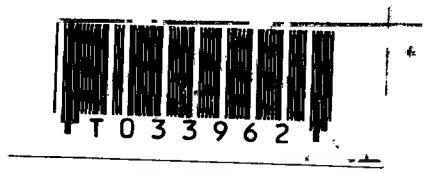
การพัฒนาระบบเชื่อมต่อกับผู้ใช้สำหรับการควบคุมแบบหลายแกน

(Development of Man-machine Interface System for Multi-axis Control)



อาจารย์ที่ปรึกษา

อาจารย์ เทพจิตร์ เชยโภคา

ดร.สุธี ผู้เจริญชนะชัย

ผู้จัดทำ

นาย วุฒิกร เชาว์ประมวลกุล 38014480

นาย อรรถพล กัณหเวก 38014627

รายงานส่วนนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2541

ปริญญานิพนธ์ปีการศึกษา 2541

ภาควิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

การพัฒนาระบบเชื่อมต่อกับผู้ใช้สำหรับการควบคุมแบบหลายแกน

(Development of Man-machine Interface System for Multi-axis Control)

ผู้จัดทำ

นาย วุฒิกร เชาว์ประมวลกุล 38014480

นาย อรรถพล กัณหเวก 38014627

อาจารย์ที่ปรึกษา

(อาจารย์ เทพจิตร์ เซยโภคา)

(ดร.สุธี ผู้เจริญชนะชัย)

บทคัดย่อ

รายงานฉบับนี้กล่าวถึงการพัฒนาระบบเชื่อมต่อกับผู้ใช้ โดยอ้างอิงถึงการสื่อสารที่เป็นมาตรฐานซึ่งใช้กันอยู่ในปัจจุบัน ได้แก่ ISA , PCI , VME , STD bus เป็นต้น โดยระบบเชื่อมต่อกับผู้ใช้นี้สามารถรับ G-Code ตามมาตรฐาน ISO แล้วแปลงเป็นคำสั่งที่ต้องกระทำ (action command) จากนั้นส่งไปให้วงจรควบคุม LM628 เพื่อการควบคุมแบบหลายแกน

ABSTRACT

This report presents the development of Man - Machine Interface (MMI) . System using communication standard such as ISA , PCI , VME , STD bus ETC. MMI system obtains G-Code according to ISO standard for converting it to action commands then , the system sends them to LM628 card in order to generate trajectory signal for multi - axis motion control

สารบัญ

สารบัญรูป

หน้า

บทที่ 1
บทนำ

ปัจจุบันวงการอุตสาหกรรมในประเทศไทย ได้เจริญเติบโตเป็นอย่างมาก เครื่องจักรกลในโรงงานต่าง ๆ มีการพัฒนาอย่างต่อเนื่อง มีการนำเอาเทคโนโลยีใหม่ ๆ จากต่างประเทศเข้ามามากมาย ซึ่งเทคโนโลยีเหล่านี้มีราคาค่อนข้างแพง และมีความซับซ้อนสูง จนทำให้ผู้ใช้งานทั่ว ๆ ไปไม่สามารถใช้เทคโนโลยีเหล่านั้นได้ จึงได้มีการพัฒนาระบบเชื่อมต่อกับผู้ใช้ขึ้นมา เพื่อลดความแตกต่างระหว่างความสามารถของผู้ใช้กับเครื่องจักรกล กล่าวคือทำให้ฟังก์ชันที่ยุ่งยากของเครื่องจักรกลแสดงออกมาในรูปแบบที่ผู้ใช้สามารถเข้าใจได้ง่ายขึ้น

สำหรับโครงงานนี้มีความมุ่งหวังที่จะพัฒนาระบบเชื่อมกับผู้ใช้บนแนวคิดของเทคโนโลยีระบบเปิดเพื่อให้เกิดความประหยัดและสามารถพัฒนาต่อไปในอนาคต สำหรับภายในระบบเชื่อมต่อกับผู้ใช้ในจะประกอบด้วยส่วนต่าง ๆ ที่ช่วยให้ผู้ใช้ สามารถทำงานได้ง่ายยิ่งขึ้น ได้แก่ Tuning ,Simulation ,GUI เป็นต้น ตลอดจนเมื่อต่อกับอุปกรณ์ต่าง ๆ เช่น DC Servo Motor ,Driver ,Encoder ,Card Motion Control ,Card 8255 เพื่อการทำงานแบบ Motion Control และ On/Off Control โดยผลของการทำงาน และสถานะต่าง ๆ จากภายนอก สามารถแสดงและควบคุมจาก PC Computer ซึ่งทำให้สะดวกต่อการใช้งาน

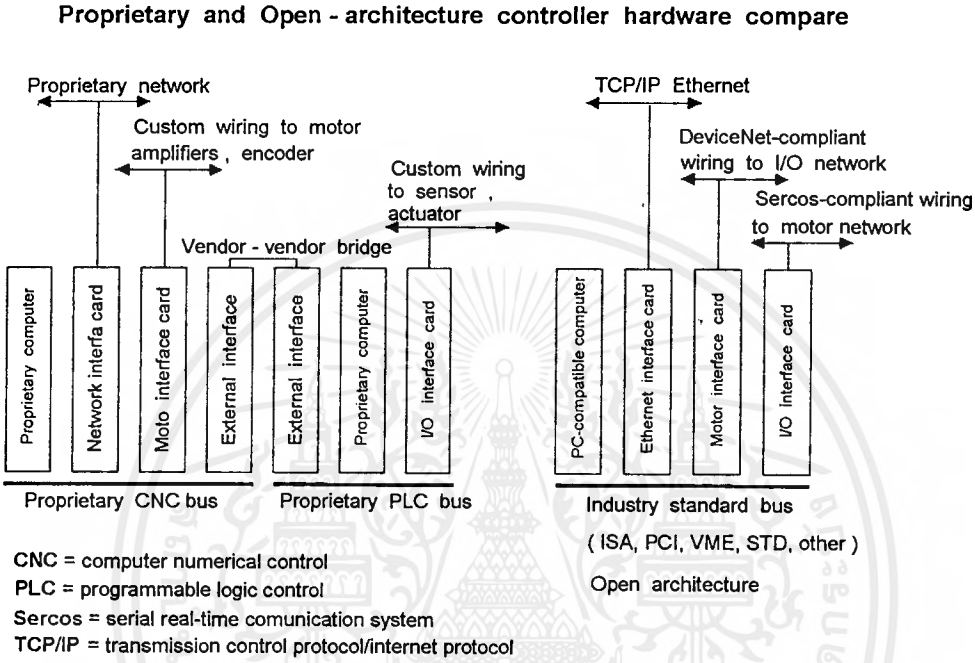
ระบบเชื่อมต่อกับผู้ใช้นี้เมื่อพัฒนาแล้วจะสามารถประยุกต์เป็นชุดทดลอง Servo Motor , ชุดทดลองเกี่ยวกับ เครื่องจักรกล CNC ได้ ทำให้สามารถพัฒนาบุคลากรในสาขานี้ได้ และในอนาคตเมื่อเราสามารถพัฒนาจนมีประสิทธิภาพสูงพอ ก็จะเป็นการลดค่าใช้จ่ายของประเทศลง และก่อให้เกิดประโยชน์ต่อภาคอุตสาหกรรมในประเทศด้วย

1.1 วัตถุประสงค์ของโครงงาน

จุดมุ่งหมายของปริญญานิพนธ์นี้ สามารถแบ่งออกเป็นหัวข้อได้ดังนี้

1. รับ NC code ตามมาตรฐาน ISO
2. สามารถทำการซดเซย การทำงานของเครื่องกัด และเครื่องกลึงได้
3. เชื่อมต่อ (Interface) ผ่าน ISA BUS หรือ Serial Port
4. ทำ Simulation สำหรับดู Tool Path ก่อนส่งให้เครื่องจักร (MAC) ได้
5. ทำการควบคุม แบบหลายแกนได้

บทที่ 2
ทฤษฎีและข้อมูลพื้นฐาน



รูปที่ 1. ภาพเปรียบเทียบระหว่างระบบที่เป็นเทคโนโลยีปิดและเทคโนโลยีเปิด

ปัจจุบันเครื่องจักร CNC ได้เข้ามามีบทบาทในงานอุตสาหกรรมภายในประเทศมากยิ่งขึ้น เมื่อแบ่งประเภทตามเทคโนโลยีสามารถแบ่งออกได้ 2 ประเภทคือ

1. เครื่องจักรกล CNC ที่เป็นเทคโนโลยีปิด หมายถึง เครื่องจักรกล CNC ที่มาจากผู้ผลิต ซึ่งมีมาตรฐานการสื่อสารเป็นของตนเอง ซึ่งเป็นความลับไม่เปิดเผย จึงทำให้การเพิ่มหรือเสริมอุปกรณ์ต่างๆ รวมทั้งโปรแกรมควบคุมการทำงานตลอดจนการบำรุงรักษาต้องทำโดยบริษัทผู้ผลิตเท่านั้น ซึ่งมักจะเสียค่าใช้จ่ายที่สูง
2. เครื่องจักรกล CNC ที่เป็นเทคโนโลยีเปิด หมายถึง เครื่องจักรกล CNC ที่ผลิตขึ้นโดยตั้งอยู่บนมาตรฐานการสื่อสารอุตสาหกรรม เช่น ISA , PCI , VME , STD bus เป็นต้น ดังนั้นจึงทำให้สามารถเลือกอุปกรณ์ต่างๆ ที่เหมาะสมกับงานได้ ในต้นทุนที่ต่ำกว่าแบบแรก

2.1 NC Code

ลักษณะโครงสร้างของ NC Code

NC code เป็นภาษาที่ประกอบด้วยตัวอักษร และค่าตัวเลขที่ตามหลังตัวอักษรนั้น ซึ่งจะให้ความหมายที่แน่ชัด โดยแบ่งออกเป็น 7 กลุ่ม ดังนี้

1. Block Number

Block Number เป็นส่วนที่ปรากฏอยู่ในหัวแถวของแต่ละบล็อค โดยเรียงลำดับกัน Block Number จะขึ้นด้วยตัวอักษร N แล้วตามด้วยตัวเลขจำนวนเต็มบวกซึ่งอยู่ในช่วง 1 ถึง 2,147,483,648 (32 bit integer)

2. Preparatory Functions

Preparatory Functions จะเป็นส่วนที่พบเห็นใน NC code โดยขึ้นต้นด้วยอักษร G และตามด้วยตัวเลขจำนวนเต็ม ซึ่งโดยปกติจะเรียกว่า G-code ซึ่งเป็นส่วนกำหนดลักษณะ รูปแบบ และ state การทำงานของ เครื่องจักร (Machine) มีทั้ง G-code ที่เป็นแบบ modal คือ แม้จะไม่มีการใส่ G-code อีก ในบรรทัดต่อไป แต่ก็จะถือว่า G-code ดังกล่าว ยังคงการทำงานอยู่ นอกจากจะมีการเปลี่ยน G-code ตัวอื่นเข้ามา ส่วน G-code อีกประเภท คือ Nonmodal จะเป็น G-code ที่จะมีการทำงานเฉพาะในบรรทัดที่มี G-code ดังกล่าวปรากฏอยู่เท่านั้น และตารางต่อไปนี้ เป็นตารางแสดงการทำงานและรูปแบบ ของ G-code

G-code	ลักษณะ และ ความหมายของ G-code	รูปแบบ
G00	Positioning In Rapid	Modal
G01	Linear Interpolation	Modal
G02	Circular Interpolation (CW)	Modal
G03	Circular Interpolation (CCW)	Modal
G04	Dwell	Nonmodal
G17	XY Plane	Modal
G18	XZ Plane	Modal
G19	YZ plane	Modal
G20/G70	Inch Units	Modal
G21/G71	Metric Units	Modal
G28	Automatic return to reference point	Nonmodal

G29	Automatic return from reference point	Nonmodal
G40	Cutter Compensation Cancel	Modal
G41	Cutter Compensation Left	Modal
G42	Cutter Compensation Right	Modal
G43	Tool Length Compensation (Plus)	Modal
G44	Tool Length Compensation (Minus)	Modal
G49	Tool Length Compensation Cancel	Modal
G80	Cancel Canned Cycles	Modal
G81	Drilling Cycle	Modal
G82	Counter Boring Cycle	Modal
G83	Deep Hole Drilling Cycle	Modal
G90	Absolute Positioning	Modal
G91	Incremental Positioning	Modal
G92	Reposition Origin Point	Nonmodal
G98	Set Initial Plane default	Nonmodal
G99	Return to Retract (Rapid) Plane	Nonmodal

3.Coordinate values

Coordinate values หรือ Axes values เป็นค่าจริงที่บอกให้เครื่องจักรทราบว่าจะต้องเคลื่อนที่ Tool ไปที่ตำแหน่งใด Coordinate values จะขึ้นต้นด้วยตัวอักษรที่แทนความหมายของแต่ละแกน อันประกอบด้วย X, Y, Z, I, J, K แล้วตามด้วยตัวเลข ซึ่งก็คือค่าจุดสิ้นสุดของการเคลื่อนที่

ตัวอักษร X, Y, Z แทน ending point ในระบบแกน 3 มิติ

ตัวอักษร I, J, K แทน center point ของเส้นโค้ง หรือ วงกลม

4.Feed Rate

Feed Rate คือ อัตราในการเคลื่อนที่ของ Tool ซึ่งจะขึ้นต้นด้วยอักษร F แล้วตามด้วยจำนวนจริง Feed Rate จะต้องมีค่ามากกว่าศูนย์ และมีหน่วยเป็น IPM (inches per minute) หรือ MMPM (millimeter per minute)

5.Spindle Speed

Spindle Speed คือ อัตราเร็วในการหมุนของ Tool ซึ่งจะขึ้นด้วยอักษร S แล้วตามด้วยจำนวนเต็มบวก และมีหน่วยเป็น RPM (Revolution per minute)

6.Tool Number

Tool Number จะแทนด้วยอักษร T แล้วตามด้วยจำนวนเต็ม ซึ่งหมายถึง เบอร์ของ Tool ที่ต้องการเลือกใช้ สำหรับข้อมูลของ Tool แต่ละเบอร์ จะถูกเก็บในหน่วยความจำส่วนหนึ่ง เมื่อมีการเปลี่ยน Tool ก็จะต้องเซ็ต Parameter ที่เกี่ยวข้องกับ Tool เบอร์นั้น ได้แก่ ขนาดเส้นผ่านศูนย์กลาง , ขนาดความยาวของ Tool นั้น เป็นต้น

7.Miscellaneous Function

Miscellaneous Function หรือ เรียกว่า M-code จะขึ้นต้นด้วยอักษร M แล้วตามด้วยจำนวนเต็ม และ M-code ที่มีหน้าที่เกี่ยวกับการควบคุมแบบ on/off จะถูกส่งให้ PLC เป็นตัวจัดการ

M code	ลักษณะ และ ความหมายของ M code
M00	Program Stop
M01	Optional Program Stop
M02	Program End
M03	Spindle On CW
M04	Spindle ON CCW
M05	Spindle Stop
M06	Tool change
M08	Coolant On
M09	Coolant Off
M10	Clamps On
M11	Clamps Off
M30	Program End, Reset to Start

2.2 ทฤษฎีทางคณิตศาสตร์สำหรับ CNC (Mathematic theory for CNC)

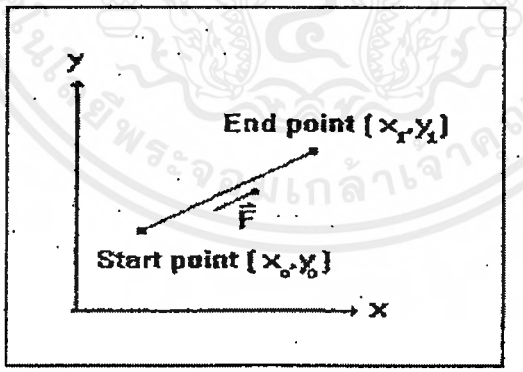
ทฤษฎีทางการคำนวณทิศทางการเคลื่อนที่ (Feed Rate Speed)อัตราการป้อนกลับของคำสั่ง G01 , G02 , G03 จะถูกกำหนดโดยตัวเลขหลังคำสั่ง F___และจะคงค่าเดิมตลอดเวลาจนกว่าจะมีการเปลี่ยนแปลงค่าใหม่ ค่าอัตราการป้อนนี้เรียกว่า Tangential Speed constant

1. Linear Interpolation

เมื่อมี Straight cut linear interpolation (G01) ส่วนควบคุมจะควบคุมแกนของเครื่องจักร 2 แกน หรือมากกว่านั้นพร้อม ๆ กัน ในส่วนการตัดเชิงมุมส่วนควบคุมจะใช้ข้อมูลจากส่วนที่ได้รับจากโปรแกรมเข้ามาเก็บไว้เพื่อคำนวณ องศาหรือความชันของการตัดชิ้นส่วนของเส้นตรง ความยาวที่เปลี่ยนแปลงจากจุดเริ่มต้นจนถึงจุดสุดท้ายจะเป็นตัวกำหนด การแบ่งเส้นและความชันของแต่ละแกน เพื่อทำการควบคุมการเคลื่อนที่ของ Cutter ให้เคลื่อนที่เสมือนกับ การเคลื่อนที่ไปในทิศทางเดียว ในโครงงานนี้กำหนดให้ Linear Interpolation เคลื่อนที่ในระนาบเดียวเท่านั้น

การใช้งานอย่างอื่นของ Linear Interpolation คือใช้ในการประมาณเส้นโค้ง หรือวงกลม (Circular Interpolation) ซึ่งจะกล่าวในช่วงต่อไป

การคำนวณใน Linear Interpolation



รูปที่ 2 การเคลื่อนที่แบบ Linear Interpolation

F – อัตราป้อนในแนวการเคลื่อนที่

F_x – อัตราป้อนในแนวแกน X

F_y – อัตราป้อนในแนวแกน Y

โดยที่

F_x = F(ระยะทางที่เคลื่อนที่ในแนวแกน X) / ระยะทางที่เคลื่อนที่ทั้งหมด

$$= \frac{F.(x_1 - x_0)}{\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}}$$

Fy = F(ระยะทางที่เคลื่อนที่ในแนวแกน Y) / ระยะทางที่เคลื่อนที่ทั้งหมด

$$= \frac{F.(y_1 - y_0)}{\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}}$$

2. Circular Interpolation

Circular Interpolation เป็นความสามารถในการตัดส่วนโค้งของวงกลม ซึ่งจำนวนที่ตัดแปรผันตามขนาดของส่วนโค้งซึ่งขึ้นอยู่กับรัศมีของส่วนโค้งส่วนควบคุมจะคำนวณทิศทางของ Cutter จากข้อมูลที่โปรแกรมมา

ข้อมูลที่ต้องการได้แก่

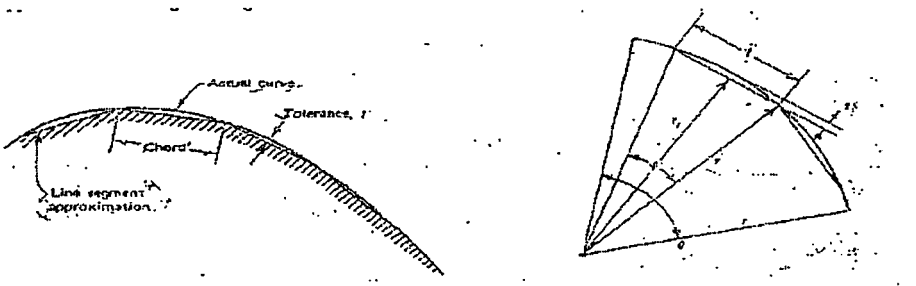
1. ทิศทางของการเคลื่อนที่(CW หรือ CCW) ตรวจสอบได้จาก G code ว่าเป็น G02 หรือ G03

2. จุดเริ่มต้นของส่วนโค้ง สามารถหาค่าได้จากการอ่านค่าตำแหน่งของแท่นใบมีดขณะนั้น

3. จุดสิ้นสุดของส่วนโค้ง ได้จากค่าที่ผู้ใช้ป้อนเข้ามา

4. รัศมีส่วนโค้ง (radius) ได้จากค่าที่ผู้ใช้ป้อนเข้ามา

หลังจากที่ทราบค่าต่างๆ แล้วจะนำมาคำนวนโดยอาศัยหลักการของ Contour Appoximation Using Line Segment ดังนี้



รูปที่ 3. Contour Approximation Using Line Segment

หลักการของ Contour Approximation Using Line Segment อาศัยหลักการของการประมาณค่าส่วนโค้งด้วยเส้นตรง โดยมีขั้นตอนดังนี้

1.หาความยาวของส่วนโค้งทั้งหมด ($r\theta$)

2.กำหนดความละเอียดในการแบ่งส่วนโค้ง (n)

3.หามุมของเส้นตรงที่เกิดจากการแบ่งส่วนโค้ง

$$\varphi = \frac{\theta}{n}$$

เมื่อได้ค่าต่าง ๆ ที่ต้องการแล้วก็ทำการเดินแทนใบมีดไปตามเส้นตรงนั้น ๆ จะทำให้เกิดการเดินแบบกลึงโค้ง

2.3 ข้อควรรู้เกี่ยวกับกระบวนการควบคุมแบบ PID

ในแต่ละปีมีกระบวนการควบคุมแบบ PID เป็นร้อยๆ พันๆ ตัวที่ถูกใช้ไปเพื่อควบคุมในทางอุตสาหกรรม ด้วยเหตุนี้จึงเป็นสิ่งสำคัญที่ควรทำความคุ้นเคยกับการปฏิบัติการของมัน

กระบวนการควบคุมคืออะไร

ถ้าจะอธิบายกันอย่างง่ายๆ ก็คือ การที่อุปกรณ์อินพุตได้รับค่ามาค่าหนึ่งและพยายามที่จะคงค่านั้นไว้ที่ค่าที่เราต้องการออกมาทางเอาต์พุตโดยการปรับค่าอุปกรณ์ต่างๆ เพื่อควบคุมค่านั้นไว้ การควบคุม อย่างที่เป็นที่รู้จักกันในชีวิตประจำวันของเราก็คือ เครื่องปรับอากาศและอุปกรณ์ควบคุมการขับเคลื่อนของรถยนต์ (automotive cruise control) ปริมาณที่ถูกวัดและควบคุมจะถูกเรียกว่า ตัวแปรกระบวนการ (process variable , PV) สำหรับค่าตัวแปรกระบวนการของเครื่องปรับอากาศคือ อุณหภูมิของบ้าน ส่วนของอุปกรณ์ควบคุมการขับเคลื่อนของรถยนต์ คือความเร็วของรถยนต์

ค่าที่เราต้องการให้ตัวแปรกระบวนการคงไว้นั้น เรียกว่า ค่าเป้าหมาย (setpoint , SP) เราปรับค่าเป้าหมายของเครื่องปรับอากาศที่บ้านของเรา โดยบ่งชี้อุณหภูมิที่เราต้องการและกำหนดค่าความเร็วที่เราต้องการ ให้กับอุปกรณ์ควบคุมการขับเคลื่อนของรถยนต์

ตัวควบคุมจะปรับเปลี่ยนค่าตัวแปรกระบวนการโดยการปรับ ค่าควบคุมเอาต์พุต (control output, CO) โดยที่เครื่องปรับอากาศจะสั่งให้คอมเพรสเซอร์ทำงานหรือหยุดทำงานเพื่อควบคุมอุณหภูมิ ส่วนอุปกรณ์ควบคุมการขับเคลื่อนรถยนต์จะผ่อนหรือเพิ่มคันเร่ง (throttle) เพื่อควบคุมความเร็ว โดยปกติแล้วกระบวนการควบคุมทางอุตสาหกรรม จะใช้ค่าตัวแปรควบคุมไปขับเคลื่อนวาล์วควบคุม เพื่อควบคุมค่าตัวแปรกระบวนการอย่างเช่น ระดับน้ำในถัง อัตราการไหลของของไหล ความดัน หรืออุณหภูมิให้เข้าสู่ค่าเป้าหมายที่เราต้องการ

กระบวนการควบคุมส่วนใหญ่ ไม่ได้ทำการควบคุมโดยตรงที่ค่าตัวแปรกระบวนการหรือค่าเป้าหมาย แต่ทำที่ค่าสัญญาณ ความผิดพลาด (error signal, e) ที่ตัวมันคำนวณออกมา ค่าความผิดพลาดจะแสดงค่าตัวแปรควบคุมที่เบี่ยงเบนไปจากค่าเป้าหมาย ซึ่งคำนวณได้ตามสมการ

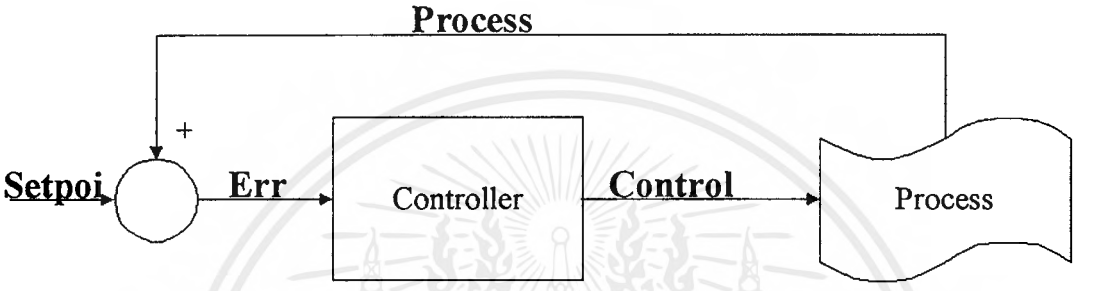
$$e = PV - SP$$

โดยที่ e = สัญญาณความผิดพลาด

PV = ตัวแปรกระบวนการ

SP = ค่าเป้าหมาย

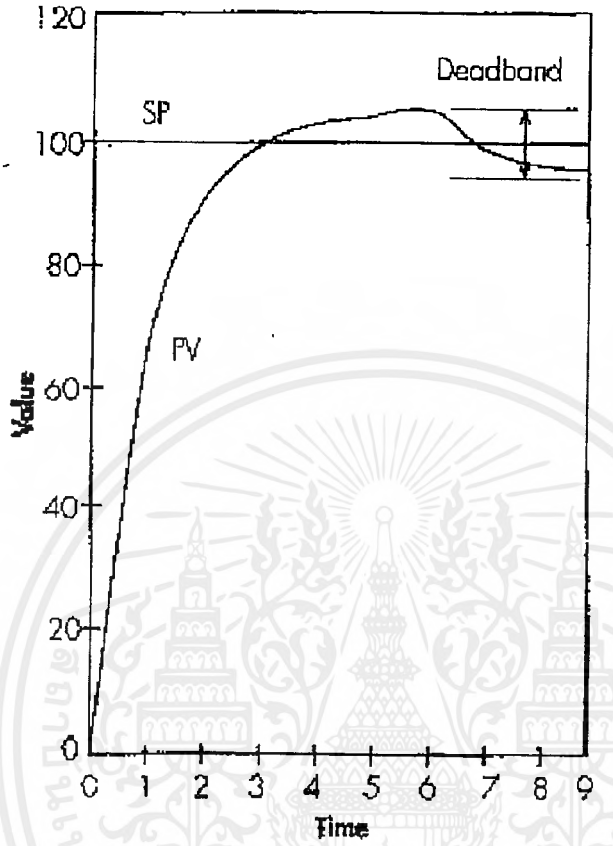
ค่าความผิดพลาดที่เป็นบวกแสดงถึง ค่าตัวแปรกระบวนการที่เกินค่าเป้าหมายไป ส่วนค่าที่เป็นค่าลบจะแสดงว่ายังไม่ถึงค่าเป้าหมาย เนื่องจากข้อมูลของตัวแปรกระบวนการถูกป้อนกลับมาจากกระบวนการที่ถูกควบคุมแล้ว ดังนั้นบางครั้งกระบวนการควบคุมประเภทนี้จึงถูกเรียกว่า กระบวนการควบคุมแบบป้อนกลับโดยอาศัยค่าสัญญาณความผิดพลาด (error feedback controller) ตัวควบคุมจะใช้ค่าความผิดพลาดมาประเมินค่าตัวแปรควบคุม ซึ่งจำเป็นในการจะคงค่าตัวแปรกระบวนการไว้ที่ค่าเป้าหมาย รูปแบบของตัวควบคุมโดยทั่วไปสามารถดูได้จากรูปที่ 4



รูปที่ 4 รูปแบบโครงสร้างของกระบวนการควบคุม

การควบคุมแบบเปิด/ปิด(ON/OFF controllers)

การควบคุมแบบเปิด/ปิดถือว่าเป็นตัวควบคุมอย่างง่ายที่สุดเนื่องจากง่ายต่อการควบคุมอุปกรณ์ที่รับทางเอาต์พุต ซึ่งจะเปิดหรือ ปิดขึ้นกับค่าสัญญาณความผิดพลาด เครื่องปรับอากาศตามบ้านส่วนใหญ่ก็เป็นตัวควบคุมแบบเปิด/ปิดคือ คอมเพรสเซอร์จะทำงาน (เปิด) เมื่อค่าความผิดพลาดเป็นลบ (อุณหภูมิสูงเกินไป) และจะหยุดทำงาน(ปิด)เมื่อค่าความผิดพลาดเป็นบวก (อุณหภูมิต่ำเกินไป) เพื่อหลีกเลี่ยงการเปิดปิดอย่างรวดเร็วของตัวคบคุม ตัวควบคุมแบบเปิด/ปิดส่วนใหญ่จึงมี deadband ซึ่งค่าเอาต์พุตของตัวควบคุมจะคงไว้ที่สถานะปัจจุบันจนกระทั่งค่าความผิดพลาดหลุดออกจากช่วง deadband ตัวอย่างเช่นในกรณีของเครื่องปรับอากาศมี deadband เป็น 4 องศาฟาเรนไฮต์ ตัวควบคุมจะสั่งให้คอมเพรสเซอร์ ทำงานก็ต่อเมื่อค่าความผิดพลาดทางด้านลบถึง 2 องศาฟาเรนไฮต์ และตัวควบคุมจะสั่งให้หยุดทำงานก็ต่อเมื่อค่าความผิดพลาดทางด้านบวกถึง 2 องศาฟาเรนไฮต์ ด้วยเหตุนี้ตัวแปรกระบวนการที่ถูกควบคุมโดยตัวควบคุมแบบเปิด/ปิดจึงเปลี่ยนแปลงค่าขึ้นลงอยู่รอบๆค่าเป้าหมาย ดังที่แสดงไว้ในรูปที่ 5 ตัวควบคุมแบบเปิด/ปิดนี้มักจะถูกเรียกว่า ตัวควบคุมแบบ “bang-bang” เพราะค่าตัวแปรควบคุมมักจะเปลี่ยนไปมาระหว่างค่า 2 ค่าที่ แตกต่างกัน



รูปที่ 5 การตอบสนองของตัวควบคุมแบบเปิด/ปิด

การควบคุมแบบต่อเนื่อง (Continuous controllers)

กระบวนการทางอุตสาหกรรมส่วนใหญ่ใช้การควบคุมแบบต่อเนื่อง ค่าตัวแปรควบคุมจะเป็นแบบอนาลอก ซึ่งจะถูกปรับอย่างต่อเนื่อง ถือเป็นข้อดีที่เด่นชัดที่สามารถกำจัดปัญหาการเปลี่ยนแปลงขึ้น/ลงของค่าตัวแปรควบคุม อย่างที่เกิดขึ้นในตัวควบคุมแบบเปิด/ปิดได้ ตัวควบคุมแบบต่อเนื่องที่ใช้ในอุตสาหกรรมทุกวันนี้จะเป็นแบบ proportional แบบ integral หรือแบบ derivative ดังนั้นจึงมีการเรียกตัวควบคุมประเภทนี้ว่าเป็น ตัวควบคุมแบบ PID (proportional integral derivative) ผลของตัวควบคุมในแต่ละแบบจะบรรยายอย่างดังต่อไปนี้

การควบคุมแบบ Proportional

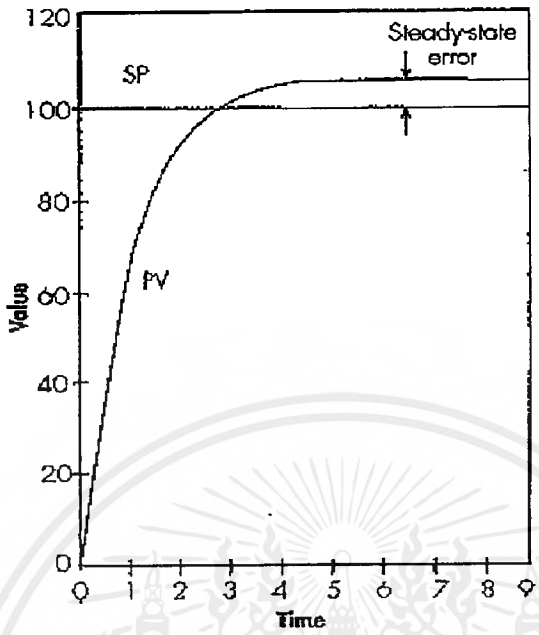
การควบคุมประเภทนี้ได้ชื่อมาจากข้อเท็จจริงที่ว่า การควบคุมค่าเอาต์พุตเป็นสัดส่วนโดยตรงกับค่าสัญญาณความผิดพลาดค่าสัญญาณความผิดพลาดจำนวนมากจะสร้างค่าควบคุมเอาต์พุตที่มาก ในทางตรงกันข้ามค่าสัญญาณความผิดพลาดที่น้อยๆ จะให้ค่าควบคุมเอาต์พุตที่น้อย

ถ้าค่าสัญญาณความผิดพลาดที่เป็นบวกทำให้ค่าควบคุมเอาต์พุตเพิ่มขึ้น ตัวควบคุมตัวนั้นจะถือว่าเป็นตัวคุมแบบให้ผลกระทำตรง (direct acting) ในทางกลับกัน ถ้าค่าสัญญาณความผิดพลาดที่เป็นบวกทำให้ค่าควบคุมเอาต์พุตลดลงตัวควบคุมตัวนั้นจะเป็นแบบที่ให้ผลกระทำผกผัน (reverse acting)

ผลกระทำของตัวควบคุมจะเป็นแบบตรงหรือแบบผกผันขึ้นอยู่กับองค์ประกอบของกระบวนการ อย่างเช่นในกรณีการควบคุมระดับของของเหลวในถัง การกระทำจะขึ้นอยู่กับตำแหน่งของวาล์วควบคุม ถ้าวาล์วควบคุมอยู่ในตำแหน่งที่ทำการควบคุมอัตราการไหลของของเหลวออกจากถัง ความต้องการของเราก็คือ ต้องการให้ค่าความผิดพลาดที่เป็นบวก (ระดับของของเหลวสูงเกินไป) ไปเพิ่มค่าตัวแปรควบคุม ก็คือการเปิดวาล์วและให้ของเหลวไหลออกจากถังเพิ่มขึ้นนั่นเอง ดังนั้นตัวควบคุมที่ใช้จะต้องใช้ตัวควบคุมแบบให้ผลกระทำตรงในทางกลับกันถ้าวาล์วทำการควบคุมอัตราการไหลเข้าถังของของเหลวเราจะเลือกใช้ตัวควบคุมแบบที่ให้ผลกระทำผกผัน ซึ่งจะให้ผลการตอบสนองต่อระดับของของเหลวสูงเกินไป โดยการปิดวาล์วเพื่อลดปริมาณของของเหลวที่จะไหลเข้าถังนั่นเอง

ค่าออฟเซตจะเป็นตัวกำหนดว่าค่าตัวแปรควบคุมจะเป็นเท่าไร ที่ค่าความผิดพลาดเป็นศูนย์โดยปกติแล้วจะตั้งค่าไว้ที่ 50 เปอร์เซ็นต์ นั่นคืออุปกรณ์ควบคุมจะเปิด 50 เปอร์เซ็นต์ เมื่อค่าความผิดพลาดเป็นศูนย์ ส่วนค่าอัตราการขยายจะเป็นตัวกำหนดปริมาณการปฏิบัติของการควบคุม (control action) ที่เกิดขึ้นจากค่าสัญญาณความผิดพลาดที่ให้ออกมา ที่ค่าสัญญาณความผิดพลาดที่เปลี่ยนแปลงไปค่าหนึ่ง ถ้าอัตราการขยายมีค่าน้อยจะทำให้ตัวแปรควบคุมเปลี่ยนแปลงไปน้อยกว่าที่อัตราการขยายมีค่ามากๆ

ปัญหาเบื้องต้นของการควบคุมแบบนี้คือ ค่าสัญญาณความผิดพลาดที่ไม่เป็นศูนย์ซึ่งจำเป็นต้องมี เพื่อสร้างตัวแปรควบคุมสำหรับทำให้กระบวนการเสถียรที่ค่าเป้าหมาย ดังนั้นเราจึงไม่สามารถควบคุมกระบวนการได้อย่างเที่ยงตรง ค่าตัวแปรควบคุมที่จะทำให้ค่าตัวแปรกระบวนการเข้าสู่ค่าเป้าหมายได้จะต้องเท่ากับค่าออฟเซต (50 เปอร์เซ็นต์ ดังตัวอย่างข้างต้น) เพื่อให้ได้มาซึ่งการควบคุมที่ไม่ผิดพลาดในช่วงค่าเป้าหมายอย่างไรก็ดีในสภาวะใดๆ ก็ตามของกระบวนการ ค่าสัญญาณความผิดพลาดที่ไม่เป็นศูนย์จำต้องมีเสมอ เพื่อสร้างค่าตัวแปรควบคุมที่เหมาะสม ด้วย

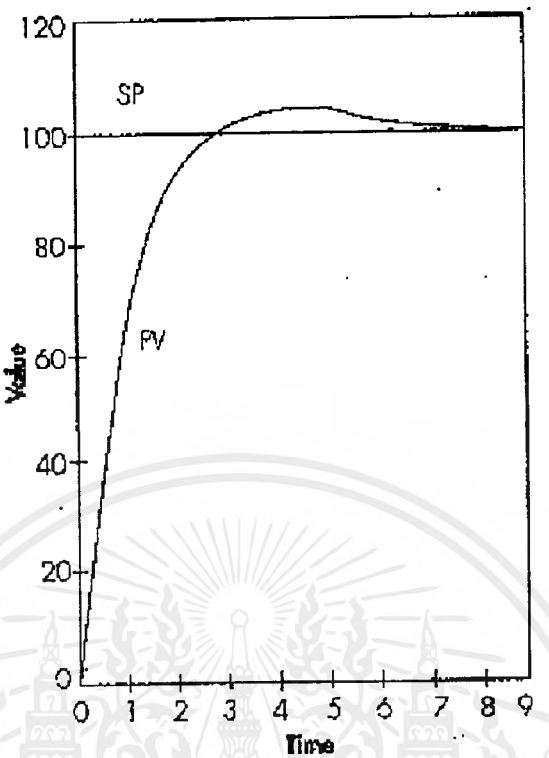


รูปที่ 6 การตอบสนองของการควบคุมแบบ proportional

เหตุนี้ค่าตัวแปรกระบวนการจึงต่างจากค่าเป้าหมายเสมอ ดังแสดงในรูปที่ 6 เราเรียกปรากฏการณ์นี้ว่า ความผิดพลาดที่สภาวะคงตัว (steady-state error)

การควบคุมแบบ Proportional-integral (PI controllers)

ในเบื้องต้นการปฏิบัติแบบ integral จะถูกเพิ่มให้กับตัวควบคุมแบบ proportional เพื่อแก้ปัญหาความผิดพลาดที่สภาวะคงตัวที่พร้อมจะกำจัดความต้องการค่าออฟเซต และยังแก้ปัญหาความผิดพลาดที่สภาวะคงตัวด้วย เนื่องด้วยค่าความผิดพลาดที่เกิดขึ้นเป็นค่าที่ถูกสะสมอย่างต่อเนื่องตลอดเวลา ทำให้ค่าตัวแปรควบคุมของตัวควบคุมที่มีการปฏิบัติแบบ integral เปลี่ยนแปลงอย่างต่อเนื่อง ตราบเท่าที่ค่าความผิดพลาดยังไม่เป็นศูนย์ และจะหยุดเมื่อค่าความผิดพลาดเป็นศูนย์ ถ้ากระบวนการที่ถูกควบคุมมีความเสถียร การควบคุมแบบนี้จะรับประกันได้ว่าค่าความผิดพลาดที่สภาวะคงตัวจะกลายเป็นศูนย์ ดังแสดงในรูปที่ 7



รูปที่ 7 การตอบสนองของการควบคุมแบบ proportional-integral

การควบคุมแบบ Proportional-integral-derivative (PID controllers)

