 ซม.



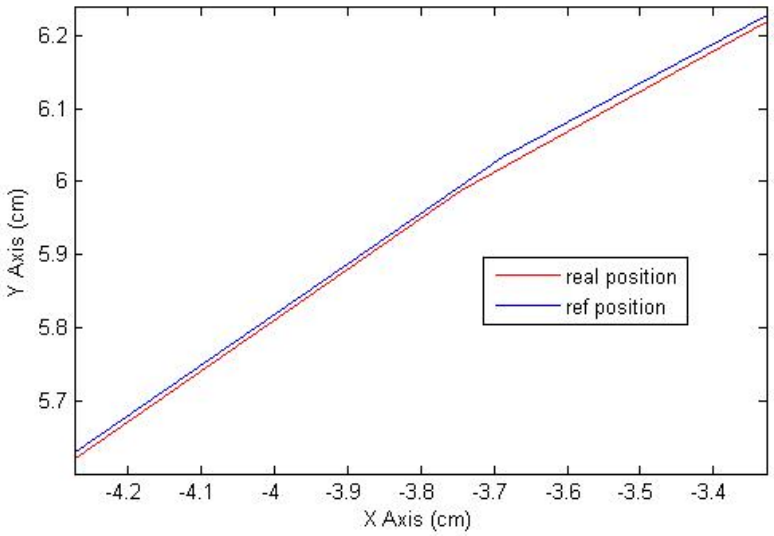
ภาพที่ 91 ค่าความผิดพลาดของแกน X เมื่อเคลื่อนที่เป็นรูปวงกลมรัศมี 7.0711 ซม.



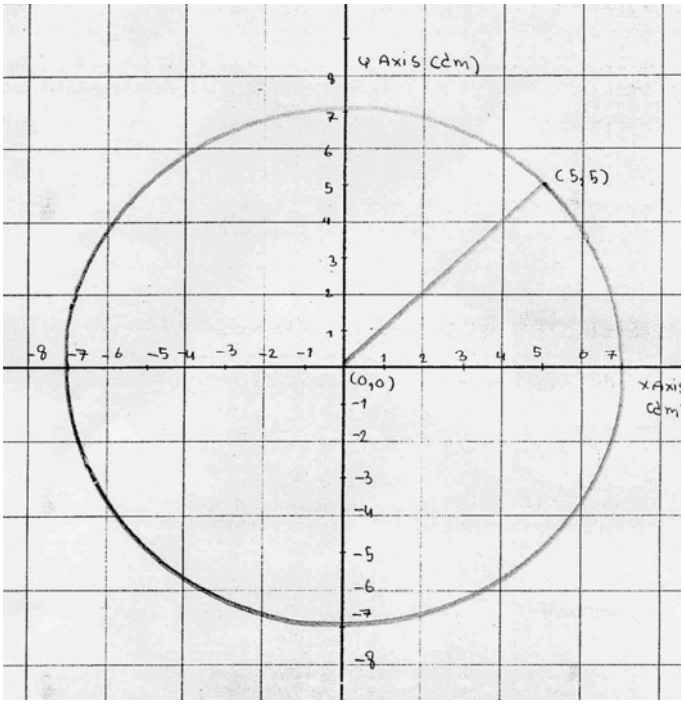
ภาพที่ 92 ค่าความผิดพลาดของแกน Y เมื่อเคลื่อนที่เป็นรูปวงกลมรัศมี 7.0711 ซม.



ภาพที่ 93 เปรียบเทียบระยะที่เคลื่อนที่เป็นวงกลมรัศมี 7.0711 ซม.



ภาพที่ 94 ค่าความผิดพลาดของการเคลื่อนที่เป็นวงกลมรัศมี 7.0711 ซม.



ภาพที่ 95 การเคลื่อนที่จริงเป็นวงกลมรัศมี 7.0711 ซม.

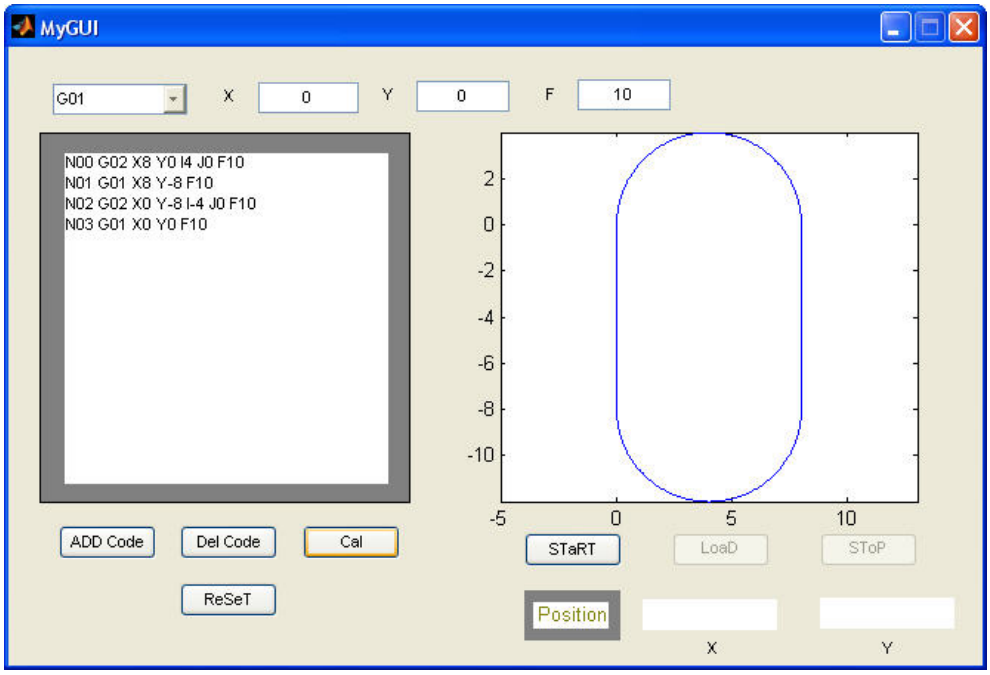
1.5 รูปประยุกต์อื่นๆ 1 (รูปแคปซูล)

เราจะใส่รหัสจีใน GUI ของ Matlab โดยใช้รหัสจีดังต่อไปนี้

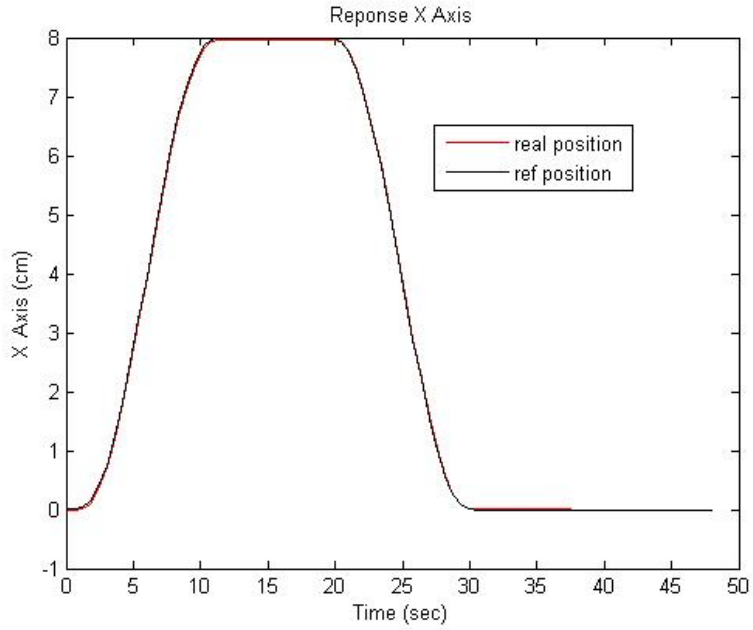
เราเริ่มที่จุด Origin (0, 0)

G02 X8 Y0 I4 J0 F10	เป็นการเคลื่อนที่เป็นวงกลมไปที่จุด (8, 0)
G01 X8 Y-8 F10	เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (8, -8)
G02 X0 Y-8 I4 J0 F10	เป็นการเคลื่อนที่เป็นวงกลมไปที่จุด (0, -8)
G01 X0 Y0 F10	เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (0, 0)

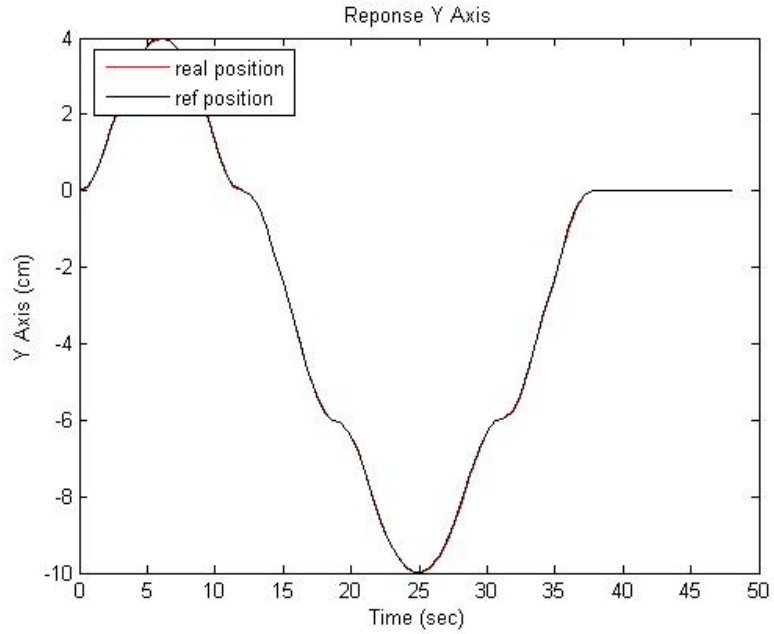
ซึ่งแสดงดังภาพที่ 96 และได้ผลตอบสนองการเคลื่อนของแกน X,Y ดังภาพที่ 97, 98 ตามลำดับ ส่วนในภาพที่ 99, 100 แสดงให้เห็นค่าความผิดพลาดของการเคลื่อนที่ของ X, Y ตามลำดับ ภาพที่ 101, 102 แสดงการเปรียบเทียบระยะที่เคลื่อนที่ได้จริงเทียบกับระยะที่ต้องการให้เคลื่อนที่พร้อมกับแสดงค่าความผิดพลาดตามลำดับ ภาพที่ 103 แสดงการเคลื่อนที่จริงบนแท่น XYZ Table