การปฏิบัติแบบ PID จะเพิ่มให้กับตัวควบคุมแบบ Proportional เพื่อแก้ปัญหากระบวนการเฉื่อย (sluggish process) อย่างเช่นกระบวนการที่จัดการกับสารปริมาณมากๆ ซึ่งต้องถูกเร่งหรือ ลดความเร่ง เพิ่มความร้อนหรือให้ความเย็น ต่างมีแนวโน้มในการใช้ตัวควบคุมแบบ PID ทั้งสิ้น

การควบคุมจะมีผลกับอัตราการเปลี่ยนแปลงของค่าความผิดพลาดที่เกิดขึ้น ซึ่งจะมีผลกระทบต่อการลดค่าตัวแปรควบคุมเพื่อทำให้เกิดโอเวอร์ชูต (overshoot) ที่น้อยที่สุด นั้นก็หมายถึงว่า กระบวนการจะเข้าถึงค่าเป้าหมายได้ในไม่ช้า ตัวอย่างเช่น เมื่อควบคุมอุณหภูมิในถังที่มีขนาดใหญ่มากถังหนึ่ง การเพิ่มค่าเป้าหมายอาจจะต้องการค่าตัวแปรควบคุมถึง 100 เปอร์เซ็นต์ เป็นระยะเวลานาน เพื่อเพิ่มความร้อนให้กับถังอย่างรวดเร็ว แต่อย่างไรก็ดีเมื่อความร้อนที่ให้ เริ่มมีผลต่ออุณหภูมิของถัง เราก็จำเป็นที่จะต้องลดความร้อนอย่างรวดเร็วเช่นกันเพื่อหลีกเลี่ยงการเกิดโอเวอร์ชูตของอุณหภูมิ การปฏิบัติแบบ PID จะทำให้การควบคุมกระบวนการประเภทนี้เป็นไปอย่างมีประสิทธิภาพ

การปรับการควบคุม (Controller Tuning)

กระบวนการในการเลือกค่าอัตราการขยายเพื่อการควบคุมที่ดีที่สุด เรียกว่า การปรับการควบคุม ค่าอัตราการขยายที่สูงเกินไปจะทำให้ค่าตัวแปรควบคุมแกว่งเป็นวงกว้าง และพฤติกรรมของค่าตัวแปรควบคุมไม่เสถียร ในขณะที่ค่าอัตราการขยายที่ต่ำเกินไปจะทำให้ค่าตัวแปรควบคุมมีการตอบสนองช้าและการควบคุมจะให้ผลที่ไม่ดีเท่าที่ควร ถ้าค่าอัตราการขยายได้รับการปรับแต่งอย่างดีแล้ว กระบวนการภายใต้การควบคุมนี้จะให้ผลตอบสนองอย่างราบเรียบและรวดเร็ว เพื่อเปลี่ยนแปลงในช่วงค่าเป้าหมายและทำให้กระบวนการกลับสู่สภาวะที่ต้องการ

อย่างไรก็ดีแนวโน้มของการเปลี่ยนแปลงค่าอัตราการขยายไม่ใช่ว่าต้องเป็นไปในทิศทางเดียวกันจึงจะให้ผลดี การผสมผสานที่เหมาะสมของการปฏิบัติแบบ proportional แบบ PI และแบบ PID จึงจะนำไปสู่การควบคุมที่ดีได้ ซึ่งต้องขึ้นอยู่กับลักษณะของกระบวนการที่จะควบคุม ด้วย การปฏิบัติแบบ proportional ที่มีค่าสูงๆ จะเหมาะกับการควบคุมกระบวนการ ที่ เราสามารถคาดเดาการเปลี่ยนแปลงอันเนื่องมาจากค่าตัวแปรควบคุมไว้ สำหรับกระบวนการที่มีสัญญาณรบกวน (noise) ขณะวัดค่าตัวแปรกระบวนการ การควบคุมด้วยค่าอัตราการขยายสัดส่วนสูงๆ ได้ดี ขณะที่การควบคุมอัตราการไหลหรือการควบคุมความดันซึ่งให้ผลตอบสนองรวดเร็วและมีสัญญาณรบกวนในค่าตัวแปรกระบวนการมาก จะต้องทำการควบคุมโดยใช้ค่าอัตราการขยายสัดส่วนต่ำๆ จึงจะให้ผลดี

สำหรับการปฏิบัติแบบ PI ถือได้ว่ามีพฤติกรรมที่เกือบจะตรงกันข้ามกับการปฏิบัติแบบ proportional เนื่องจากการปฏิบัติแบบ PI มีคุณสมบัติเป็นเหมือนตัวกรองสัญญาณความถี่ต่ำชนิดหนึ่ง เพราะฉะนั้นสัญญาณรบกวนจึงไม่มีผลกระทบต่อตัวแปรกระบวนการ (สัญญาณรบกวนมีค่าน้อยเมื่อเทียบกับค่าความผิดพลาดสะสม) การปฏิบัติที่เพิ่มเข้ามาก็หมายถึงว่า การควบคุมจะไม่เกิดผลเฉียบพลันต่อการเปลี่ยนแปลงในกระบวนการ แต่จะเป็นแบบค่อยเป็นค่อยไป ความจริงที่ว่านี้จึงทำให้เป็นผลดีต่อกระบวนการที่มีการตอบสนองที่รวดเร็ว (fast-responding) อาจกล่าวได้ว่าเป็นกระบวนการเหมาะกับสัญญาณรบกวน อย่างเช่น การควบคุมความดันและอัตราการไหล ซึ่งในการควบคุม ค่าอัตราการขยาย PI ควรจะมีค่าที่ค่อนข้างสูงกว่าค่าอัตราการขยาย proportional

ในการควบคุมอุณหภูมิและระดับของของไหลในถัง ผลกระทบของตัวแปรควบคุมภายในกระบวนการ การจะถูกสะสมตลอดเวลา ความร้อนที่ถูกเพิ่มให้กับกระบวนการ ท้ายที่สุดก็จะทำให้อุณหภูมิเพิ่มขึ้น ขณะที่ของไหลที่เพิ่มเข้าไปในถังท้ายที่สุดก็จะเพิ่มระดับของของไหลเช่นกัน เนื่องกันด้วยกระบวนการประเภทนี้มีการปฏิบัติแบบ PI อยู่แล้ว ดังนั้นในการควบคุมจึงต้องการตัวควบคุมที่มีการปฏิบัติแบบ PI เพียงน้อยนิดถึงไม่มีเลย และอัตราการขยาย PI ที่ใช้ก็ควรจะมีค่าต่ำเช่นกัน

ในการควบคุมโดยการใช้การปฏิบัติแบบ PID จำเป็นจะต้องใช้ความระมัดระวังอย่างสูง ขั้นแรกอัตราการขยาย PID ควรจะถูกปรับอย่างระมัดระวัง ลำดับที่สองควรหลีกเลี่ยงการใช้ในกระบวนการที่สัญญาณรบกวนในค่าตัวแปรกระบวนการ สัญญาณรบกวนในค่าตัวแปรกระบวนการจะทำให้เกิดค่าสัญญาณความผิดพลาดเปลี่ยนแปลงขึ้น-ลงอย่างรุนแรง เป็นผลให้ค่าความชันเปลี่ยนแปลงทั้งทางบวกและทางลบอย่างรวดเร็ว การเปลี่ยนแปลงนี้เป็นผลมาจากการปฏิบัติแบบ PID ซึ่งการควบคุมมักจะไม่เสถียรเมื่อมีสัญญาณรบกวนเกิดขึ้นในค่าตัวแปรกระบวนการ ดังนั้นในการเลือกใช้ควรเป็นไปอย่างรอบคอบ กระบวนการที่สามารถใช้การปฏิบัติแบบนี้ได้ต้องเป็นกระบวนการที่มีการวัดที่เชื่อถือได้และไม่มีสัญญาณรบกวน (dependable and noise-free) ซึ่งกระบวนการที่เป็นตามนี้ก็คือการควบคุมอุณหภูมิ

ดังนั้นในการที่คุณจะใช้กระบวนการควบคุมแบบไหน เริ่มแรกคุณต้องมีความรู้เฉพาะทางในเรื่องของตัวควบคุมที่ใช้และกระบวนการการภายใต้การควบคุมนั้นเพื่อคุณจะสามารถตั้งค่าอัตราการขยายของตัวควบคุมที่ให้ค่าเป็นที่น่าพอใจได้ยิ่งไปกว่านั้นมีผู้ผลิตตัวควบคุมแต่ละตัวกำหนดสมการที่ใช้กับตัวควบคุมแตกต่างกันไป ดังนั้นค่าอัตราการขยายจึงไม่จำเป็นต้องมีพฤติกรรมเหมือนดังตัวอย่างนี้เสมอไป ตัวควบคุมบางตัว เราจะปรับการปฏิบัติแบบ propotional โดยการปรับค่าเปอร์เซ็นต์ของแบบสัดส่วน และการปฏิบัติของ PI โดยการปรับค่าหน่วยจำนวนครั้งต่อนาทีหรือนาทีต่อจำนวนครั้ง (repeats per minute or minutes per repeat) ซึ่งจะให้ผลตรงข้ามกัน เพื่อเพิ่มความสับสนยิ่งขึ้นไปอีก การปฏิบัติแบบ proportional แบบ PI และแบบ PID มักจะถูกเรียกว่าเป็นค่าอัตราการขยายค่ารีเซ็ท (reset) และค่าอัตรา (rate) ในหนังสือแบบเก่า ดังนั้นเพื่อความปลอดภัย ควรขอข้อมูลจากผู้ผลิตตัวควบคุมทุกครั้ง

Implementing The PID Controller

จาก transfer function ของ PID Controller

$$G(s) = \frac{M(s)}{E(s)} = K_c\left(1 + \frac{1}{T_i s} + T_d s\right) \qquad \text{(1)}$$

เมื่อ Kc = controller gain

Ti = integral action time

Td = derivative action time

และเมื่อกระจาย ในรูป time - domain จะได้

$$m(t) = K_c T_d \frac{de(t)}{dt} + K_c e(t) + \frac{K_c}{T_i}\int e(t)dt \quad \text{_____(2)}$$

เราสามารถแทน อนุพันธ์ และ อินทิเกรต ด้วย first order finite differences ดังต่อไปนี้

$$\frac{df}{dt}\Big|_k = \frac{f_k - f_{k-1}}{\Delta t}, \int e(t)dt = \sum_{k=0}^{n} e_k . \Delta t$$

ดังนั้นเราสามารถเขียนสมการที่(2) ใหม่ในรูป Discrete form ได้ดังนี้

$$m_n = K_c\left[T_d \frac{(e_n - e_{n-1})}{\Delta t} + e_n + \frac{1}{T_i}\sum_{k=0}^{n} e_k . \Delta t\right] \quad \text{_____(3)}$$

เมื่อกำหนดให้

$$K_p = K_c$$
$$K_i = K_c * T_s / T_i$$
$$K_d = K_c * T_d / T_s$$
$$s_n = s_{n-1} + e_n$$
$$T_s = \Delta t = sampling \quad time$$
$$s_n = sum \quad of \quad the \quad error$$

จะได้

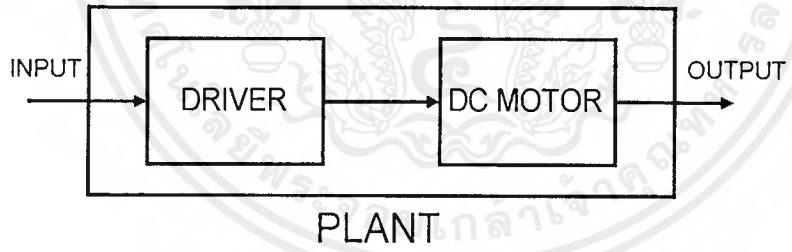
$$m_n = K_p * e_n + K_i * s_n + K_d\left(e_n - e_{n-1}\right) \quad \text{_____ (4)}$$

ซึ่งเป็น อัลกอริทึม ของ PID Controller ที่นำไปเขียนโปรแกรมควบคุมการหมุนของ motor ให้ได้ตำแหน่งตามที่เราต้องการ

2.4 การประมาณค่าพารามิเตอร์ในระบบ

ในการควบคุมระบบต่าง ๆ เราจำเป็นที่จะต้องทราบรายละเอียดเกี่ยวกับระบบที่เราจะทำการควบคุมนั้นเพื่อสามารถที่จะออกแบบตัวควบคุมต่อไป และในการควบคุมการหมุนของมอเตอร์เราต้องการตัวควบคุมที่เหมาะสมเช่นกันในการควบคุมเพื่อให้ระบบทำงานได้ตามต้องการในทางทฤษฎีการที่จะหาตัวควบคุมที่เหมาะสมนั้นสามารถทำได้โดยการเลือกและลองใช้ตัวควบคุมกับระบบจริงแล้วทำการปรับค่าพารามิเตอร์ (Parameter) ของตัวควบคุมจนกว่าจะได้ผลตอบสนองของระบบที่ต้องการ แต่ในทางปฏิบัติอาจทำให้เกิดความเสียหายแก่ระบบจริงในช่วงการปรับได้ เพื่อลดความเสี่ยงดังกล่าว จึงใช้วิธีการจำลอง (Simulation) ระบบการทำงานจริง ด้วยสมการทางคณิตศาสตร์ (Mathematical Model) แทนระบบของมอเตอร์แล้วทำการปรับค่าตัวควบคุมได้ผลตอบสนองของระบบที่ต้องการ แล้วจึงนำผลที่ได้ทดสอบกับระบบจริง ซึ่งวิธีนี้ต้องทราบแบบจำลองทางคณิตศาสตร์ที่มีความสมมูลกับระบบจริงก่อนจึงสามารถหาแบบจำลองในการควบคุม

การหาแบบจำลองของมอเตอร์เป็นการประมาณโดยใช้ฟังก์ชั่นถ่ายโอน (Transfer Function) โดยที่มอเตอร์ที่ใช้เป็นมอเตอร์กระแสตรง การประมาณค่าที่ใช้เป็นการประมาณค่าด้วยฟังก์ชั่นการถ่ายโอนอันดับต่ำ ซึ่งง่ายต่อการออกแบบ สามารถแสดงได้ดังรูป



รูปที่ 8 แสดงภาพระบบควบคุม

จากภาพที่แสดงข้างต้นสามารถเขียนเป็นฟังก์ชั่นการถ่ายโอนได้ดังนี้

$$\frac{\Theta(s)}{V(s)} = \frac{\alpha}{s(s+\beta)}$$

โดยที่ α และ βเป็นค่าคงที่

ในการหาค่าของ α และ β สามารถทำได้โดยการป้อนแรงดันแบบขั้น (Step Input) โดยใช้ค่าในช่วงของการทำงาน (Operating Range) ของระบบจริง และผลตอบสนองต่อเวลา (Time Response) ของความเร็วเชิงมุม $\omega(t)$ สร้างกราฟแล้วทำการหาค่า β จากสมการ

$$\beta = \frac{-\ln(1-k)}{T_k}$$

โดยที่ T_k คือเวลาที่ความเร็วเชิงมุมมีค่าเป็น k เท่าของความเร็วเชิงมุมที่สภาวะคงตัว (Steady State)

$$\omega_{ss} = \frac{\alpha V_0}{\beta}$$

ส่วนค่า α สามารถหาได้จากความสัมพันธ์ดังนี้

โดยที่ ω_{ss} เป็นความเร็วเชิงมุมที่สภาวะคงตัว และ V_0 เป็นค่าแอมปลิจูด (Amplitude) ของแรงดันแบบขั้น ให้กับวงจรขับ

2.5 การเขียนกราฟฟิกด้วยโปรแกรมภาษา C

การเขียนกราฟฟิกด้วยภาษา C บนระบบปฏิบัติ DOS เป็นเรื่องที่ยุ่งยากและน่าเบื่อจน programmer หลายคนหนีไปใช้บน Windows แทน แต่ถึงอย่างไรบนระบบปฏิบัติการ DOS ก็ยังมีข้อดีในการ interface PC เข้ากับอุปกรณ์ภายนอกซึ่งง่ายกว่าระบบปฏิบัติการบน Windows มาก

1. ก่อนทำการเขียนกราฟฟิกต้องทำการ Initial Graphic ก่อน เพราะโดยปกติการทำงานของโปรแกรมจะอยู่ในโหมดตัวอักษร (text mode) ดังนั้นหากเราต้องการจะให้โปรแกรมเปลี่ยนไปทำงานในโหมดภาพ จะต้องใช้ฟังก์ชันสำหรับเปลี่ยนโหมดให้มาอยู่ในโหมดภาพ และเมื่อต้องการเปลี่ยนโหมดกลับไปสู่โหมดตัวอักษร จะต้องใช้ฟังก์ชันปิดโหมดภาพ ฟังก์ชันที่ใช้ในการเปลี่ยนโหมดมีดังนี้

```
void far initgraph(int far *graphdriver,int far *graphmode,char far *pathtodriver);
void far closegraph(void);
```

ซึ่งทั้ง 2 ฟังก์ชันนี้อยู่ใน graphics.h โดย ฟังก์ชัน initgraph() เป็นฟังก์ชันสำหรับเปลี่ยนโหมดการทำงานมาเป็นโหมดภาพ ส่วนฟังก์ชัน closegraph() ทำหน้าที่ปิดโหมดภาพให้กลับไปอยู่โหมดตัวอักษร พารามิเตอร์ใน initgraph() มีรายละเอียดดังนี้

graphdriver เป็นตัวชี้สำหรับชี้ไปยังหมายเลขโหมดของจอภาพที่ต้องการ โดยทั่วไปเราจะกำหนดให้เท่ากับ DETECT ซึ่งเป็นค่าคงที่ที่ได้นิยามไว้ใน graphics.h หมายถึงให้ใช้โหมดภาพที่มีความละเอียดสูงสุดที่การ์ดจะสามารถทำงานได้

graphmode เป็นตัวชี้สำหรับชี้ไปยังหมายเลขโหมดย่อย บอกระดับความละเอียดที่ต้องการ ในการใช้งานโดยทั่วไปเราจะไม่เซ็ตค่านี้เมื่อเราใช้ค่าคงที่ DETECT ในการกำหนดโหมดจอให้สูงสุดอยู่แล้ว

pathtodriver คือข้อมูลสตริง ที่หมายถึงชื่อไดเรกตอรีของไดรเวอร์จอภาพ ในการเขียนโปรแกรมภาษาซีและปาสคาลโดยใช้คอมไพเลอร์จากบริษัทบอร์แลนด์ซีนั้น จะแยกโมดูลส่วนควบคุมการแสดงผลออกมาต่างหาก เราเรียกส่วนควบคุมนี้ว่า กราฟิกไดรเวอร์(Graphic Driver) มีนามสกุลเป็น BGI แต่ถ้าเราไม่กำหนดค่าดังในตัวอย่างข้างล่าง โปรแกรมจะค้นหาจากไดเรกตอรีปัจจุบันแทน. ดังนั้น ผู้เขียนโปรแกรมจะต้องตรวจสอบไดเรกตอรีปัจจุบันว่ามีไฟล์นามสกุล BGI หรือไม่ กราฟิกไดรเวอร์ที่เราจะใช้งานประจำ จะเป็นของจอ EGA และ VGA ซึ่งรวมอยู่ด้วยกันมีชื่อว่า EGAVGA.BGI

ตัวอย่างการเขียนโปรแกรมในโหมดภาพโดยใช้ฟังก์ชันของภาษาซี

```
#include <conio.h>
#include <graphics.h>

void main()
{
        int  gdriver=DETECT , gmode;
        initgraph(&gdriver,&gmode,"");          // เปลี่ยนไปสู่กราฟิกโหมด
        // ส่วนวาดกราฟฟิก
        closegraph();                           // เปลี่ยนไปสู่โหมดตัวอักษร
}
```

2. การรับค่าจากคีย์บอร์ด เมื่อเรากดปุ่มใดๆบนคีย์บอร์ด ตำแหน่งของคีย์บอร์ดก็จะให้หมายเลขหนึ่งๆ ซึ่งเรียกว่า scan code ซึ่งมีขนาด 2 ไบต์ โดยปุ่มฟังก์ชันพิเศษ F1-F12 จะมีรูปแบบการส่งค่า scan code ที่แตกต่างจากปุ่มทั่วไปนั่นคือ เมื่อกดปุ่มฟังก์ชันพิเศษจะได้ข้อมูลไบต์แรกเป็น 00F ข้อมูลไบต์ถัดมาก็จะเป็นรหัสของฟังก์ชันพิเศษแต่ละคีย์ โดยใช้ ฟังก์ชัน getch() เป็นฟังก์ชันที่รับค่ามาจากคีย์บอร์ด

ตัวอย่างการรับค่าจากคีย์บอร์ด

```
#define F1          59      // เป็นค่า scan code ของ F1
#define F2          60      // เป็นค่า scan code ของ F2
#define F3          61      // เป็นค่า scan code ของ F3
main()
{
    char ch;
    ch=getch();             // รับค่าไบต์แรกของการกดคีย์
    if (ch==0) ch=getch();  // ถ้าค่าที่รับมาเป็น 0 (แสดงว่าเป็นปุ่มฟังก์ชันพิเศษ)
    switch(ch)
      {
          case F1 : // ไปทำฟังก์ชันที่ต้องการเมื่อกด F1 ;
          case F2 : // ไปทำฟังก์ชันที่ต้องการเมื่อกด F2 ;
          case F3 : // ไปทำฟังก์ชันที่ต้องการเมื่อกด F3 ;
      }
}
```

3. การเขียนปุ่มง่ายๆบน DOS ให้เหมือนกับปุ่มบน Windows โดยใช้หลักการสร้างฉากหลังให้มีสีเทา แล้วลากเส้นสีขาว 2 เส้นที่มุมบนซ้าย และลากเส้นสีดำ 2 เส้นที่มุมล่างขวาก็จะได้ปุ่มที่รอการกดคล้ายกับปุ่มบน Windows ดังแสดงในรูปของโปรแกรมที่รันแล้วใน Main Result ในทางตรงกันข้ามถ้าต้องการวาดปุ่มที่ถูกกดก็ใช้หลักการเดียวกันเพียงแต่สลับเส้นสีขาวและสีดำก็จะได้ปุ่มที่ผ่านการกดแล้ว

2.6 การจัดการอินเตอร์รัพต์

การ interrupt นั้นนับได้ว่าเป็นลักษณะการทำงาน ที่มีความสำคัญมาก ในระบบคอมพิวเตอร์ คือช่วยลดเวลาของ CPU ที่สูญเสียไปในการคอยตรวจสอบว่ามีอุปกรณ์ใดที่ต้องการให้ CPU ทำงานให้บ้าง ดังนั้นการนำเอาวิธีการ interrupt เข้ามาใช้ในระบบ จึงเป็นการทำให้ CPU สามารถทำงานอื่นๆ ได้โดยไม่จำเป็นต้องคอยตรวจสอบอุปกรณ์ต่างๆ ที่เชื่อมโยงกับระบบ และยังทำให้ CPU สามารถทำงานให้อุปกรณ์ต่างๆ เหล่านั้นได้ เมื่ออุปกรณ์นั้นๆ ต้องการ

สำหรับเครื่องพีซี แหล่งที่จะกำเนิดสัญญาณที่จะบอกการเกิดการอินเตอร์รัพต์นี้อาจมาจากฮาร์ดแวร์ เช่นเมื่อเกิดการหารด้วยศูนย์ ซีพียูจะส่งสัญญาณอินเตอร์รัพต์เพื่อให้ระบบปฏิบัติการทราบว่า ค่าที่ได้จากการคำนวณนั้นเป็นอนันต์ เป็นต้น นอกจากนี้ สัญญาณอินเตอร์รัพต์อาจมาจากโปรเซสที่กำลังทำงานอยู่สั่งขัดจังหวะตนเอง อันเป็นวิธีที่เราจะใช้ในการเรียกอินเตอร์รัพต์ฟังก์ชันของระบบปฏิบัติการ ดังนั้นเราอาจแยกประเภทของการอินเตอร์รัพต์ตามแหล่งที่มาได้สองชนิดคือ **ฮาร์ดแวร์อินเตอร์รัพต์** (Hardware Interrupt) เป็นอินเตอร์รัพต์ที่ถูกร้องขอโดยฮาร์ดแวร์ และ **ซอฟต์แวร์อินเตอร์รัพต์** (Software Interrupt) จะถูกร้องขอจากโพรเซสใดโพรเซสหนึ่งที่กำลังทำงานภายในระบบ

Timer Interrupt

ภายใน IBM PC ปกติจะมี interrupt หมายเลข ICH ซึ่งเป็นหมายเลขของ timer interrupt โดยที่ปกติภายใน IBM PC จะทำการนับลงค่าใน Register counter (ปกติจะเก็บค่า maximum คือค่า FFFFH) ด้วยความถี่ 1.19318 MHz เมื่อค่าใน Register counter นับลงจนมีค่าเป็นศูนย์แล้ว ก็จะส่งสัญญาณ interrupt ให้เกิดขึ้นโดย สัญญาณ interrupt ที่เกิดขึ้นจะเกิดขึ้นด้วยความถี่ 18.2 ครั้งต่อวินาที หรือเกิดทุกๆ 55 msec ดังนั้นเราสามารถนำ timer interrupt ค่าปกติของ IBM PC นี้ไปใช้ได้ แต่ก็มีข้อควรระวังคือขนาดของ อินเตอร์รัพต์ฟังก์ชันไม่ควรใหญ่เกินไป ถ้าใหญ่จนใช้เวลามากกว่า 55 msec จะทำให้เกิดอินเตอร์รัพต์ซ้ำซ้อนได้

แต่ถ้าเราต้องการให้มีการ interrupt ที่เวลามากกว่า 55 msec ก็สามารถทำได้โดยการตั้งค่าตัวนับไว้ในโปรแกรมเช่น ต้องการให้เกิดการ interrupt ทุกๆ เวลา 1 sec ก็ทำได้โดยการตั้งตัวนับให้มีค่าเท่ากับ 1/0.055 หรือประมาณ 18 ครั้ง ซึ่งเมื่อเกิดการ interrupt แต่ละครั้งก็จะทำการลดค่าตัวนับลง 1 จนกระทั่งครบ 18 ครั้งก็จะทำ sub-routine ตอบสนองต่อการ interrupt นั้น

แต่ถ้าเราต้องการให้มีการ interrupt ที่เวลาน้อยกว่า 55 msec สามารถทำได้โดยเปลี่ยนค่าใน Register counter โดยมีวิธีการดังนี้

การกำหนด sampling ให้มีค่าน้อยกว่า 55 msec.

ดังนั้นก่อนที่จะมีการตั้งค่า Sampling Time จะต้องมีการกำหนดโหมดการทำงานให้กับ Register counter ก่อนโดยการส่ง Control Byte ไปใส่ที่หมายเลข พอร์ต 43H (Register mode control) ซึ่ง Control Byte นี้แต่ละ Bit จะมีความหมายต่าง ๆ กัน ดังนี้

SC1	SC0	RL1	RL2	M2	M1	M0	BCD

(1) SC คือ SELECT CHANNEL

SC1	SC0	CHANNEL ที่ถูกเลือก
0	0	CHANNEL 0
0	1	CHANNEL 1
1	0	CHANNEL 2
1	1	NOT USED

ในการเซ็ต sampling time นั้นเราจะเลือกใช้แชลเนล 0

(2) READ / WRITE OR LATCH

RL1	RL0	ลักษณะและวิธีเข้าถึงข้อมูล
0	0	ทำการ Latch ค่าใน Register Counter
0	1	read/write data เฉพาะ 8 bit ล่าง
1	0	read/write data เฉพาะ 8 bit บน
1	1	read/write data ทั้ง 16 bit โดยเริ่มจาก low byte ก่อนและตามด้วย high byte

(3) MODE การทำงาน

M2	M1	M0	Mode การทำงาน
0	0	0	mode 0 : Single Time Out
0	0	1	mode 1 : Retriggerable One shot
0	1	0	mode 2 : Rate Generater
0	1	1	mode 3 : Square wave mode
1	0	0	mode 4 : Softwave Triggered Strobe
1	0	1	mode 5 : Hardwave Retriggerable Strobe

และในการทดลองนี้ ได้กำหนดการทำงานเป็น channel 0 .. mode 3 : square wave mode และ มีการอ่านเขียน data ขนาด 16 bit จะได้ control byte เป็น 36H

เมื่อมีการ load ค่า 36H ไปที่พอร์ตหมายเลข 43H เพื่อกำหนดลักษณะการทำงานแล้ว จะต้องมีการ load ค่า counter ให้ Register counter เพื่อเป็นการกำหนด Sampling time โดยค่า counter ในกรณี modde 3 : Square wave mode จะมีค่าเท่ากับ 1.19318MHz คูณกับค่า Sampling time ที่ต้องการ ก็จะได้ค่า counter ออกมา จากนั้นให้ load ค่า counter ที่ต้องการ ไปที่พอร์ตหมายเลข 40H (Reigister counter)

พอร์ตหมายเลข 40H คือ Register counter channel 0

พอร์ตหมายเลข 41H คือ Register counter channel 1

พอร์ตหมายเลข 42H คือ Register counter channel 2

พอร์ตหมายเลข 43H คือ Register mode control

2.7 สัญญาณต่าง ๆบนสล็อตของ IBM/PC

เนื่องจากในการทำ MMI นี้ได้ทำการ์ดตัวทดสอบโปรแกรม MMI ซึ่งประกอบด้วยการ์ด LM628 ซึ่งใช้ทดสอบส่วน Motion Control และการ์ด 8255 ซึ่งใช้ทดสอบส่วน On-Off Control ซึ่งการ์ดทั้งสองนี้มีการอินเตอร์เฟสโดยผ่านสล็อต จึงขอกล่าวถึงรายละเอียดของสล็อต IBM/PC ดังนี้

สำหรับสล็อตบนเมนบอร์ดนั้นจะมีจำนวนขาทั้งสิ้น 62 ขา แบ่งออกเป็น 2 ข้างๆละ 31 ขา ส่วนการเรียกตำแหน่งขาของสล็อตเหล่านี้จะขึ้นอยู่กับว่าขานั้นอยู่ข้างใด (ซ้ายหรือขวา) ของสล็อต โดยขาที่อยู่ทางด้านซ้ายของสล็อตจะเรียกโดยใช้อักษร "B" นำหน้าเลขตำแหน่งขา ส่วนขาที่อยู่ทางด้านขวาของสล็อตจะเรียกโดยใช้อักษร "A" นำหน้าเลขตำแหน่งขา แต่ละขาของสล็อตเหล่านี้จะเชื่อมต่อกับเส้นสัญญาณต่างๆ บนเมนบอร์ด ทำให้การสร้างวงจรอินเตอร์เฟสกับ IBM/PC สามารถทำได้โดยสะดวก ซึ่งเส้นสัญญาณที่เชื่อมต่อกับขาของสล็อตเหล่านี้ประกอบไปด้วย Address Bus , Data Bus , บัสควบคุมการเขียน/อ่านข้อมูลจากหน่วยความจำ หรือพอร์ต I/O , เส้นสัญญาณสำหรับการขออินเตอร์รัพท์ของวงจรอินเตอร์เฟส , เส้นสัญญาณสำหรับการขอ DMA , สัญญาณฐานเวลา (Timing Signal) ต่างๆที่ใช้ในระบบ , เส้นสัญญาณแสดงการรีเฟรชหน่วยความจำ และสัญญาณสำหรับการตรวจสอบความผิดพลาด (I/O CHCK)

นอกจากนี้ สล็อตบนเมนบอร์ดยังเชื่อมต่อกับแหล่งจ่ายไฟต่างๆที่ใช้ในระบบอีกด้วย คือ +5Vdc , -5Vdc , +12Vdc และ –12Vdc

รายละเอียดเกี่ยวกับสัญญาณต่าง ๆ

เนื่องจากในการทำการ์ดอินเตอร์เฟสไม่ได้ต่อสัญญาณกับทุกขาสล็อตดังนั้นจะขอกล่าวรายละเอียดเฉพาะขาสัญญาณที่ใช้กันบ่อยๆ

- RESET DRV (ขา B2) ใช้ในการรีเซ็ทวงจรอินเตอร์เฟสหรืออุปกรณ์ I/O ต่างๆในช่วงที่เริ่มจ่ายไฟให้กับระบบ ซึ่งจะเป็นการทำให้วงจรหรืออุปกรณ์เหล่านั้นถูกปรับให้อยู่ในสภาวะที่แน่นอน ก่อนที่จะเริ่มต้นทำงานในระบบ แต่ถ้าระดับแรงดันของแหล่งจ่ายไฟตก สัญญาณนี้ก็จะถูกทำให้แอคทีฟเช่นกัน

- A0 – A19 (Address Bus : ขา A31 – A12) ใช้กำหนดแอดเดรสของหน่วยความจำ หรืออุปกรณ์ I/O ที่ซีพียูต้องการติดต่อด้วย จะเห็นได้ว่าจำนวนเส้นแอดเดรสนี้มีอยู่ 20 เส้น ซึ่งสามารถอ้างแอดเดรสของหน่วยความจำได้ถึง 1 Mbyte แต่อย่างไรก็ตามจะมีแอดเดรสบางแอดเดรสที่ถูกใช้งานโดย IBM/PC อยู่ก่อนแล้ว สำหรับการอ้างอิงแอดเดรสของพอร์ท I/O นั้นจะใช้แอดเดรสเพียง 16 เส้นคือ A0-A15 ทำให้สามารถอ้างแอดเดรสของพอร์ทได้ 64K พอร์ท แต่อย่างไรก็ตาม

ภายใน IBM/PC จะใช้เส้นแอดเดรสในการอ้างแอดเดรสของพอร์ทเพียง 10 เส้น คือจาก A0-A9 และค่าแอดเดรสที่ใช้งานจะต้องอยู่ในช่วง 0200H-03FFH เท่านั้น

- D0 – D7 (Data Bus : ขา A9 – A2) ขาสัญญาณนี้จะเป็นแบบ Bi-Direction ซึ่งต่อกับบัสข้อมูลของระบบ
- ALE (Address Latch Enable : ขา B28) แสดงให้อุปกรณ์ภายนอกทราบว่าแอดเดรสที่ซีพียูต้องการจะติดต่อด้วยนั้นถูกส่งออกมาบนบัสแอดเดรสแล้ว โดยที่สัญญาณ ALE นี้จะเปลี่ยนจากลอจิก "1" เป็น "0" เมื่อค่าแอดเดรสที่ถูกต้องถูกส่งออกมาบนบัสข้อมูลเรียบร้อยแล้ว
- IRQ2 - IRQ7 (Interrupt Request 2 Through 7 : ขา B4 และ B25 – B21) ใช้สำหรับการขออินเตอร์รัพท์จากซีพียู โดยสัญญาณเหล่านี้จะต่อเข้ากับ 8259A บนเมนบอร์ดโดยตรง ซึ่ง IRQ2 มีลำดับความสำคัญสูงสุด (Highest Priority) และ IRQ7 มีลำดับความสำคัญต่ำสุด
- $\overline{\text{IOR}}$ (I/O Read : ขา B14) เป็นการอ่านข้อมูลจากพอร์ท I/O เพื่อให้พอร์ท I/O ที่มีแอดเดรสตรงกับแอดเดรสบนบัสแอดเดรสนั้น ส่งข้อมูลออกมาบนบัสข้อมูล
- $\overline{\text{IOW}}$ (I/O Write : ขา B13) เป็นการเขียนข้อมูลลงพอร์ท I/O เพื่อให้พอร์ท I/O ที่มีแอดเดรสตรงกับแอดเดรสบนบัสแอดเดรสนั้น รับข้อมูลที่อยู่บนบัสข้อมูลไปเก็บไว้
- $\overline{\text{MEMW}}$ (Memory Write : ขา B11) ส่งออกมาเพื่อให้หน่วยความจำที่แอดเดรสตรงกันกับค่าแอดเดรสบนแอดเดรสนั้น ทำการรับข้อมูลที่อยู่บนบัสข้อมูลไปเก็บไว้
- $\overline{\text{MEMR}}$ (Memory Read : ขา B12) ส่งออกมาเพื่อให้หน่วยความจำที่แอดเดรสตรงกันกับค่าแอดเดรสบนแอดเดรสนั้น ทำการส่งอมูลที่ออกมาบนบัสข้อมูล

AEN (Address Enable) สัญญาณนี้เป็นเอาท์พุทที่ใช้ในการแสดงว่าบัสไซเคิลที่เกิดขึ้นในช่วงเวลาที่สัญญาณ AEN แอคทีฟ (ลอจิก "1") นั้นเป็นบัสไซเคิลของขบวนการ DMA สำหรับบนเมนบอร์ดของ IBM/PC นั้น จะใช้สัญญาณนี้ในการดิสเอเบิ้ล (Disable) 8288 Bus Controll และใช้ดิสเอเบิ้ลพอร์ท I/Oต่างๆที่ไม่เกี่ยวข้องกับขบวนการ DMA

บทที่ 3
โครงสร้างของฮาร์ดแวร์

ในโครงงานการพัฒนาระบบเชื่อมต่อนี้มีส่วนของฮาร์ดแวร์ ช่วยในการเชื่อมต่อกับอุปกรณ์ภายนอกที่ต้องการจะควบคุม 2 ส่วนคือ การ์ด LM628 เชื่อมต่อกับมอเตอร์ใช้ในส่วนควบคุมการเคลื่อนที่ (Motion Control) และ การ์ด 8255 เพื่อเชื่อมต่อกับอุปกรณ์ที่เกี่ยวข้องกับการเปิดปิด (On-Off Control)

การ์ด LM628

การ์ด LM628 ทำหน้าที่เป็นตัวควบคุม(Controller) จุดเปรียบเทียบความแตกต่าง (Summing point) และจุดอ้างอิง(Set point) ใน Block diagram ของระบบควบคุมวงปิด ซึ่งภายในจะมีชิป LM628 ซึ่งมีรายละเอียดดังนี้

Chip LM628

ใช้เป็น motion-control processors มี 8-bit output สามารถส่งข้อมูลให้กับทั้ง 8-bit หรือ 12-bit DAC โดยภายใน chip LM628 นั้นมี PID Compensation Filter เป็นตัวชดเชยของ control loop เป็นไปตามสมการ

$$u(n) = kp*e(n) + ki \sum_{n=0}^{N} e(n) + kd[\, e(n') - e(n'-1) \,]$$

เมื่อ u(n) คือ output ของสัญญาณควบคุมมอเตอร์ ณ. sampling time n

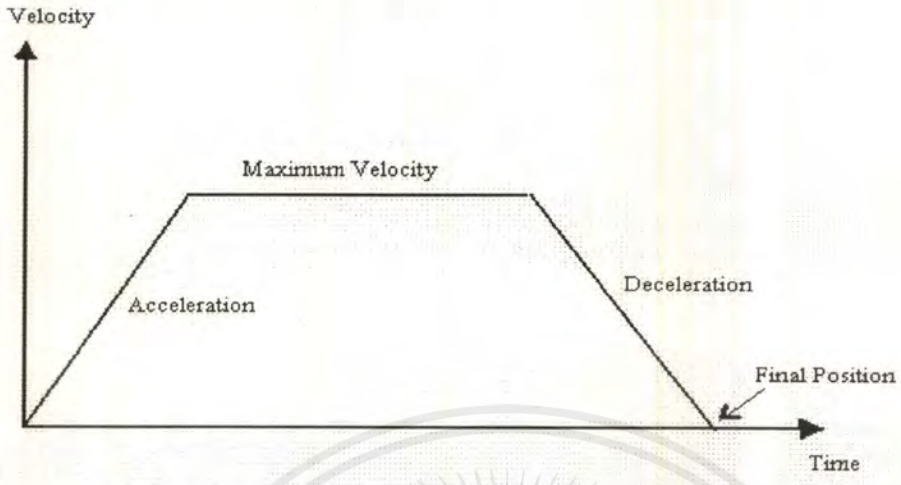
e(n) คือ position error ณ. sampling time n

n' คือ การ sampling ณ. derivative sampling rate

ส่วน kp , ki และ kd คือ ค่า discrete-time filter parameters ถูกกำหนดโดยผู้ใช้

นอกจากนี้ภายใน chip LM628 ยังมีอีกส่วนหนึ่งที่เรียกว่า Trajectory Generation ซึ่งทำหน้าที่สร้างเส้นทาง trajectory โดยเมื่อควบคุมใน velocity mode นั้น host processor จะต้องทำการระบุค่า ความเร่ง (acceleration) , ความเร็วสูงสุด (maximum velocity) และ ตำแหน่งสุดท้าย (final position) LM628 จะใช้ข้อมูลนี้ในการกำหนดการเคลื่อนที่ โดยเริ่มต้นจะเคลื่อนที่ตามความเร่งที่ระบุจนกระทั่งถึงค่าความเร็วสูงสุด และคงค่าความเร็วนั้น แล้วก็จะเคลื่อนที่ด้วย

ความหน่วงซึ่งมีค่าเท่ากับความเร่งจนความเร็วเป็นศูนย์ ณ. จุดที่ความเร็วเป็นศูนย์นี้คือ final position ที่กำหนดไว้ตั้งแต่ตอนแรก แสดงดังรูปที่ 9



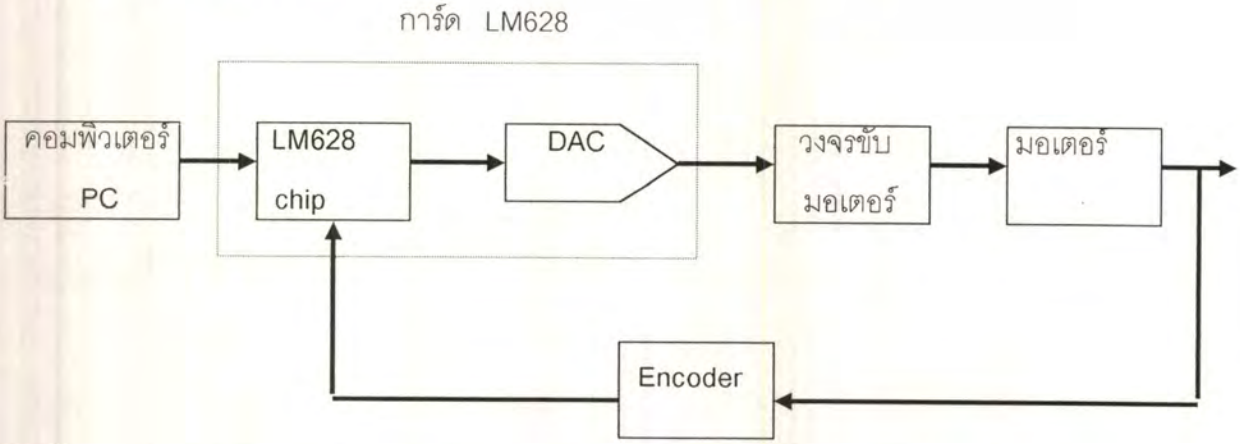
รูปที่ 9 แสดง Trajectory ใน velocity mode

แต่ถ้า motor ไม่สามารถคงค่าความเร็วที่ระบุไว้ได้ นั่นคือจะทำให้เกิดความคลาดเคลื่อนของตำแหน่งมากขึ้น ส่วนรายละเอียดอื่นๆ ของ LM628 แสดงอยู่ใน ภาคผนวก

เมื่อนำการ์ด LM628 ไปใช้งาน ตัวการ์ดจะเสียบติดอยู่กับ slot ของ PC โดยที่ PC จะทำหน้าที่เป็น Host processor ควบคุมการทำงานอีกทอดหนึ่ง การ์ด LM628 จะส่งสัญญาณควบคุม Analog เอาท์พุทซึ่งอยู่ในช่วง ±10 โวลต์ ให้กับวงจรขับมอเตอร์ วงจรขับมอเตอร์จะขยายสัญญาณควบคุมที่ได้รับ เพื่อขับมอเตอร์ต่อไป

<u>การนำการ์ด LM628 ไปใช้งาน</u>

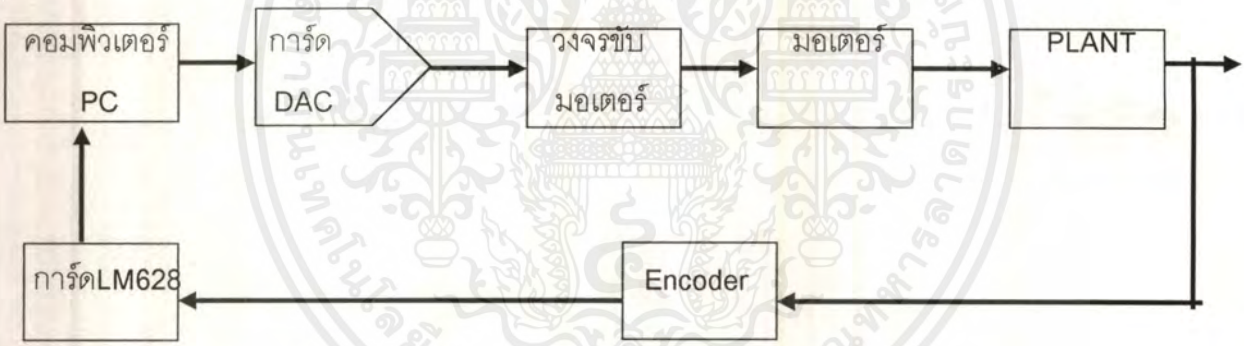
1. ใช้การ์ด LM628 ทำหน้าที่เป็นตัวกำเนิดสัญญาณอ้างอิงทราเจ็คทอรี่ ตัวควบคุม PID และตัวนับสัญญาณพัลส์ของ Encoder เพื่อเก็บข้อมูลตำแหน่งของ motor ในการควบคุมการหมุนของ motor ดังแสดงในรูปที่ 10.