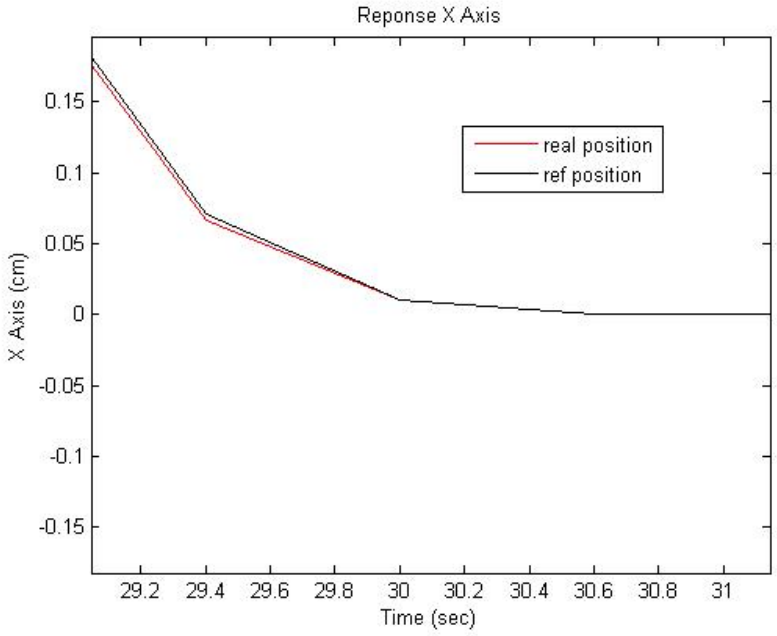
ภาพที่ 96 GUI รูปแคปซูล



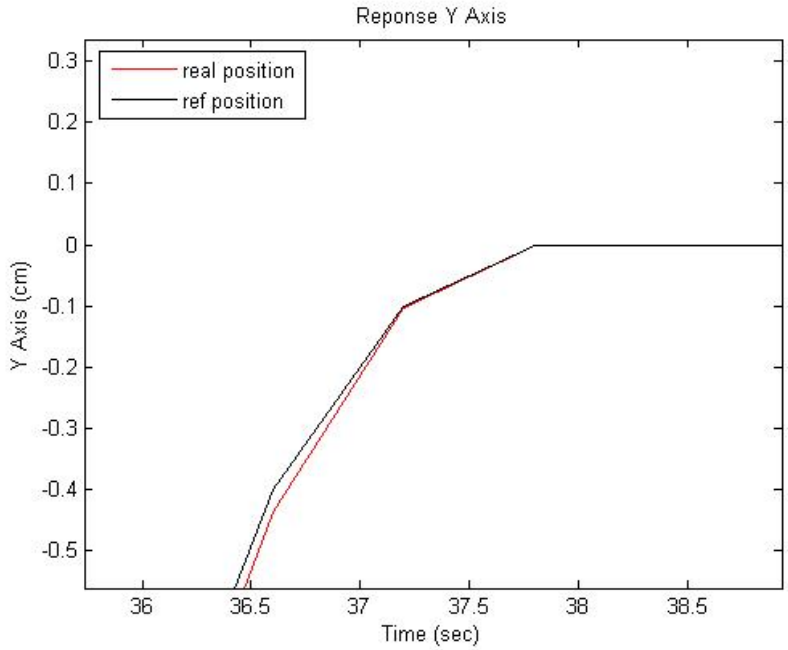
ภาพที่ 97 ผลตอบสนองของแกน X เมื่อเคลื่อนที่เป็นรูปแคปซูล



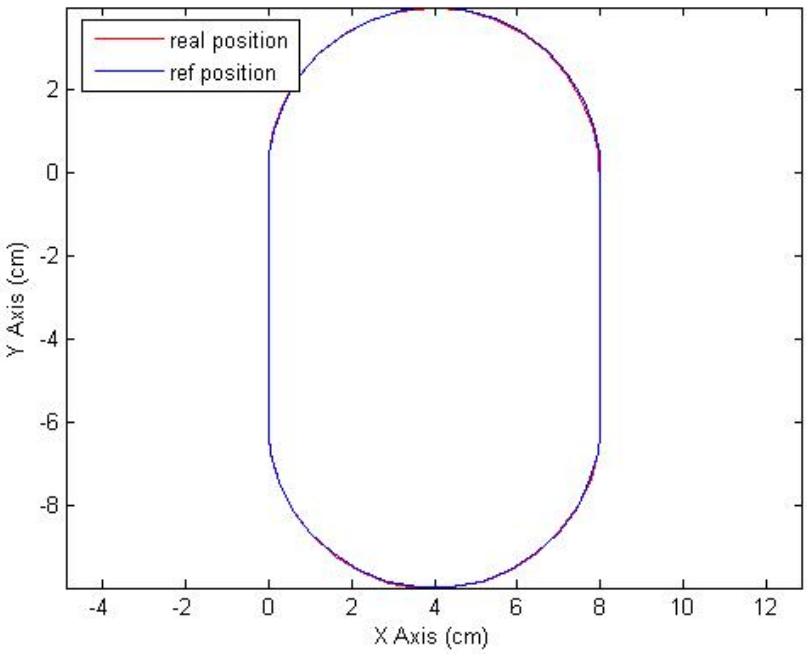
ภาพที่ 98 ผลตอบสนองของแกน Y เมื่อเคลื่อนที่เป็นรูปแคปซูล



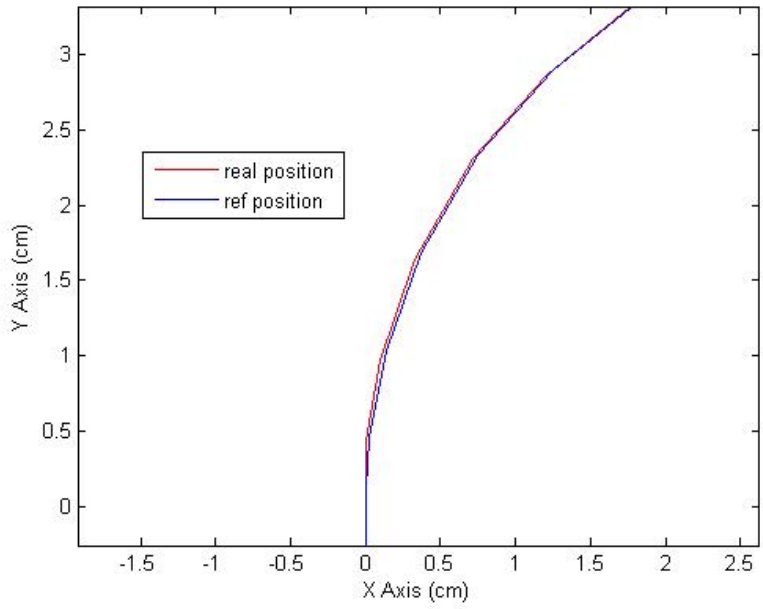
ภาพที่ 99 ค่าความผิดพลาดของแกน X เมื่อเคลื่อนที่เป็นรูปแคปซูล



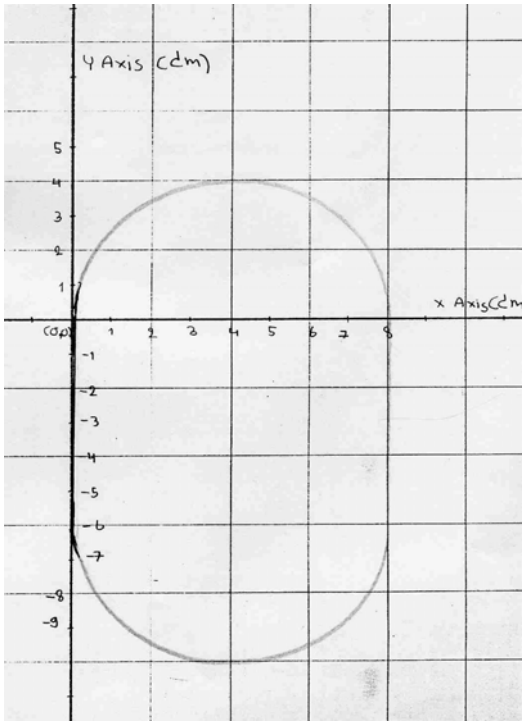
ภาพที่ 100 ค่าความผิดพลาดของแกน Y เมื่อเคลื่อนที่เป็นรูปแคปซูล



ภาพที่ 101 เปรียบเทียบระยะที่เคลื่อนที่เป็นรูปแคปซูล



ภาพที่ 102 ค่าความผิดพลาดของการเคลื่อนที่เป็นรูปแคปซูล



ภาพที่ 103 การเคลื่อนที่จริงเป็นรูปแคปซูล

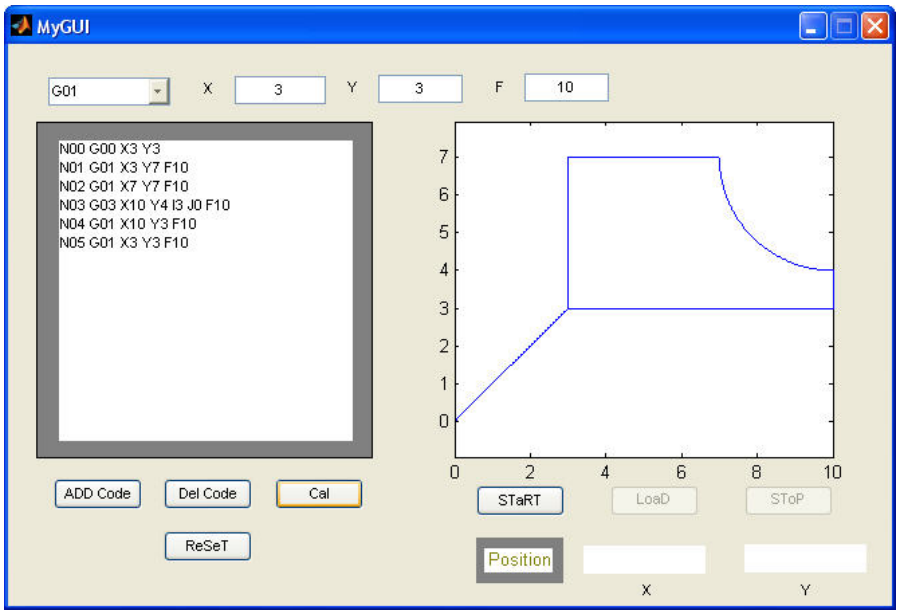
1.6 รูปประยุกต์อื่นๆ 2

เราจะใส่รหัสจีใน GUI ของ Matlab โดยใช้รหัสจีดังต่อไปนี้

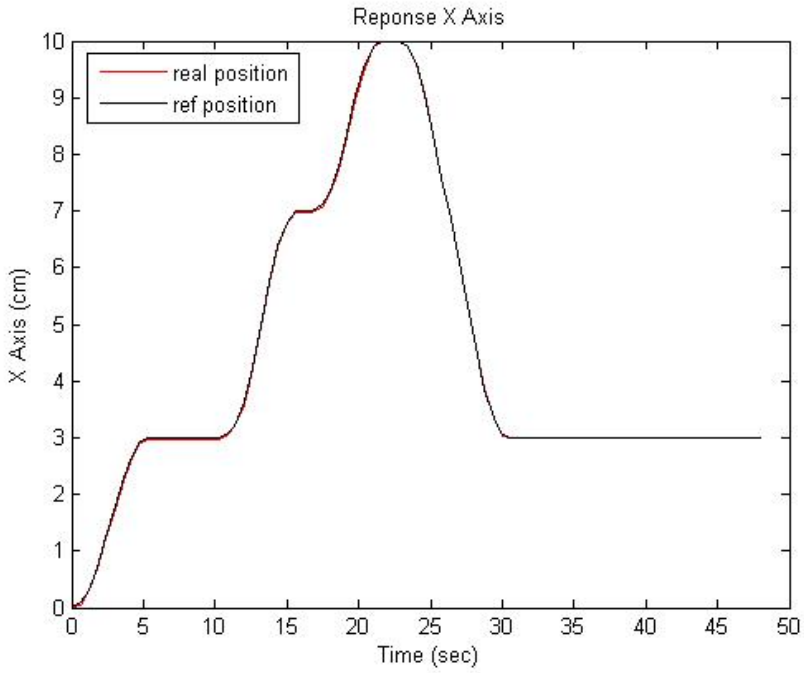
เราเริ่มที่จุด Origin (0, 0)

G00 X3 Y3	เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (3, 3)
G01 X3 Y7 F10	เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (3, 7)
G01 X7 Y7 F10	เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (7, 7)
G03 X10 Y4 I3 J0 F10	เป็นการเคลื่อนที่เป็นวงกลมไปที่จุด (10, 4)
G01 X10 Y3 F10	เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (10, 3)
G01 X3 Y3 F10	เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (3, 3)

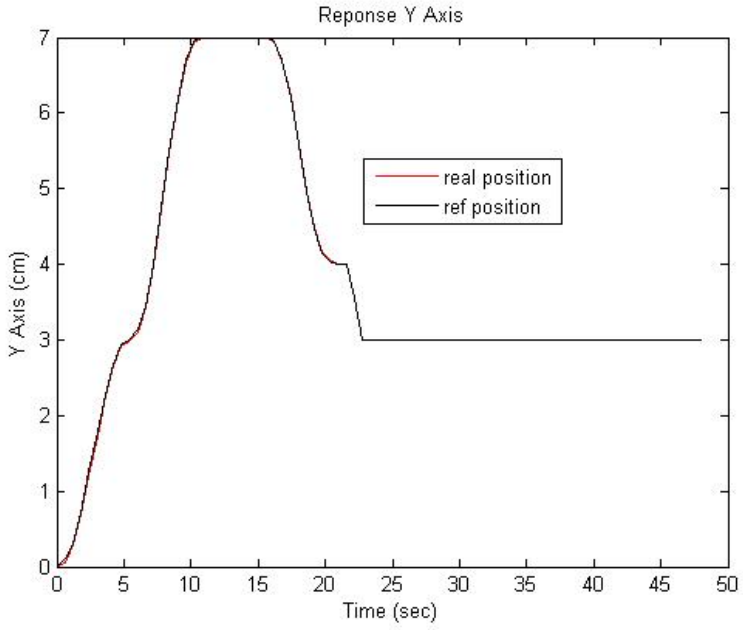
ซึ่งแสดงดังภาพที่ 104 และได้ผลตอบสนองการเคลื่อนของแกน X,Y ดังภาพที่ 105, 106 ตามลำดับ ส่วนในภาพที่ 107, 108 แสดงให้เห็นค่าความผิดพลาดของการเคลื่อนที่ของ X, Y ตามลำดับ ภาพที่ 109, 110 แสดงการเปรียบเทียบระยะที่เคลื่อนที่ได้จริงเทียบกับระยะที่ต้องการให้เคลื่อนที่พร้อมกับแสดงค่าความผิดพลาดตามลำดับ ภาพที่ 111 แสดงการเคลื่อนที่จริงบนแท่น XYZ Table



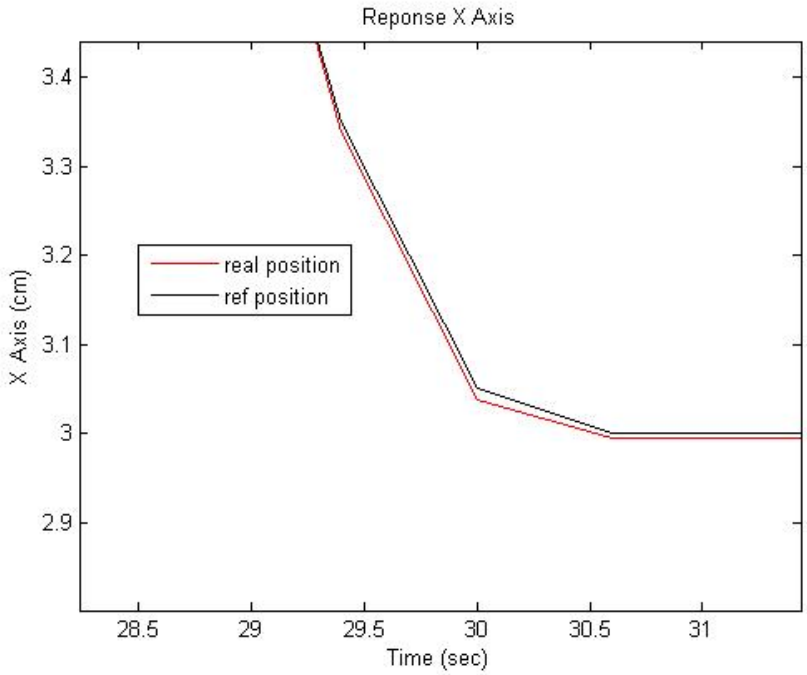
ภาพที่ 104 GUI รูปประยุกต์อื่นๆ 2



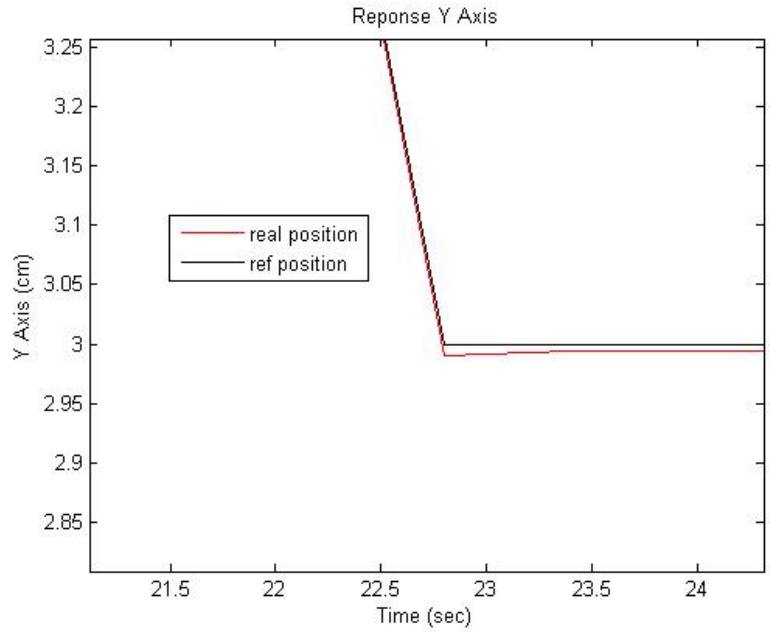
ภาพที่ 105 ผลตอบสนองของแกน X เมื่อเคลื่อนที่เป็นรูปประยุกต์อื่นๆ 2



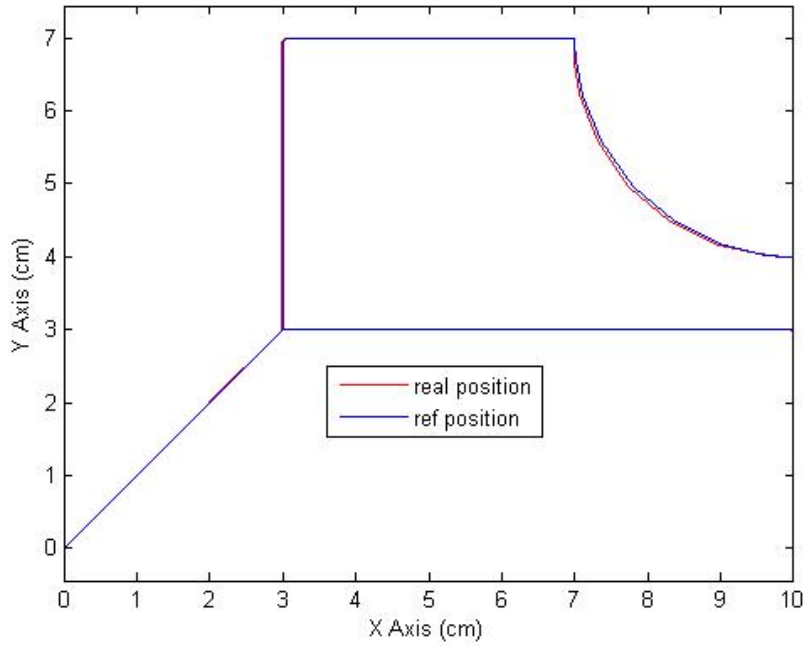
ภาพที่ 106 ผลตอบสนองของแกน Y เมื่อเคลื่อนที่เป็นรูปประยุกต์อื่นๆ 2