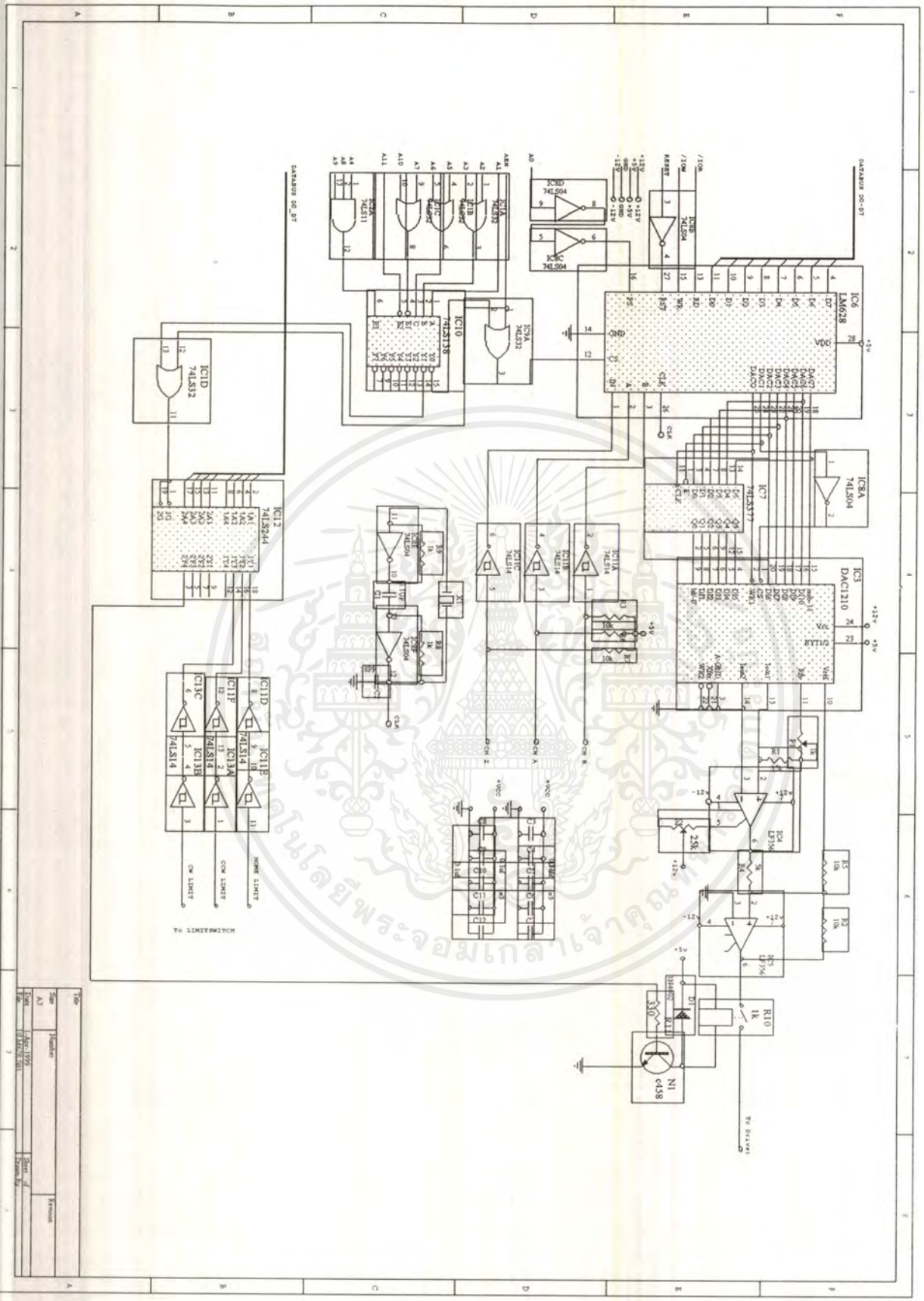
รูปที่ 10. แสดงการใช้งานการ์ด LM628 ในการควบบคุมแบบ PID

2. ใช้การ์ด LM628 นับสัญญาณพัลส์ของ Encoder เท่านั้น ในการเก็บข้อมูลตำแหน่งของ motor ในขณะหมุน เพื่อใช้ Iden plant ซึ่งงานนี้ PC จะทำหน้าที่สร้างสัญญาณอ้างอิง sinusoid และส่งสัญญาณอ้างอิง sinusoid ที่สร้างได้ให้การ์ด DAC ในรูปของสัญญาณ digital ต่อมาการ์ด DAC จะส่งสัญญาณ Analog ซึ่งอยู่ในช่วง ±5 โวลต์ ให้วงจรขับมอเตอร์ต่อไป ดังแสดงในรูปที่ 11.

3. ใช้การ์ด LM628 ทำหน้าที่เป็นตัวกำเนิดสัญญาณอ้างอิงทราเจ็คทอรี่ และนับสัญญาณพัลส์ของ Encoder ในการควบคุมแบบปรับค่าได้ ซึ่งงานนี้ PC จะทำหน้าที่อ่านตำแหน่งและความเร็วของสัญญาณอ้างอิงทราเจ็คทอรี่และตำแหน่งและความเร็วจริงของ Plant จากการ์ด LM628 เพื่อนำข้อมูลเหล่านี้มาคำนวณหาความเร่งอ้างอิงของสัญญาณทราเจ็คทอรี่ สร้างสัญญาณควบคุมและปรับค่าพารามิเตอร์ของระบบ ต่อมาส่งสัญญาณควบคุมให้การ์ด DAC ในรูปของสัญญาณ digital การ์ด DAC จะส่งสัญญาณ Analog ซึ่งอยู่ในช่วง ±10 โวลต์ ให้วงจรขับมอเตอร์ต่อไป ดังแสดงในรูปที่ 11.



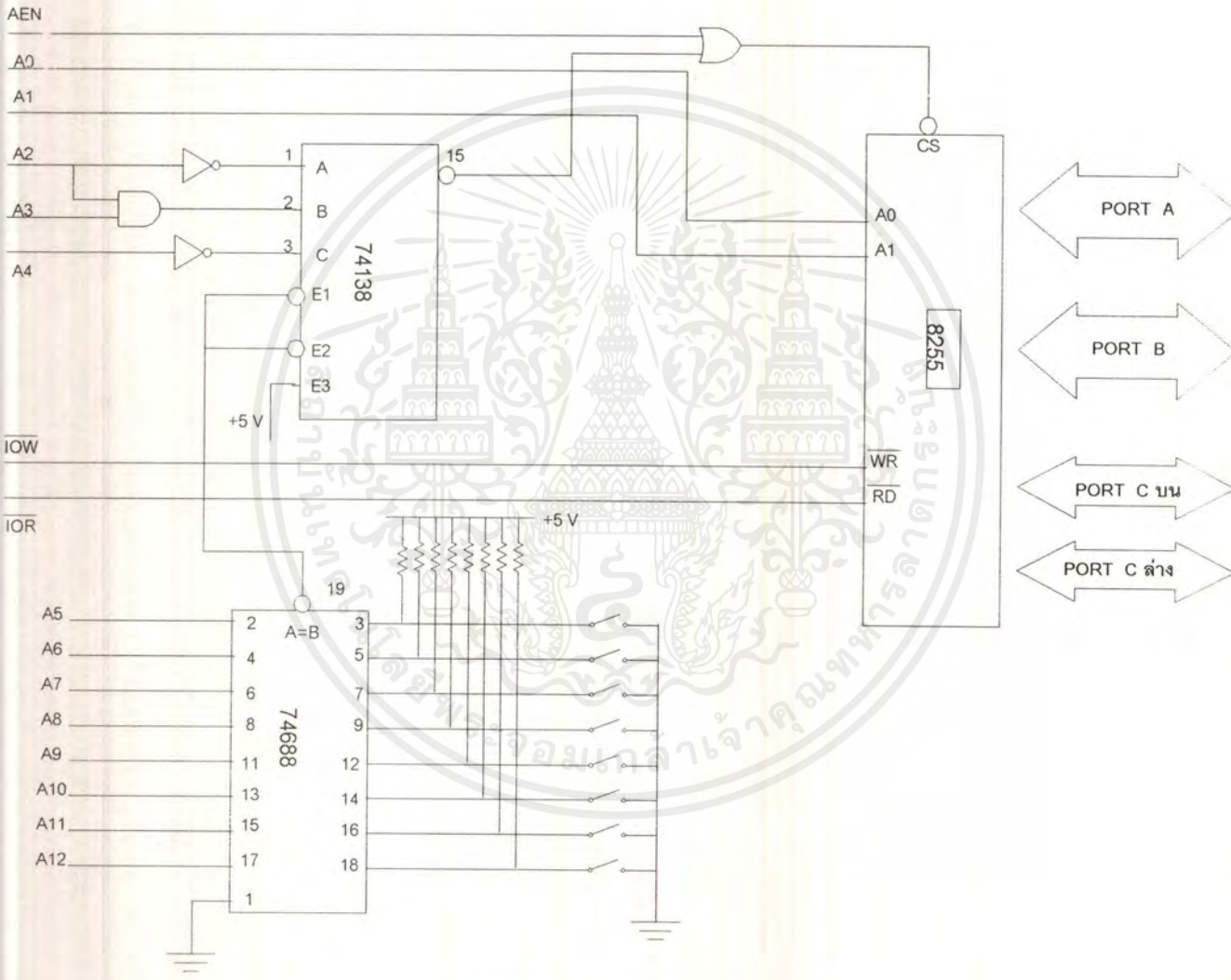
รูปที่ 11. แสดงการใช้งานการ์ด LM628 ในงานเก็บข้อมูล



รูปที่ 12 แสดงวงจรของการ์ด LM628

การ์ด 8255

ในการใช้งานเครื่อง CNC จริงๆนั้นนอกจากจะมีส่วน motion control เช่นในการควบคุมการเคลื่อนที่ของหัวกัดแล้วยังมีส่วนของ on/off control ในส่วนของการควบคุมการเปิดปิดอุปกรณ์ต่างๆ เช่น Coolant , Limit Switch X , Limit Switch Y , Power , Emergency ฯลฯ จึงใช้ชิป 8255 มาใช้ในส่วน on/off (รายละเอียดของ 8255 ได้อธิบายไว้ในภาคผนวก) ทำเป็นการ์ด 8255 ใช้อินเตอร์เฟสกับ MMI เพื่อใช้ในการ control การ on/off โดยวงจรการ์ด 8255 มีรายละเอียดดังรูป



รูปที่ 13 แสดงวงจรการ์ด 8255

ในวงจรจะใช้ชิป 74688 เป็นตัวเปรียบเทียบค่าลอจิกของทั้งสองฝั่ง ถ้าทั้งสองฝั่งมีลอจิกเหมือนกันก็จะส่งสัญญาณแอคทีฟ "0" นั่นคือเราจะใช้ dip switch ที่ต่อกับ 74688 เป็นตัวปรับย่านแอดเดรส และใช้ชิป 74138 เป็นตัว Decoder เพื่อเลือก 8255 ที่จะติดต่อ

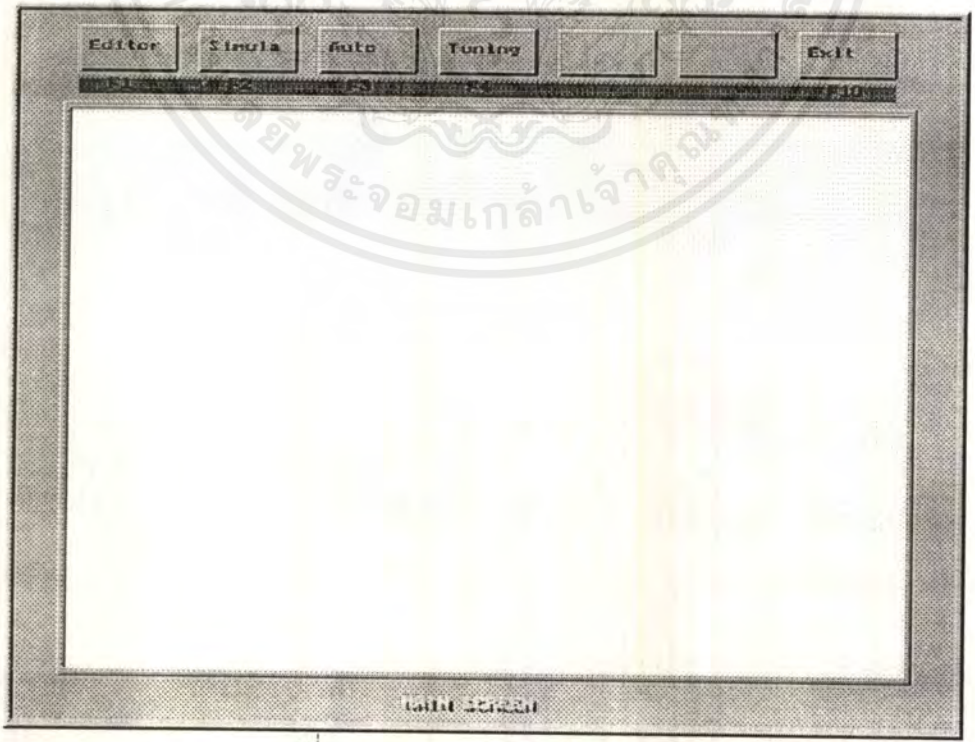
ในการติดต่อกับโปรแกรม MMI นั้นได้กำหนดให้พอร์ท A เป็นเอาท์พุทพอร์ท เพื่อติดต่อกับอุปกรณ์ที่ต้องการการ on/off จาก MMI ได้แก่ Coolant , Enable Drive X , Enable Drive Y และกำหนดให้พอร์ท B เป็นอินพุทพอร์ทเพื่อดูสถานะอุปกรณ์ต่างๆ ได้แก่ Limti Switch X + , Limit Switch X - , Limti Switch Y + , Limit Switch Y - , Emergency , Power

บทที่ 4
โครงสร้างของโปรแกรม

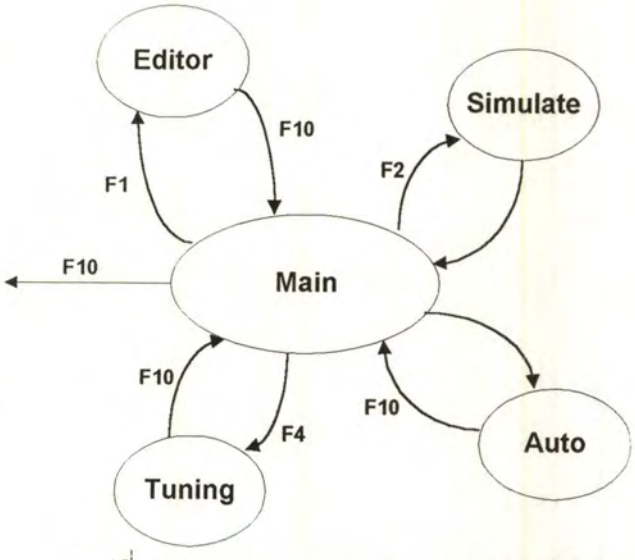
จากทฤษฎีที่ได้กล่าวมาข้างต้นทั้งหมดข้างต้นนี้คือทฤษฎีในการทำMMIในการเขียนโปรแกรม ได้ใช้โปรแกรมภาษาซีเขียนบนระบบปฏิบัติการดอสเขียนโปรแกรมโดยมี ความสามารถดังนี้

1. Editor มีความสามารถในการ โหลดไฟล์ G-Code และ แปลงจาก G-Code มาเป็น action command
2. Simulation สามารถดูผลการเคลื่อนที่ 2 แกนตามไฟล์ G-Code ก่อนส่งออกไปเคลื่อนที่จริง
3. Auto มีหน้าที่ส่ง action command ที่แปลงมาจาก Editor เพื่อไปควบคุมจริง และมีส่วนของการควบคุมแบบเปิดปิด
4. Tuning มีความสามารถในการหา Transfer Function ของมอเตอร์ และมีส่วนกำหนดค่าเกน PID

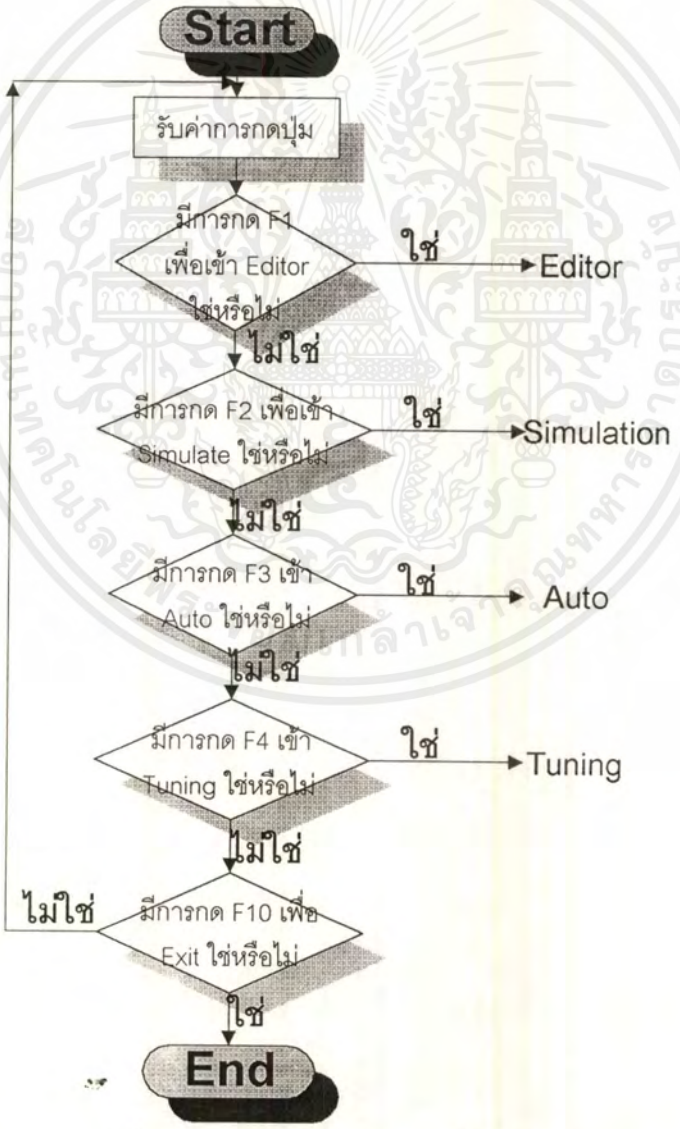
แสดงผลการรันหน้าจอ main , dataflow diagram และ flowchart ดังรูปที่ 14 รูปที่ 15 และรูปที่ 16 ตามลำดับ



รูปที่ 14 แสดงผลการรันหน้าจอ main



รูปที่ 15 แสดง dataflow diagram ของหน้าจอ main



รูปที่ 16 แสดง flowchart ของหน้าจอ main

Editor

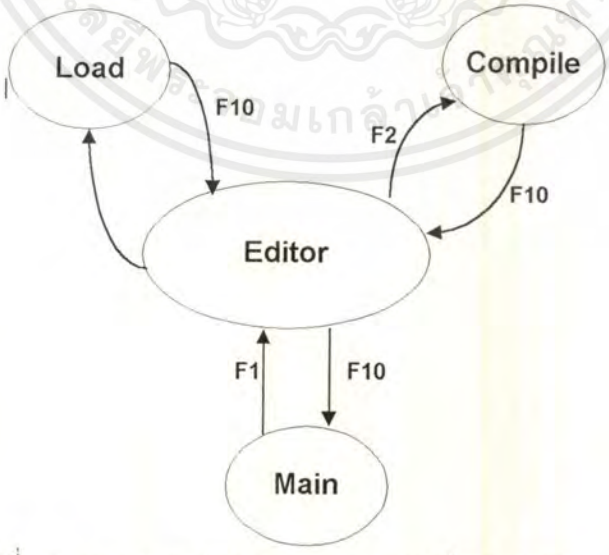
เมื่อกด F1 จากหน้าจอ main จะเข้าสู่หน้าจอ editor ซึ่งมีความสามารถดังนี้

- โหลดไฟล์ G-Code ที่เขียนขึ้นจาก editor ใดๆ แล้ว save เป็นนามสกุล .cnc แสดงตัวอย่างการโหลดไฟล์ G-Code ที่ชื่อ test.cnc ดังรูปที่ 18

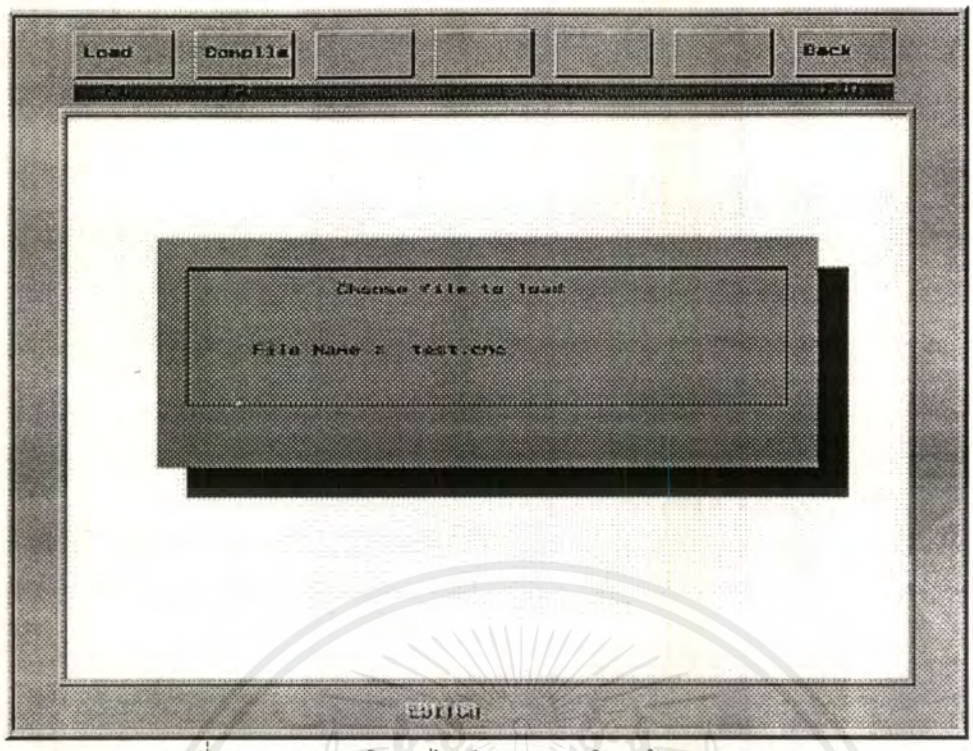
- NC Imterpreter ในการใช้งานเครื่องจักรกล CNC เพื่อผลิตชิ้นงานจำเป็นต้องมีคำสั่ง (Command) หรือรหัส (Code) ซึ่งใช้แทนลักษณะของการทำงานนั้น ๆ และต้องเป็นรหัส ที่ผู้ใช้งานเครื่องจักรกล CNC สามารถใช้งานได้ง่าย สำหรับโครงงานนี้ จะใช้ NC code หรือนิยมเรียกสั้น ๆว่า G code เป็นรหัสในการปฏิบัติงาน บางคนอาจสงสัยว่าทำไมจึงเลือก G code เป็นรหัสในการปฏิบัติงาน ? โดยเหตุผลสำคัญที่เลือก G code เป็นรหัสในการปฏิบัติงาน ก็เพราะ G code เป็นภาษา หรือรหัสซึ่งเป็นที่นิยมใช้กันในปัจจุบัน อีกทั้งโครงงานนี้มุ่งเน้นที่จะพัฒนาระบบเชื่อมต่อกับผู้ใช้ บนรากฐานของระบบเปิด จึงต้องคำนึงถึงมาตรฐาน และความนิยมแพร่หลายเป็นหลัก

แต่สำหรับการทำงานจริง เครื่องจักรกล CNC ไม่สามารถที่จะเข้าใจและปฏิบัติงานตาม G code ได้ทันที ซึ่งจำเป็นต้องมีส่วนที่ทำหน้าที่นำเอา G code ดังกล่าว มาตีความเพื่อให้เกิดเป็นคำสั่งในการปฏิบัติงาน (Action command) โดย G code 1 ตัว อาจประกอบด้วยคำสั่งในการปฏิบัติงาน เพียง หนึ่ง คำสั่งหรือมากกว่าก็ได้ ซึ่งส่วนนี้คือ "NC Interpreter"

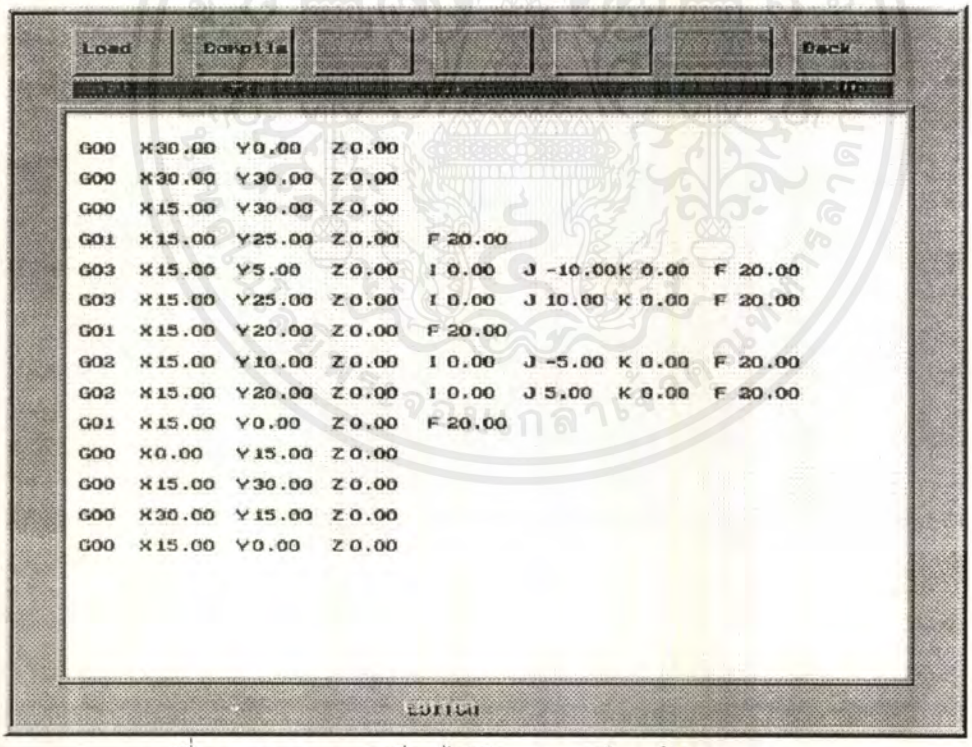
แสดง dataflow diagram และ flowchart ของหน้าจอ editor ดังรูปที่ 17 และ 20



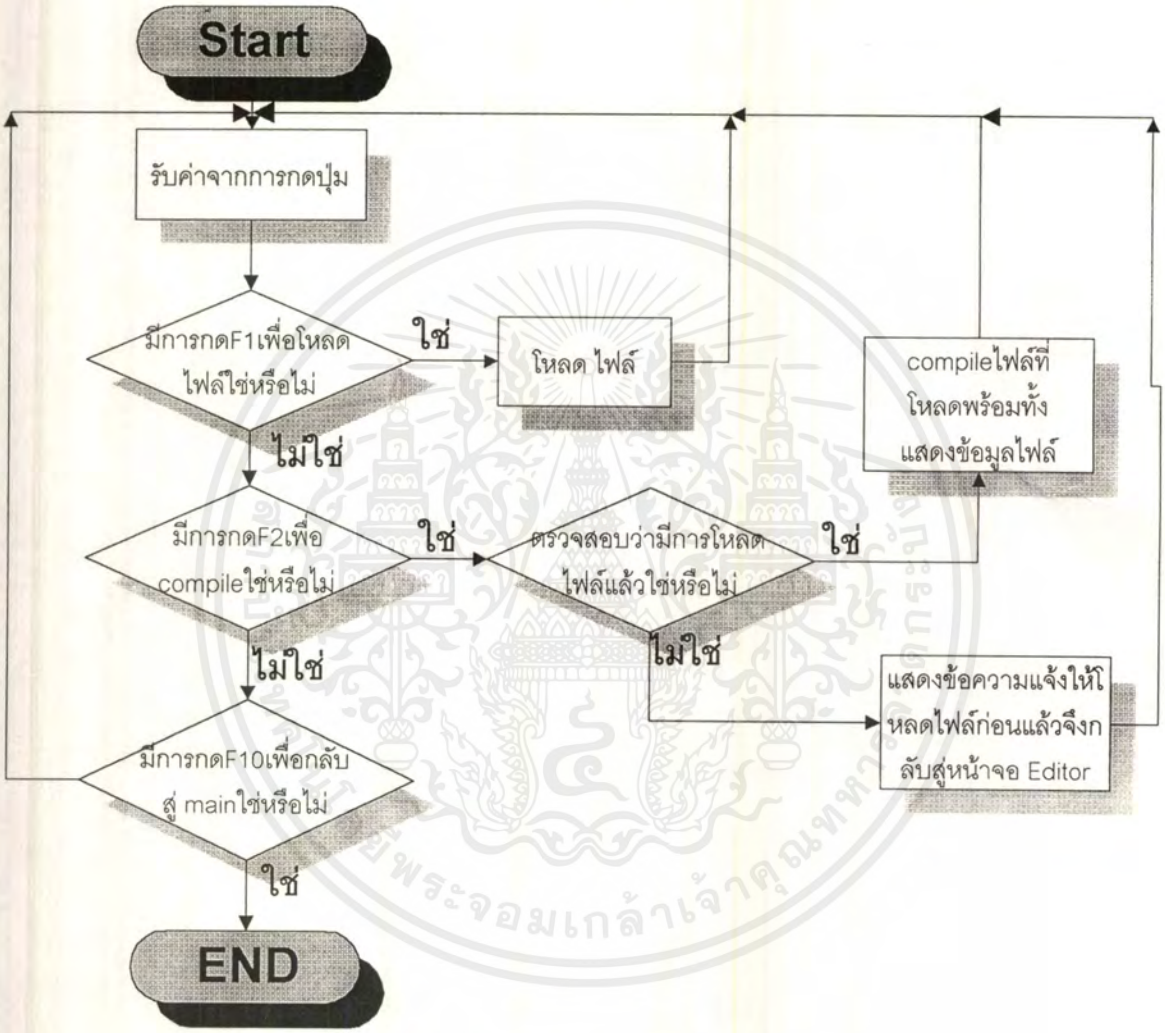
รูปที่ 17 แสดง dataflow diagram ของหน้าจอ editor



รูปที่ 18 แสดงการโหลดไฟล์ test.cnc ในหน้าจอ Editor



รูปที่ 19 แสดงรายละเอียดไฟล์ test.cnc ในหน้าจอ Editor

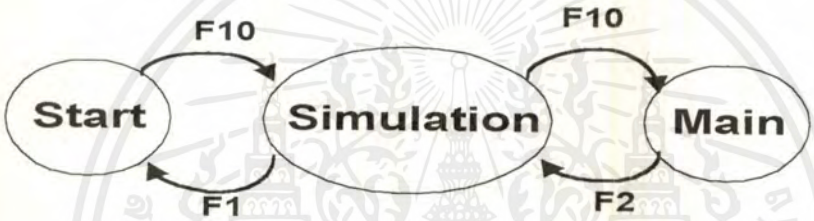


รูปที่ 20 แสดง flowchart ของหน้าจอ Editor

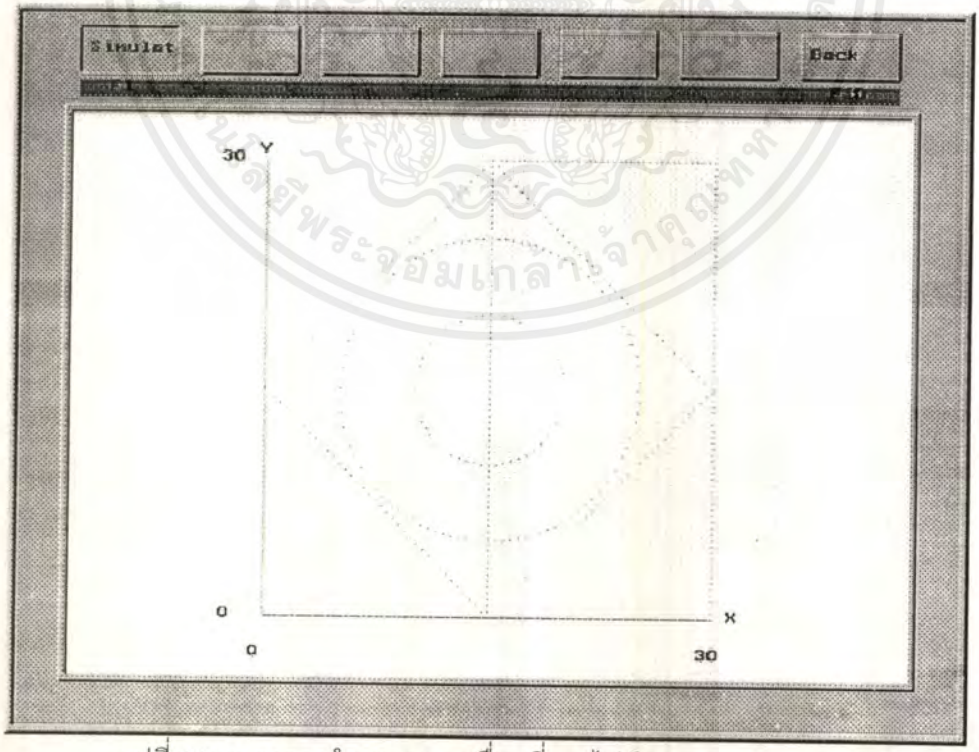
Simulation

ในการทำงานจริง ถึงแม้ผู้ใช้เครื่องจักรกล CNC จะมีความชำนาญเกี่ยวกับการเขียน G code มากเพียงใดก็ตาม ก็มักจะเกิดข้อผิดพลาดอยู่เสมอ ๆ และถ้าโปรแกรมที่มีข้อผิดพลาด ถูกนำไปใช้งาน ก็จะทำให้ชิ้นงานที่ผลิตออกมาเสียหาย จึงจำเป็นต้องมี " Simulation" ที่ใช้จำลองการทำงานของ G code ดังกล่าวเพื่อให้ทราบถึง Tool path ของมีดกัด ก่อนส่งให้เครื่องจักรกล CNC ซึ่งจะช่วยให้เห็นข้อผิดพลาด ก่อนความผิดพลาดจริง