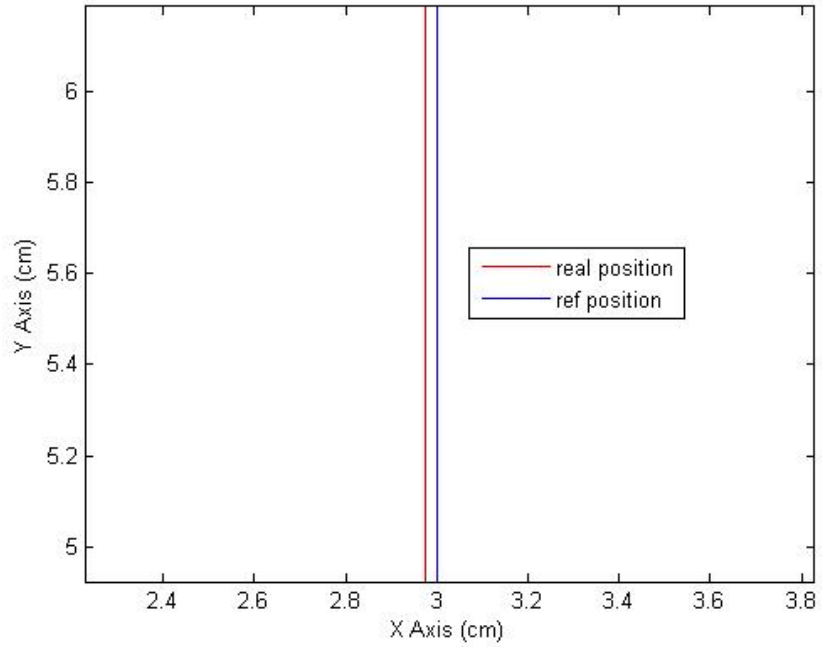
ภาพที่ 107 ค่าความผิดพลาดของแกน X เมื่อเคลื่อนที่เป็นรูปประยุกต์อื่นๆ 2



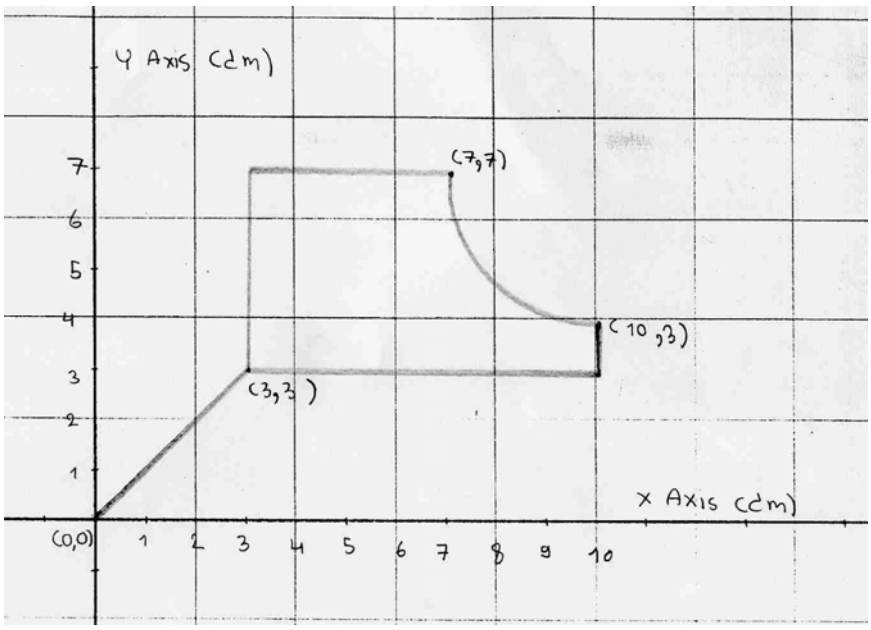
ภาพที่ 108 ค่าความผิดพลาดของแกน Y เมื่อเคลื่อนที่เป็นรูปประยุกต์อื่นๆ 2



ภาพที่ 109 เปรียบเทียบระยะที่เคลื่อนที่เป็นรูปประยุกต์อื่นๆ 2



ภาพที่ 110 ค่าความผิดพลาดของการเคลื่อนที่เป็นรูปประยุกต์อื่นๆ 2



ภาพที่ 111 การเคลื่อนที่จริงเป็นรูปประยุกต์อื่นๆ 2

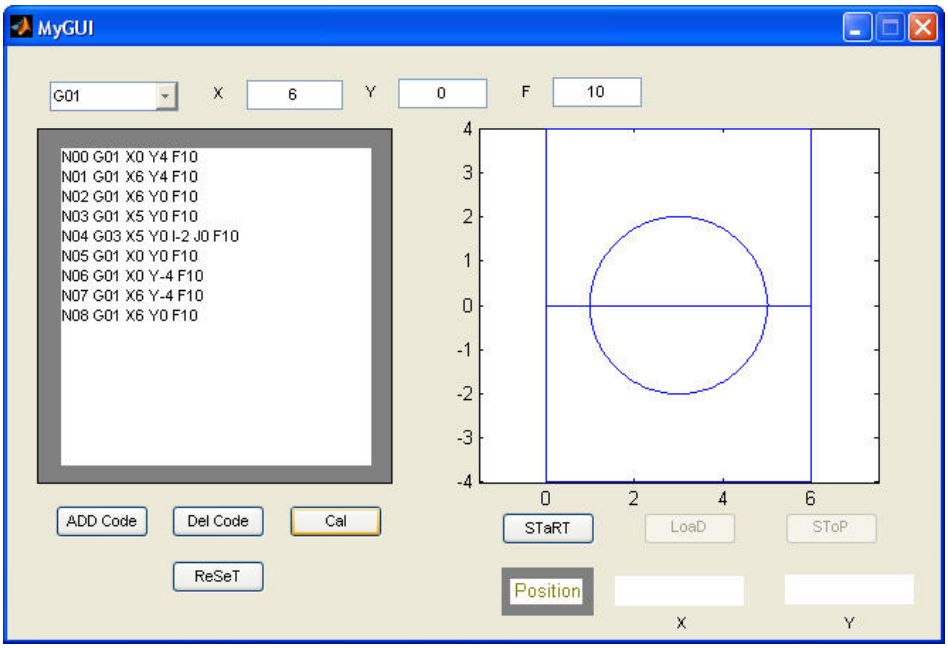
1.7 รูปประยุกต์อื่นๆ 3 (รูปสนามฟุตบอล)

เราจะใส่รหัสจีใน GUI ของ Matlab โดยใช้รหัสจีดังต่อไปนี้

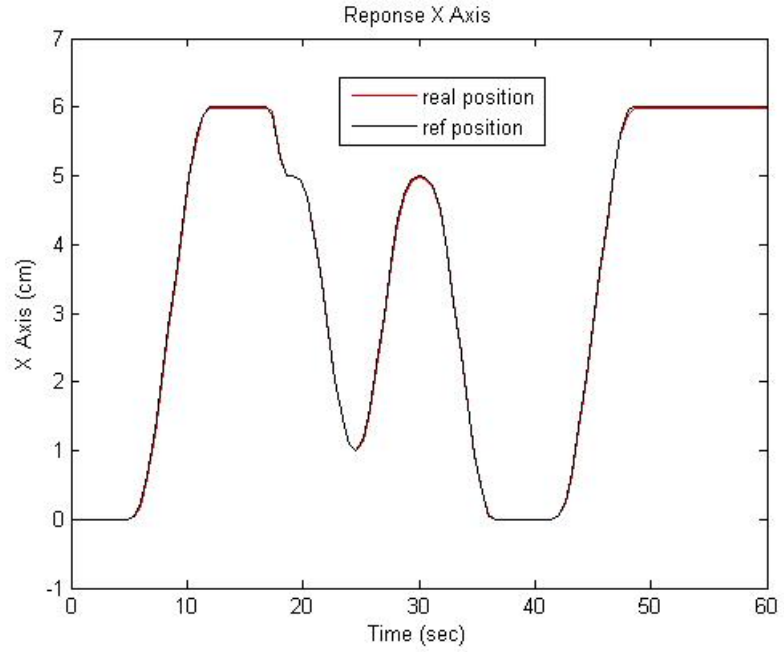
เราเริ่มที่จุด Origin (0, 0)

G01 X0 Y4 F10	เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (0, 4)
G01 X6 Y4 F10	เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (6, 4)
G01 X6 Y0 F10	เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (6, 0)
G01 X5 Y0 F10	เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (5, 0)
G03 X5 Y0 I-2 J0 F10	เป็นการเคลื่อนที่เป็นวงกลมทิศทางทวนเข็มนาฬิกา
G01 X0 Y0 F10	เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (0, 0)
G01 X0 Y-4 F10	เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (0, -4)
G01 X6 Y-4 F10	เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (6, -4)
G01 X6 Y0 F10	เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (6, 0)

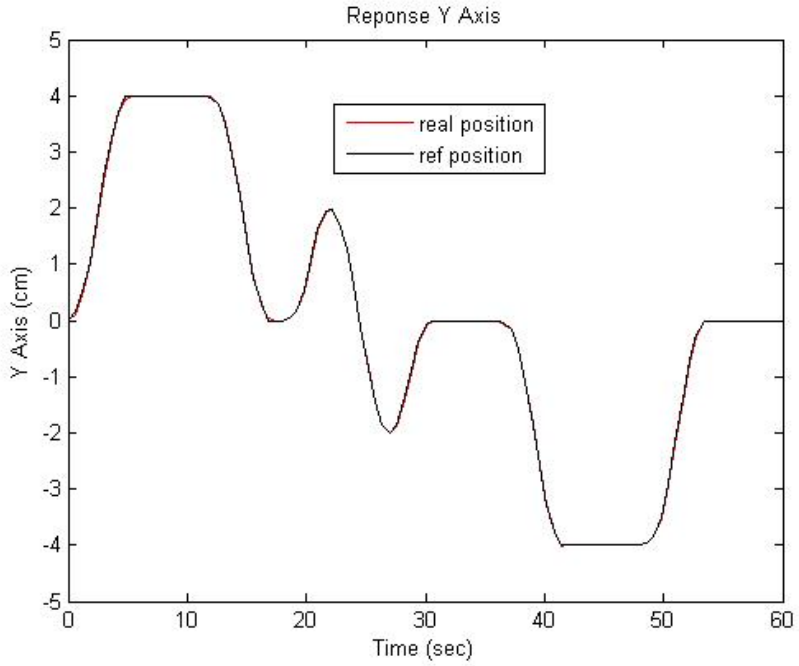
ซึ่งแสดงดังภาพที่ 112 และได้ผลตอบสนองการเคลื่อนของแกน X,Y ดังภาพที่ 113, 114 ตามลำดับ ส่วนในภาพที่ 115, 116 แสดงให้เห็นค่าความผิดพลาดของการเคลื่อนที่ของ X, Y ตามลำดับ ภาพที่ 117, 118 แสดงการเปรียบเทียบระยะที่เคลื่อนที่ได้จริงเทียบกับระยะที่ต้องการให้เคลื่อนที่พร้อมกับแสดงค่าความผิดพลาดตามลำดับ ภาพที่ 119 แสดงการเคลื่อนที่จริงบนแท่น XYZ Table



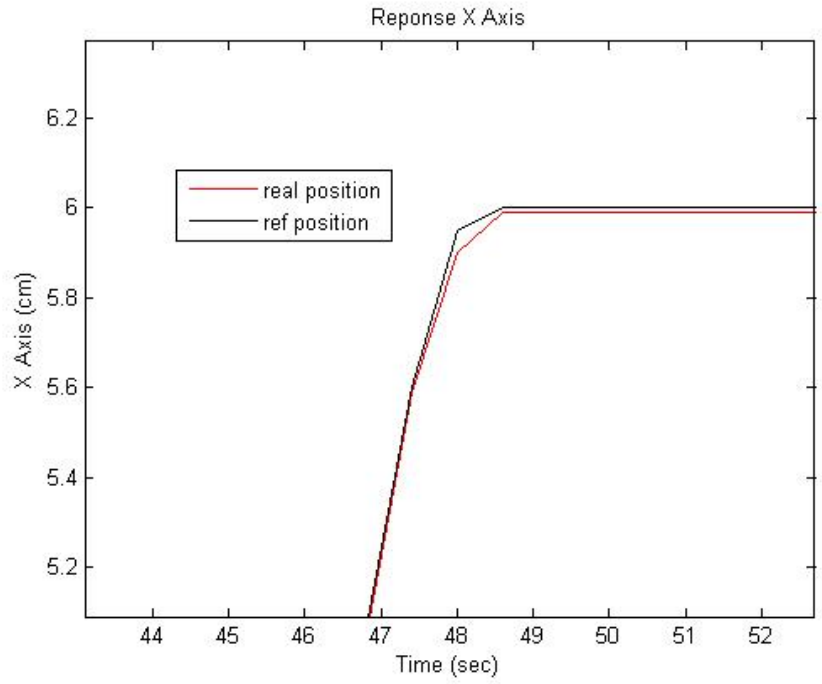
ภาพที่ 112 GUI รูปสนามฟุตบอล



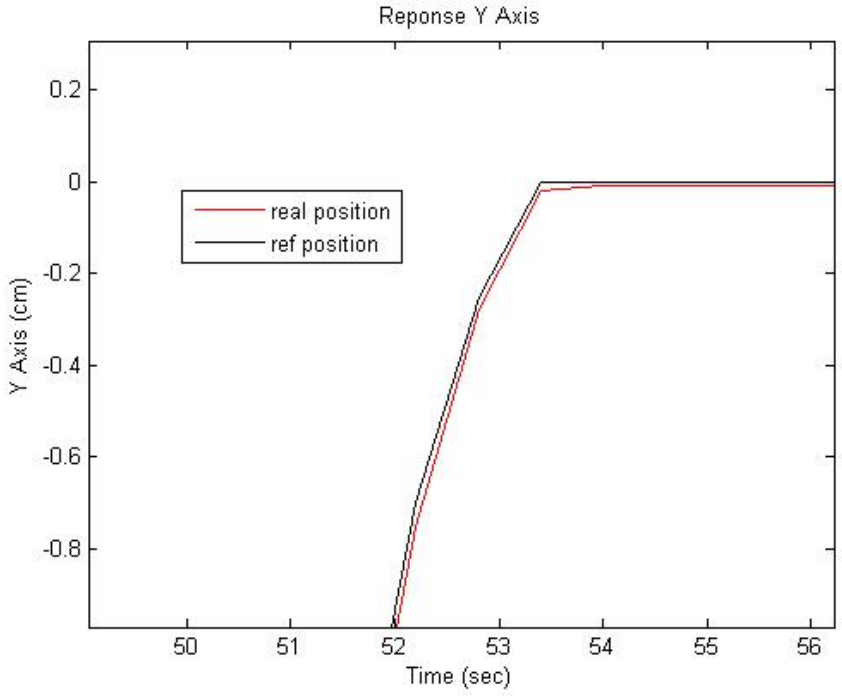
ภาพที่ 113 ผลตอบสนองของแกน X เมื่อเคลื่อนที่เป็นรูปสนามฟุตบอล



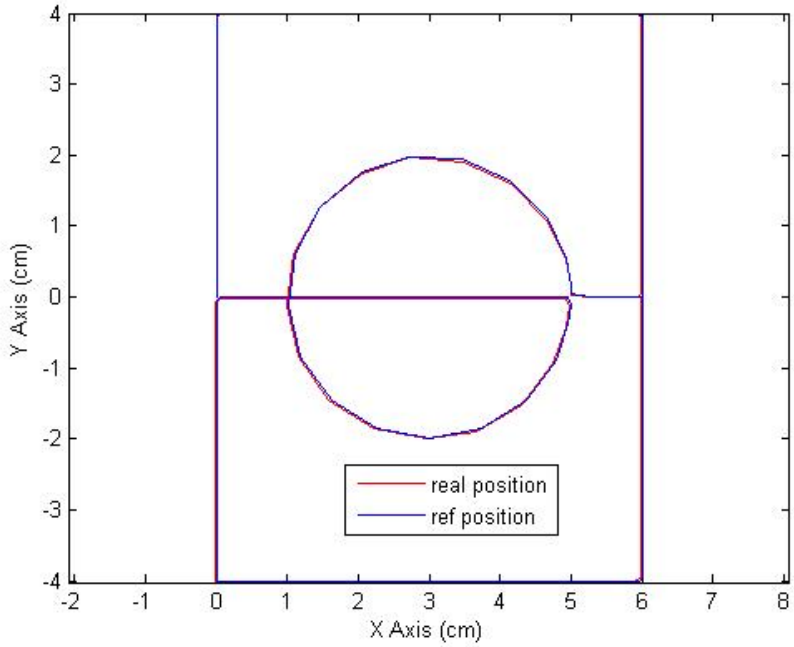
ภาพที่ 114 ผลตอบสนองของแกน Y เมื่อเคลื่อนที่เป็นรูปสนามฟุตบอล



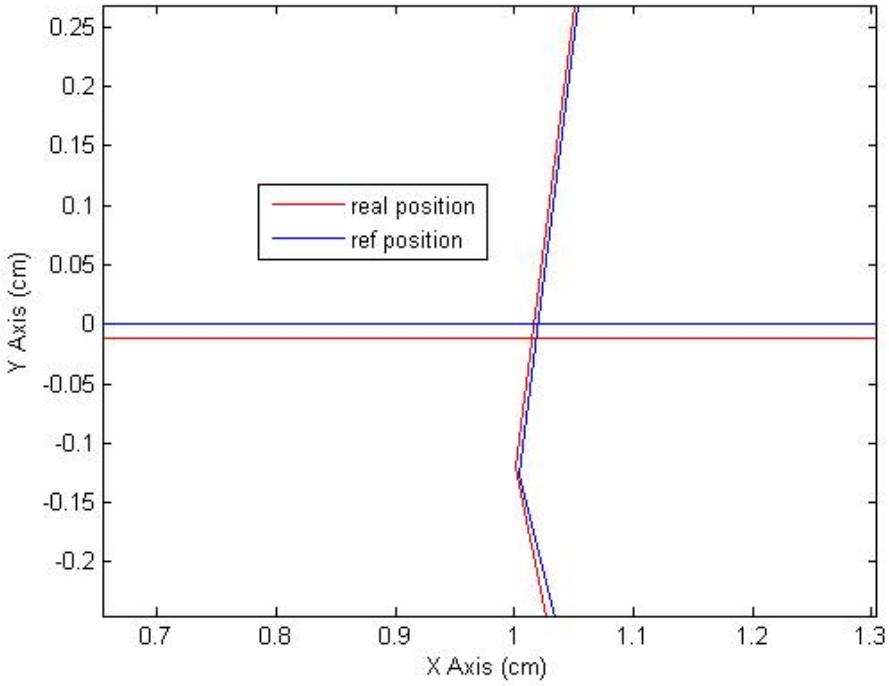
ภาพที่ 115 ค่าความผิดพลาดของแกน X เมื่อเคลื่อนที่เป็นรูปสนามฟุตบอล



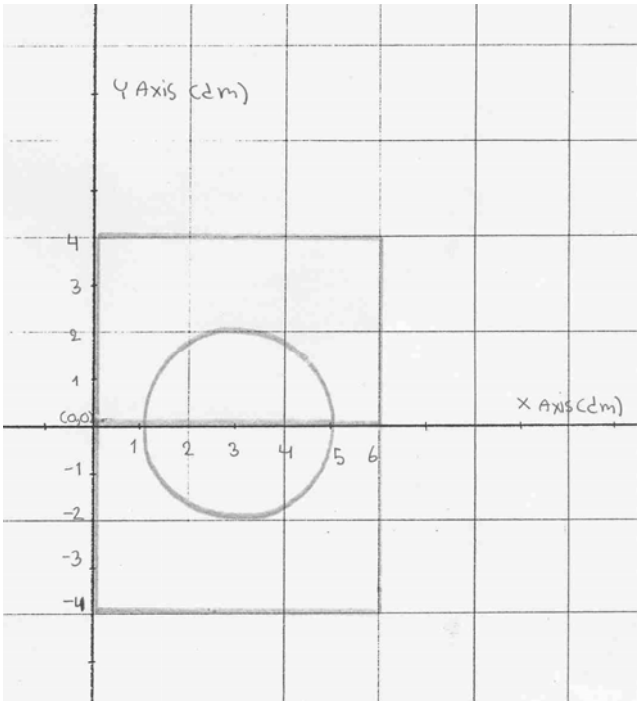
ภาพที่ 116 ค่าความผิดพลาดของแกน Y เมื่อเคลื่อนที่เป็นรูปสนามฟุตบอล