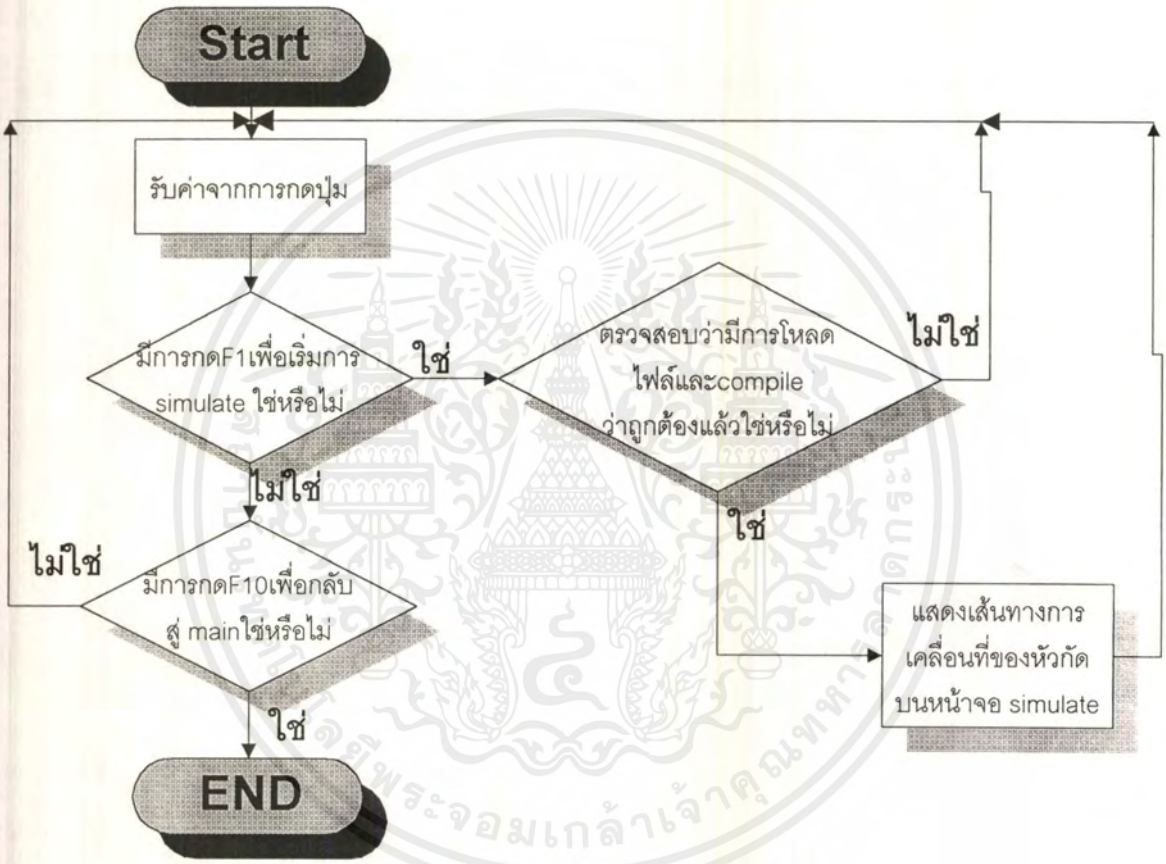
เมื่อกด F2 จากหน้าจอ main ก็จะเข้าสู่หน้าจอ simulation สำหรับจำลองการเคลื่อนที่ของไฟล์ที่โหลดมา โดยไฟล์ที่จะทำการจำลองการเคลื่อนที่นั้นต้องผ่านการ แปลงเป็น action command จากหน้าจอ editor เสียก่อน รูปที่ 22 แสดงตัวอย่างการจำลองการเคลื่อนที่ของไฟล์ test.cnc รูปที่ 21 และรูปที่ 23 แสดง dataflow diagram และ flowchart ของ หน้าจอ simulation



รูปที่ 21 แสดง dataflow diagram ของหน้าจอ simulation



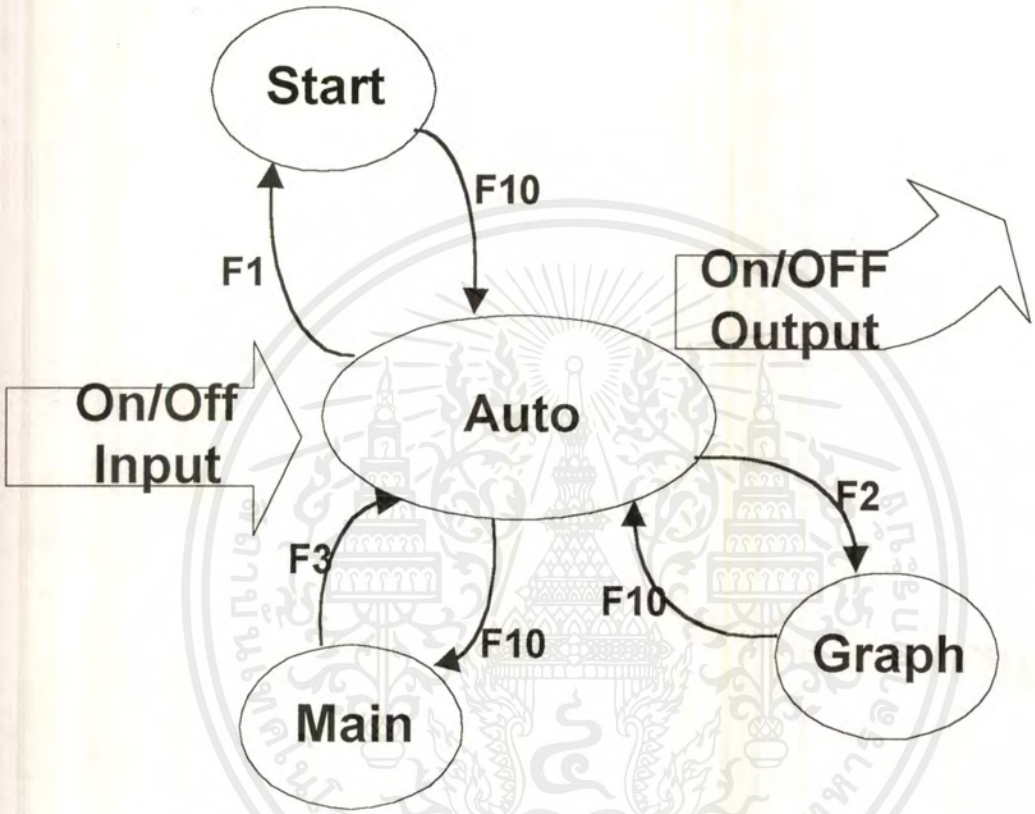
รูปที่ 22 แสดงการจำลองการเคลื่อนที่ของไฟล์ test.cnc



รูปที่ 23 แสดง flowchart ของหน้าจอ simulation

Auto

เมื่อกดปุ่ม F3 จากหน้าจอ main ก็จะเข้าสู่หน้าจอ auto โดยแสดง dataflow diagram ของหน้าจอ auto ดังรูปที่ 24



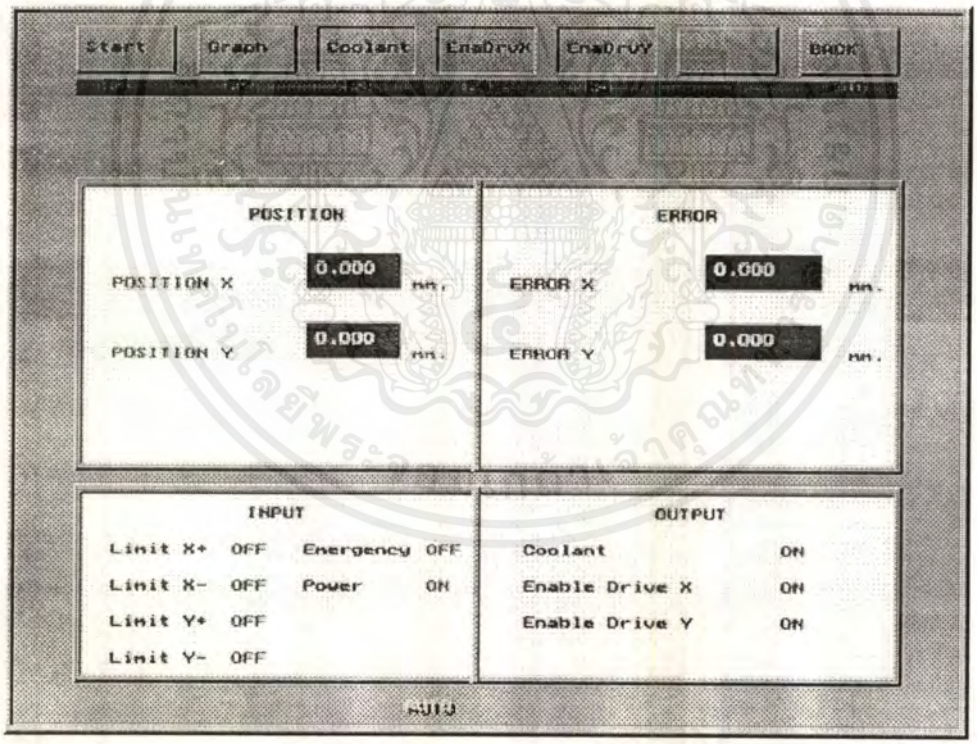
รูปที่ 24 แสดง dataflow diagram ของหน้าจอ auto

โดยในส่วน auto จะเป็นส่วนที่รับส่งค่ากับอุปกรณ์ภายนอก แบ่งออกเป็น

1. on/off control ในส่วนนี้หน้าจอ auto จะส่งค่าเป็นดิจิตอลเพื่อใช้เป็นสัญญาณเปิดปิดอุปกรณ์ภายนอก โดยจะใช้การ์ด 8255 เป็นตัวอินเตอร์เฟสระหว่างโปรแกรมและอุปกรณ์ภายนอก ซึ่งจะกำหนดให้พอร์ต A ของการ์ด 8255 เป็นเอาท์พุทพอร์ทใช้ส่งค่า on/off ให้กับ coolant , drive_x , drive_y โดยการกด F3,F4 และ F5 ตามลำดับ และกำหนดให้พอร์ท B เป็นอินพุทพอร์ท ใช้รับค่า on/off จาก limit switch x , limit switch y , emergency , power

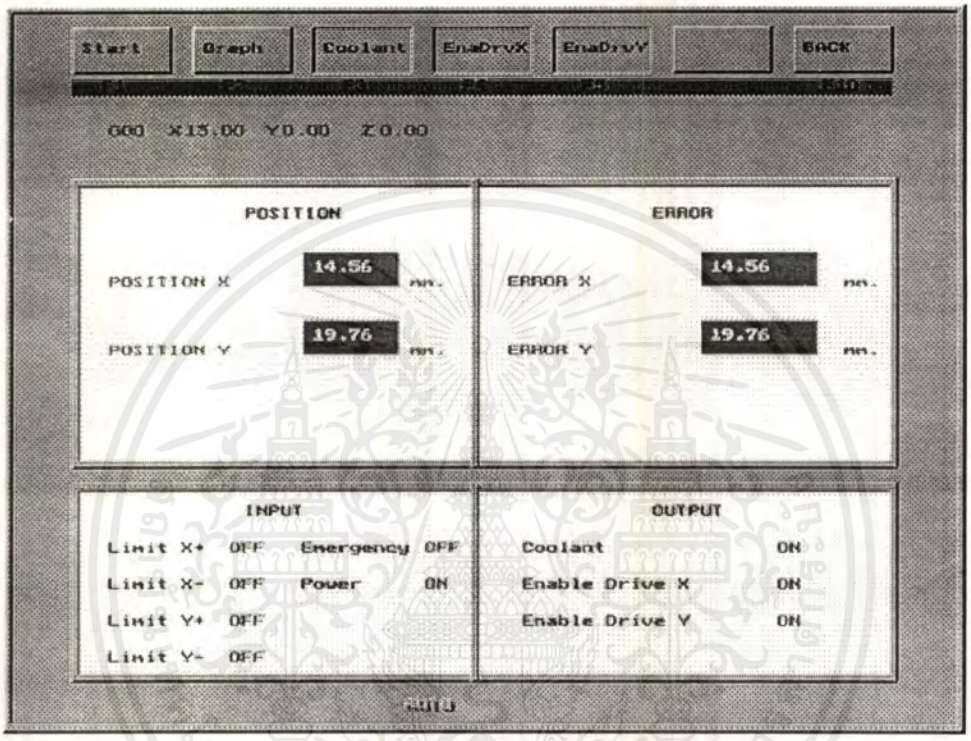
โดยในการทดสอบได้ต่อพอร์ท A เข้ากับ LED เมื่อมีการเปิด coolant , drive_x , drive_y ก็จะติด LED กดปุ่มอีกครั้งก็เป็นการปิด LED และต่อพอร์ท B เข้ากับสวิตซ์ เมื่อมีการเปิด/ปิดสวิตซ์ภายนอกก็จะเกิดการเปลี่ยนแปลงสถานะ on/off ภายในหน้าจอ auto

ก่อนการส่งค่าในการควบคุมการเคลื่อนที่ของมอเตอร์ ต้องทำการเปิด power , coolant ,drive_x , drive_y แสดงดังรูปที่ 25



รูปที่ 25 แสดงรูปการ on/off อุปกรณ์ภายนอกก่อนเริ่ม motion control

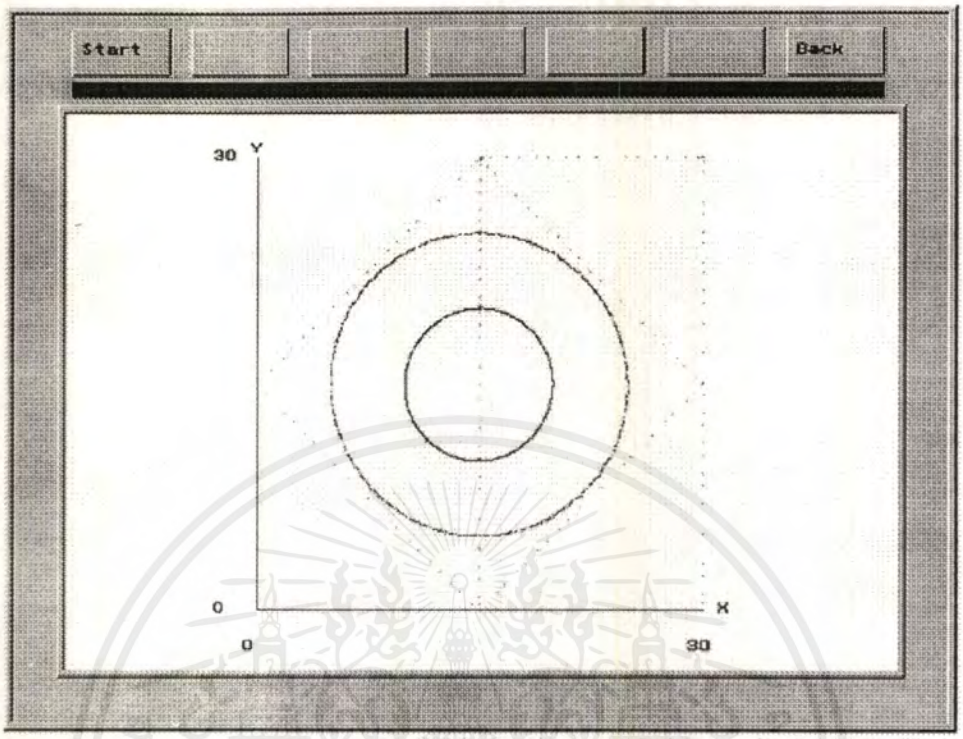
2. motion control โดยการกดปุ่ม start โปรแกรมก็จะนำ action command มาแปลงเป็น count ซึ่งเป็นสัญญาณที่ชิป LM628 เข้าใจและจะใช้หมุนมอเตอร์ต่อไป พร้อมกับอ่านค่า count มาจาก encoder เพื่อเปรียบเทียบตำแหน่ง set point ที่ส่งไปว่ามีค่าเท่ากันแล้วหรือไม่ ถ้าเท่ากัน (แสดงว่ามอเตอร์วิ่งไปถึงตำแหน่งที่ต้องการ) ก็จะนำค่าตำแหน่งต่อไปส่งไปควบคุมมอเตอร์อีก โดย รูปที่ 26 แสดง motion control เมื่อกดปุ่ม start



รูปที่ 26 แสดงรูป motion control ของไฟล์ test.cnc

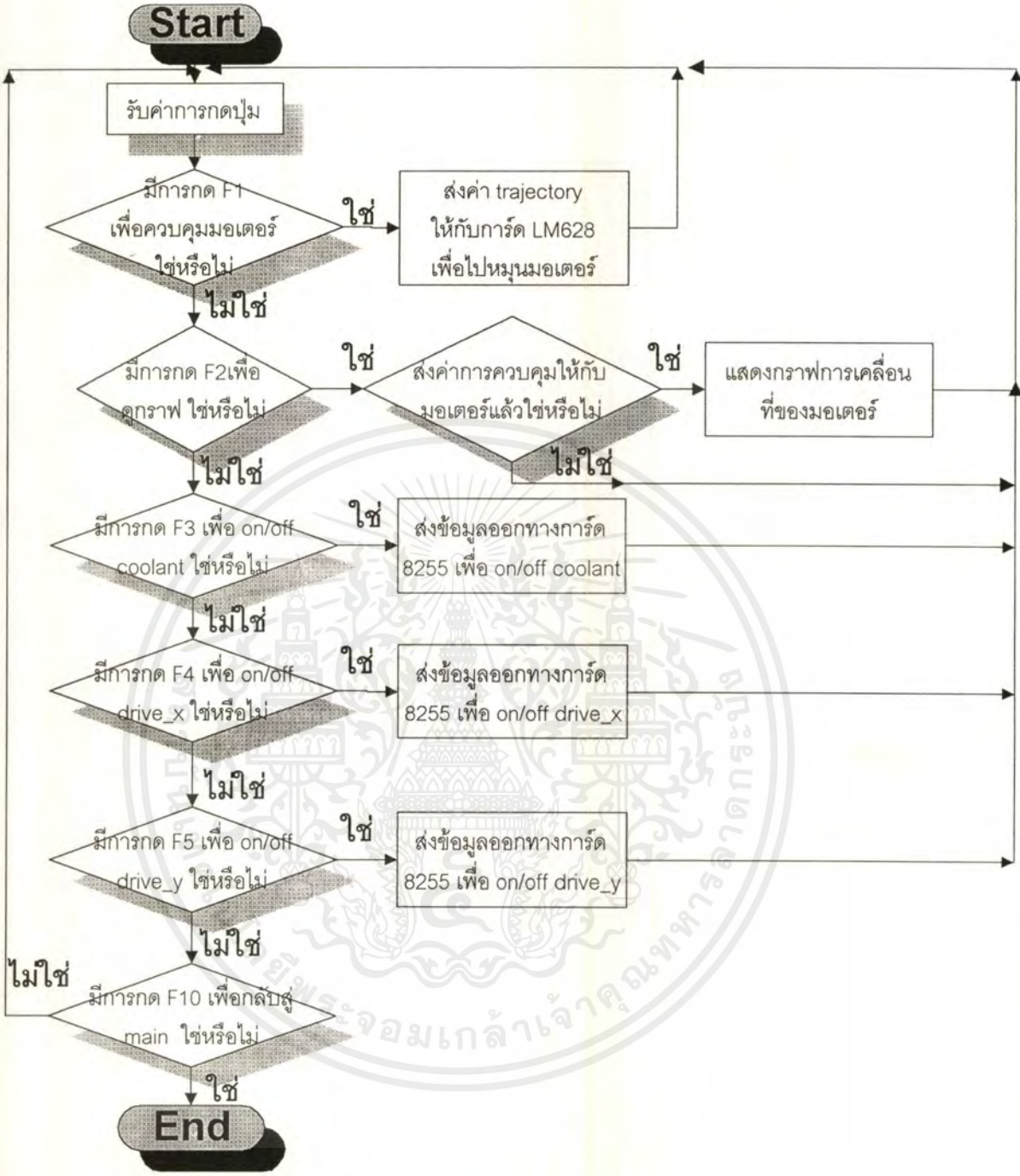
จากรูปที่ 26 ยังไม่ได้ต่อกับมอเตอร์จริงไม่ได้มีการอ่านค่าจริง (real position) เข้ามา เป็นเพียงการอ่าน set point (design position) จากการ์ดเท่านั้น ดังนั้นค่า error (คือค่าแตกต่างกันระหว่าง real position และ design position) จึงมีค่าเท่ากับ set point (design position)

เมื่อกด F2 ก็จะเป็นการแสดงกราฟการเคลื่อนที่ของมอเตอร์ 2 แกน (มีดกัด) ที่ได้จากการอ่านค่า design position โดยแสดงดังรูปที่ 27



รูปที่ 27 แสดงกราฟของไฟล์ test.cnc เมื่อยังไม่ได้ต่อกับมอเตอร์จริง

จากรูปจะเห็นว่ากราฟของ set point มีลักษณะเหมือนกับกราฟที่ได้จากการ simulate นั่นคือในอุดมคติหัวกัดควรที่จะมีการเคลื่อนที่ตามรูปที่ 22

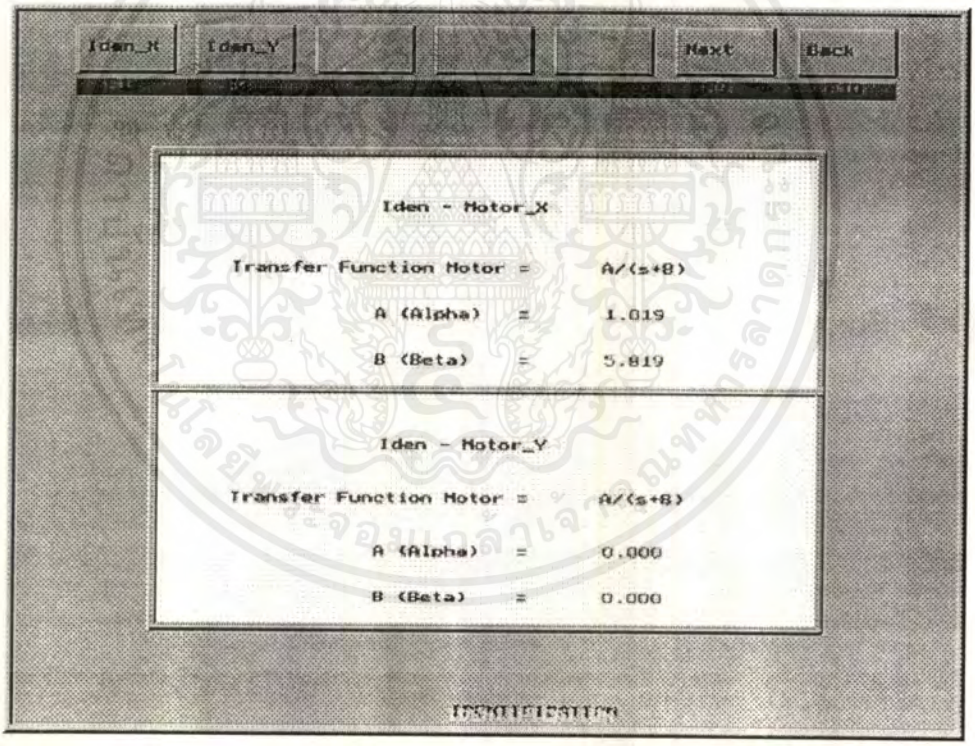


รูปที่ 28 แสดง flowchart ของหน้าจอ auto

Tuning

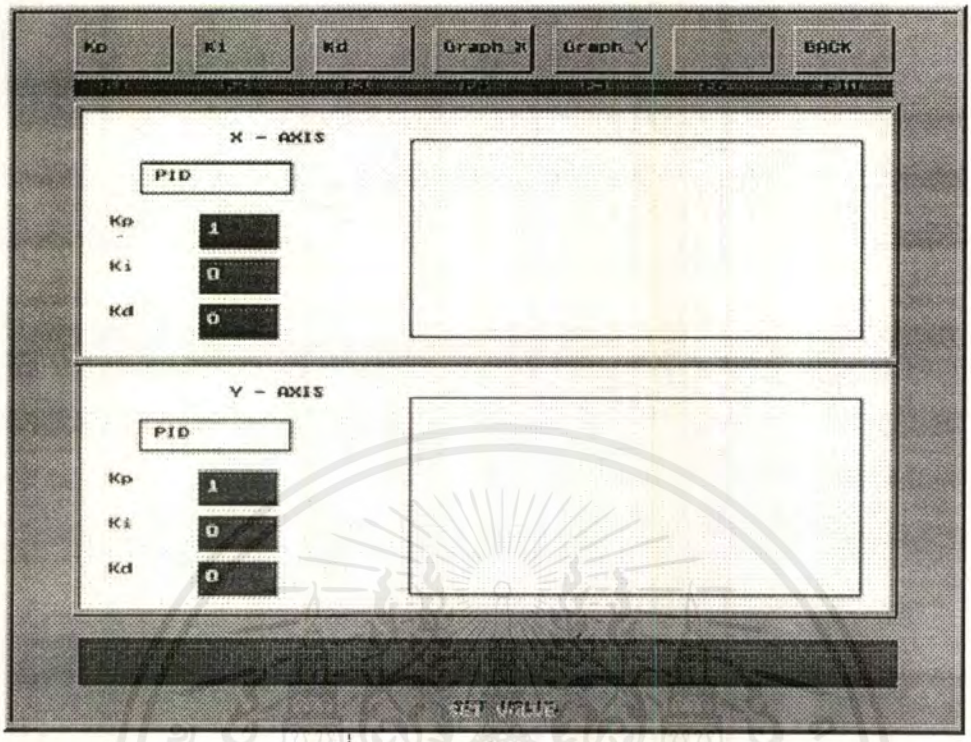
Tuning เป็นส่วนที่มุ่งเน้นให้ผู้ใช้ สามารถหาค่าเกน (Gain) ของตัวควบคุมแบบ PID ได้ง่ายขึ้น ประกอบด้วยความสามารถ 2 ส่วน

1. ส่วน Identification ภายในประกอบด้วย อัลกอริทึม ที่ใช้ในการประมาณค่าพารามิเตอร์ของฟังก์ชันถ่ายโอนของมอเตอร์ จากนั้นจึงนำค่าพารามิเตอร์ที่ได้ ไปหาค่าเกนที่เหมาะสมด้วยโปรแกรมอื่น เช่น MATLAB เป็นต้น ซึ่งเรียกวิธีการหาเกนแบบนี้ว่า "การหาเกนแบบ off line" รูปที่ 29 แสดงค่าฟังก์ชันถ่ายโอนที่หาได้ เมื่อกดปุ่ม Iden_X

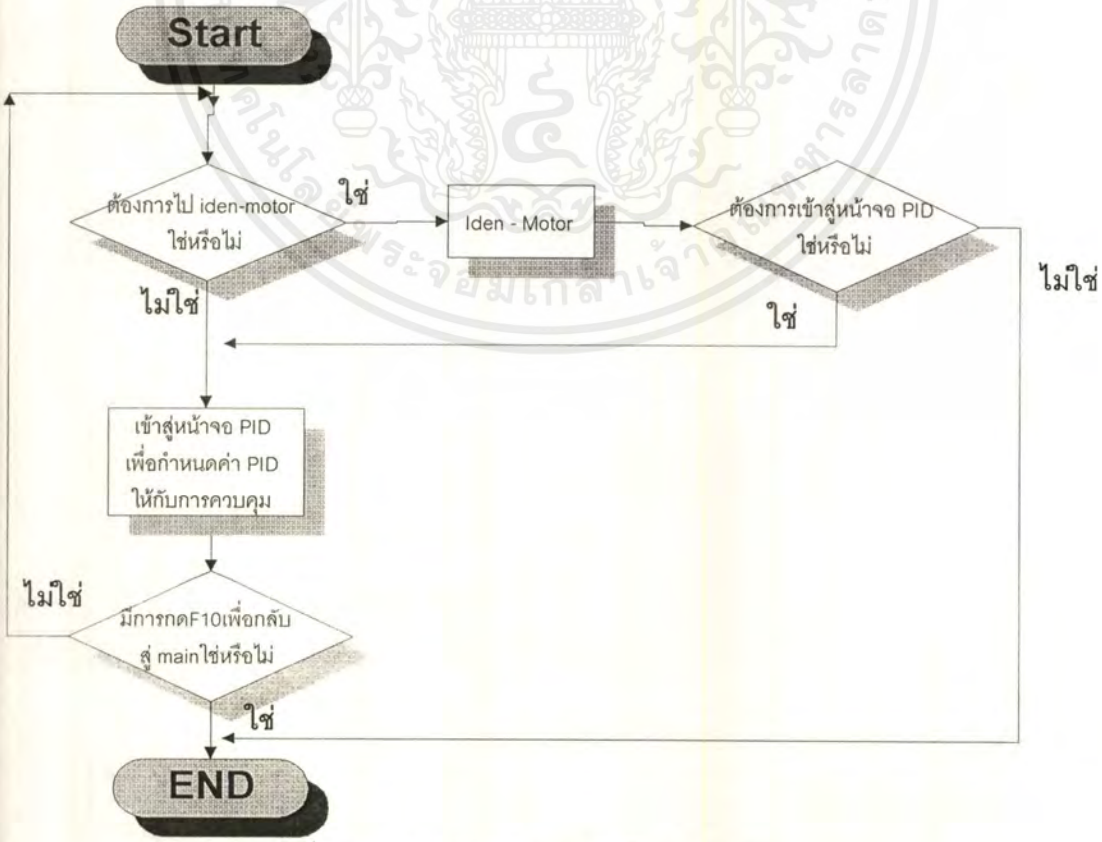


รูปที่ 29 แสดงหน้าจอ Identification

2. ส่วน PID Control เป็นส่วนที่กำหนดค่า PID ให้กับการควบคุมแสดงดังรูปที่ 30



รูปที่ 30 แสดงหน้าจอ PID



รูปที่ 31 แสดง flowchart ของหน้าจอ tuning

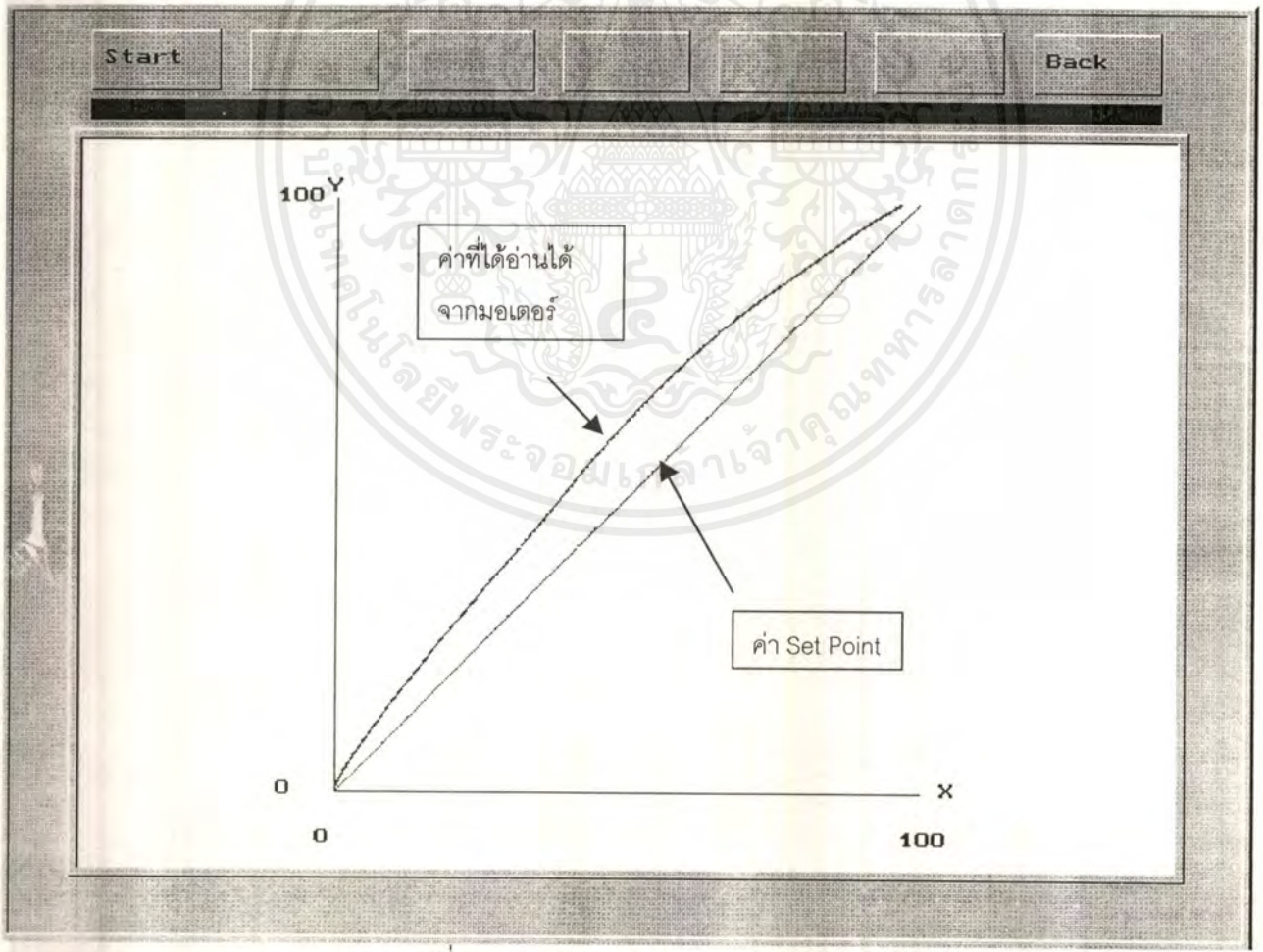
บทที่ 5
การทดลอง

ในการทดลองจะทำการทดลองโดยเคลื่อนที่เป็นเส้นตรง และเคลื่อนที่เป็นวงกลม โดยจากรูปเส้นที่อ่านได้จากมอเตอร์นั้นได้มาจากการต่อกับมอเตอร์จริงเพียง 1 แกน คือแกน X แล้วทำการอ่านค่าตำแหน่งจริงของแกน X จากเอ็นโคดเดอร์กลับมา ส่วนแกน Y นั้นเราจะอ่านค่า Design Position ที่คำนวณได้จากการ์ด LM628 กลับมา เพื่อเปรียบเทียบดูการเคลื่อนที่ที่ได้ กับเส้น set point ซึ่งได้จากการอ่านค่า Design Position ทั้งแกน X และแกน Y

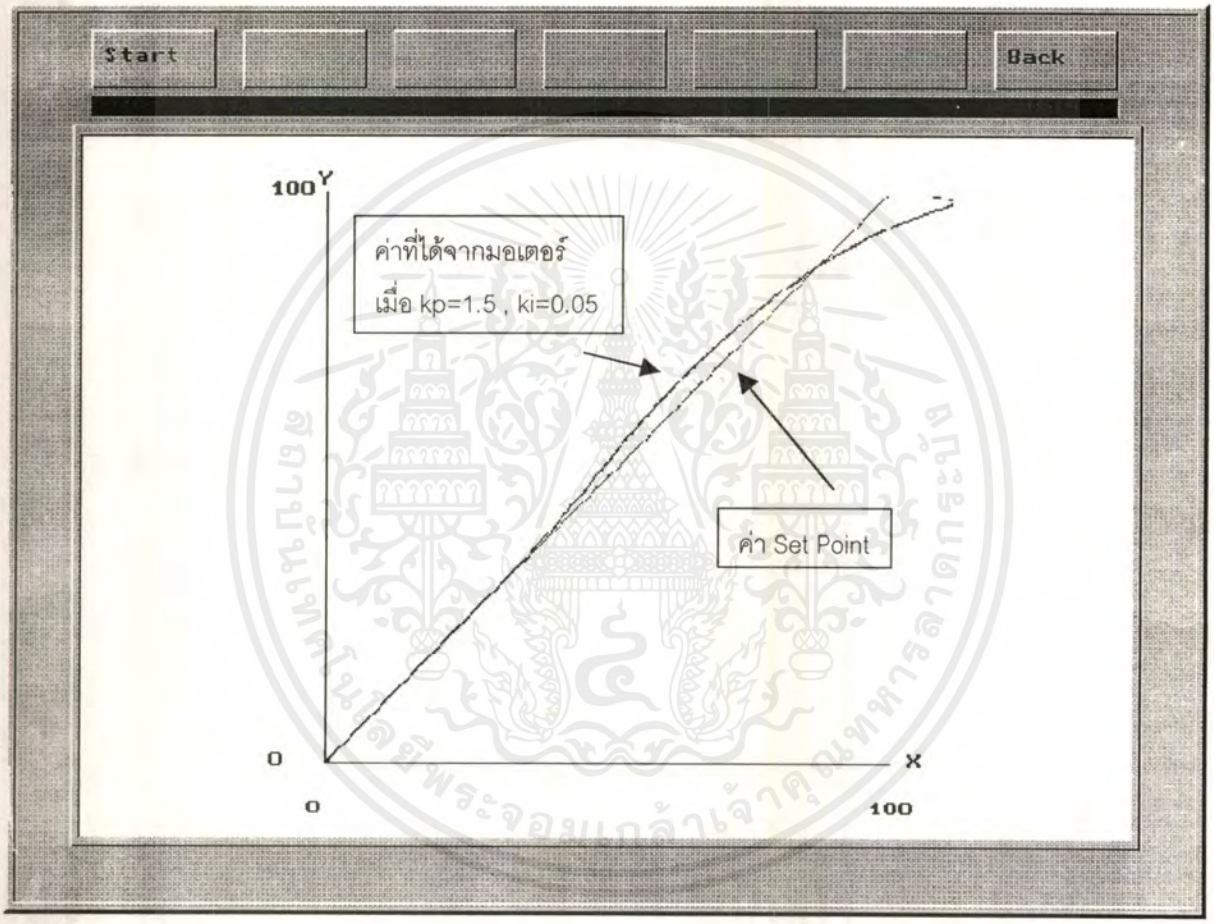
โดยมอเตอร์ที่ใช้ในการทดลองนั้น เป็น DC Motor 180 โวลท์,กระแส 1.15 Amps, 42 RPM , ทอร์ค 30.5 นิวตัน – เมตร และเอ็นโคดเดอร์ที่ใช้มี resolution 2500 P/R