ภาพที่ 117 เปรียบเทียบระยะที่เคลื่อนที่เป็นรูปสนามฟุตบอล



ภาพที่ 118 ค่าความผิดพลาดของการเคลื่อนที่เป็นรูปสนามฟุตบอล



ภาพที่ 119 การเคลื่อนที่จริงเป็นรูปสนามฟุตบอล

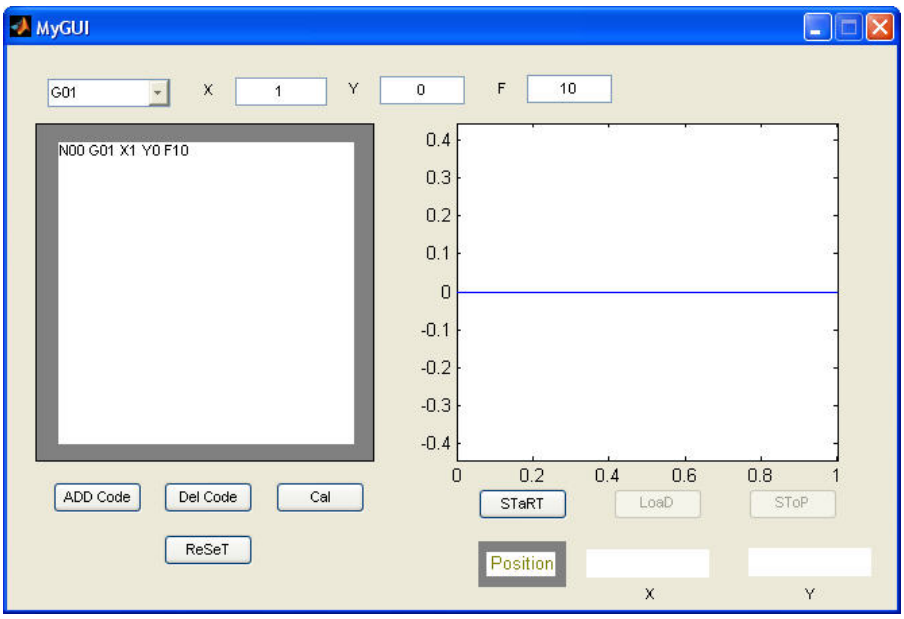
2. การทดลองสร้างการเคลื่อนที่โดยใช้ AC servo motor

2.1 รูปเส้นตรงระยะ 1 ซม.

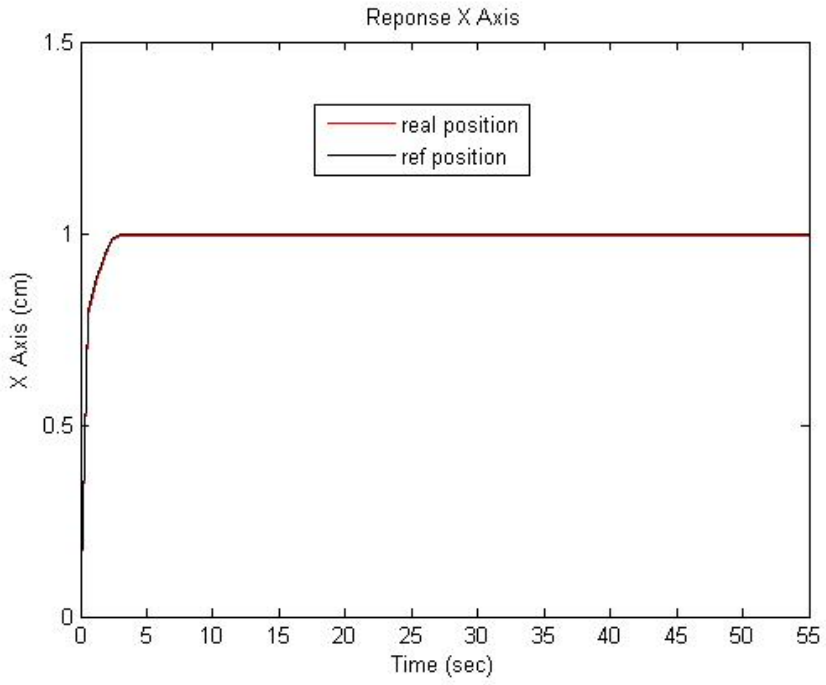
เราจะใส่รหัสจีใน GUI ของ Matlab

G01 X1 Y0 F10 เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (1, 0)

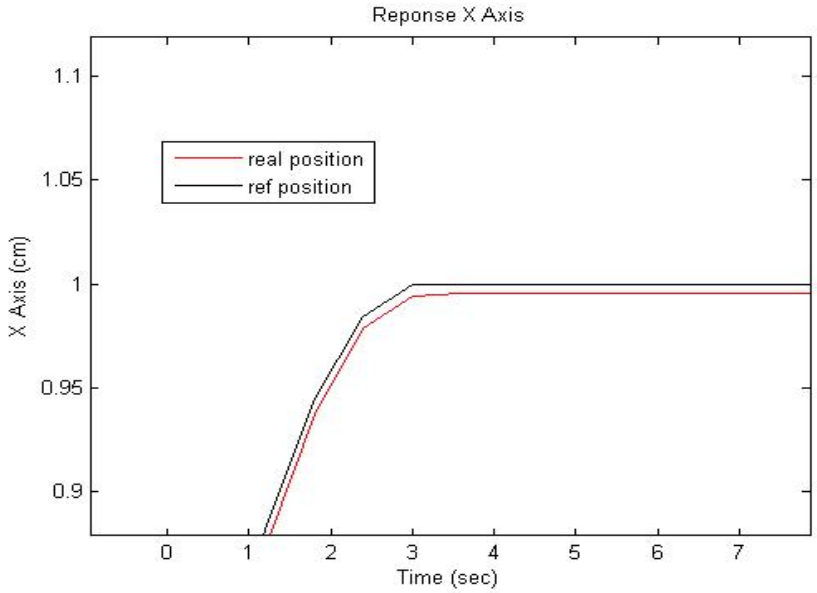
ซึ่งแสดงดังภาพที่ 120 และได้ผลตอบสนองการเคลื่อนของมอเตอร์ดังภาพที่ 121 ส่วนในภาพที่ 122 แสดงให้เห็นค่าความผิดพลาดของการเคลื่อนที่ของมอเตอร์



ภาพที่ 120 GUI การเคลื่อนที่ของ AC servo motor ระยะ 1 ซม.



ภาพที่ 121 ผลตอบสนองของ AC servo motor เมื่อเคลื่อนที่เป็นระยะ 1 ซม.



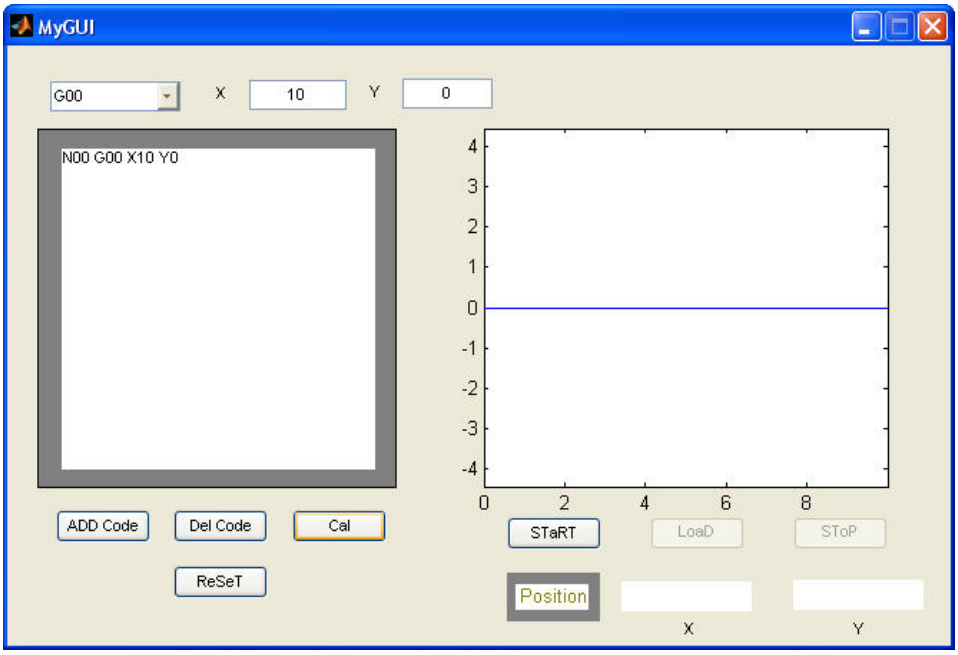
ภาพที่ 122 ค่าความผิดพลาดของ AC servo motor เมื่อเคลื่อนที่เป็นระยะ 1 ซม.

2.2 รูปเส้นตรงระยะ 10 ซม.

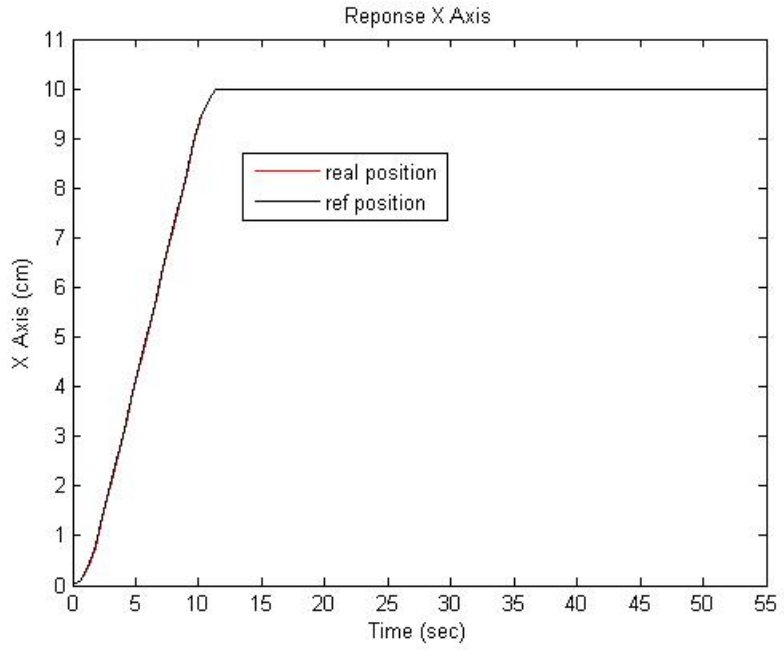
เราจะใส่รหัสจีใน GUI ของ Matlab

G01 X10 Y0 F10 เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (10, 0)

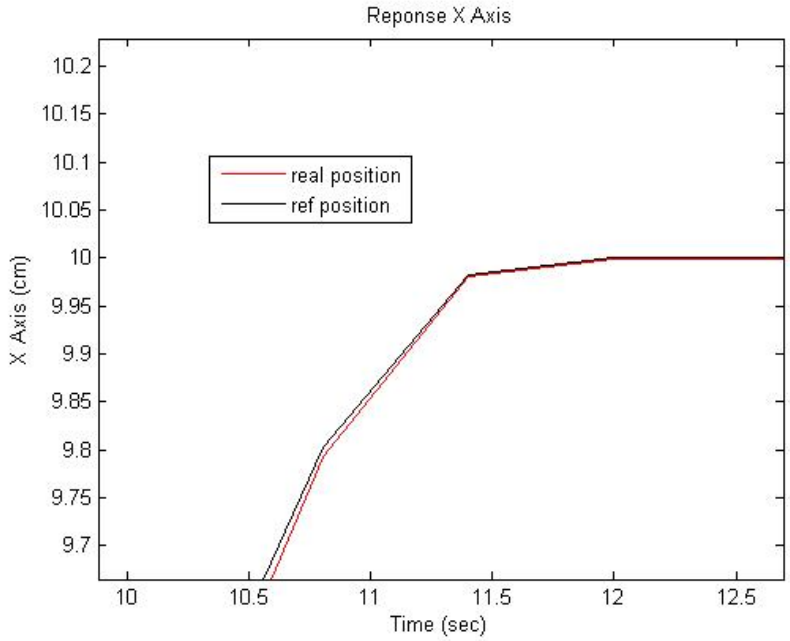
ซึ่งแสดงดังภาพที่ 123 และได้ผลตอบสนองการเคลื่อนของมอเตอร์ดังภาพที่ 124 ส่วนในภาพที่ 125 แสดงให้เห็นค่าความผิดพลาดของการเคลื่อนที่ของมอเตอร์



ภาพที่ 123 GUI การเคลื่อนที่ของ AC servo motor ระยะ 10 ซม.



ภาพที่ 124 ผลตอบสนองของ AC servo motor เมื่อเคลื่อนที่เป็นระยะ 10 ซม.



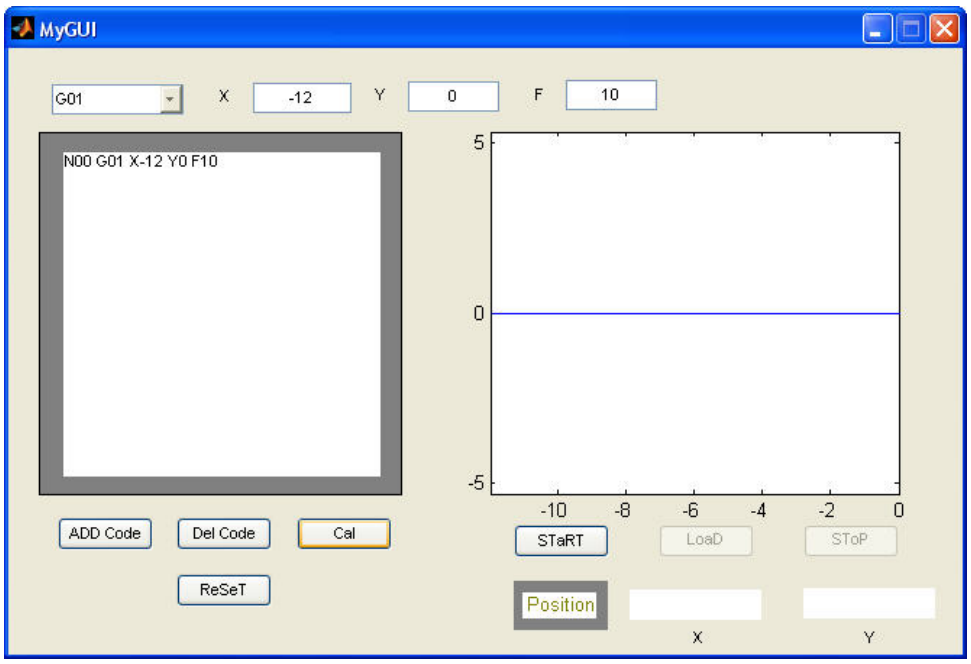
ภาพที่ 125 ค่าความผิดพลาดของ AC servo motor เมื่อเคลื่อนที่เป็นระยะ 10 ซม.

2.3 รูปเส้นตรงระยะ -12 ซม.

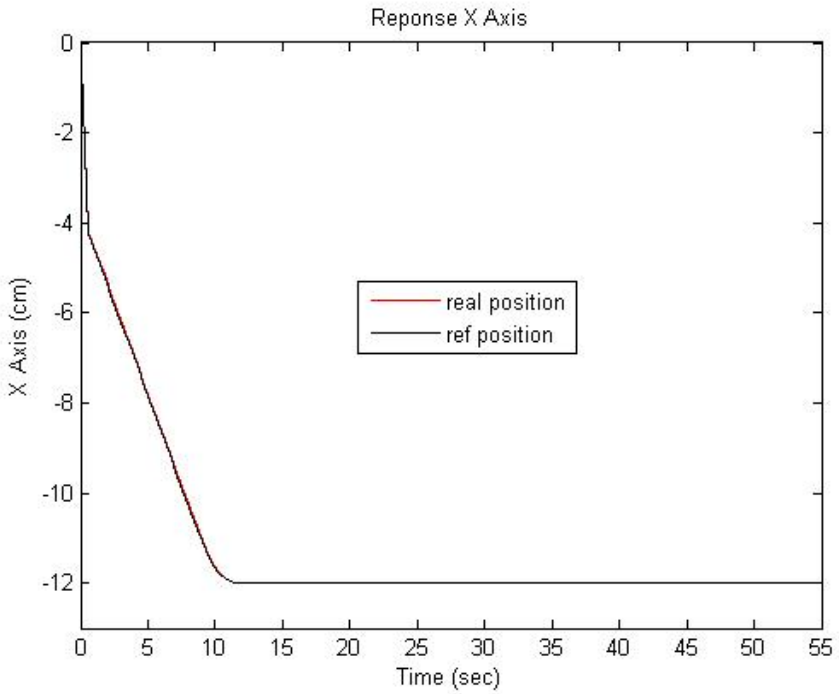
เราจะใส่รหัสจีใน GUI ของ Matlab

G01 X-12 Y0 F10 เป็นการเคลื่อนที่เป็นเส้นตรงไปที่จุด (-12, 0)

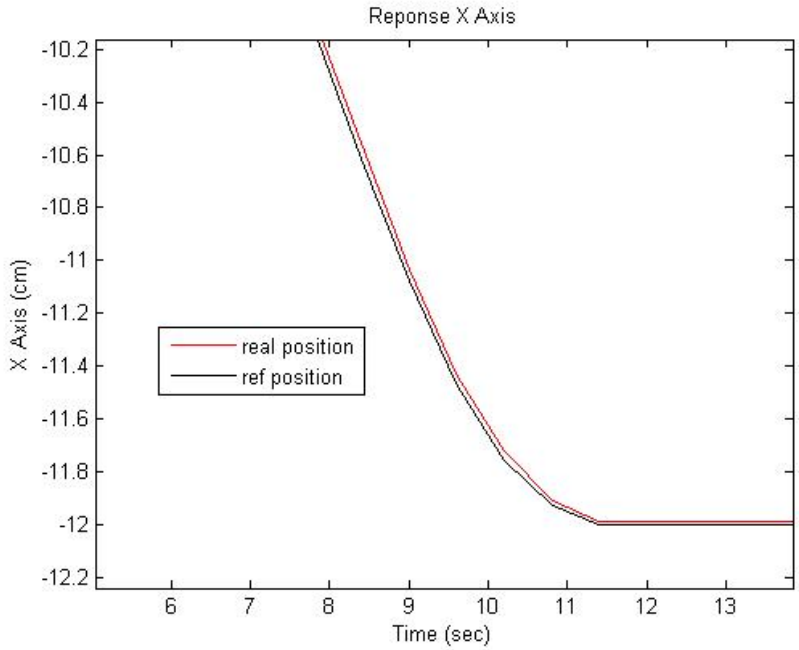
ซึ่งแสดงดังภาพที่ 126 และได้ผลตอบสนองการเคลื่อนของมอเตอร์ดังภาพที่ 127 ส่วนในภาพที่ 128 แสดงให้เห็นค่าความผิดพลาดของการเคลื่อนที่ของมอเตอร์



ภาพที่ 126 GUI การเคลื่อนที่ของ AC servo motor ระยะ -12 ซม.



ภาพที่ 127 ผลตอบสนองของ AC servo motor เมื่อเคลื่อนที่เป็นระยะ -12 ซม.



ภาพที่ 128 ค่าความผิดพลาดของ AC servo motor เมื่อเคลื่อนที่เป็นระยะ -12 ซม.