1. การเคลื่อนที่เป็นเส้นตรง ตาม G code ต่อไปนี้

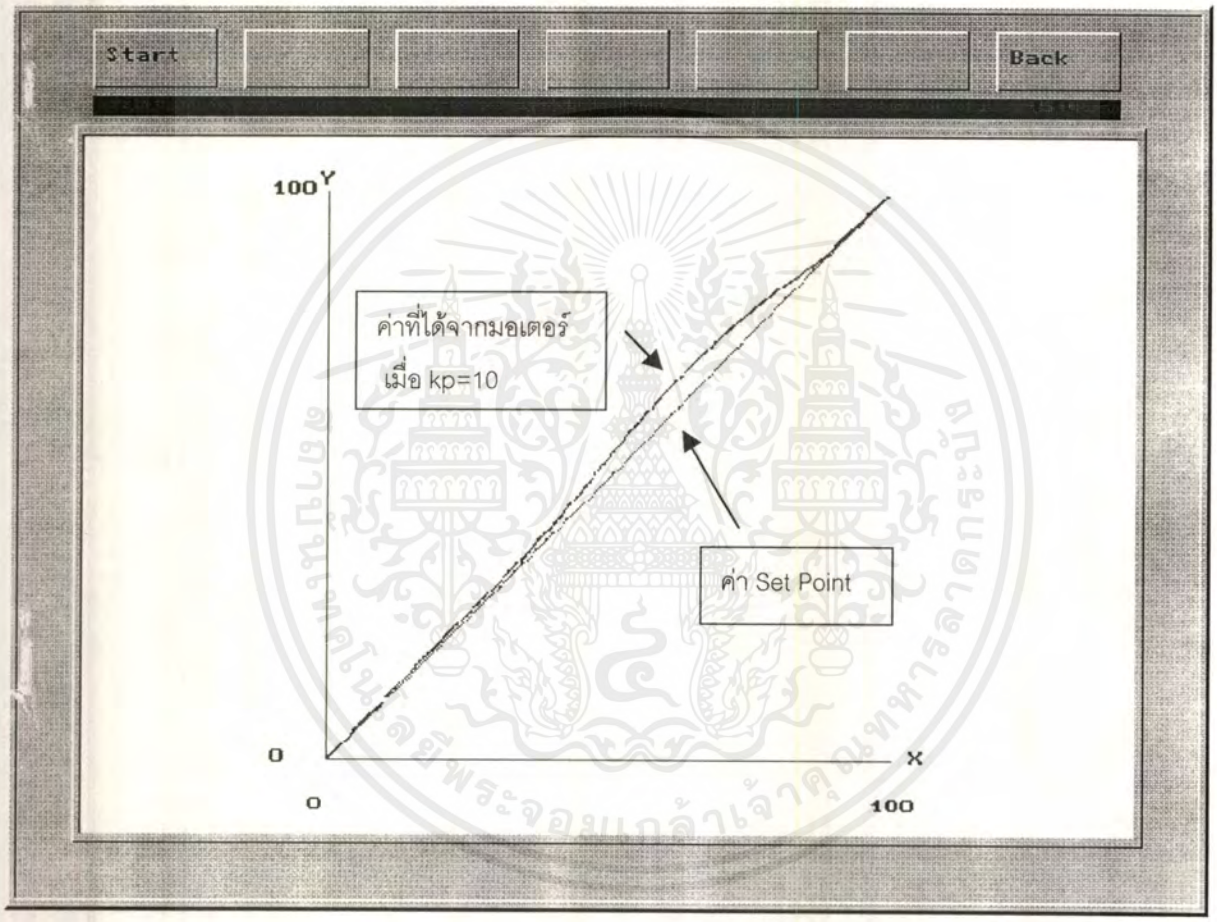
N10 G00 X100 Y100 Z0



รูปที่ 32 Kp=1 Offset=2.88



รูปที่ 33 Kp= 1.5 Ki=0.05



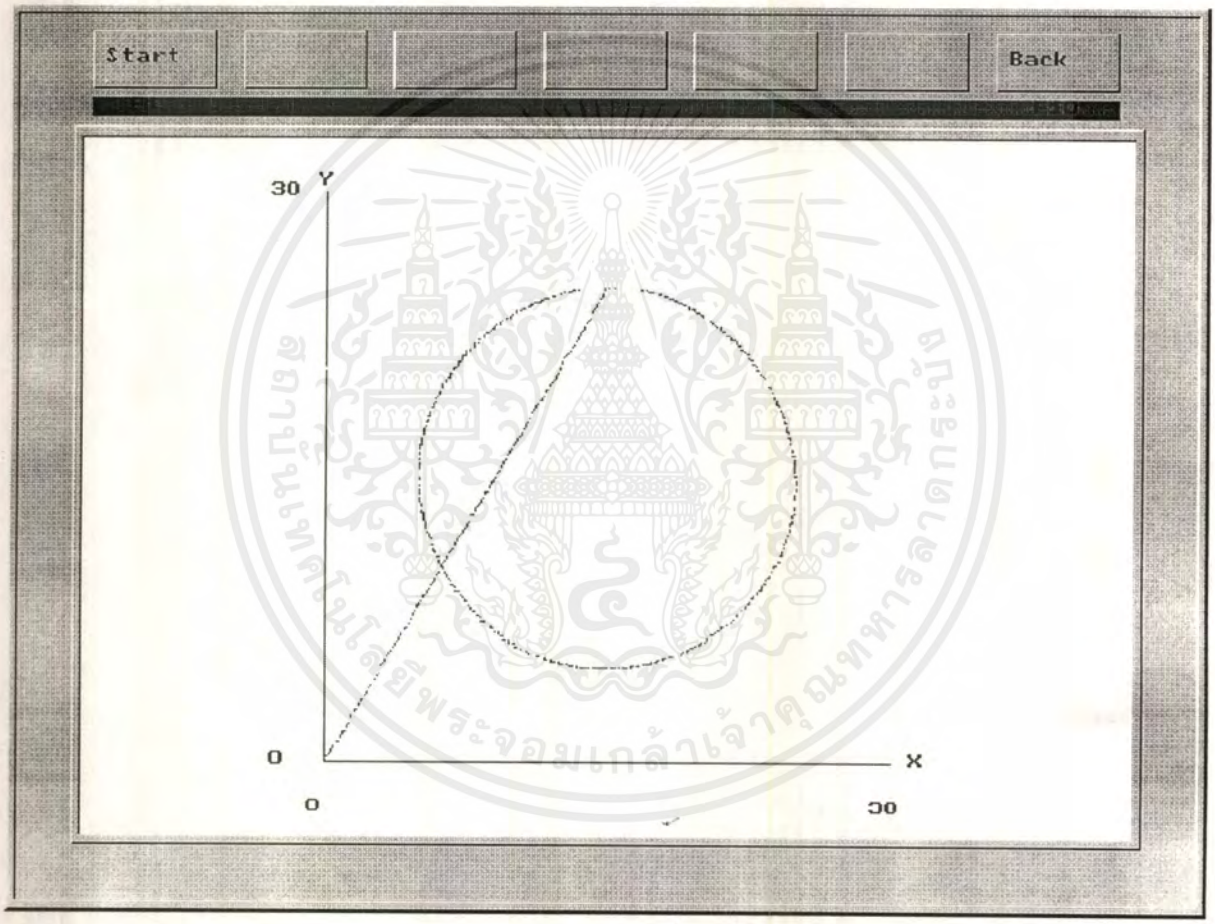
รูปที่ 34 Kp= 10 Offset=0.61

2 การเคลื่อนที่เป็นวงกลม ตาม G code ต่อไปนี้

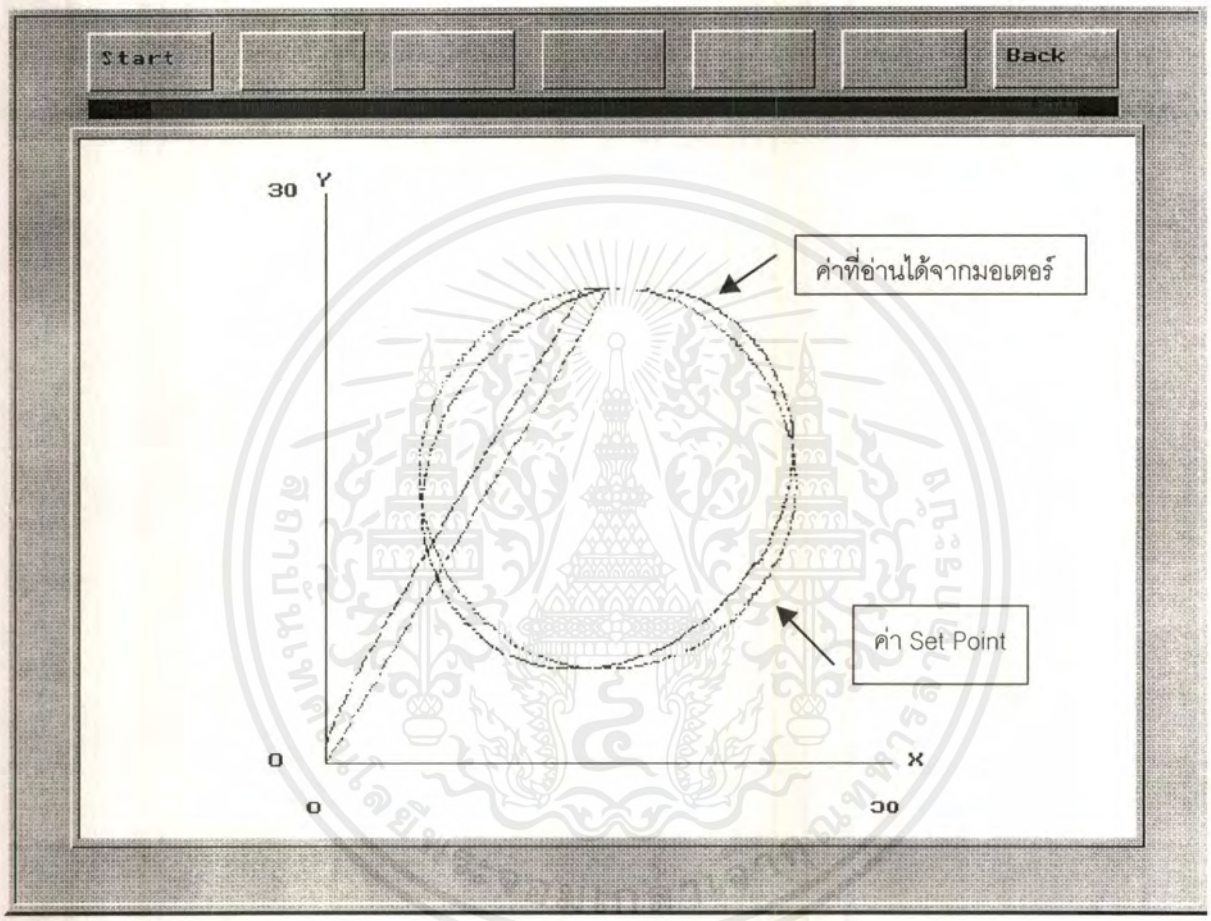
N10 G00 X15 Y25 Z0

N20 G03 X15 Y5 Z0 I0 J-10 K0 F20

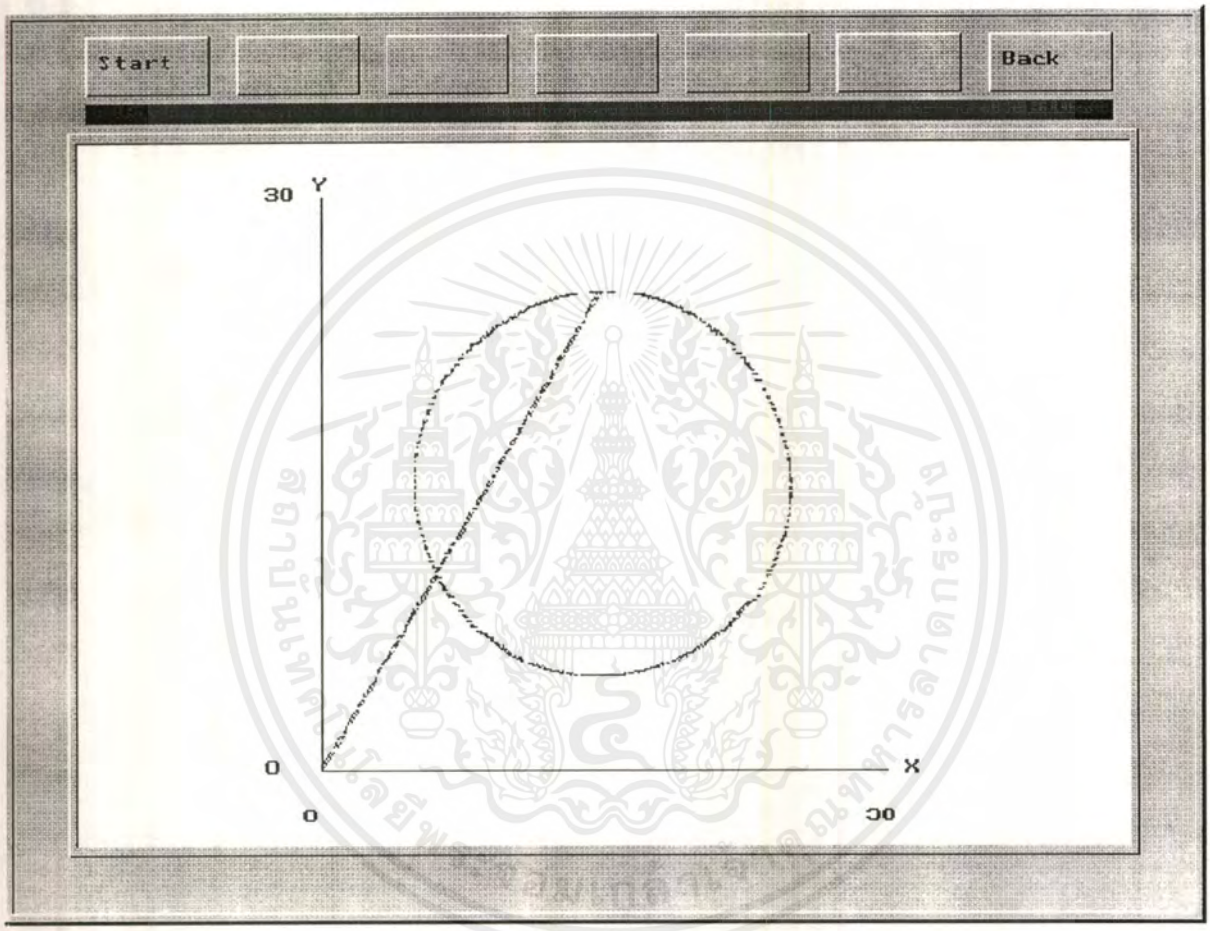
N30 G03 X15 Y25 Z0 I0 J10 K0 F20



รูปที่ 35 สร้างจาก Desire Position



รูปที่ 36 Desire Position และ Real Position Kp=1



รูปที่ 37 Desire Position และ Real Position Kp=10

บทที่ 6
สรุปผลการทดลองและวิจารณ์

สรุป

เราจะนำกราฟที่ได้จากการต่อกับมอเตอร์จริงโดยแกน X อ่านค่ามาจากตำแหน่งของมอเตอร์จริง ส่วนแกน Y นั้นอ่านค่า Design Position มาจากการ์ด LM628 มาเปรียบเทียบกับกราฟ Set Point ที่อ่านค่า Design Position มาจากการ์ด X และการ์ด Y

1. การเคลื่อนที่เป็นเส้นตรง

- จากผลการทดลองรูปที่ 32 จะพบว่าเส้นกราฟที่ได้จากการต่อกับมอเตอร์จริงนั้นจะมี error เกิดขึ้นตลอดทางและมี error เกิดขึ้นที่ตำแหน่งปลายทางอีกด้วยจากค่าที่อ่านมาได้จะพบว่ามี error ที่จุดปลาย 2.88 มิลลิเมตร
- จากผลการทดลองรูปที่ 33 จะชดเชยรูปที่ 32 โดยใช้ PI Controller โดยป้อนค่า Kp = 1.5 , Ki = 0.05 จากกราฟจะพบว่าค่า error ระหว่างทางลดลงแต่จะเกิด overshoot ขึ้นในแกน X
- จากผลการทดลองรูปที่ 34 จะชดเชยรูปที่ 32 โดยใช้ P Controller โดยป้อนค่า Kp = 10 จากกราฟจะพบว่าค่า error ระหว่างทางและปลายทางลดลง รวมทั้ง errorที่จุดปลายก็ลดลงด้วย โดยมีค่า error = 0.61

2. การเคลื่อนที่เป็นวงกลม

- จากผลการทดลองรูปที่ 35 เป็นการเคลื่อนที่เป็นเส้นตรงของ Set Point คือทั้งแกน X และแกน Y อ่านค่ามาจากการ์ด LM628 ทั้งคู่ จากกราฟจะพบว่าการเคลื่อนที่ที่ได้เป็นวงกลมจริงๆ
- จากผลการทดลองรูปที่ 36 เป็นการเปรียบเทียบการเคลื่อนที่ที่อ่านค่ามาจากมอเตอร์จริงกับค่า Set Point จะเห็นว่าค่าที่อ่านมาจากมอเตอร์จริงนั้นจะไม่เป็นวงกลมแต่จะมีการเคลื่อนที่เป็นวงรี
- จากผลการทดลองรูปที่ 37 เป็นการชดเชยการเคลื่อนที่ของรูปที่ 36 โดยใช้ P Controller ซึ่งป้อนค่าเกน Kp=10 จากกราฟจะเห็นว่าการเคลื่อนที่อ่านมาจากมอเตอร์จริงสามารถเป็นเส้นตรงได้ เหมือนกับกราฟของ Set Point

วิจารณ์

ในการต่อมอเตอร์จริงในการเคลื่อนที่นั้น ค่าการเคลื่อนที่ที่ได้จะไม่เป็นไปตาม Set Point เพราะมอเตอร์มีปัจจัยทางกายภาพที่ทำให้มอเตอร์ไม่สามารถทำงานได้เหมือนกับมอเตอร์ในอุดมคติ นั่นคือเมื่อมีการเปลี่ยนความเร็วในการเคลื่อนที่นั้น มอเตอร์จะยังมีความหน่วงอยู่จึงไม่สามารถเปลี่ยนแปลงความเร็วได้โดยทันที จึงเป็นเหตุให้เกิด error และเมื่อมีการกลับทิศมอเตอร์นั้น มอเตอร์ก็ไม่สามารถกลับทิศการเคลื่อนที่ได้ทันทีจึงทำให้เกิด error อีกเช่นเดียวกัน ด้วยเหตุนี้จึงต้องใช้ Tuning เข้ามาช่วยชดเชยการเคลื่อนที่ของมอเตอร์ให้เป็นไปตามเซ็ตพอย

บทที่7
สรุปปริญญานิพนธ์และข้อเสนอแนะ

ปริญญานิพนธ์ฉบับนี้เป็นการพัฒนาระบบเชื่อมต่อกับผู้ใช้ สำหรับการควบคุมแบบหลายแกน บนแนวคิดของระบบเปิด ซึ่งมุ่งเน้นความนิยมแพร่หลายและความเป็นมาตรฐานสากลเป็นหลัก โดยประโยชน์จากแนวคิดนี้ ได้แก่ ความประหยัดเนื่องจากการเลือกซื้อวัสดุอุปกรณ์ที่เหมาะสมกับขนาดของงาน และ ยังง่ายต่อการพัฒนาต่อไปในอนาคต สำหรับระบบเชื่อมต่อกับผู้ใช้ในปริญญานิพนธ์ฉบับนี้จะประกอบด้วยส่วนต่าง ๆ ที่ช่วยให้ผู้ใช้ สามารถทำงานได้ง่ายยิ่งขึ้น ได้แก่ Tuning ,Simulation ,GUI เป็นต้น ตลอดจนเมื่อต่อกับอุปกรณ์ต่าง ๆ เช่น DC Servo Motor ,Driver ,Encoder ,Card Motion Control ,Card 8255 เพื่อการทำงานแบบ Motion Control และ On/Off Control โดยผลของการทำงาน และสถานะต่าง ๆ จากภายนอกสามารถแสดงและควบคุมจาก PC Computer ซึ่งทำให้สะดวกต่อการใช้งาน

ระบบเชื่อมต่อกับผู้ใช้นี้เมื่อพัฒนาแล้วจะสามารถประยุกต์เป็นชุดทดลอง Servo Motor , ชุดทดลองเกี่ยวกับ เครื่องจักรกล CNC ได้ ทำให้สามารถพัฒนาบุคลากรในสาขานี้ได้ และในอนาคตเมื่อเราสามารถพัฒนาจนมีประสิทธิภาพสูงพอ ก็จะเป็นการลดค่าใช้จ่ายของประเทศลง และก่อให้เกิดประโยชน์ต่อภาคอุตสาหกรรมในประเทศด้วย

ข้อเสนอแนะ

1. ระบบเชื่อมต่อกับผู้ใช้นี้ เขียนโดยภาษา C ทำให้การใช้งานผ่าน GUI (Graphic User Interface) ขาดความสะดวกในการใช้งาน ดังนั้นในการพัฒนาขั้นต่อไป ควรเขียนโดยโปรแกรมสำหรับเขียน Windows เช่น Visaul C++ ในการเขียน

2. ในส่วนของ Tuning ควรพัฒนาให้สามารถหาเกน PID ได้แบบ Auto Tuning และแสดงค่าเกน เพื่อเป็นการแนะนำให้แก่ผู้ใช้

3. ในการทดลองมักมีสัญญาณรบกวนเข้ามาทำให้ค่า Count ของ Encoder ที่วัดมาผิดพลาดดังนั้นควรหาทางป้องกันในส่วนนี้

เอกสารอ้างอิง

1. Frank Nanfara, Tony Uccello, Derek Murphy, " The CNC Workbook" , ADDISON-WESLEY PUBLISHING COMPANY
2. Stuart Bennett, "Real-Time Computer Control An Introduction" , Prentice-Hall, 1988
3. Mark Nelson, " Serial Communication : A C++ Developer's Guide " ,Prentic-Hall,
3. ชุดสื่อการเรียนการสอน (IMP) งานกัด CNC,สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ
4. ธันวา ศรีประโมง, "การเขียนโปรแกรมภาษา C สำหรับวิศวกรรม" , มหาวิทยาลัยมหานคร
5. เขียนโปรแกรมกราฟิกและเกมด้วยภาษา C ,COMPUTER AGE TECHNOLOGY CO. LTD.

กิตติกรรมประกาศ

ขอขอบพระคุณ ดร.สุธี ผู้เจริญชนะชัย และอาจารย์เทพจิตร์ เชยโภคา ที่ให้คำปรึกษาและสนับสนุนในด้านต่าง ๆ ตลอดจนช่วยจัดหาอุปกรณ์ที่ใช้ในการทำปริญญานิพนธ์นี้

ขอขอบคุณ พี่อดิศักดิ์ แข็งสารีกิจ, พี่วิโรจน์ แสงธงทอง, พี่สัมฤทธิ์ เตชะวงศ์ธรรม ที่คอยให้คำแนะนำ และคอยช่วยเหลืออยู่เสมอมา จนทำให้ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี

ขอขอบคุณ เพื่อน ๆ CT-LAB ทุกคนที่ให้กำลังใจและความคิดเห็นต่าง ๆ เพื่อเป็นแนวทางในการทำปริญญานิพนธ์ฉบับนี้

ภาคผนวก

National Semiconductor

February 1995

LM628/LM629 Precision Motion Controller

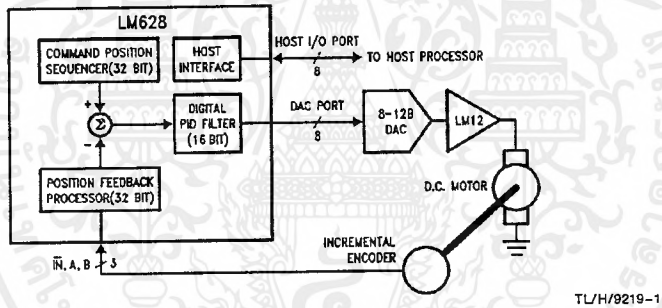
General Description

The LM628/LM629 are dedicated motion-control processors designed for use with a variety of DC and brushless DC servo motors, and other servomechanisms which provide a quadrature incremental position feedback signal. The parts perform the intensive, real-time computational tasks required for high performance digital motion control. The host control software interface is facilitated by a high-level command set. The LM628 has an 8-bit output which can drive either an 8-bit or a 12-bit DAC. The components required to build a servo system are reduced to the DC motor/actuator, an incremental encoder, a DAC, a power amplifier, and the LM628. An LM629-based system is similar, except that it provides an 8-bit PWM output for directly driving H-switches. The parts are fabricated in NMOS and packaged in a 28-pin dual in-line package or a 24-pin surface mount package (LM629 only). Both 6 MHz and 8 MHz maximum frequency versions are available with the suffixes -6 and -8, respectively, used to designate the versions. They incorporate an SDA core processor and cells designed by SDA.

Features

- 32-bit position, velocity, and acceleration registers
- Programmable digital PID filter with 16-bit coefficients
- Programmable derivative sampling interval
- 8- or 12-bit DAC output data (LM628)
- 8-bit sign-magnitude PWM output data (LM629)
- Internal trapezoidal velocity profile generator
- Velocity, target position, and filter parameters may be changed during motion
- Position and velocity modes of operation
- Real-time programmable host interrupts
- 8-bit parallel asynchronous host interface
- Quadrature incremental encoder interface with index pulse input
- Available in a 28-pin dual in-line package or a 24-pin surface mount package (LM629 only)

TRI-STATE® is a registered trademark of National Semiconductor Corporation.

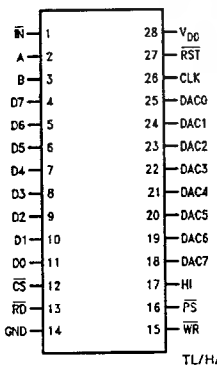


TL/H/9219–1

FIGURE 1. Typical System Block Diagram

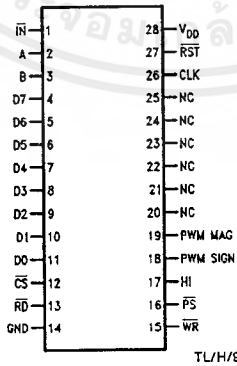
Connection Diagrams

LM628N



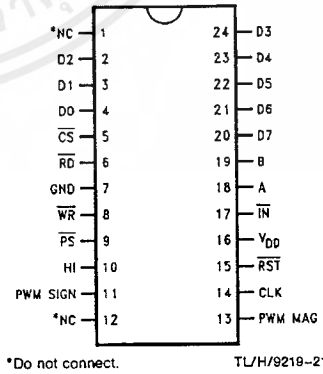
TL/H/9219–2

LM629N



TL/H/9219–3

LM629M



*Do not connect.

TL/H/9219–21

Order Number LM629M-6, LM629M-8, LM628N-6, LM628N-8, LM629N-6 or LM629N-8
See NS Package Number M24B or N28B

RRD-B30M115/Printed in U. S. A.

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin with Respect to GND	−0.3V to +7.0V
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature	
28-pin Dual In-Line Package (Soldering, 4 sec.)	260°C
24-pin Surface Mount Package (Soldering, 10 sec.)	300°C
Maximum Power Dissipation ($T_A \leq 85°C$, Note 2)	605 mW
ESD Tolerance (C_{ZAP} = 120 pF, R_{ZAP} = 1.5k)	2000V

Operating Ratings

Temperature Range	$-40°C < T_A < +85°C$
Clock Frequency:	
LM628N-6, LM629N-6, LM629M-6	$1.0\ MHz < f_{CLK} < 6.0\ MHz$
LM628N-8, LM629N-8, LM629M-8	$1.0\ MHz < f_{CLK} < 8.0\ MHz$
V_{DD} Range	$4.5V < V_{DD} < 5.5V$

DC Electrical Characteristics (V_{DD} and T_A per Operating Ratings; f_{CLK} = 6 MHz)

Symbol	Parameter	Conditions	Tested Limits		Units
			Min	Max	
I_{DD}	Supply Current	Outputs Open		110	mA
INPUT VOLTAGES					
V_{IH}	Logic 1 Input Voltage		2.0		V
V_{IL}	Logic 0 Input Voltage			0.8	V
I_{IN}	Input Currents	$0 \leq V_{IN} \leq V_{DD}$	−10	10	μA
OUTPUT VOLTAGES					
V_{OH}	Logic 1	I_{OH} = −1.6 mA	2.4		V
V_{OL}	Logic 0	I_{OL} = 1.6 mA		0.4	V
I_{OUT}	TRI-STATE® Output Leakage Current	$0 \leq V_{OUT} \leq V_{DD}$	−10	10	μA

AC Electrical Characteristics

(V_{DD} and T_A per Operating Ratings; f_{CLK} = 6 MHz; C_{LOAD} = 50 pF; Input Test Signal $t_r = t_f$ = 10 ns)

Timing Interval	Tⓝ	Tested Limits		Units
		Min	Max	
ENCODER AND INDEX TIMING (See *Figure 2*)				
Motor-Phase Pulse Width	T1	$\frac{16}{f_{CLK}}$		μs
Dwell-Time per State	T2	$\frac{8}{f_{CLK}}$		μs
Index Pulse Setup and Hold (Relative to A and B Low)	T3	0		μs
CLOCK AND RESET TIMING (See *Figure 3*)				
Clock Pulse Width				
LM628N-6, LM629N-6, LM629M-6	T4	78		ns
LM628N-8, LM629N-8, LM629M-8	T4	57		ns
Clock Period				
LM628N-6, LM629N-6, LM629M-6	T5	166		ns
LM628N-8, LM629N-8, LM629M-8	T5	125		ns
Reset Pulse Width	T6	$\frac{8}{f_{CLK}}$		μs

AC Electrical Characteristics (Continued)

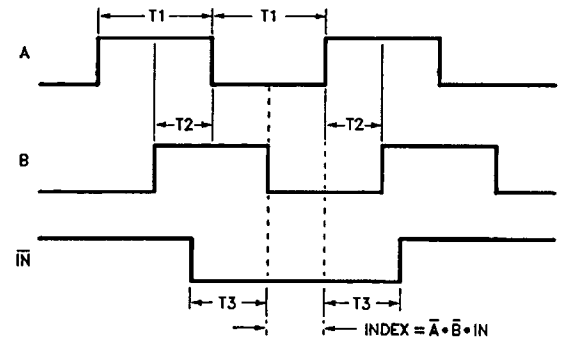
(V_{DD} and T_A per Operating Ratings; f_{CLK} = 6 MHz; C_{LOAD} = 50 pF; Input Test Signal $t_r = t_f$ = 10 ns)

Timing Interval	T#	Tested Limits		Units
		Min	Max	
STATUS BYTE READ TIMING (See *Figure 4*)				
Chip-Select Setup/Hold Time	T7	0		ns
Port-Select Setup Time	T8	30		ns
Port-Select Hold Time	T9	30		ns
Read Data Access Time	T10		180	ns
Read Data Hold Time	T11	0		ns
$\overline{RD}$ High to Hi-Z Time	T12		180	ns
COMMAND BYTE WRITE TIMING (See *Figure 5*)				
Chip-Select Setup/Hold Time	T7	0		ns
Port-Select Setup Time	T8	30		ns
Port-Select Hold Time	T9	30		ns
Busy Bit Delay	T13		(Note 3)	ns
$\overline{WR}$ Pulse Width	T14	100		ns
Write Data Setup Time	T15	50		ns
Write Data Hold Time	T16	120		ns
DATA WORD READ TIMING (See *Figure 6*)				
Chip-Select Setup/Hold Time	T7	0		ns
Port-Select Setup Time	T8	30		ns
Port-Select Hold Time	T9	30		ns
Read Data Access Time	T10		180	ns
Read Data Hold Time	T11	0		ns
$\overline{RD}$ High to Hi-Z Time	T12		180	ns
Busy Bit Delay	T13		(Note 3)	ns
Read Recovery Time	T17	120		ns
DATA WORD WRITE TIMING (See *Figure 7*)				
Chip-Select Setup/Hold Time	T7	0		ns
Port-Select Setup Time	T8	30		ns
Port-Select Hold Time	T9	30		ns
Busy Bit Delay	T13		(Note 3)	ns
$\overline{WR}$ Pulse Width	T14	100		ns
Write Data Setup Time	T15	50		ns
Write Data Hold Time	T16	120		ns
Write Recovery Time	T18	120		ns

Note 1: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond the above Operating Ratings.

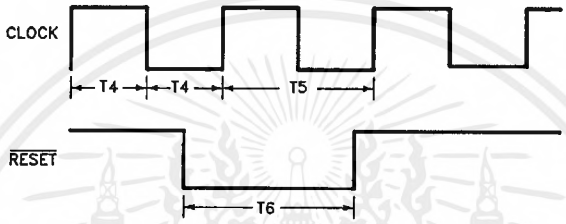
Note 2: When operating at ambient temperatures above 70°C, the device must be protected against excessive junction temperatures. Mounting the package on a printed circuit board having an area greater than three square inches and surrounding the leads and body with wide copper traces and large, uninterrupted areas of copper, such as a ground plane, suffices. The 28-pin DIP (N) and the 24-pin surface mount package (M) are molded plastic packages with solid copper lead frames. Most of the heat generated at the die flows from the die, through the copper lead frame, and into copper traces on the printed circuit board. The copper traces act as a heat sink. Double-sided or multi-layer boards provide heat transfer characteristics superior to those of single-sided boards.

Note 3: In order to read the busy bit, the status byte must first be read. The time required to read the busy bit far exceeds the time the chip requires to set the busy bit. It is, therefore, impossible to test actual busy bit delay. The busy bit is guaranteed to be valid as soon as the user is able to read it.



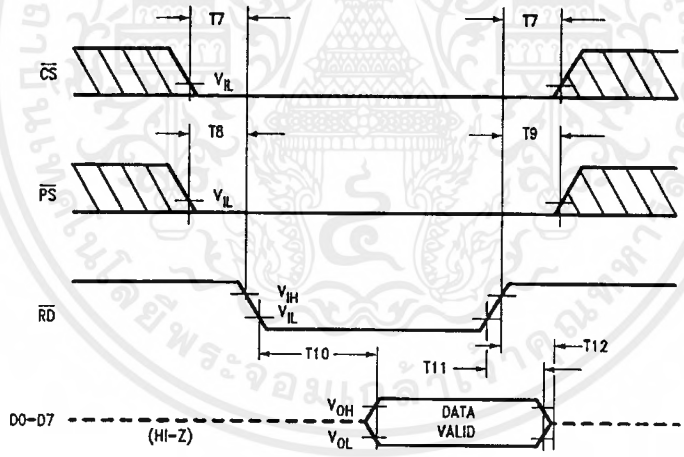
TL/H/9219–4

FIGURE 2. Quadrature Encoder Input Timing



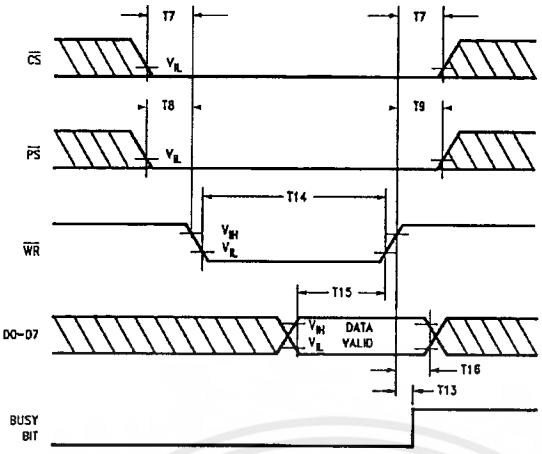
TL/H/9219–5

FIGURE 3. Clock and Reset Timing



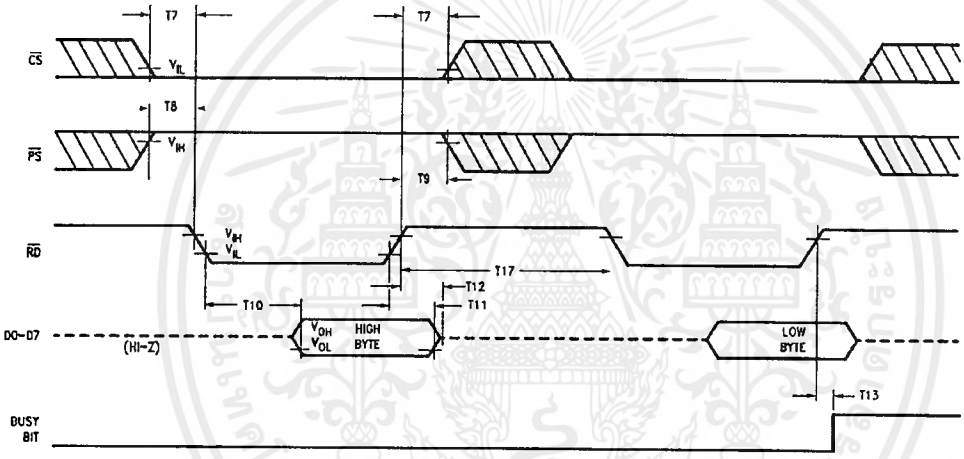
TL/H/9219–6

FIGURE 4. Status Byte Read Timing



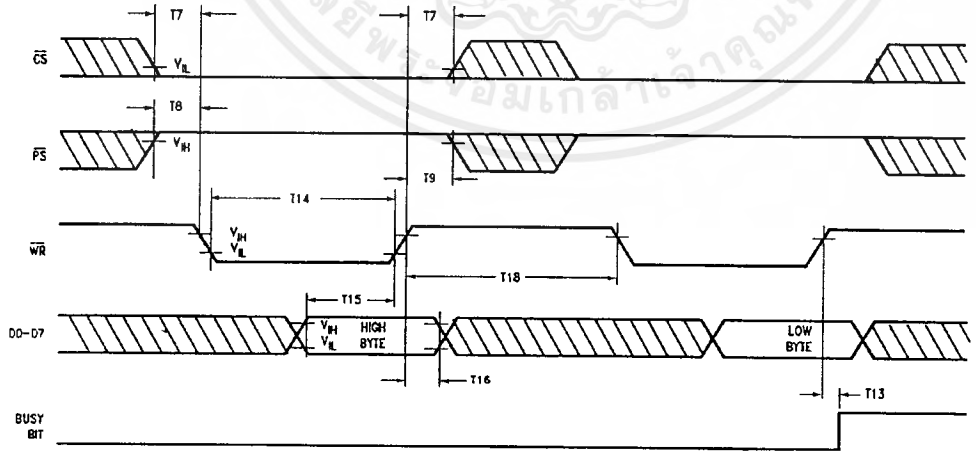
TL/H/9219–7

FIGURE 5. Command Byte Write Timing



TL/H/9219–8

FIGURE 6. Data Word Read Timing



TL/H/9219–9

FIGURE 7. Data Word Write Timing

Pinout Description

(See Connection Diagrams) Pin numbers for the 24-pin surface mount package are indicated in parentheses.

Pin 1 (17), Index ($\overline{IN}$) Input: Receives optional index pulse from the encoder. Must be tied high if not used. The index position is read when Pins 1, 2, and 3 are low.

Pins 2 and 3 (18 and 19), Encoder Signal (A, B) Inputs: Receive the two-phase quadrature signals provided by the incremental encoder. When the motor is rotating in the positive ("forward") direction, the signal at Pin 2 leads the signal at Pin 3 by 90 degrees. Note that the signals at Pins 2 and 3 must remain at each encoder state (See *Figure 9*) for a minimum of 8 clock periods in order to be recognized. Because of a four-to-one resolution advantage gained by the method of decoding the quadrature encoder signals, this corresponds to a maximum encoder-state capture rate of 1.0 MHz (f_{CLK} = 8.0 MHz) or 750 kHz (f_{CLK} = 6.0 MHz). For other clock frequencies the encoder signals must also remain at each state a minimum of 8 clock periods.

Pins 4 to 11 (20 to 24 and 2 to 4), Host I/O Port (D0 to D7): Bi-directional data port which connects to host computer/processor. Used for writing commands and data to the LM628, and for reading the status byte and data from the LM628, as controlled by $\overline{CS}$ (Pin 12), $\overline{PS}$ (Pin 16), $\overline{RD}$ (Pin 13), and $\overline{WR}$ (Pin 15).

Pin 12 (5), Chip Select ($\overline{CS}$) Input: Used to select the LM628 for writing and reading operations.

Pin 13 (6), Read ($\overline{RD}$) Input: Used to read status and data.

Pin 14 (7), Ground (GND): Power-supply return pin.

Pin 15 (8), Write ($\overline{WR}$) Input: Used to write commands and data.

Pin 16 (9), Port Select ($\overline{PS}$) Input: Used to select command or data port. Selects command port when low, data port when high. The following modes are controlled by Pin 16:

1. Commands are written to the command port (Pin 16 low),
2. Status byte is read from command port (Pin 16 low), and
3. Data is written and read via the data port (Pin 16 high).

Pin 17 (10), Host Interrupt (HI) Output: This active-high signal alerts the host (via a host interrupt service routine) that an interrupt condition has occurred.

Pins 18 to 25, DAC Port (DAC0 to DAC7): Output port which is used in three different modes:

1. LM628 (8-bit output mode): Outputs latched data to the DAC. The MSB is Pin 18 and the LSB is Pin 25.
2. LM628 (12-bit output mode): Outputs two, multiplexed 6-bit words. The less-significant word is output first. The MSB is on Pin 18 and the LSB is on Pin 23. Pin 24 is used to demultiplex the words; Pin 24 is low for the less-significant word. The positive-going edge of the signal on Pin 25 is used to strobe the output data. *Figure 8* shows the timing of the multiplexed signals.
3. LM629 (sign/magnitude outputs): Outputs a PWM sign signal on Pin 18 (11 for surface mount), and a PWM magnitude signal on Pin 19 (13 for surface mount). Pins 20 to 25 are not used in the LM629. *Figure 11* shows the PWM output signal format.

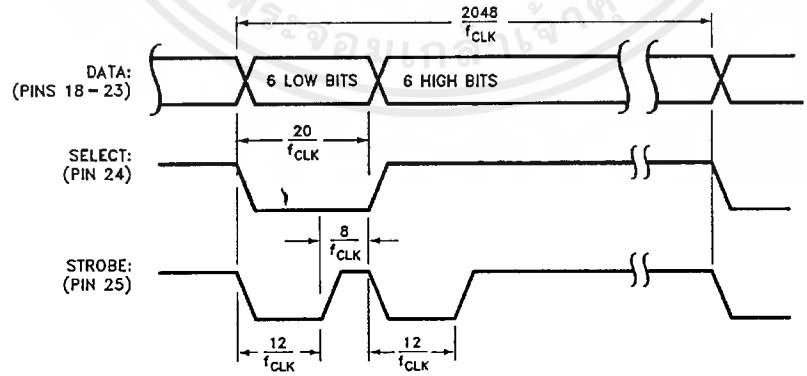
Pin 26 (14), Clock (CLK) Input: Receives system clock.

Pin 27 (15), Reset ($\overline{RST}$) Input: Active-low, positive-edge triggered, resets the LM628 to the internal conditions shown below. Note that the reset pulse must be logic low for a minimum of 8 clock periods. Reset does the following:

1. Filter coefficient and trajectory parameters are zeroed.
2. Sets position error threshold to maximum value (7FFF hex), and effectively executes command LPEI.
3. The SBPA/SBPR interrupt is masked (disabled).
4. The five other interrupts are unmasked (enabled).
5. Initializes current position to zero, or "home" position.
6. Sets derivative sampling interval to $2048/f_{CLK}$ or 256 μs for an 8.0 MHz clock.
7. DAC port outputs 800 hex to "zero" a 12-bit DAC and then reverts to 80 hex to "zero" an 8-bit DAC.

Immediately after releasing the reset pin from the LM628, the status port should read '00'. If the reset is successfully completed, the status word will change to hex '84' or 'C4' within 1.5 ms. If the status word has not changed from hex '00' to '84' or 'C4' within 1.5 ms, perform another reset and repeat the above steps. To be certain that the reset was properly performed, execute a **RSTI** command. If the chip has reset properly, the status byte will change from hex '84' or 'C4' to hex '80' or 'C0'. If this does not occur, perform another reset and repeat the above steps.