3. ค่าความผิดพลาดทางตัวเลขของการทดลอง

เราสามารถสรุปค่าความผิดพลาดในการติดตามการเคลื่อนที่ดังตารางที่ 2 และ 3 โดยแสดงค่าความผิดพลาดในการติดตามมากสุด, ความผิดพลาดในการติดตามน้อยสุด และความผิดพลาดในการติดตามแบบ rms (root mean squares) ซึ่งแสดงด้วยสมการ

$$
\begin{aligned}
e(n) &= real\ position - referance\ position\ at\ time\ n \\
e_{\min} &= \min_{n} |e(n)| \\
e_{\max} &= \max_{n} |e(n)| \\
e_{rms} &= \sqrt{\frac{\sum_{i=1}^{n} e^2(n)}{N}}
\end{aligned}
\tag{53}
$$

โดยตารางที่ 2 เป็นการเคลื่อนที่ของ DC servo motor และตารางที่ 3 เป็นการเคลื่อนที่ของ AC servo motor

ตารางที่ 2 สรุปค่าความผิดพลาดในการเคลื่อนที่ของ DC servo motor

การทดลองที่	รูปแบบการเคลื่อนที่ DC servo motor	ค่าความผิดพลาดของแกน X			ค่าความผิดพลาดของแกน Y		
		min error (cm)	max error (cm)	rms error (cm)	min error (cm)	max error (cm)	rms error (cm)
1.1	รูปเส้นตรงแกน X 5 ซม. แกน Y 5 ซม.	0.0027	0.0280	0.0056	0.0006	0.03	0.0054
1.2	รูปเส้นตรงแกน X -8 ซม. แกน Y -8 ซม.	0.00055	0.0196	0.0024	0.000075	0.0177	0.0025
1.3	รูปวงกลมรัศมี 5.6569 ซม.	0.00032	0.0390	0.0029	0.00045	0.034	0.0039
1.4	รูปวงกลมรัศมี 7.0711 ซม.	0.00005	0.032	0.002	0.00017	0.031	0.0025
1.5	รูปประยุกต์อื่นๆ 1 (รูปแคปซูล)	0	0.029	0.0031	0.000027	0.033	0.0028
1.6	รูปประยุกต์อื่นๆ 2	0.000075	0.043	0.0035	0.000037	0.048	0.0028
1.7	รูปประยุกต์อื่นๆ 3 (รูปสนามฟุตบอล)	0	0.041	0.0023	0.000036	0.043	0.0022

ตารางที่ 3 สรุปค่าความผิดพลาดในการเคลื่อนที่ของ AC servo motor

การทดลองที่	รูปแบบการเคลื่อนที่ AC servo motor	ค่าความผิดพลาดในการติดตาม		
		min error (cm)	max error (cm)	rms error (cm)
2.1	รูปเส้นตรงระยะ 1 ซม.	0.000040	0.035	0.0024
2.2	รูปเส้นตรงระยะ 10 ซม.	0.000012	0.031	0.0032
2.3	รูปเส้นตรงระยะ -12 ซม.	0.000032	0.037	0.0022

จากตารางที่ 2 เราจะเห็นว่าค่าความผิดพลาดในการติดตามมากที่สุดของการเคลื่อนที่แกน X ของ DC servo motor เป็น 0.043 ซม. และค่าความผิดพลาดในการติดตามมากที่สุดของการเคลื่อนที่แกน Y ของ DC servo motor เป็น 0.048 ซม. ส่วนในตารางที่ 3 เราจะเห็นว่าค่าความผิดพลาดในการติดตามมากที่สุดของการเคลื่อนที่ของ AC servo motor เป็น 0.037 ซม.

สรุป

การศึกษาการออกแบบชุดควบคุม CNC 3 แกนนี้จะช่วยให้ผู้ศึกษาเข้าใจทฤษฎีควบคุมมากขึ้น นอกจากนี้ยังสามารถที่จะนำทฤษฎีควบคุมอื่นมาออกแบบเพื่อให้สามารถควบคุมระบบนี้ได้ อีกทั้งยังสามารถนำชุดทดลองอุปกรณ์ต้นแบบนี้มาศึกษาหาเอกลักษณ์ของระบบได้อีกด้วย

ในงานวิจัยนี้ได้ทำการออกแบบชุดควบคุม CNC 3 แกนเพื่อที่จะได้พัฒนาเป็นเครื่องต้นแบบในการอุตสาหกรรมขนาดกลางและขนาดเล็ก และยังเป็นชุดทดลองสำหรับนิสิตอีกด้วย ผลการทดลองที่ได้จากการของสร้างเส้นทางการเคลื่อนที่มีค่าความผิดพลาดที่ค่าสุดท้ายมากสุด (steady-state error) ไม่เกิน 0.03 เซนติเมตร ซึ่งให้ผลที่น่าพอใจระดับหนึ่ง แต่ก็ยังพบปัญหาในส่วนของ Software และ Hardware

1. Connector ที่ใช้ในการต่อระหว่าง DSP board และ Extensive board เป็นสายไม่มีค่อยมีความคงทน ขาดได้ง่าย และก็มีสัญญาณรบกวน ทำให้การส่งสัญญาณบางครั้งมีความผิดพลาดบ้าง
2. ส่วนของวงจร PWM board ที่ใช้ในการขับเคลื่อนเซอร์โวมอเตอร์นั้นมีแรงบิด (torque) ที่ต่ำ ซึ่งอาจส่งผลทำให้มีความผิดพลาดที่เกิดขึ้นในการเคลื่อนที่
3. ในวงจร Extensive board สำหรับ AC servo motor นั้นมีปัญหาในการจัดการกับสัญญาณรบกวนที่มาจากตัวมอเตอร์เองโดยต้องแยก GND โดย Supply ด้าน analog และด้าน digital แยกกัน โดยใช้ชิพ isolator และ ferrite beat และจะต้องใส่ EMI filter ในการป้องการสัญญาณที่รบกวนออกมาจาก AC motor โดยตรงที่สาย power ของ supply แต่ละตัวด้วย
4. การติดต่อส่งข้อมูลระหว่าง DSP board และ Matlab ด้วยวิธี RTDX ส่งผลทำให้ การทำงานใน software ช้าลงเนื่องจากมี interrupt routine มากขึ้นและการทำงานของ RTDX ต้องผ่านทาง Library ของ CCS ด้วย การพัฒนาในอนาคตอาจจะเปลี่ยนมาใช้การติดต่อแบบ serial โดยผ่านทาง McBSP ของ DSP board
5. ค่า K_P, K_I, K_D ในสมการ PID Controller นั้นเราประมาณค่ามาจากสมการเอกลักษณ์ทั่วไปของมอเตอร์ซึ่งอาจจะมีผลทำให้มีค่าความผิดพลาดในการเคลื่อนที่ ถ้าเราใช้หาเอกลักษณ์ของระบบมาช่วยในการหาสมการเอกลักษณ์ที่แท้จริงของมอเตอร์รวมกับชุดขับที่มีความไม่เป็นเชิงเส้นคาดว่าจะส่งผลทำให้ค่าความผิดพลาดน้อยลงได้

เอกสารและสิ่งอ้างอิง

ชาลี ตระการกูล. 2548. **เทคโนโลยีซีเอ็นซี**. พิมพ์ครั้งที่ 12. สำนักพิมพ์ ส.ส.ท., กรุงเทพฯ

วโรดม ตู้จินดา. 2547. **การออกแบบสร้างและรีโทรฟิตชุดควบคุมเครื่องซีเอนซี.** 1st ed. สำนักพิมพ์ ส.ส.ท., กรุงเทพฯ.

Acosta Calderon, C.A., Rosales Pena Alfaro, Gan, E.M. and Hu, J.Q. 2003. **TRAJECTORY GENERATION AND TRACKING OF A 5-DOF ROBOTIC ARM**.

Anonymous, Brewer Science. Inc. 2003. PID Control. Available Source: http://www.brewer science.com/cee/ otherprods/cee_pid.html, November 14, 2004.

Aoki, K., Iwazawa, N., Tsujisawa, T., Sakaguchi Y. and Nakawatase, K. 1994. **High-accuracy path-tracking by multi-axis coordination in CNC machines.** Industrial Electronics, Control and Instrumentation, 1994. IECON '94., 20th International Conference on , Volume: 3 , 5-9 Sept. 1994 Pages:1842 - 1847 vol.3

Barello, Larry. 2002. Motion Control Information. Available Source: http://www.barello.net/Papers/Motion_Control/index.htm

David Bruce Newman. 1960. **The analysis of cross-coupling effects on the stability of two-dimensional, orthogonal, feedback control systems.** Automatic Control, IRE Transactions on , Volume: 5 , Issue: 4 , Sep 1960 Pages:314 – 320

Decotignie, J.-D. 1991. **Distributed path and speed control in machine-tool axis motion**. Industrial Electronics, Control and Instrumentation , 1991. Proceedings. IECON '91., 1991 International Conference on , 28 Oct.-1 Nov. 1991 Pages:772 - 777 vol.1

Frew, Eric W. and Rock, Stephen M. 2003. **Trajectory Generation for Constant Velocity Target MotionEstimation Using Monocular Vision.**

Gon-Jen Wang. and Tzong-Jing Lee. 1999. **Neural-network cross-coupled control system with application on circular tracking of linear motor X-Y table**. Neural Networks, 1999. IJCNN '99. International Joint Conference on , 10-16 July 1999 Pages:2194 - 2199 vol.3

Yamazaki, Jiancheng Li. and Yokoyama, K.Y. 1998. **Dynamic gain motion control with multi-axis trajectory monitoring for machine tool systems**. Advanced Motion Control, 1998. AMC '98-Coimbra., 1998 5th International Workshop on , 29 June-1 July 1998 Pages:316 – 321

Craig Simal, John J. Inc. 1955. **Introduction to Robotics, Mechanic And Control Second EDITION**. Addison-Wesley.

Lewis, Christopher L. and Maciejewski, Anthony A. 1990. **Trajectory generation for cooperating robots**. Systems Engineering, IEEE International Conference on, 9-11 Aug 1990, Pages: 300 – 303.

Matsui, N. 1995. **DSP-based intelligent motor/motion control**. American Control Conference, Volume: 1, 21-23 June 1995 Pages: 490 – 494.

Pierre Rouchon. 2001. **Motion planing, equivalence, infinite dimensional systems**. International Journals Applied Math Computer Science, Volume 11, Pages:165-188

Syh-Shiuh Yeh and Pau-Lo Hsu. 2000. **A new approach to bi-axial cross-coupled control.** Control Applications, 2000. Proceedings of the 2000 IEEE International Conference on 25-27 Sept. 2000, Pages: 168 – 173.

_____ and _____ 2000. **Design of precise multi-axis motion control systems.** Advanced Motion Control, 2000, Proceedings, 6th International Workshop on 30 March - 1 April 2000 Pages: 234 – 239.

_____ and _____ 2003. **Analysis and design of integrated control for multi-axis motion systems**. Control Systems Technology, IEEE Transactions on Volume: 11, Issue: 3, May 2003 Pages: 375 – 382.