Pin 28 (16), Supply Voltage (V_{DD}): Power supply voltage (+5V).



TL/H/9219–10

FIGURE 8. 12-Bit Multiplexed Output Timing

Theory of Operation

INTRODUCTION

The typical system block diagram (See *Figure 1*) illustrates a servo system built using the LM628. The host processor communicates with the LM628 through an I/O port to facilitate programming a trapezoidal velocity profile and a digital compensation filter. The DAC output interfaces to an external digital-to-analog converter to produce the signal that is power amplified and applied to the motor. An incremental encoder provides feedback for closing the position servo loop. The trapezoidal velocity profile generator calculates the required trajectory for either position or velocity mode of operation. In operation, the LM628 subtracts the actual position (feedback position) from the desired position (profile generator position), and the resulting position error is processed by the digital filter to drive the motor to the desired position. Table I provides a brief summary of specifications offered by the LM628/LM629:

POSITION FEEDBACK INTERFACE

The LM628 interfaces to a motor via an incremental encoder. Three inputs are provided: two quadrature signal inputs, and an index pulse input. The quadrature signals are used to keep track of the absolute position of the motor. Each time a logic transition occurs at one of the quadrature inputs, the LM628 internal position register is incremented or decremented accordingly. This provides four times the resolution over the number of lines provided by the encoder. See *Figure 9*. Each of the encoder signal inputs is synchronized with the LM628 clock.

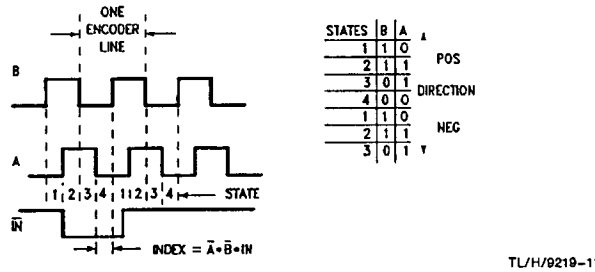
The optional index pulse output provided by some encoders assumes the logic-low state once per revolution. If the LM628 is so programmed by the user, it will record the absolute motor position in a dedicated register (the index register) at the time when all three encoder inputs are logic low.

If the encoder does not provide an index output, the LM628 index input can also be used to record the home position of the motor. In this case, typically, the motor will close a switch which is arranged to cause a logic-low level at the index input, and the LM628 will record motor position in the index register and alert (interrupt) the host processor. Permanently grounding the index input will cause the LM628 to malfunction.

TABLE I. System Specifications Summary

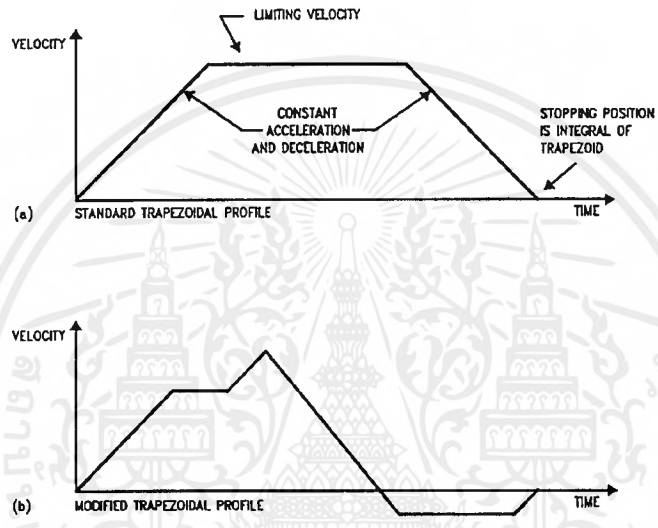
Position Range	−1,073,741,824 to 1,073,741,823 counts
Velocity Range	0 to 1,073,741,823/2^{16} counts/sample; ie, 0 to 16,383 counts/sample, with a resolution of $1/2^{16}$ counts/sample
Acceleration Range	0 to 1,073,741,823/2^{16} counts/sample/sample; ie, 0 to 16,383 counts/sample/sample, with a resolution of $1/2^{16}$ counts/sample/sample
Motor Drive Output	LM628: 8-bit parallel output to DAC, or 12-bit multiplexed output to DAC LM629: 8-bit PWM sign/magnitude signals
Operating Modes	Position and Velocity
Feedback Device	Incremental Encoder (quadrature signals; support for index pulse)
Control Algorithm	Proportional Integral Derivative (PID) (plus programmable integration limit)
Sample Intervals	Derivative Term: Programmable from $2048/f_{CLK}$ to $(2048 * 256)/f_{CLK}$ in steps of $2048/f_{CLK}$ (256 to 65,536 μs for an 8.0 MHz clock). Proportional and Integral: $2048/f_{CLK}$

Theory of Operation (Continued)



TL/H/9219–11

FIGURE 9. Quadrature Encoder Signals



TL/H/9219–12

FIGURE 10. Typical Velocity Profiles

VELOCITY PROFILE (TRAJECTORY) GENERATION

The trapezoidal velocity profile generator computes the desired position of the motor versus time. In the position mode of operation, the host processor specifies acceleration, maximum velocity, and final position. The LM628 uses this information to affect the move by accelerating as specified until the maximum velocity is reached or until deceleration must begin to stop at the specified final position. The deceleration rate is equal to the acceleration rate. At any time during the move the maximum velocity and/or the target position may be changed, and the motor will accelerate or decelerate accordingly. *Figure 10* illustrates two typical trapezoidal velocity profiles. *Figure 10 (a)* shows a simple trapezoid, while *Figure 10 (b)* is an example of what the trajectory looks like when velocity and position are changed at different times during the move.

When operating in the velocity mode, the motor accelerates to the specified velocity at the specified acceleration rate and maintains the specified velocity until commanded to stop. The velocity is maintained by advancing the desired position at a constant rate. If there are disturbances to the motion during velocity mode operation, the long-time average velocity remains constant. If the motor is unable to maintain the specified velocity (which could be caused by a locked rotor, for example), the desired position will continue to be increased, resulting in a very large position error. If this condition goes undetected, and the impeding force on the motor is subsequently released, the motor could reach a very high velocity in order to catch up to the desired position (which is still advancing as specified). This condition is easily detected; see commands LPEI and LPES.

All trajectory parameters are 32-bit values. Position is a signed quantity. Acceleration and velocity are specified as 16-bit, positive-only integers having 16-bit fractions. The integer portion of velocity specifies how many counts per sampling interval the motor will traverse. The fractional portion designates an additional fractional count per sampling interval. Although the position resolution of the LM628 is limited to integer counts, the fractional counts provide increased average velocity resolution. Acceleration is treated in the same manner. Each sampling interval the commanded acceleration value is added to the current desired velocity to generate a new desired velocity (unless the command velocity has been reached).

One determines the trajectory parameters for a desired move as follows. If, for example, one has a 500-line shaft encoder, desires that the motor accelerate at one revolution per second per second until it is moving at 600 rpm, and then decelerate to a stop at a position exactly 100 revolutions from the start, one would calculate the trajectory parameters as follows:

Theory of Operation (Continued)

let P = target position (units = encoder counts)

let R = encoder lines * 4 (system resolution)

then R = 500 * 4 = 2000

and P = 2000 * desired number of revolutions

P = 2000 * 100 revs = 200,000 counts (value to load)

P (coding) = 00030D40 (hex code written to LM628)

let V = velocity (units = counts/sample)

let T = sample time (seconds) = 341 μs (with 6 MHz clock)

let C = conversion factor = 1 minute/60 seconds

then V = R * T * C * desired rpm

and V = 2000 * 341E−6 * 1/60 * 600 rpm

V = 6.82 counts/sample

V (scaled) = 6.82 * 65,536 = 446,955.52

V (rounded) = 446,956 (value to load)

V (coding) = 0006D1EC (hex code written to LM628)

let A = acceleration (units = counts/sample/sample)

A = R * T * T * desired acceleration (rev/sec/sec)

then A = 2000 * 341E−6 * 341E-6 * 1 rev/sec/sec

and A = 2.33E−4 counts/sample/sample

A (scaled) = 2.33E−4 * 65,536 = 15.24

A (rounded) = 15 (value to load)

A (coding) = 0000000F (hex code written to LM628)

The above position, velocity, and acceleration values must be converted to binary codes to be loaded into the LM628. The values shown for velocity and acceleration must be multiplied by 65,536 (as shown) to adjust for the required integer/fraction format of the input data. Note that after scaling the velocity and acceleration values, literal fractional data cannot be loaded; the data must be rounded and converted to binary. The factor of four increase in system resolution is due to the method used to decode the quadrature encoder signals, see *Figure 9*.

PID COMPENSATION FILTER

The LM628 uses a digital Proportional Integral Derivative (PID) filter to compensate the control loop. The motor is held at the desired position by applying a restoring force to the motor that is proportional to the position error, plus the integral of the error, plus the derivative of the error. The following discrete-time equation illustrates the control performed by the LM628:

$$u(n) = kp*e(n) + ki\sum_{N=0}^{n} e(n) + kd[e(n') - e(n'-1)] \qquad \text{(Eq.1)}$$

where u(n) is the motor control signal output at sample time n, e(n) is the position error at sample time n, n' indicates sampling at the derivative sampling rate, and kp, ki, and kd are the discrete-time filter parameters loaded by the users.

The first term, the proportional term, provides a restoring force porportional to the position error, just as does a spring obeying Hooke's law. The second term, the integration term, provides a restoring force that grows with time, and thus ensures that the static position error is zero. If there is a constant torque loading, the motor will still be able to achieve zero position error.

The third term, the derivative term, provides a force proportional to the rate of change of position error. It acts just like viscous damping in a damped spring and mass system (like a shock absorber in an automobile). The sampling interval associated with the derivative term is user-selectable; this capability enables the LM628 to control a wider range of inertial loads (system mechanical time constants) by providing a better approximation of the continuous derivative. In general, longer sampling intervals are useful for low-velocity operations.

In operation, the filter algorithm receives a 16-bit error signal from the loop summing-junction. The error signal is saturated at 16 bits to ensure predictable behavior. In addition to being multiplied by filter coefficient kp, the error signal is added to an accumulation of previous errors (to form the integral signal) and, at a rate determined by the chosen *derivative* sampling interval, the previous error is subtracted from it (to form the derivative signal). All filter multiplications are 16-bit operations; only the bottom 16 bits of the product are used.

The integral signal is maintained to 24 bits, but only the top 16 bits are used. This scaling technique results in a more usable (less sensitive) range of coefficient ki values. The 16 bits are right-shifted eight positions and multiplied by filter coefficient ki to form the term which contributes to the motor control output. The absolute magnitude of this product is compared to coefficient il, and the lesser, appropriately signed magnitude then contributes to the motor control signal.

The derivative signal is multiplied by coefficient kd each *derivative* sampling interval. This product contributes to the motor control output *every* sample interval, independent of the user-chosen *derivative* sampling interval.

The kp, limited ki, and kd product terms are summed to form a 16-bit quantity. Depending on the output mode (wordsize), either the top 8 or top 12 bits become the motor control output signal.

LM628 READING AND WRITING OPERATIONS

The host processor writes commands to the LM628 via the host I/O port when Port Select ($\overline{PS}$) input (Pin 16) is logic low. The desired command code is applied to the parallel port line and the Write ($\overline{WR}$) input (Pin 15) is strobed. The command byte is latched into the LM628 on the rising edge of the $\overline{WR}$ input. When writing command bytes it is necessary to first read the status byte and check the state of a flag called the "busy bit" (Bit 0). If the busy bit is logic high, no command write may take place. The busy bit is never high longer than 100 μs, and typically falls within 15 μs to 25 μs.

The host processor reads the LM628 status byte in a similar manner: by strobing the Read ($\overline{RD}$) input (Pin 13) when $\overline{PS}$ (Pin 16) is low; status information remains valid as long as $\overline{RD}$ is low.

Writing and reading data to/from the LM628 (as opposed to writing commands and reading status) are done with $\overline{PS}$ (Pin 16) logic high. These writes and reads are always an integral number (from one to seven) of two-byte words, with the first byte of each word being the more significant. Each byte requires a write ($\overline{WR}$) or read ($\overline{RD}$) strobe. When transferring data words (byte-pairs), it is necessary to first read the status byte and check the state of the busy bit. When the

Theory of Operation (Continued)

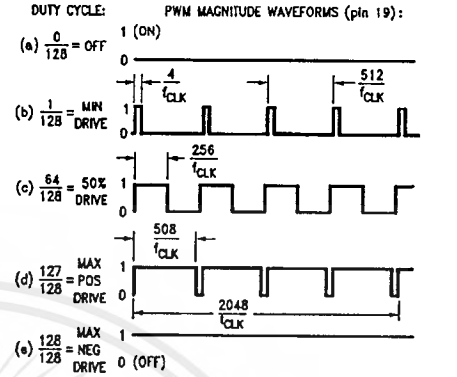
busy bit is logic low, the user may then sequentially transfer both bytes comprising a data word, but the busy bit must again be checked and found to be low before attempting to transfer the next byte pair (when transferring multiple words). Data transfers are accomplished via LM628-internal interrupts (which are not nested); the busy bit informs the host processor when the LM628 may not be interrupted for data transfer (or a command byte). If a command is written when the busy bit is high, the command will be ignored.

The busy bit goes high immediately after writing a command byte, or reading or writing a second byte of data (See *Figures 5* thru *7*).

MOTOR OUTPUTS

The LM628 DAC output port can be configured to provide either a latched eight-bit parallel output or a multiplexed 12-bit output. The 8-bit output can be directly connected to a flow-through (non-input-latching) D/A converter; the 12-bit output can be easily demultiplexed using an external 6-bit latch and an input-latching 12-bit D/A converter. The DAC output data is offset-binary coded; the 8-bit code for zero is 80 hex and the 12-bit code for zero is 800 hex. Values less than these cause a negative torque to be applied to the motor and, conversely, larger values cause positive motor torque. The LM628, when configured for 12-bit output, provides signals which control the demultiplexing process. See *Figure 8* for details.

The LM629 provides 8-bit, sign and magnitude PWM output signals for directly driving switch-mode motor-drive amplifiers. *Figure 11* shows the format of the PWM magnitude output signal.



Note: Sign output (pin 18) not shown

FIGURE 11. PWM Output Signal Format

TABLE II. LM628 User Command Set

Command	Type	Description	Hex	Data Bytes	Note
RESET	Initialize	Reset LM628	00	0	1
PORT8	Initialize	Select 8-Bit Output	05	0	2
PORT12	Initialize	Select 12-Bit Output	06	0	2
DFH	Initialize	Define Home	02	0	1
SIP	Interrupt	Set Index Position	03	0	1
LPEI	Interrupt	Interrupt on Error	1B	2	1
LPES	Interrupt	Stop on Error	1A	2	1
SBPA	Interrupt	Set Breakpoint, Absolute	20	4	1
SBPR	Interrupt	Set Breakpoint, Relative	21	4	1
MSKI	Interrupt	Mask Interrupts	1C	2	1
RSTI	Interrupt	Reset Interrupts	1D	2	1
LFIL	Filter	Load Filter Parameters	1E	2 to 10	1
UDF	Filter	Update Filter	04	0	1
LTRJ	Trajectory	Load Trajectory	1F	2 to 14	1
STT	Trajectory	Start Motion	01	0	3
RDSTAT	Report	Read Status Byte	None	1	1, 4
RDSIGS	Report	Read Signals Register	0C	2	1
RDIP	Report	Read Index Position	09	4	1
RDDP	Report	Read Desired Position	08	4	1
RDRP	Report	Read Real Position	0A	4	1
RDDV	Report	Read Desired Velocity	07	4	1
RDRV	Report	Read Real Velocity	0B	2	1
RDSUM	Report	Read Integration Sum	0D	2	1

Note 1: Commands may be executed "On the Fly" during motion.

Note 2: Commands not applicable to execution during motion.

Note 3: Command may be executed during motion if acceleration parameter was not changed.

Note 4: Command needs no code because the command port status-byte read is totally supported by hardware.

User Command Set

GENERAL

The following paragraphs describe the user command set of the LM628. Some of the commands can be issued alone and some require a supporting data structure. As examples, the command STT (STarT motion) does not require additional data; command LFIL (Load FILter parameters) requires additional data (derivative-term sampling interval and/or filter parameters).

Commands are categorized by function: initialization, interrupt control, filter control, trajectory control, and data reporting. The commands are listed in Table II and described in the following paragraphs. Along with each command name is its command-byte code, the number of accompanying data bytes that are to be written (or read), and a comment as to whether the command is executable during motion.

Initialization Commands

The following four LM628 user commands are used primarily to initialize the system for use.

RESET COMMAND: RESET the LM628

Command Code:	00 Hex
Data Bytes:	None
Executable During Motion:	Yes

This command (and the hardware reset input, Pin 27) results in setting the following data items to zero: filter coefficients and their input buffers, trajectory parameters and their input buffers, and the motor control output. A zero motor control output is a half-scale, offset-binary code: (80 hex for the 8-bit output mode; 800 hex for 12-bit mode). During reset, the DAC port outputs 800 hex to "zero" a 12-bit DAC and reverts to 80 hex to "zero" an 8-bit DAC. The command also clears five of the six interrupt masks (only the SBPA/SBPR interrupt is masked), sets the output port size to 8 bits, and defines the current absolute position as home. Reset, which may be executed at any time, will be completed in less than 1.5 ms. Also see commands PORT8 and PORT12.

PORT8 COMMAND: Set Output PORT Size to 8 Bits

Command Code:	05 Hex
Data Bytes:	None
Executable During Motion:	Not Applicable

The default output port size of the LM628 is 8 bits; so the PORT8 command need not be executed when using an 8-bit DAC. This command must not be executed when using a 12-bit converter; it will result in erratic, unpredictable motor behavior. The 8-bit output port size is the required selection when using the LM629, the PWM-output version of the LM628.

PORT12 COMMAND: Set Output PORT Size to 12 Bits

Command Code:	06 Hex
Data Bytes:	None
Executable During Motion:	Not Applicable

When a 12-bit DAC is used, command PORT12 should be issued very early in the initialization process. Because use of this command is determined by system hardware, there is only one foreseen reason to execute it later: if the RESET command is issued (because an 8-bit output would then be selected as the default) command PORT12 should be immediately executed. This command must not be issued when using an 8-bit converter or the LM629, the PWM-output version of the LM628.

DFH COMMAND: DeFine Home

Command Code:	02 Hex
Data Bytes:	None
Executable During Motion:	Yes

This command declares the current position as "home", or absolute position 0 (Zero). If DFH is executed during motion it will not affect the stopping position of the on-going move unless command STT is also executed.

Interrupt Control Commands

The following seven LM628 user commands are associated with conditions which can be used to interrupt the host computer. In order for any of the potential interrupt conditions to actually interrupt the host via Pin 17, the corresponding bit in the interrupt mask data associated with command MSKI must have been set to logic high (the non-masked state).

The identity of all interrupts is made known to the host via reading and parsing the status byte. Even if all interrupts are masked off via command MSKI, the state of each condition is still reflected in the status byte. This feature facilitates polling the LM628 for status information, as opposed to interrupt driven operation.

SIP COMMAND: Set Index Position

Command Code:	03 Hex
Data Bytes:	None
Executable During Motion:	Yes

After this command is executed, the absolute position which corresponds to the occurrence of the next index pulse input will be recorded in the index register, and bit 3 of the status byte will be set to logic high. The position is recorded when both encoder-phase inputs and the index pulse input are logic low. This register can then be read by the user (see description for command RDIP) to facilitate aligning the definition of home position (see description of command DFH) with an index pulse. The user can also arrange to have the LM628 interrupt the host to signify that an index pulse has occurred. See the descriptions for commands MSKI and RSTI.

LPEI COMMAND: Load Position Error for Interrupt

Command Code:	1B Hex
Data Bytes:	Two
Data Range:	0000 to 7FFF Hex
Executable During Motion:	Yes

An excessive position error (the output of the loop summing junction) can indicate a serious system problem; e.g., a stalled rotor. Instruction LPEI allows the user to input a threshold for position error detection. Error detection occurs when the absolute magnitude of the position error exceeds the threshold, which results in bit 5 of the status byte being set to logic high. If it is desired to also stop (turn off) the motor upon detecting excessive position error, see command LPES, below. The first byte of threshold data written with command LPEI is the more significant. The user can have the LM628 interrupt the host to signify that an excessive position error has occurred. See the descriptions for commands MSKI and RSTI.

Interrupt Control Commands (Continued)

LPES COMMAND: Load Position Error for Stopping

Command Code:	1A Hex
Data Bytes:	Two
Data Range:	0000 to 7FFF Hex
Executable During Motion:	Yes

Instruction LPES is essentially the same as command LPEI above, but adds the feature of turning off the motor upon detecting excessive position error. The motor drive is not actually switched off, it is set to half-scale, the offset-binary code for zero. As with command LPEI, bit 5 of the status byte is also set to logic high. The first byte of threshold data written with command LPES is the more significant. The user can have the LM628 interrupt the host to signify that an excessive position error has occurred. See the descriptions for commands MSKI and RSTI.

SBPA COMMAND:

Command Code:	20 Hex
Data Bytes:	Four
Data Range:	C0000000 to 3FFFFFFF Hex
Executable During Motion:	Yes

This command enables the user to set a breakpoint in terms of absolute position. Bit 6 of the status byte is set to logic high when the breakpoint position is reached. This condition is useful for signaling trajectory and/or filter parameter updates. The user can also arrange to have the LM628 interrupt the host to signify that a breakpoint position has been reached. See the descriptions for commands MSKI and RSTI.

SBPR COMMAND:

Command Code:	21 Hex
Data Bytes:	Four
Data Range:	See Text
Executable During Motion:	Yes

This command enables the user to set a breakpoint in terms of relative position. As with command SBPA, bit 6 of the status byte is set to logic high when the breakpoint position (relative to the current commanded target position) is reached. The relative breakpoint input value must be such that when this value is added to the target position the result remains within the absolute position range of the system (C0000000 to 3FFFFFFF hex). This condition is useful for signaling trajectory and/or filter parameter updates. The user can also arrange to have the LM628 interrupt the host to signify that a breakpoint position has been reached. See the descriptions for commands MSKI and RSTI.

MSKI COMMAND: MaSK Interrupts

Command Code:	1C Hex
Data Bytes:	Two
Data Range:	See Text
Executable During Motion:	Yes

The MSKI command lets the user determine which potential interrupt condition(s) will interrupt the host. Bits 1 through 6 of the status byte are indicators of the six conditions which are candidates for host interrupt(s). When interrupted, the host then reads the status byte to learn which condition(s) occurred. Note that the MSKI command is immediately followed by two data bytes. Bits 1 through 6 of the second (less significant) byte written determine the masked/unmasked status of each potential interrupt. Any zero(s) in this 6-bit field will mask the corresponding interrupt(s); any one(s) enable the interrupt(s). Other bits comprising the two bytes have no effect. The mask controls only the host interrupt process; reading the status byte will still reflect the actual conditions independent of the mask byte. See Table III.

TABLE III. Mask and Reset Bit Allocations for Interrupts

Bit Position	Function
Bits 15 thru 7	Not Used
Bit 6	Breakpoint Interrupt
Bit 5	Position-Error Interrupt
Bit 4	Wrap-Around Interrupt
Bit 3	Index-Pulse Interrupt
Bit 2	Trajectory-Complete Interrupt
Bit 1	Command-Error Interrupt
Bit 0	Not Used

RSTI COMMAND: ReSeT Interrupts

Command Code:	1D Hex
Data Bytes:	Two
Data Range:	See Text
Executable During Motion:	Yes

When one of the potential interrupt conditions of Table III occurs, command RSTI is used to reset the corresponding interrupt flag bit in the status byte. The host may reset one or all flag bits. Resetting them one at a time allows the host to service them one at a time according to a priority programmed by the user. As in the MSKI command, bits 1 through 6 of the second (less significant) byte correspond to the potential interrupt conditions shown in Table III. Also see description of RDSTAT command. Any zero(s) in this 6-bit field reset the corresponding interrupt(s). The remaining bits have no effect.

Filter Control Commands

The following two LM628 user commands are used for setting the derivative-term sampling interval, for adjusting the filter parameters as required to tune the system, and to control the timing of these system changes.

LFIL COMMAND: Load FILter Parameters

Command Code:	1E Hex
Data Bytes:	Two to Ten
Data Ranges . . .	
Filter Control Word:	See Text
Filter Coefficients:	0000 to 7FFF Hex (Pos Only)
Integration Limit:	0000 to 7FFF Hex (Pos Only)
Executable During Motion:	Yes

The filter parameters (coefficients) which are written to the LM628 to control loop compensation are: kp, ki, kd, and il (integration limit). The integration limit (il) constrains the contribution of the integration term

$$\left[ki * \sum_{N=0}^{n} e(n) \right]$$

(see Eq. 1) to values equal to or less than a user-defined maximum value; this capability minimizes integral or reset "wind-up" (an overshooting effect of the integral action). The positive-only input value is compared to the absolute

Filter Control Commands (Continued)

magnitude of the integration term; when the magnitude of integration term value exceeds il, the il value (with appropriate sign) is substituted for the integration term value.

The derivative-term sampling interval is also programmable via this command. After writing the command code, the first two data bytes that are written specify the derivative-term sampling interval and which of the four filter parameters is/are to be written via any forthcoming data bytes. The first byte written is the more significant. Thus the two data bytes constitute a filter control word that informs the LM628 as to the nature and number of any following data bytes. See Table IV.

TABLE IV. Filter Control word Bit Allocation

Bit Position	Function
Bit 15	Derivative Sampling Interval Bit 7
Bit 14	Derivative Sampling Interval Bit 6
Bit 13	Derivative Sampling Interval Bit 5
Bit 12	Derivative Sampling Interval Bit 4
Bit 11	Derivative Sampling Interval Bit 3
Bit 10	Derivative Sampling Interval Bit 2
Bit 9	Derivative Sampling Interval Bit 1
Bit 8	Derivative Sampling Interval Bit 0
Bit 7	Not Used
Bit 6	Not Used
Bit 5	Not Used
Bit 4	Not Used
Bit 3	Loading kp Data
Bit 2	Loading ki Data
Bit 1	Loading kd Data
Bit 0	Loading il Data

Bits 8 through 15 select the derivative-term sampling interval. See Table V. The user must locally save and restore these bits during successive writes of the filter control word.

Bits 4 through 7 of the filter control word are not used.

Bits 0 to 3 inform the LM628 as to whether any or all of the filter parameters are about to be written. The user may choose to update any or all (or none) of the filter parameters. Those chosen for updating are so indicated by logic one(s) in the corresponding bit position(s) of the filter control word.

The data bytes specified by and immediately following the filter control word are written in pairs to comprise 16-bit words. The order of sending the data words to the LM628 corresponds to the descending order shown in the above description of the filter control word; i.e., beginning with kp, then ki, kd and il. The first byte of each word is the more-significant byte. Prior to writing a word (byte pair) it is necessary to check the busy bit in the status byte for readiness. The required data is written to the primary buffers of a double-buffered scheme by the above described operations; it is not transferred to the secondary (working) registers until the UDF command is executed. This fact can be used advantageously; the user can input numerous data ahead of their actual use. This simple pipeline effect can relieve potential host computer data communications bottlenecks, and facilitates easier synchronization of multiple-axis controls.

UDF COMMAND: UpDate Filter

Command Code: 04 Hex
Data Bytes: None
Executable During Motion: Yes

The UDF command is used to update the filter parameters, the specifics of which have been programmed via the LFIL command. Any or all parameters (derivative-term sampling interval, kp, ki, kd, and/or il) may be changed by the appropriate command(s), but command UDF must be executed to affect the change in filter tuning. Filter updating is synchronized with the calculations to eliminate erratic or spurious behavior.

Trajectory Control Commands

The following two LM628 user commands are used for setting the trajectory control parameters (position, velocity, acceleration), mode of operation (position or velocity), and direction (velocity mode only) as required to describe a desired motion or to select the mode of a manually directed stop, and to control the timing of these system changes.

LTRJ COMMAND: Load TRaJectory Parameters

Command Code: 1F Hex
Data Bytes: Two to Fourteen
Data Ranges . . .
Trajectory Control Word: See Text
Position: C0000000 to 3FFFFFFF Hex
Velocity: 00000000 to 3FFFFFFF Hex (Pos Only)
Acceleration: 00000000 to 3FFFFFFF Hex (Pos Only)
Executable During Motion: Conditionally, See Text

TABLE V. Derivative-Term Sampling Interval Selection Codes

	Bit Position								Selected Derivative Sampling Interval
	15	14	13	12	11	10	9	8	
	0	0	0	0	0	0	0	0	256 μs
	0	0	0	0	0	0	0	1	512 μs
	0	0	0	0	0	0	1	0	768 μs
	0	0	0	0	0	0	1	1	1024 μs, etc . . .
thru	1	1	1	1	1	1	1	1	65,536 μs

Note: Sampling intervals shown are when using an 8.0 MHz clock. The 256 corresponds to 2048/8 MHz; sample intervals must be scaled for other clock frequencies.

Trajectory Control Commands (Continued)

The trajectory control parameters which are written to the LM628 to control motion are: acceleration, velocity, and position. In addition, indications as to whether these three parameters are to be considered as absolute or relative inputs, selection of velocity mode and direction, and manual stopping mode selection and execution are programmable via this command. After writing the command code, the first two data bytes that are written specify which parameter(s) is/are being changed. The first byte written is the more significant. Thus the two data bytes constitute a trajectory control word that informs the LM628 as to the nature and number of any following data bytes. See Table VI.

TABLE VI. Trajectory Control Word Bit Allocation

Bit Position	Function
Bit 15	Not Used
Bit 14	Not Used
Bit 13	Not Used
Bit 12	Forward Direction (Velocity Mode Only)
Bit 11	Velocity Mode
Bit 10	Stop Smoothly (Decelerate as Programmed)
Bit 9	Stop Abruptly (Maximum Deceleration)
Bit 8	Turn Off Motor (Output Zero Drive)
Bit 7	Not Used
Bit 6	Not Used
Bit 5	Acceleration Will Be Loaded
Bit 4	Acceleration Data Is Relative
Bit 3	Velocity Will Be Loaded
Bit 2	Velocity Data Is Relative
Bit 1	Position Will Be Loaded
Bit 0	Position Data Is Relative

Bit 12 determines the motor direction when in the velocity mode. A logic one indicates forward direction. This bit has no effect when in position mode.

Bit 11 determines whether the LM628 operates in velocity mode (Bit 11 logic one) or position mode (Bit 11 logic zero).

Bits 8 through 10 are used to select the method of *manually stopping* the motor. These bits are *not* provided for one to merely specify the desired *mode* of stopping, in position mode operations, normal stopping is always smooth and occurs automatically at the end of the specified trajectory. Under exceptional circumstances it may be desired to manually intervene with the trajectory generation process to affect a premature stop. In velocity mode operations, however, the normal means of stopping *is* via bits 8 through 10 (usually bit 10). Bit 8 is set to logic one to stop the motor by turning off motor drive output (outputting the appropriate offset-binary code to apply zero drive to the motor); bit 9 is set to one to stop the motor abruptly (at maximum available acceleration, by setting the target position equal to the current position); and bit 10 is set to one to stop the motor smoothly by using the current user-programmed acceleration value. Bits 8 through 10 are to be used *exclusively;* only one bit should be a logic one at any time.

Bits 0 through 5 inform the LM628 as to whether any or all of the trajectory controlling parameters are about to be written, and whether the data should be interpreted as absolute or relative. The user may choose to update any or all (or none) of the trajectory parameters. Those chosen for updating are so indicated by logic one(s) in the corresponding bit position(s). Any parameter may be changed while the motor is in motion; however, if acceleration is changed then the next STT command must not be issued until the LM628 has completed the current move or has been manually stopped.

The data bytes specified by and immediately following the trajectory control word are written in pairs which comprise 16-bit words. Each data item (parameter) requires two 16-bit words; the word and byte order is most-to-least significant. The order of sending the parameters to the LM628 corresponds to the descending order shown in the above description of the trajectory control word; i.e., beginning with acceleration, then velocity, and finally position.

Acceleration and velocity are 32 bits, positive only, but range only from 0 (00000000 hex) to $[2^{30}]-1$ (3FFFFFFF hex). The bottom 16 bits of both acceleration and velocity are scaled as fractional data; therefore, the least-significant integer data bit for these parameters is bit 16 (where the bits are numbered 0 through 31). To determine the coding for a given velocity, for example, one multiplies the desired velocity (in counts per sample interval) times 65,536 and converts the result to binary. The units of acceleration are counts per sample per sample. The value loaded for acceleration must not exceed the value loaded for velocity. Position is a signed, 32-bit integer, but ranges only from $-[2^{30}]$ (C0000000 hex) to $[2^{30}]-1$ (3FFFFFFF Hex).

The required data is written to the primary buffers of a double-buffered scheme by the above described operations; it is not transferred to the secondary (working) registers until the STT command is executed. This fact can be used advantageously; the user can input numerous data ahead of their actual use. This simple pipeline effect can relieve potential host computer data communications bottlenecks, and facilitates easier synchronization of multiple-axis controls.

STT COMMAND: STarT Motion Control

Command Code:	01 Hex
Data Bytes:	None
Executable During Motion:	Yes, if acceleration has not been changed

The STT command is used to execute the desired trajectory, the specifics of which have been programmed via the LTRJ command. Synchronization of multi-axis control (to within one sample interval) can be arranged by loading the required trajectory parameters for each (and every) axis and then simultaneously issuing a single STT command to all axes. This command may be executed at any time, unless the acceleration value has been changed and a trajectory has not been completed or the motor has not been manually stopped. If STT is issued during motion and acceleration has been changed, a command error interrupt will be generated and the command will be ignored.

Data Reporting Commands

The following seven LM628 user commands are used to obtain data from various registers in the LM628. Status, position, and velocity information are reported. With the exception of RDSTAT, the data is read from the LM628 data port after first writing the corresponding command to the command port.

Data Reporting Commands (Continued)

RDSTAT COMMAND: ReaD STATus Byte

Command Code: None
Byte Read: One
Data Range: See Text
Executable During Motion: Yes

The RDSTAT command is really not a command, but is listed with the other commands because it is used very frequently to control communications with the host computer. There is no identification code; it is directly supported by the hardware and may be executed at any time. The single-byte status read is selected by placing $\overline{CS}$, $\overline{PS}$ and $\overline{RD}$ at logic zero. See Table VII.

TABLE VII. Status Byte Bit Allocation

Bit Position	Function
Bit 7	Motor Off
Bit 6	Breakpoint Reached [Interrupt]
Bit 5	Excessive Position Error [Interrupt]
Bit 4	Wraparound Occurred [Interrupt]
Bit 3	Index Pulse Observed [Interrupt]
Bit 2	Trajectory Complete [Interrupt]
Bit 1	Command Error [Interrupt]
Bit 0	Busy Bit

Bit 7, the motor-off flag, is set to logic one when the motor drive output is off (at the half-scale, offset-binary code for zero). The motor is turned off by any of the following conditions: power-up reset, command RESET, excessive position error (if command LPES had been executed), or when command LTRJ is used to manually stop the motor via turning the motor off. Note that when bit 7 is set in conjunction with command LTRJ for producing a manual, motor-off stop, the actual setting of bit 7 does not occur until command STT is issued to affect the stop. Bit 7 is cleared by command STT, except as described in the previous sentence.

Bit 6, the breakpoint-reached interrupt flag, is set to logic one when the position breakpoint loaded via command SBPA or SBPR has been exceeded. The flag is functional independent of the host interrupt mask status. Bit 6 is cleared via command RSTI.

Bit 5, the excessive-position-error interrupt flag, is set to logic one when a position-error interrupt condition exists. This occurs when the error threshold loaded via command LPEI or LPES has been exceeded. The flag is functional independent of the host interrupt mask status. Bit 5 is cleared via command RSTI.

Bit 4, the wraparound interrupt flag, is set to logic one when a numerical "wraparound" has occurred. To "wraparound" means to exceed the position address space of the LM628, which could occur during velocity mode operation. If a wraparound has occurred, then position information will be in error and this interrupt helps the user to ensure position data integrity. The flag is functional independent of the host interrupt mask status. Bit 4 is cleared via command RSTI.

Bit 3, the index-pulse acquired interrupt flag, is set to logic one when an index pulse has occurred (if command SIP had been executed) and indicates that the index position register has been updated. The flag is functional independent of the host interrupt mask status. Bit 3 is cleared by command RSTI.

Bit 2, the trajectory complete interrupt flag, is set to logic one when the trajectory programmed by the LTRJ command and initiated by the STT command has been completed. Because of overshoot or a limiting condition (such as commanding the velocity to be higher than the motor can achieve), the motor may not yet be at the final commanded position. This bit is the logical OR of bits 7 and 10 of the Signals Register, see command RDSIGS below. The flag functions independently of the host interrupt mask status. Bit 2 is cleared via command RSTI.

Bit 1, the command-error interrupt flag, is set to logic one when the user attempts to read data when a write was appropriate (or vice versa). The flag is functional independent of the host interrupt mask status. Bit 1 is cleared via command RSTI.

Bit 0, the busy flag, is frequently tested by the user (via the host computer program) to determine the busy/ready status prior to writing and reading any data. Such writes and reads may be executed only when bit 0 is logic zero (not busy). Any command or data writes when the busy bit is high will be ignored. Any data reads when the busy bit is high will read the current contents of the I/O port buffers, not the data expected by the host. Such reads or writes (with the busy bit high) will not generate a command-error interrupt.

RDSIGS COMMAND: ReaD SIGnalS Register

Command Code: 0C Hex
Bytes Read: Two
Data Range: See Text
Executable During Motion: Yes

The LM628 internal "signals" register may be read using this command. The first byte read is the more significant. The less significant byte of this register (with the exception of bit 0) duplicates the status byte. See Table VIII.

TABLE VIII. Signals Register Bit Allocation

Bit Position	Function
Bit 15	Host Interrupt
Bit 14	Acceleration Loaded (But Not Updated)
Bit 13	UDF Executed (But Filter Not yet Updated)
Bit 12	Forward Direction
Bit 11	Velocity Mode
Bit 10	On Target
Bit 9	Turn Off upon Excessive Position Error
Bit 8	Eight-Bit Output Mode
Bit 7	Motor Off
Bit 6	Breakpoint Reached [Interrupt]
Bit 5	Excessive Position Error [Interrupt]
Bit 4	Wraparound Occurred [Interrupt]
Bit 3	Index Pulse Acquired [Interrupt]
Bit 2	Trajectory Complete [Interrupt]
Bit 1	Command Error [Interrupt]
Bit 0	Acquire Next Index (SIP Executed)

Bit 15, the host interrupt flag, is set to logic one when the host interrupt output (Pin 17) is logic one. Pin 17 is set to logic one when any of the six host interrupt conditions occur (if the corresponding interrupt has not been masked). Bit 15 (and Pin 17) are cleared via command RSTI.

Bit 14, the acceleration-loaded flag, is set to logic one when acceleration data is written to the LM628. Bit 14 is cleared by the STT command.

Data Reporting Commands (Continued)

Bit 13, the UDF-executed flag, is set to logic one when the UDF command is executed. Because bit 13 is cleared at the end of the sampling interval in which it has been set, this signal is very short-lived and probably not very profitable for monitoring.

Bit 12, the forward direction flag, is meaningful only when the LM628 is in velocity mode. The bit is set to logic one to indicate that the desired direction of motion is "forward"; zero indicates "reverse" direction. Bit 12 is set and cleared via command LTRJ. The actual setting and clearing of bit 12 does not occur until command STT is executed.

Bit 11, the velocity mode flag, is set to logic one to indicate that the user has selected (via command LTRJ) velocity mode. Bit 11 is cleared when position mode is selected (via command LTRJ). The actual setting and clearing of bit 11 does not occur until command STT is executed.

Bit 10, the on-target flag, is set to logic one when the trajectory generator has completed its functions for the last-issued STT command. Bit 10 is cleared by the next STT command.

Bit 9, the turn-off on-error flag, is set to logic one when command LPES is executed. Bit 9 is cleared by command LPEI.

Bit 8, the 8-bit output flag, is set to logic one when the LM628 is reset, or when command PORT8 is executed. Bit 8 is cleared by command PORT12.

Bits 0 through 7 replicate the status byte (see Table VII), with the exception of bit 0. Bit 0, the acquire next index flag, is set to logic one when command SIP is executed; it then remains set until the next index pulse occurs.

RDIP COMMAND: ReaD Index Position

Command Code:	09 Hex
Bytes Read:	Four
Data Range:	C0000000 to 3FFFFFFF Hex
Executable During Motion:	Yes

This command reads the position recorded in the index register. Reading the index register can be part of a system error checking scheme. Whenever the SIP command is executed, the new index position minus the old index position, divided by the incremental encoder resolution (encoder lines times four), should always be an integral number. The RDIP command facilitates acquiring these data for host-based calculations. The command can also be used to identify/verify home or some other special position. The bytes are read in most-to-least significant order.

RDDP COMMAND: ReaD Desired Position

Command Code:	08 Hex
Bytes Read:	Four
Data Range:	C0000000 to 3FFFFFFF Hex
Executable During Motion:	Yes

This command reads the instantaneous desired (current *temporal*) position output of the profile generator. This is the "setpoint" input to the position-loop summing junction. The bytes are read in most-to-least significant order.

RDRP COMMAND: ReaD Real Position

Command Code:	0A Hex
Bytes Read:	Four
Data Range:	C0000000 to 3FFFFFFF Hex
Executable During Motion:	Yes

This command reads the current actual position of the motor. This is the feedback input to the loop summing junction. The bytes are read in most-to-least significant order.

RDDV COMMAND: ReaD Desired Velocity

Command Code:	07 Hex
Bytes Read:	Four
Data Range:	C0000001 to 3FFFFFFF
Executable During Motion:	Yes

This command reads the integer and fractional portions of the instantaneous desired (current *temporal*) velocity, as used to generate the desired position profile. The bytes are read in most-to-least significant order. The value read is properly scaled for numerical comparison with the user-supplied (commanded) velocity; however, because the two least-significant bytes represent *fractional* velocity, only the two most-significant bytes are appropriate for comparison with the data obtained via command RDRV (see below). Also note that, although the velocity *input* data is constrained to positive numbers (see command LTRJ), the data returned by command RDDV represents a *signed* quantity where negative numbers represent operation in the reverse direction.

RDRV COMMAND: ReaD Real Velocity

Command Code:	0B Hex
Bytes Read:	Two
Data Range:	C000 to 3FFF Hex, See Text
Executable During Motion:	Yes

This command reads the *integer* portion of the instantaneous actual velocity of the motor. The internally maintained fractional portion of velocity is not reported because the reported data is derived by reading the incremental encoder, which produces only integer data. For comparison with the result obtained by executing command RDDV (or the user-supplied input value), the value returned by command RDRV must be multiplied by 2^{16} (shifted left 16 bit positions). Also, as with command RDDV above, data returned by command RDRV is a *signed* quantity, with negative values representing reverse-direction motion.

RDSUM COMMAND: ReaD Integration-Term SUMmation Value

Command Code:	0D Hex
Bytes Read:	Two
Data Range:	00000 Hex to ± the Current Value of the Integration Limit
Executable During Motion:	Yes

This command reads the value to which the integration term has accumulated. The ability to read this value may be helpful in initially or adaptively tuning the system.

Typical Applications

Programming LM628 Host Handshaking (Interrupts)

A few words regarding the LM628 host handshaking will be helpful to the system programmer. As indicated in various portions of the above text, the LM628 handshakes with the host computer in two ways: via the host interrupt output (Pin 17), or via polling the status byte for "interrupt" conditions. When the hardwired interrupt is used, the status byte is also read and parsed to determine which of six possible conditions caused the interrupt.

Typical Applications (Continued)

When using the hardwired interrupt it is very important that the host interrupt service routine does not interfere with a command sequence which might have been in progress when the interrupt occurred. If the host interrupt service routine were to issue a command to the LM628 while it is in the middle of an ongoing command sequence, the ongoing command will be aborted (which could be detrimental to the application).

Two approaches exist for avoiding this problem. If one is using hardwired interrupts, they should be disabled at the host prior to issuing any LM628 command sequence, and re-enabled after each command sequence. The second approach is to avoid hardwired interrupts and poll the LM628 status byte for "interrupt" status. The status byte always reflects the interrupt-condition status, independent of whether or not the interrupts have been masked.

Typical Host Computer/Processor Interface

The LM628 is interfaced with the host computer/processor via an 8-bit parallel bus. *Figure 12* shows such an interface and a minimum system configuration.

As shown in *Figure 12*, the LM628 interfaces with the host data, address and control lines. The address lines are decoded to generate the LM628 $\overline{CS}$ input; the host address LSB directly drives the LM628 $\overline{PS}$ input. *Figure 12* also shows an 8-bit DAC and an LM12 Power Op Amp interfaced to the LM628.

LM628 and High Performance Controller (HPC) Interface

Figure 13 shows the LM628 interfaced to a National HPC High Performance Controller. The delay and logic associated with the $\overline{WR}$ line is used to effectively increase the write-data hold time of the HPC (as seen at the LM628) by causing the $\overline{WR}$ pulse to rise early. Note that the HPC CK2 output provides the clock for the LM628. The 74LS245 is used to decrease the read-data hold time, which is necessary when interfacing to fast host busses.

Interfacing a 12-Bit DAC

Figure 14 illustrates use of a 12-bit DAC with the LM628. The 74LS378 hex gated-D flip-flop and an inverter demultiplex the 12-bit output. DAC offset must be adjusted to minimize DAC linearity and monotonicity errors. Two methods exist for making this adjustment. If the DAC1210 has been socketed, remove it and temporarily connect a 15 kΩ resistor between Pins 11 and 13 of the DAC socket (Pins 2 and 6 of the LF356) and adjust the 25 kΩ potentiometer for 0V at Pin 6 of the LF356.

If the DAC is not removable, the second method of adjustment requires that the DAC1210 inputs be presented an all-zeros code. This can be arranged by commanding the appropriate move via the LM628, but with no feedback from the system encoder. When the all-zeros code is present, adjust the pot for 0V at Pin 6 of the LF356.

A Monolithic Linear Drive Using LM12 Power Op Amp

Figure 15 shows a motor-drive amplifier built using the LM12 Power Operational Amplifier. This circuit is very simple and can deliver up to 8A at 30V (using the LM12L/LM12CL). Resistors R1 and R2 should be chosen to set the gain to provide maximum output voltage consistent with maximum input voltage. This example provides a gain of 2.2, which allows for amplifier output saturation at ±22V with a ±10V input, assuming power supply voltages of ±30V. The amplifier gain should not be higher than necessary because the system is non-linear when saturated, and because gain should be controlled by the LM628. The LM12 can also be configured as a current driver, see 1987 Linear Databook, Vol. 1, p. 2-280.

Typical PWM Motor Drive Interfaces

Figure 16 shows an LM18298 dual full-bridge driver interfaced to the LM629 PWM outputs to provide a switch-mode power amplifier for driving small brush/commutator motors. *Figure 17* shows an LM621 brushless motor commutator interfaced to the LM629 PWM outputs and a discrete device switch-mode power amplifier for driving brushless DC motors.

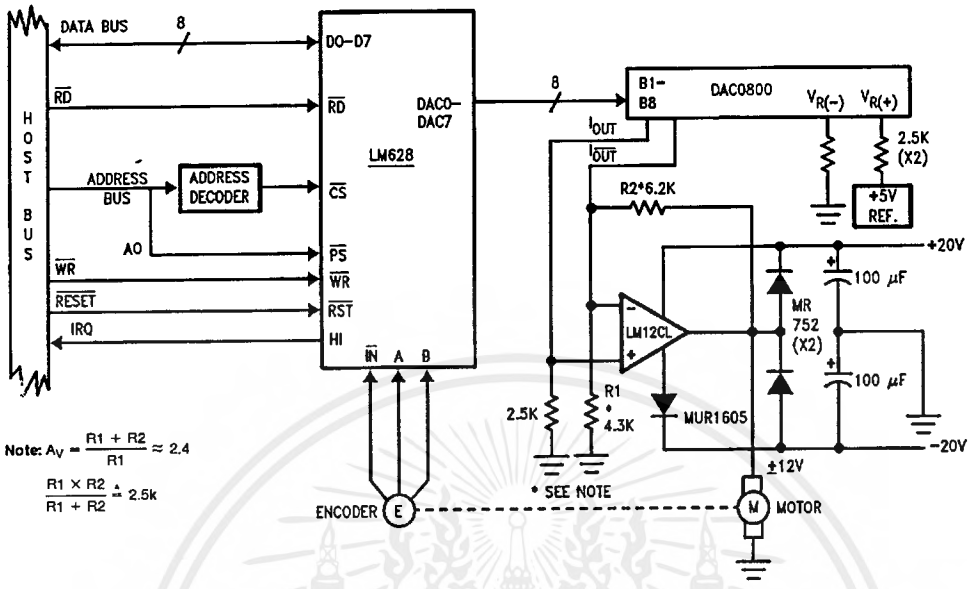
Incremental Encoder Interface

The incremental (position feedback) encoder interface consists of three lines: Phase A (Pin 2), Phase B (Pin 3), and Index (Pin 1). The index pulse output is not available on some encoders. The LM628 will work with both encoder types, but commands SIP and RDIP will not be meaningful without an index pulse (or alternative input for this input . . . be sure to tie Pin 1 high if not used).

Some consideration is merited relative to use in high Gaussian-noise environments. If noise is added to the encoder inputs (either or both inputs) and is such that it is not sustained until the next encoder transition, the LM628 decoder logic will reject it. Noise that mimics quadrature counts or persists through encoder transitions must be eliminated by appropriate EMI design.

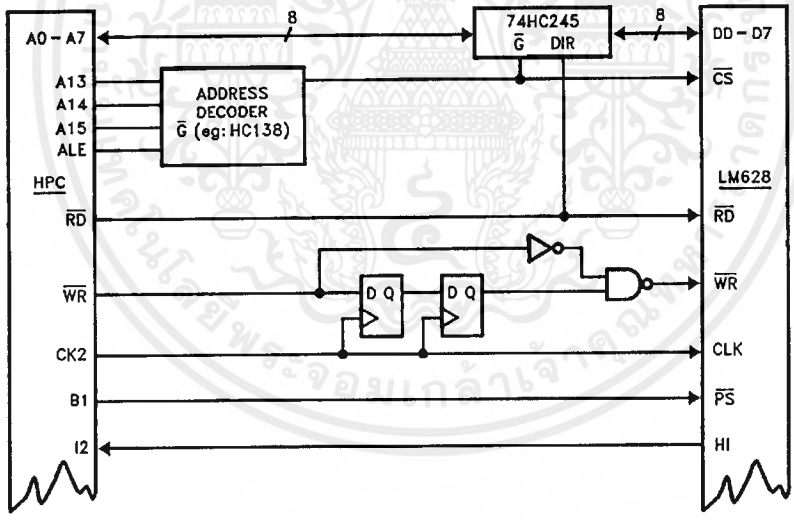
Simple digital "filtering" schemes merely reduce susceptibility to noise (there will always be noise pulses longer than the filter can eliminate). Further, any noise filtering scheme reduces decoder bandwidth. In the LM628 it was decided (since simple filtering does not eliminate the noise problem) to not include a noise filter in favor of offering maximum possible decoder bandwidth. Attempting to drive encoder signals too long a distance with simple TTL lines can also be a source of "noise" in the form of signal degradation (poor risetime and/or ringing). This can also cause a system to lose positional integrity. Probably the most effective countermeasure to noise induction can be had by using balanced-line drivers and receivers on the encoder inputs. *Figure 18* shows circuitry using the DS26LS31 and DS26LS32.

Typical Applications (Continued)



TL/H/9219–14

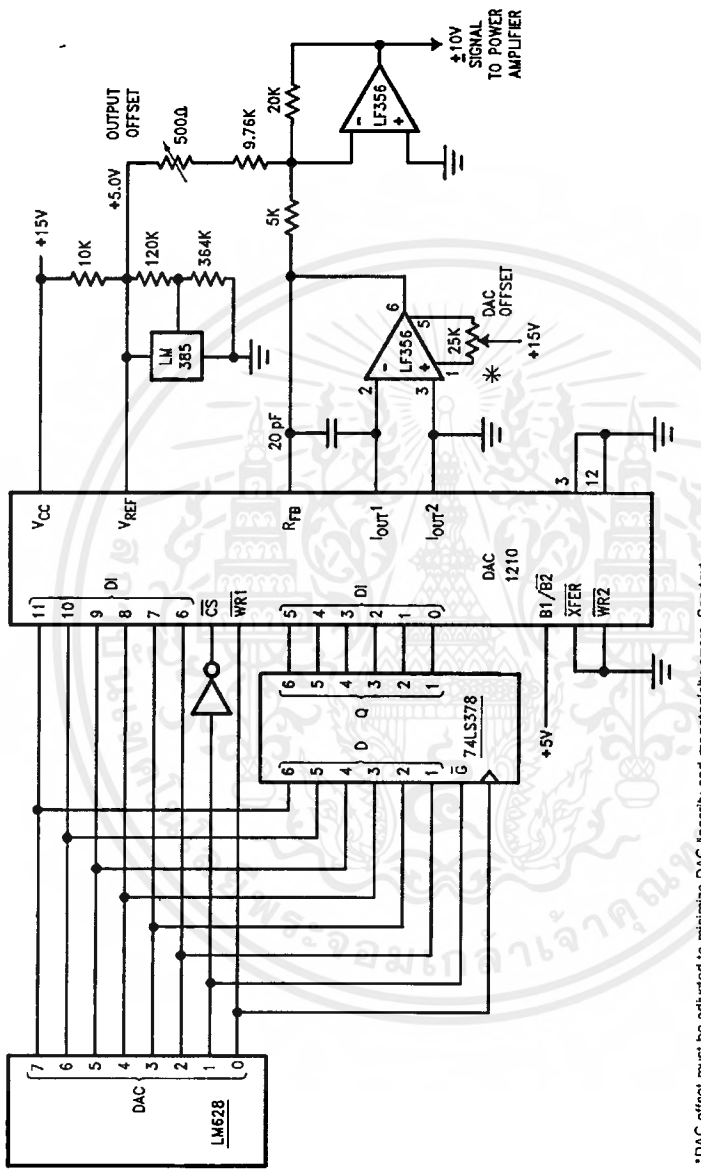
FIGURE 12. Host Interface and Minimum System Configuration



TL/H/9219–15

FIGURE 13. LM628 and HPC Interface

Typical Applications (Continued)

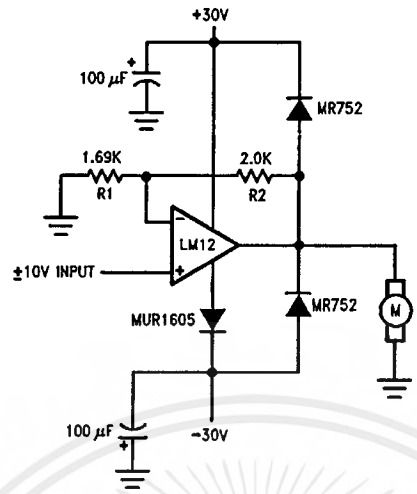


TL/H/9218–16

*DAC offset must be adjusted to minimize DAC linearity and monotonicity errors. See text.

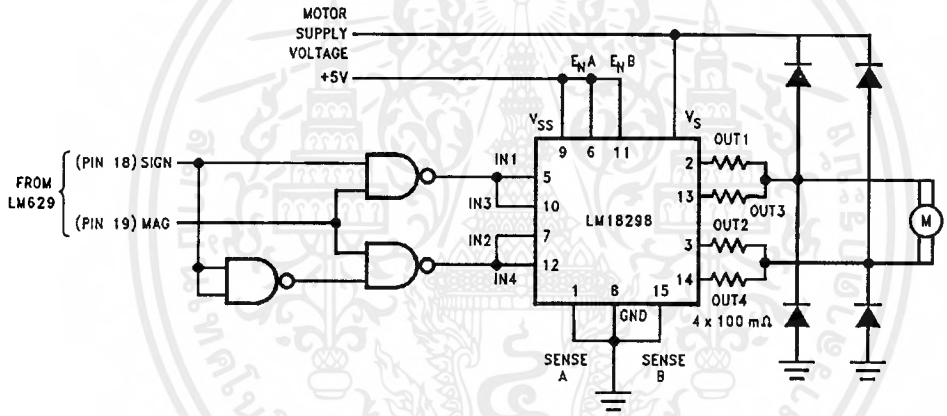
FIGURE 14. Interfacing a 12-Bit DAC and LM628

Typical Applications (Continued)



TL/H/9219–17

FIGURE 15. Driving a Motor with the LM12 Power Op Amp



TL/H/9219–18

FIGURE 16. PWM Drive for Brush/Commutator Motors

Typical Applications (Continued)

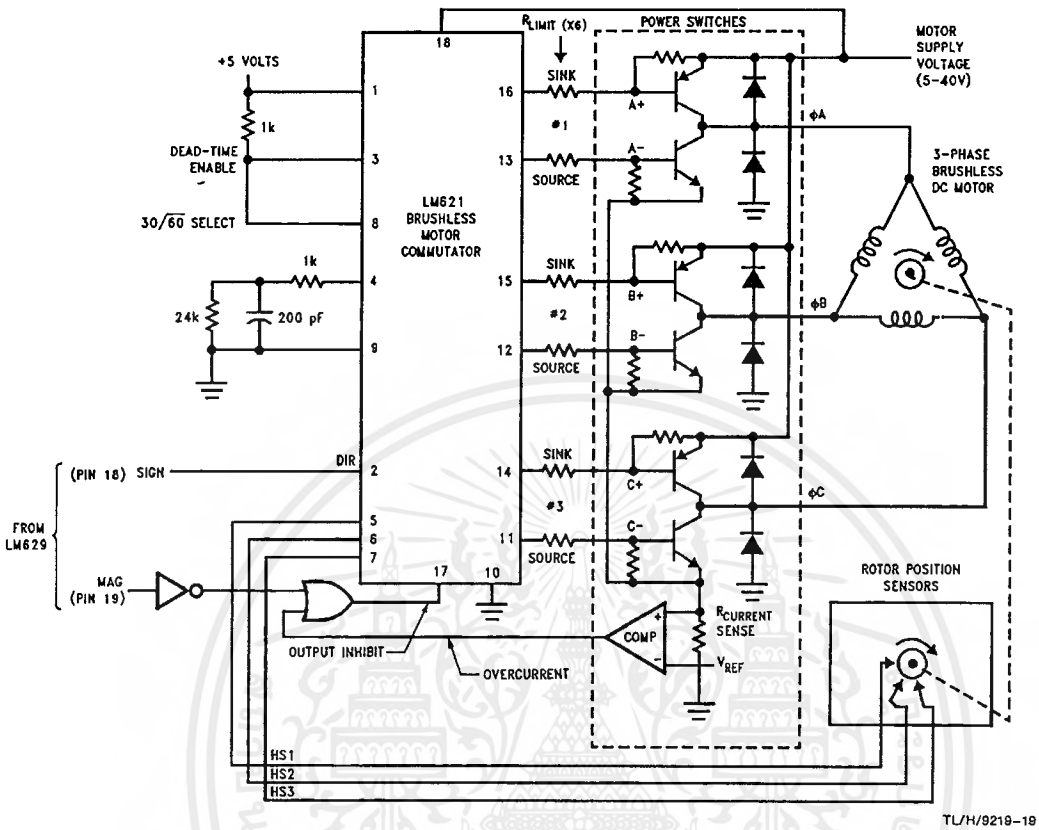


FIGURE 17. PWM Drive for Brushless Motors

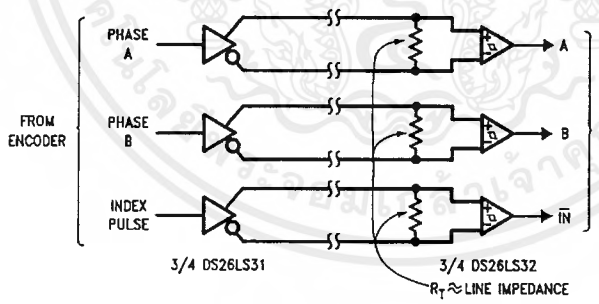
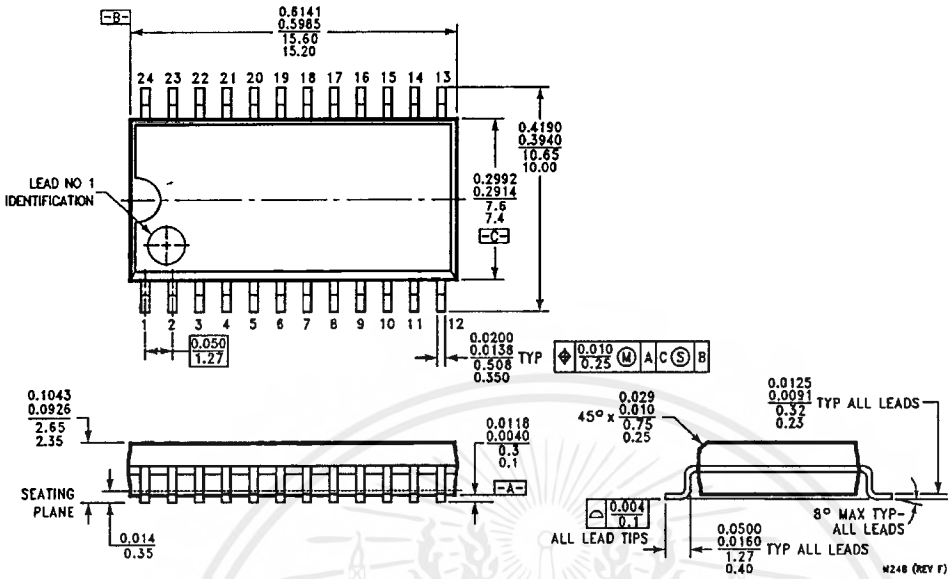


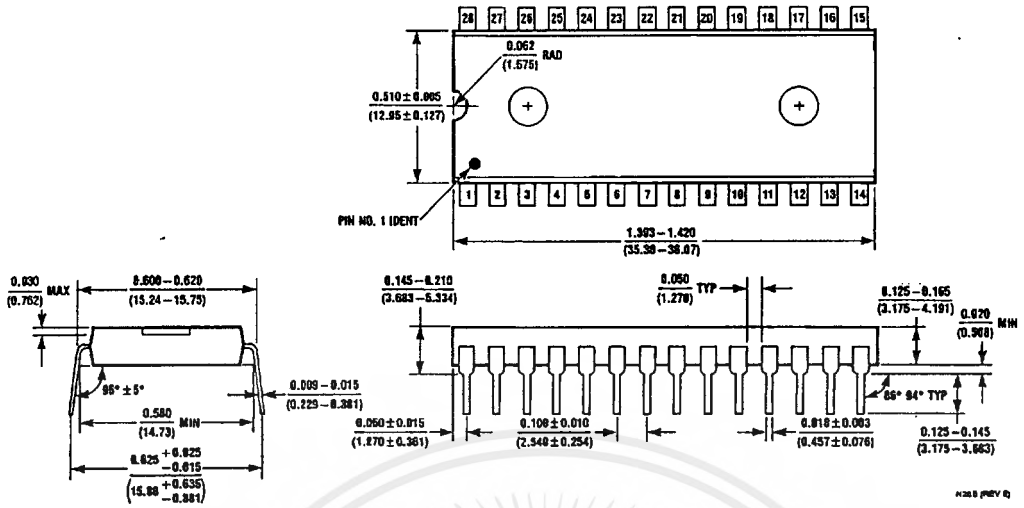
FIGURE 18. Typical Balanced-Line Encoder Input Circuit

Physical Dimensions inches (millimeters)



24-Lead Small Outline Package (M)
Order Number LM629M-6 or LM629M-8
NS Package Number M24B

Physical Dimensions inches (millimeters) (Continued)



28 Lead Molded Dual-In-Line Package (N)
Order Number LM628N-6, LM628N-8, LM629N-6 or LM629N-8
NS Package Number N28B

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408

HARRIS Semiconductor

HS-82C55ARH

Radiation Hardened CMOS Programmable Peripheral Interface

September 1995

Features

- Radiation Hardened
  - Total Dose >10^5 RAD (Si)
  - Transient Upset <10^8 RAD (Si)/s
  - Latch Up Free EPI-CMOS
- Low Power Consumption
  - IDDSB = 20μA
- Pin Compatible with NMOS 8255A and the Harris 82C55A
- High Speed, No "Wait State" Operation with 5MHz HS-80C86RH
- 24 Programmable I/O Pins
- Bus-Hold Circuitry on All I/O Ports Eliminates Pull-Up Resistors
- Direct Bit Set/Reset Capability
- Enhanced Control Word Read Capability
- Hardened Field, Self-Aligned, Junction Isolated CMOS Process
- Single 5V Supply
- 2.0mA Drive Capability on All I/O Port Outputs
- Military Temperature Range: -55°C to +125°C

Description

The Harris HS-82C55ARH is a high performance, radiation hardened CMOS version of the industry standard 8255A and is manufactured using a hardened field, self-aligned silicongate CMOS process. It is a general purpose programmable I/O device which may be used with many different microprocessors. There are 24 I/O pins which are organized into two 8-bit and two 4-bit ports. Each port may be programmed to function as either an input or an output. Additionally, one of the 8-bit ports may be programmed for bi-directional operation,and the two 4-bit ports can be programmed to provide handshaking capabilities. The high performance, radiation hardness, and industry standard configuration of the HS-82C55ARH make it compatible with the HS-80C86RH radiation hardened microprocessor.

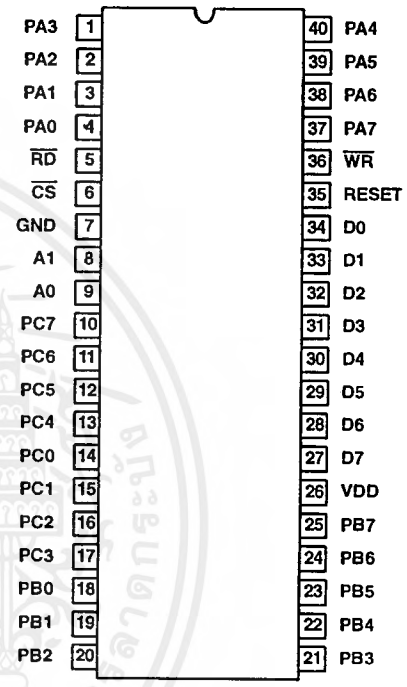
Static CMOS circuit design insures low operating power. Bus hold circuitry eliminates the need for pull-up resistors. The Harris hardened field CMOS process results in performance equal to or greater than existing radiation resistant products at a fraction of the power.

Ordering Information

PART NUMBER	TEMPERATURE	PACKAGE
HS1-82C55ARH-Q	-55°C to +125°C	40 Lead SBDIP
HS1-82C55ARH-8	-55°C to +125°C	40 Lead SBDIP
HS1-82C55ARH/Sample	+25°C	40 Lead SBDIP

Pinout

40 LEAD CERAMIC DUAL-IN-LINE
METAL SEAL PACKAGE (SBDIP)
MIL-STD-1835 CDIP2-T40
TOP VIEW



Pin Description

PIN	DESCRIPTION
D7 - D0	Data Bus (Bi-Directional
RESET	Reset Input
$\overline{CS}$	Chip Select
$\overline{RD}$	Read Input
$\overline{WR}$	Write Input
A0 - A1	Port Address
PA7 - PA0	Port A (Bit)
PB& - PB0	Port B (Bit)
PC7 - PC0	Port C (Bit)
VDD	+5 volts
GND	0 volts

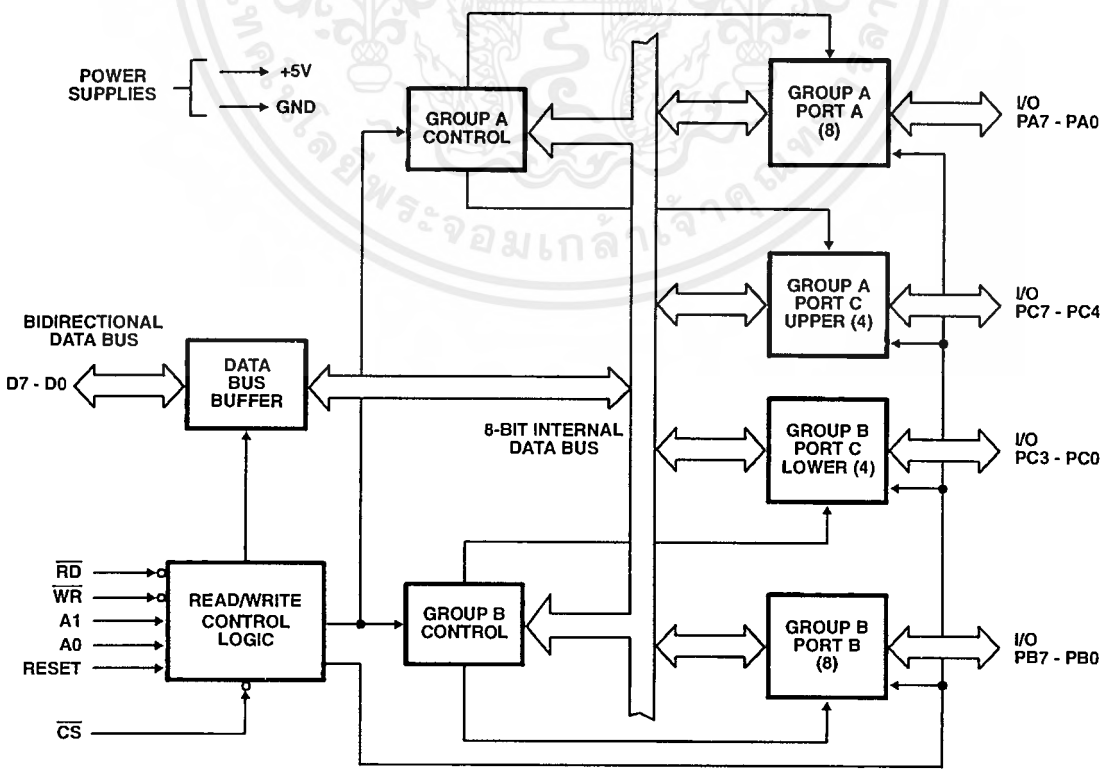
CAUTION: These devices are sensitive to electrostatic discharge. Users should follow proper I.C. Handling Procedures.

Spec Number **518060**
File Number **3191.1**

Pin Description

SYMBOL	PIN NUMBERS	TYPE	DESCRIPTION
PA0-7	1-4, 37-40	I/O	**Port A:** General purpose I/O Port. Data direction and mode is determined by the contents of the Control Word.
PB0-7	18-25	I/O	**Port B:** General purpose I/O port. See Port A.
PC0-3	14-17	I/O	**Port C (Lower):** Combination I/O port and control port associated with Port B. See Port A.
PC4-7	10-13	I/O	**Port C (Upper):** Combination I/O Port and control port associated with Port A. See Port A.
D0-7	27-34	I/O	**Bidirectional Data Bus:** Three-State data bus enabled as an input when $\overline{CS}$ and $\overline{WR}$ are low and as an output when $\overline{CS}$ and $\overline{RD}$ are low.
VDD	26	I	**VDD:** The +5V power supply pin. A 0.1µF capacitor between pins 26 and 7 is recommended for decoupling.
GND	7	I	**Ground.**
$\overline{CS}$	6	I	**Chip Select:** A "low" on this input pin enables the communication between the HS-82C55ARH and the CPU.
$\overline{RD}$	5	I	**Read:** A "low" on this input pin enables the HS-82C55ARH to send the data or status information to the CPU on the data bus. In essence, it allows the CPU to "read from" the HS-82C55ARH.
$\overline{WR}$	36	I	**Write:** A "low" on this input pin enables the CPU to write data or control words into the HS-82C55ARH.
A0 and A1	8, 9	I	**Port Select 0 and Port Select 1:** These input signals, in conjunction with the $\overline{RD}$ and $\overline{WR}$ inputs, control the selection of one of the three ports or the control word registers. They are normally connected to the Least Significant Bits of the address bus (A0 and A1).
Reset	35	I	**Reset:** A "high" on this input clears the control register and all ports (A, B, C) are set to the input mode. "Bus hold" devices internal to the HS-82C55ARH will hold the I/O port inputs to a logic "1" state with a maximum hold current of 400µA.

Functional Diagram



Absolute Maximum Ratings

Supply Voltage . . . . . +7.0V
Input, Output or I/O Voltage . . . . . VSS-0.3V to VDD+0.3V
Storage Temperature Range . . . . . -65°C to +150°C
Junction Temperature . . . . . +175°C
Lead Temperature (Soldering 10s) . . . . . +300°C
ESD Classification . . . . . Class 1

Reliability Information

Thermal Resistance	θ_{JA}	θ_{JC}
SBDIP Package	40°C/W	6°C/W

Maximum Package Power Dissipation at +125°C Ambient
SBDIP Package 1.25W
If device power exceeds package dissipation capability, provide heat sinking or derate linearly at the following rate:
SBDIP Package 25.0mW/C

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

Operating Conditions

Operating Voltage Range . . . . . +4.5V to +5.5V
Operating Temperature Range . . . . . -55°C to +125°C
Input Low Voltage . . . . . 0V to +0.8V
Input High Voltage . . . . . VDD -1.5V to VDD

TABLE 1. DC ELECTRICAL PERFORMANCE CHARACTERISTICS

PARAMETER	SYMBOL	CONDITIONS	GROUP A SUBGROUP	TEMPERATURE	LIMITS MIN	LIMITS MAX	UNITS
TTL Output High Voltage	VOH1	VDD = 4.5V, IO = -2.5mA, VIN = 0V, 4.5V	1, 2, 3	-55°C, +25°C, +125°C	3.0	-	V
CMOS Output High Voltage	VOH2	VDD = 4.5V, IO = -100µA, VIN = 0V, 4.5V	1, 2, 3	-55°C, +25°C, +125°C	VDD-0.4	-	V
Output Low Voltage	VOL	VDD = 4.5V, IO = 2.5mA, VIN = 0V, 4.5V	1, 2, 3	-55°C, +25°C, +125°C	-	0.4	V
Input Leakage Current	IIL or IIH	VDD = 5.5V, VIN = 0V, 5.5V	1, 2, 3	-55°C, +25°C, +125°C	-1.0	1.0	µA
Output Leakage Current	IOZL or IOZH	VDD = 5.5V, VIN = 0V, 5.5V	1, 2, 3	-55°C, +25°C, +125°C	-10	10	µA
Input Current Bus Hold High	IBHH	VDD = 4.5V or 5.5V, VIN = 3.0V (See Note 1) Ports A, B, C	1, 2, 3	-55°C, +25°C, +125°C	-800	-60	µA
Input Current Bus Hold Low	IBHL	VDD = 4.5V or 5.5V, VIN = 1.0V (See Note 2) Port A	1, 2, 3	-55°C, +25°C, +125°C	60	800	µA
Standby Power Supply Current	IDDSB	VDD = 5.5V, IO = 0mA, VIN =GND or VDD	1, 2, 3	-55°C, +25°C, +125°C	-	20	µA
Darlington Drive Voltage	VDAR	VDD = 4.5V, IO = -2.0mA, VIN = GND or VDD	1, 2, 3	-55°C, +25°C, +125°C	3.9	-	V
Functional Tests	FT	VDD = 4.5V and 5.5V, VIN = GND or VDD, f = 1MHz	7, 8A, 8B	-55°C, +25°C, +125°C	-	-	-
Noise Immunity Functional Test (Note 4)	FN	VDD = 5.5V, VIN = GND or VDD - 1.5V and VDD = 4.5V, VIN = 0.8V or VDD	7, 8A, 8B	-55°C, +25°C, +125°C	-	-	-

NOTES:

1. IBHH should be measured after raising VIN and then lowering to 3.0V.
2. IBHL should be measured after lowering VIN to VSS and then raising to 0.8V.
3. No internal current limiting exists on the Port Outputs. A resistor must be added externally to limit the current.
4. For VIH (VDD = 5.5V) and VIL (VDD = 4.5V) each of the following groups is tested separately with all other inputs using VIH = 2.6V, VIL = 0.4V: PA, PB, PC, Control Pins (Pins 5, 6, 8, 9, 35, 36).

TABLE 2. AC ELECTRICAL PERFORMANCE CHARACTERISTICS T_A = -55°C to +125°C

PARAMETER	SYMBOL	CONDITIONS	SUB-GROUPS	TEMPERATURE	LIMITS MIN	LIMITS MAX	UNITS
READ							
Address Stable Before $\overline{RD}$	TAVRL	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	0	-	ns
Address Stable After $\overline{RD}$	TRHAX	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	0	-	ns
$\overline{RD}$ Pulse Width	TRLRH	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	250	-	ns
Data Valid From $\overline{RD}$	TRLDV	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	-	200	ns
Data Float After $\overline{RD}$	TRHDX	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	10	-	ns
Time Between $\overline{RD}$s and/ or $\overline{WR}$s	TRWHRWL	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	300	-	ns
WRITE							
Address Stable Before $\overline{WR}$	TAVWL	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	0	-	ns
Address Stable After $\overline{WR}$	TWHAX	VDD = 4.5, 5.5V, Ports A and B	9, 10, 11	-55°C, +25°C, +125°C	20	-	ns
		VDD = 4.5, 5.5V, Port C	9, 10, 11	-55°C, +25°C, +125°C	100	-	ns
$\overline{WR}$ Pulse Width	TWLWH	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	100	-	ns
Data Valid to $\overline{WR}$ High	TDVWH	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	100	-	ns
Data Valid After $\overline{WR}$ High	TWHDX	VDD = 4.5, 5.5V, Ports A and B	9, 10, 11	-55°C, +25°C, +125°C	30	-	ns
		VDD = 4.5, 5.5V, Port C	9, 10, 11	-55°C, +25°C, +125°C	100	-	
OTHER TIMINGS							
$\overline{WR}$ = 1 to Output	TWHPV	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	-	350	ns
Peripheral Data Before $\overline{RD}$	TPVRL	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	0	-	ns
Peripheral Data After $\overline{RD}$	TRHPX	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	0	-	ns
$\overline{ACK}$ Pulse Width	TKLKH	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	200	-	ns
$\overline{STB}$ Pulse Width	TSLSH	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	100	-	ns
Peripheral Data Before $\overline{STB}$ High	TPVSH	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	20	-	ns
Peripheral Data After $\overline{STB}$ High	TSHPX	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	50	-	ns
$\overline{ACK}$ = 0 to Output	TKLPV	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	-	175	ns
$\overline{ACK}$ = 1 to output Float	TKHPZ	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	10	-	ns

TABLE 2. AC ELECTRICAL PERFORMANCE CHARACTERISTICS T_A = -55°C to +125°C (Continued)

PARAMETER	SYMBOL	CONDITIONS	SUB-GROUPS	TEMPERATURE	LIMITS		UNITS
					MIN	MAX	
$\overline{WR}$ = 1 to $\overline{OBF}$ = 0	TWHOL	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	-	150	ns
$\overline{ACK}$ = 0 to $\overline{OBF}$ = 1	TKLOH	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	-	150	ns
$\overline{STB}$ = 0 to IBF = 1	TSLIH	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	-	150	ns
$\overline{RD}$ = 1 to IBF = 0	TRHIL	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	-	150	ns
$\overline{RD}$ = 0 to INTR = 1	TRLNL	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	-	200	ns
$\overline{STB}$ = 1 t INTR = 1	TSHNH	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	-	150	ns
$\overline{ACK}$ = 1 to INTR = 1	TKHNH	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	-	150	ns
$\overline{WR}$ = 0 to INTR = 0	TWLNL	VDD = 4.5, 5.5V	9, 10, 11	-55°C, +25°C, +125°C	-	200	ns
RESET Pulse Width	TRSHRSL	VDD = 4.5, 5.5V (Note 2)	9, 10, 11	-55°C, +25°C, +125°C	500	-	ns

NOTES:

1. AC's tested at worst case VDD, guaranteed over full operating range.
2. Period of initial RESET pulse after power-on must be at least 50µs. Subsequenct RESET pulses may be 500ns minimum.

TABLE 3. ELECTRICAL PERFORMANCE CHARACTERISTICS

PARAMETER	SYMBOL	CONDITIONS	TEMPERATURE	LIMITS		UNITS
				MIN	MAX	
Input Capacitance	CIN	VDD = Open, f = 1MHz, All measurements referenced to device ground	T_A = +25°C	-	10	pF
I/O Capacitance	CI/O	VDD = Open, f = 1MHz, All measurements referenced to device ground	T_A = +25°C	-	20	pF
Data Float After $\overline{RD}$	TRHDX	VDD = 4.5V and 5.5V	-55°C < T_A < +125°C	-	75	ns
$\overline{ACK}$ = 1 to Output Float	TKHPZ	VDD = 4.5V and 5.5V	-55°C < T_A < +125°C	-	250	ns

NOTE: The parameters listed in Table 3 are controlled via design or process parameters and are not directly tested. These parameters are characterized upon initial design release and upon design changes which would affect these characteristics

TALBE 4. POST 100K RAD ELECTRICAL PERFORMANCE CHARACTERISTICS

See +25°C limits in Table 1 and Table 2 for Post RAD limits (Subgroups 1, 7, 9)

TABLE 5. BURN-IN DELTA PARAMETERS (+25°C)

PARAMETER	SYMBOL	DELTA LIMITS
Static Current	IDDSB	±10µA
Input Leakage Current	IIL, IIH	±200nA
Output Leakage Current	IOZL, IOZH	±2µA
Low Level Output Voltage	VOL	±80mV
TTL Output High Voltage	VOH1	±600mV
CMOS Output High Voltage	VOH2	±150mV

TABLE 6. APPLICABLE SUBGROUPS

CONFORMANCE GROUP	MIL-STD-883 METHOD	GROUP A SUBGROUPS			
		TESTED FOR -Q	RECORDED FOR -Q	TESTED FOR -8	RECORDED FOR -8
Initial Test	100% 5004	1, 7, 9	1 (Note 2)	1, 7, 9	
Interim Test	100% 5004	1, 7, 9, Δ	1, Δ (Note 2)	1, 7, 9	
PDA	100% 5004	1, 7, Δ	-	1, 7	
Final Test	100% 5004	2, 3, 8A, 8B, 10, 11	-	2, 3, 8A, 8B, 10, 11	
Group A (Note 1)	Sample 5005	1, 2, 3, 7, 8A, 8B, 9, 10, 11	-	1, 2, 3, 7, 8A, 8B, 9, 10, 11	
Subgroup B5	Sample 5005	1, 2, 3, 7, 8A, 8B, 9, 10, 11, Δ	1, 2, 3, Δ (Note 2)	N/A	
Subgroup B6	Sample 5005	1, 7, 9	-	N/A	
Group C	Sample 5005	N/A	N/A	1, 2, 3, 7, 8A, 8B, 9, 10, 11	
Group D	Sample 5005	1, 7, 9	-	1, 7, 9	
Group E, Subgroup 2	Sample 5005	1, 7, 9	-	1, 7, 9	

NOTES:

1. Alternate Group A testing in accordance with MIL-STD-883 method 5005 may be exercised.
2. Table 5 parameters only

Harris Space Level Product Flow -Q

Wafer Lot Acceptance (All Lots) Method 5007 (Includes SEM)

GAMMA Radiation Verification (Each Wafer) Method 1019, 2 Samples/Wafer, 0 Rejects

100% Die Attach

100% Nondestructive Bond Pull, Method 2023

Sample - Wire Bond Pull Monitor, Method 2011

Sample - Die Shear Monitor, Method 2019 or 2027

100% Internal Visual Inspection, Method 2010, Condition A

CSI and/or GSI PreCap (Note 6)

100% Temperature Cycle, Method 1010, Condition C, 10 Cycles

100% Constant Acceleration, Method 2001, Condition per Method 5004

100% PIND, Method 2020, Condition A

100% External Visual

100% Serialization

100% Initial Electrical Test (T0)

100% Static Burn-In 1, Condition A or B, 72 Hours Min, +125°C Min, Method 1015

100% Interim Electrical Test 1 (T1)

100% Delta Calculation (T0-T1)

100% PDA 1, Method 5004 (Note 1)

100% Dynamic Burn-In, Condition D, 240 Hours, +125°C or Equivalent, Method 1015

100% Interim Electrical Test 2(T2)

100% Delta Calculation (T0-T2)

100% PDA 2, Method 5004 (Note 1)

100% Final Electrical Test

100% Fine/Gross Leak, Method 1014

100% Radiographic (X-Ray), Method 2012 (Note 2)

100% External Visual, Method 2009

Sample - Group A, Method 5005 (Note 3)

Sample - Group B, Method 5005 (Note 4)

Sample - Group D, Method 5005 (Notes 4 and 5)

100% Data Package Generation (Note 7)

CSI and/or GSI Final (Note 6)

NOTES:

1. Failures from subgroup 1, 7 and deltas are used for calculating PDA. The maximum allowable PDA = 5% with no more than 3% of the failures from subgroup 7.
2. Radiographic (X-Ray) inspection may be performed at any point after serialization as allowed by Method 5004.
3. Alternate Group A testing may be performed as allowed by MIL-STD-883, Method 5005.
4. Group B and D inspections are optional and will not be performed unless required by the P.O. When required, the P.O. should include separate line items for Group B Test, Group B Samples, Group D Test and Group D Samples.
5. Group D Generic Data, as defined by MIL-I-38535, is optional and will not be supplied unless required by the P.O. When required, the P.O. should include a separate line item for Group D Generic Data. Generic data is not guaranteed to be available and is therefore not available in all cases.
6. CSI and/or GSI inspections are optional and will not be performed unless required by theP.O. When required, the P.O. should include separate line items for CSI PreCap inspection, CSI final inspection, GSI PreCap inspection, and/or GSI final inspection.
7. Data Package Contents:
   - Cover Sheet (Harris Name and/or Logo, P.O. Number, Customer Part Number, Lot Date Code, Harris Part Number, Lot Number, Quantity).
   - Wafer Lot Acceptance Report (Method 5007). Includes reproductions of SEM photos with percent of step coverage.
   - GAMMA Radiation Report. Contains Cover page, disposition, Rad Dose, Lot Number, Test Package used, Specification Numbers, Test equipment, etc. Radiation Read and Record data on file at Harris.
   - X-Ray report and film. Includes penetrometer measurements.
   - Screening, Electrical, and Group A attributes (Screening attributes begin after package seal).
   - Lot Serial Number Sheet (Good units serial number and lot number).
   - Variables Data (All Delta operations). Data is identified by serial number. Data header includes lot number and date of test.
   - Group B and D attributes and/or Generic data is included when required by the P.O.
   - The Certificate of Conformance is a part of the shipping invoice and is not part of the Data Book. The Certificate of Conformance is signed by an authorized Quality Representative.

Harris Space Level Product Flow -8

GAMMA Radiation Verification (Each Wafer) Method 1019, 2 Samples/Wafer, 0 Rejects

100% Die Attach

Periodic- Wire Bond Pull Monitor, Method 2011

Periodic- Die Shear Monitor, Method 2019 or 2027

100% Internal Visual Inspection, Method 2010, Condition B

CSI an/or GSI PreCap (Note 5)

100% Temperature Cycle, Method 1010, Condition C, 10 Cycles

100% Constant Acceleration, Method 2001, Condition per Method 5004

100% External Visual

100% Initial Electrical Test

100% Dynamic Burn-In, Condition D, 160 Hours, +125°C or Equivalent, Method 1015

100% Interim Electrical Test

100% PDA, Method 5004 (Note 1)

100% Final Electrical Test

100% Fine/Gross Leak, Method 1014

100% External Visual, Method 2009

Sample - Group A, Method 5005 (Note 2)

Sample - Group B, Method 5005 (Note 3)

Sample - Group C, Method 5005 (Notes 3 and 4)

Sample - Group D, Method 5005 (Notes 3 and 4)

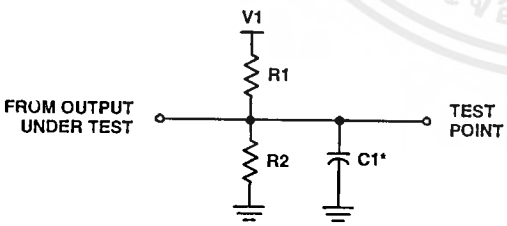
100% Data Package Generation (Note 6)

CSI and/or GSI Final (Note 5)

NOTES:

1. Failures from subgroup 1, 7 are used for calculating PDA. The maximum allowable PDA = 5%.
2. Alternate Group A testing may be performed as allowed by MIL-STD-883, Method 5005.
3. Group B, C and D inspections are optional and will not be performed unless required by the P.O. When required, the P.O. should include separate line items for Group B Test, Group C Test, Group C Samples, Group D Test and Group D Samples.
4. Group C and/or Group D Generic Data, as defined by MIL-I-38535, is optional and will not be supplied unless required by the P.O. When required, the P.O. should include a separate line item for Group C Generic Data and/or Group D Generic Data. Generic data is not guaranteed to be available and is therefore not available in all cases.
5. CSI and/or GSI inspections are optional and will not be performed unless required by theP.O. When required, the P.O. should include separate line items for CSI PreCap inspection, CSI final inspection, GSI PreCap inspection, and/or GSI final inspection.
6. Data Package Contents:
   - Cover Sheet (Harris Name and/or Logo, P.O. Number, Customer Part Number, Lot Date Code, Harris Part Number, Lot Number, Quantity).
   - GAMMA Radiation Report. Contains Cover page, disposition, Rad Dose, Lot Number, Test Package used, Specification Numbers, Test equipment, etc. Radiation Read and Record data on file at Harris.
   - Screening, Electrical, and Group A attributes (Screening attributes begin after package seal).
   - Group B, C and D attributes and/or Generic data is included when required by the P.O.
   - The Certificate of Conformance is a part of the shipping invoice and is not part of the Data Book. The Certificate of Conformance is signed by an authorized Quality Representative.

AC Test Circuit



* Includes stray and jig capacitance

TEST CONDITIONS DEFINITION TABLE

V1	R1	R2	C1
1.7V	523Ω	Open	150pF

AC Testing Input, Output Waveforms

NOTE: AC Testing: All parameters tested as per test circuits. Input rise and fall times are driven at 1V/ns.

Waveforms

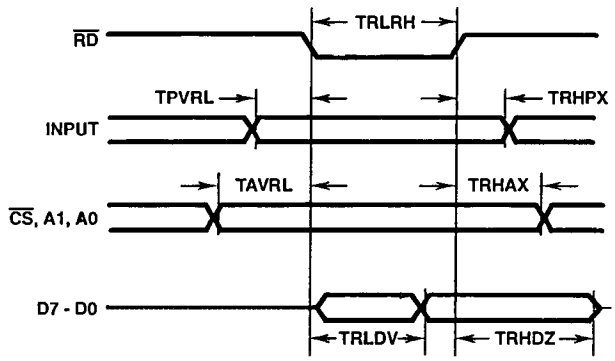


FIGURE 1. MODE 0 (BASIC INPUT)

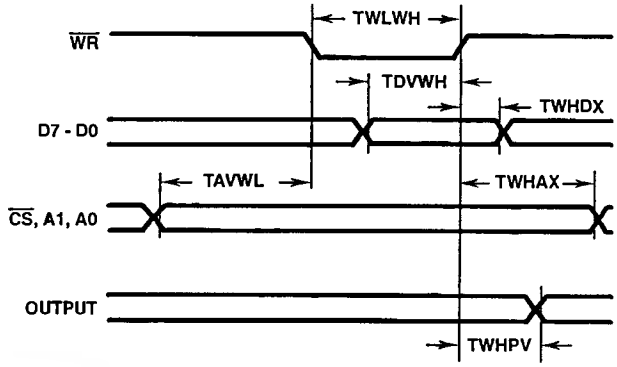


FIGURE 2. MODE 0 (BASIC OUTPUT)

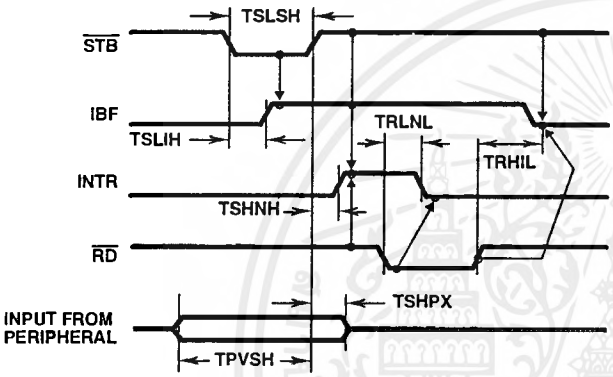


FIGURE 3. MODE 1 (STROBED INPUT)

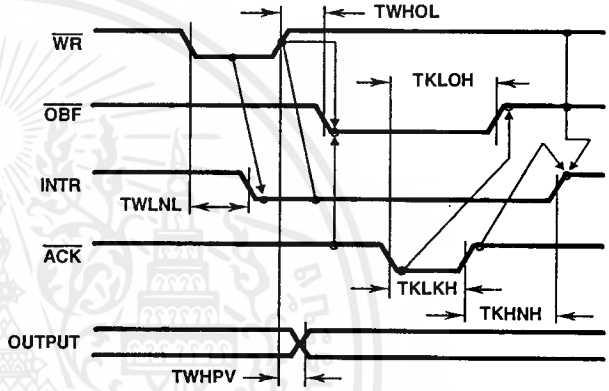


FIGURE 4. MODE 1 (STROBED OUTPUT)

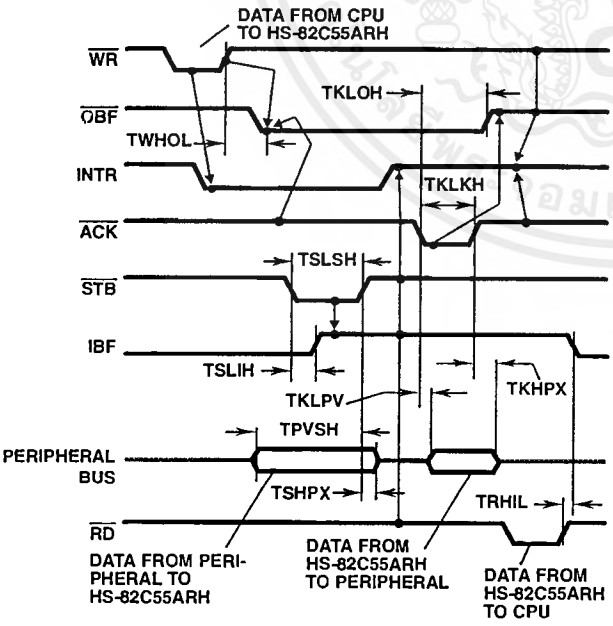


FIGURE 5. MODE 2 (BIDIRECTIONAL)

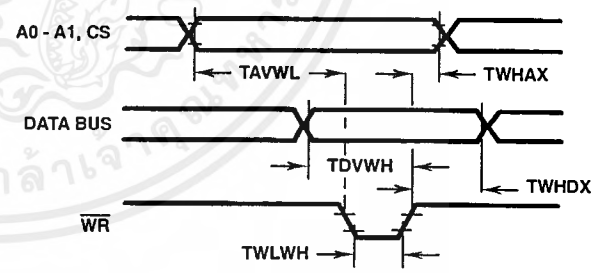


FIGURE 6. WRITE TIMING

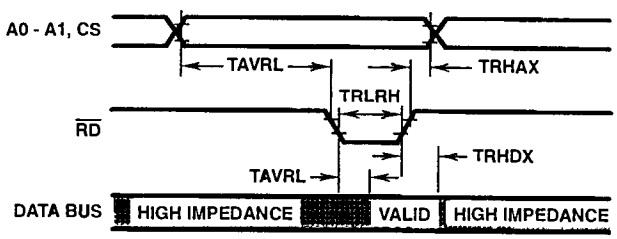
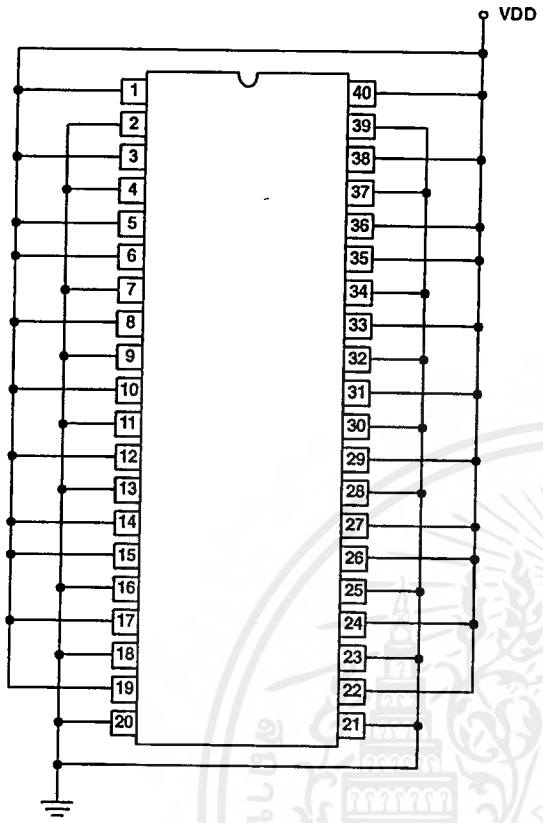


FIGURE 7. READ TIMING

NOTE: Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$ and $\overline{STB}$ occurs before $\overline{RD}$ is permissible.

Burn-In Circuits

PROGRAMMABLE PERIPHERAL INTERFACE

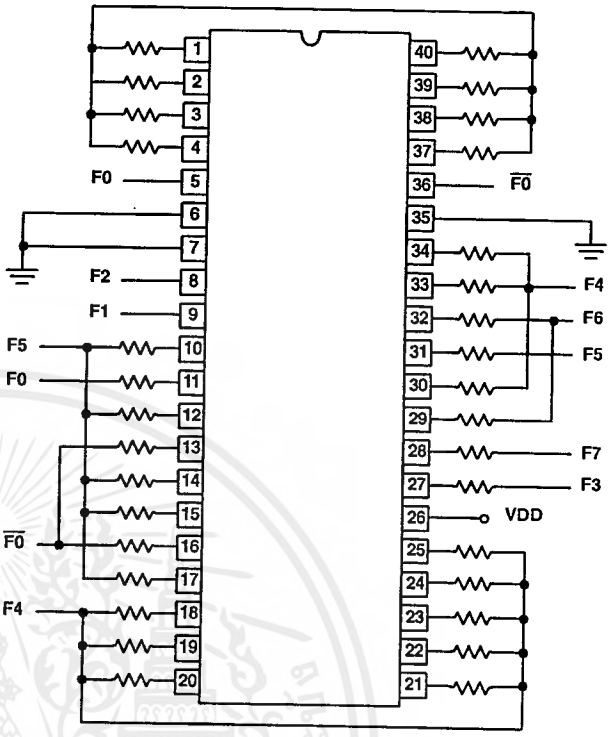


STATIC CONFIGURATION

NOTES:

1. VDD = 6.0V ± 0. 5%
2. IDD <500µA
3. T_A Min = +125°C

PROGRAMMABLE PERIPHERAL INTERFACE



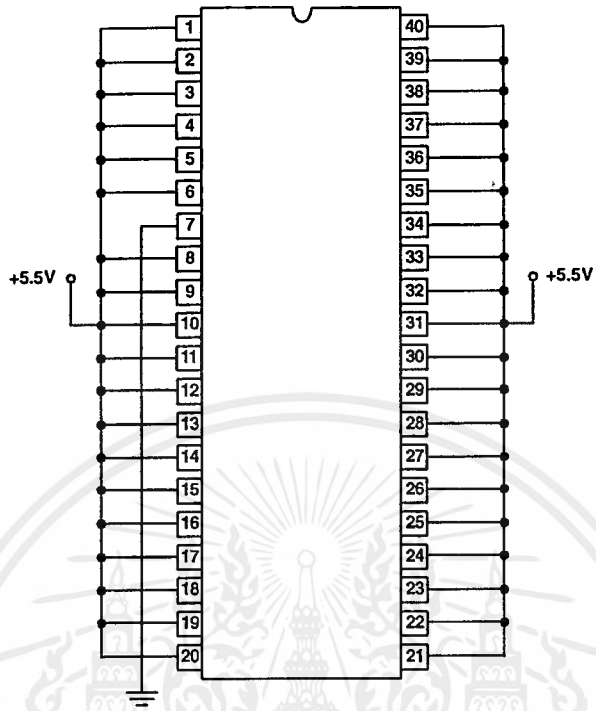
DYNAMIC CONFIGURATION

NOTES:

1. VDD = 6.0V ± 5% for Burn-In
2. VDD = 5.0V ± 5% for Life Test
3. All resistors are 10KΩ ± 5%
4. -0.3V ≤ VIL ≤ 0.8V
5. VDD - 1.0V ≤ VIH ≤ VDD
6. IDD < 5mA
7. F0 = 10KHz, 50% Duty cycle
8. F1 = F0/2; F2 = F1/2; F3 = F2/2; F4 = F3/2 . . . F7 = F6/2
9. T_A Min = +125°C

Irradiation Circuit

CMOS PROGRAMMABLE PERIPHERAL INTERFACE



NOTE:

1. V_{DD} = 5.5V

Functional Description

The HS-82C55ARH is a programmable peripheral interface designed to allow microcomputer systems to control and interface with all types of peripheral devices.It has the ability to generate and respond to all asynchronous handshaking signals necessary to transfer data to and from peripheral devices, and it can also interrupt the processor when a peripheral needs servicing. These capabilities allow the HS-82C55ARH to be used in an unlimited number of applications including EXTERNAL SYSTEM CONTROL, ASYNCHRONOUS DATA TRANSFER, and SYSTEMS MONITORING.

Data Bus Buffer

This tri-state bidirectional 8-bit buffer is used to interface the HS-82C55ARH to the system data bus (see Figure 8). Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

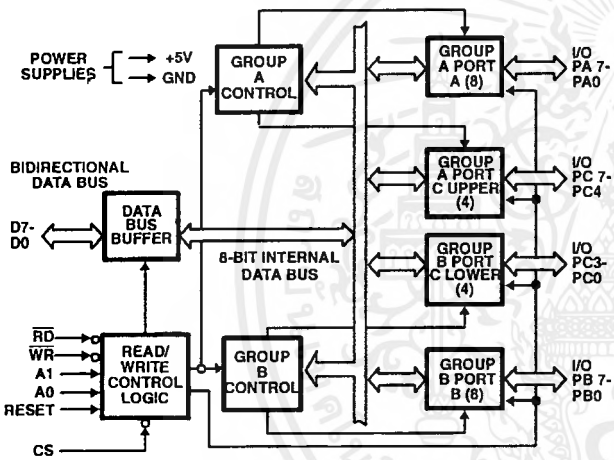


FIGURE 8. BLOCK DIAGRAM DATA BUS BUFFER, READ/WRITE, GROUP A AND B CONTROL LOGIC FUNCTIONS

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfer of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU writes a control word to the HS-82C55ARH. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the HS-82C55ARH.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group - Port A and Port C upper (C7 - C4)

Control Group - Port B and Port C lower (C3 - C0).

Ports A, B, C

The HS-82C55ARH contains three 8-bit ports (A, B and C). All can be configured to a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the HS-82C55ARH.

Port A One 8-bit data output latch/buffer and one 8-bit data input latch. Both "pull-up" and "pull-down" bus hold devices are present on Port A. See Figure 9A.

Port B One 8-bit data input/output latch/buffer and one 8-bit data input buffer. See Figure 9B.

Port C One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and can be used for the control signal outputs and status signal inputs in conjunction with Ports A and B. See Figure 9B.

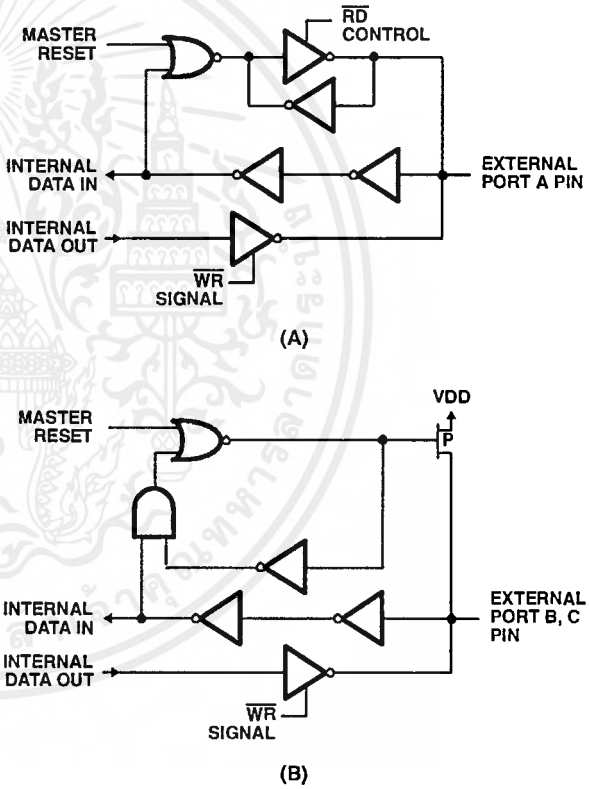


FIGURE 9. I/O PORT CONFIGURATION

Operational Description

Control Word

The data direction and mode of Ports A, B and C are determined by the contents of the Control Word. See Figure 11. The Control Word can be both written and read as shown in Table 1 and 2. During write operations, the function of the Control Word being written is determined by data bit D7. If D7 is low, the data on D0 - D3 will set or reset one of the bits of Port C. See Figure 12. During read Operations, the

Control Word will always be in the format illustrated in Figure 11 with Bit D7 high to indicate Control Word Mode Information.

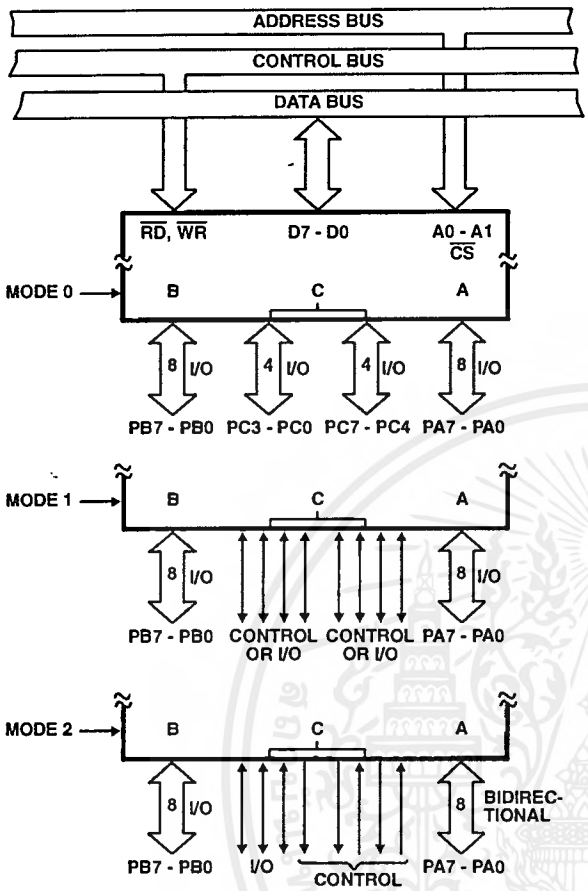


FIGURE 10. BASIC MODE DEFINITIONS & BUS INTERFACE

TABLE 1.

A1	A0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	INPUT OPERATION (READ)
0	0	0	1	0	Port A - Data Bus
0	1	0	1	0	Port B - Data Bus
1	0	0	1	0	Port C - Data Bus
1	1	0	1	0	Control Word - Data Bus

TABLE 2.

A1	A0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	OUTPUT OPERATION (WRITE)
0	0	1	0	0	Data Bus - Port A
0	1	1	0	0	Data Bus - Port B
1	0	1	0	0	Data Bus - Port C
1	1	1	0	0	Data Bus - Control Word

TABLE 3.

A1	A0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	DISABLE FUNCTION
X	X	X	X	1	Data Bus - 3-State
X	X	1	1	0	Data Bus - 3-State

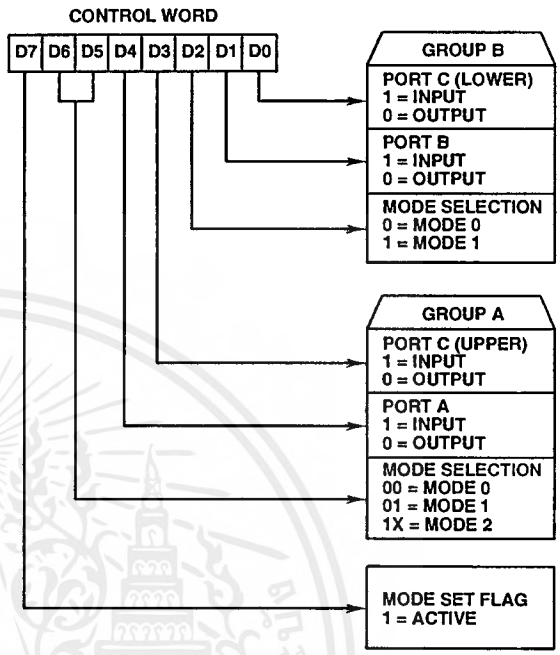


FIGURE 11. MODE SET CONTROL WORD FORMAT

Mode Selection

There are three basic modes of operation that can be selected by the system software:

Mode 0 - Basic Input/Output
Mode 1 - Strobed Input/Output
Mode 2 - Bidirectional Bus

When the RESET input goes "high", all ports will be set to the input mode with all 24 port lines held at the logic "one" level by internal bus hold devices. After reset, the HS-82C55ARH can remain in the input mode with no additional initialization required. This eliminates the need for pullup or pulldown resistors in all CMOS designs. During the execution of the system program, any of the other modes may be selected using a single output instruction. This allows a single HS-82C55ARH to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status register, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance: Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape recorder on an interrupt-driven basis.

The mode definitions and possible mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the HS-82C55ARH has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

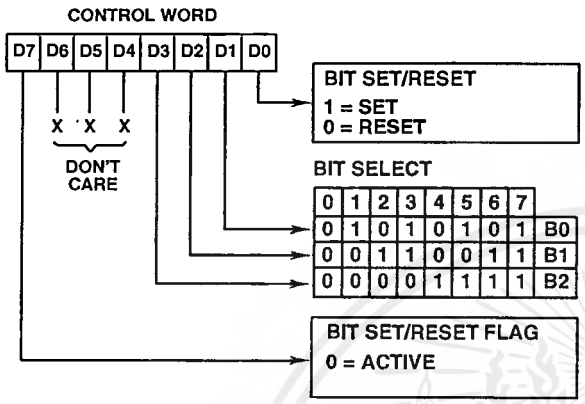


FIGURE 12. BIT SET/RESET CONTROL WORD FORMAT

Single Bit/Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTput instruction. See Figure 12. This feature reduces software requirements in control-based applications.

Interrupt Control Functions

When the HS-82C55ARH is programmed to operate in Mode 1 or Mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from Port C, can be inhibited or enable by setting or resetting the associated INTE flip-flop, using the Bit Set/Reset function of Port C.

This function allows the programmer to enable or disable a CPU interrupt by a specific I/O device without affecting any other device in the interrupt structure.

INTE Flip-Flop Definition:

(BIT-SET) - INTE is SET - Interrupt enable.

(BIT-RESET) - INTE is RESET - Interrupt disable.

NOTE: All mask flip-flops are automatically reset during mode selection and device Reset.

Operating Modes

Mode 0 (Basic Input/Output)

This functional configuration provides simple input and output operations for each of the three ports. No handshaking it required, data is simply written to or read from a specific port.

Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports
- Any port can be input or output
- Outputs are latched
- Inputs are not latched
- 16 different Input/Output configurations possible

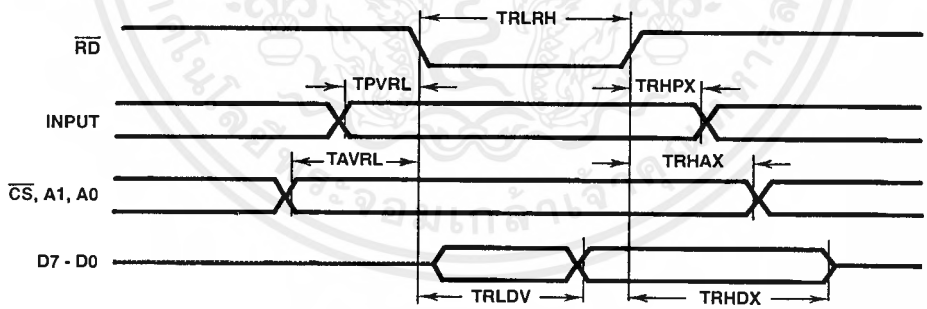


FIGURE 13. MODE 0 (BASIC INPUT)

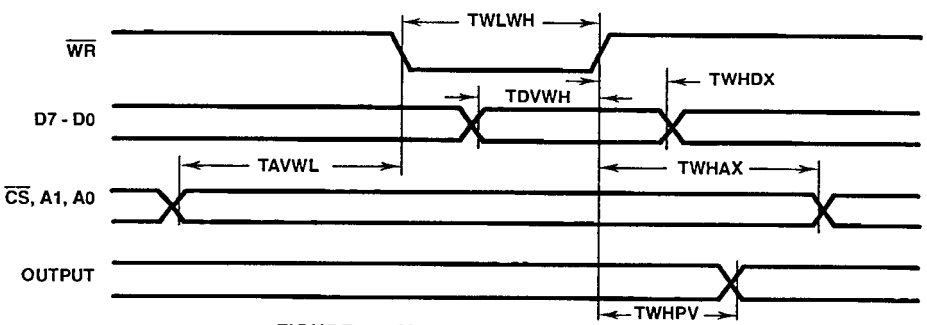


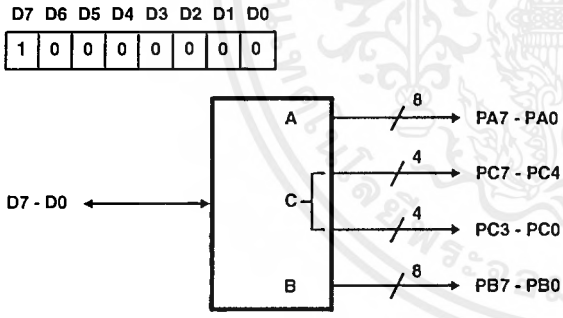
FIGURE 14. MODE 0 (BASIC OUTPUT)

Mode 0 Port Definition

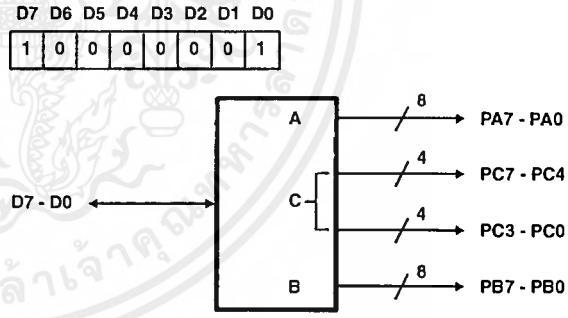
A		B		GROUP A			GROUP B	
D4	D3	D1	D0	PORT A	PORT C (UPPER)	NO.	PORT B	PORT C (LOWER)
0	0	0	0	Output	Output	0	Output	Output
0	0	0	1	Output	Output	1	Output	Input
0	0	1	0	Output	Output	2	Input	Output
0	0	1	1	Output	Output	3	Input	Input
0	1	0	0	Output	Input	4	Output	Output
0	1	0	1	Output	Input	5	Output	Input
0	1	1	0	Output	Input	6	Input	Output
0	1	1	1	Output	Input	7	Input	Input
1	0	0	0	Input	Output	8	Output	Output
1	0	0	1	Input	Output	9	Output	Input
1	0	1	0	Input	Output	10	Input	Output
1	0	1	1	Input	Output	11	Input	Input
1	1	0	0	Input	Input	12	Output	Output
1	1	0	1	Input	Input	13	Output	Input
1	1	1	0	Input	Input	14	Input	Output
1	1	1	1	Input	Input	15	Input	Input

Mode 0 Configurations

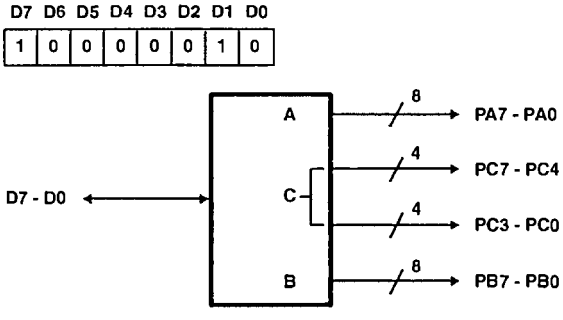
CONTROL WORD #0



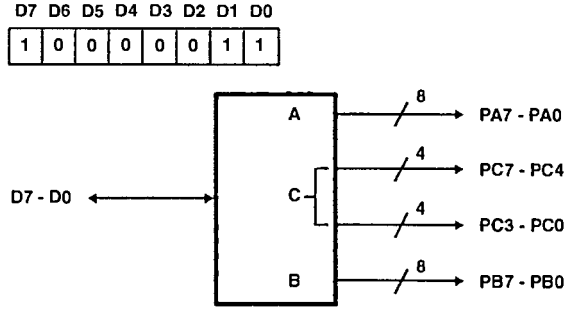
CONTROL WORD #1



CONTROL WORD #2



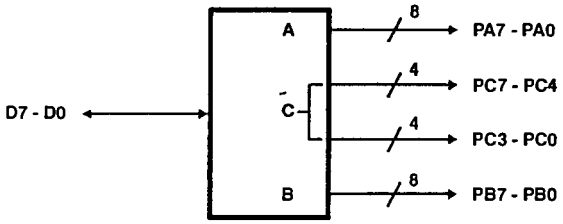
CONTROL WORD #3



Mode 0 Configurations (Continued)

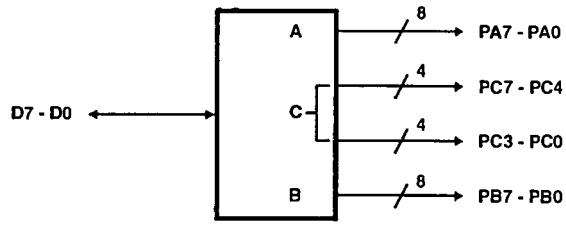
CONTROL WORD #4

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	1	0	0	0



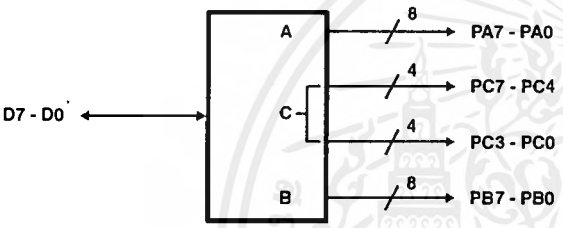
CONTROL WORD #5

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	1	0	0	1



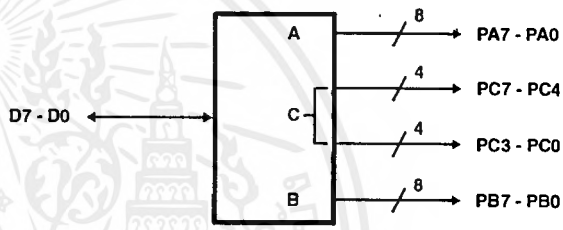
CONTROL WORD #6

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	1	0	1	0



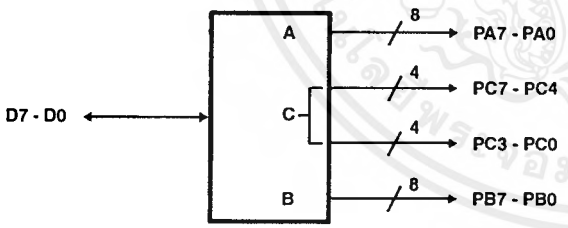
CONTROL WORD #7

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	1	0	1	1



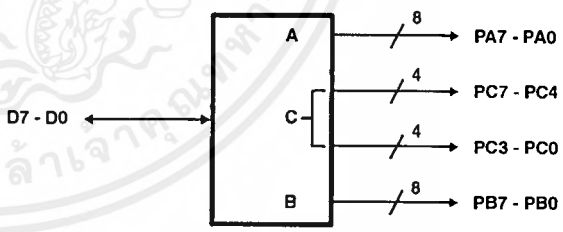
CONTROL WORD #8

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	0	0	0	0



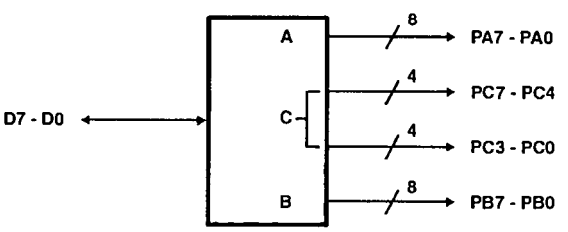
CONTROL WORD #9

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	0	0	0	1



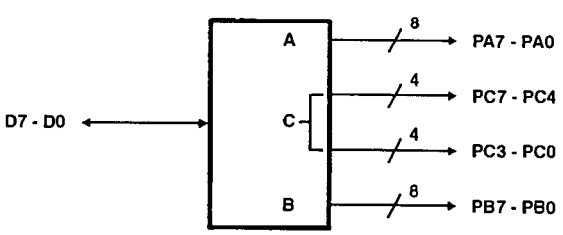
CONTROL WORD #10

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	0	0	1	0



CONTROL WORD #11

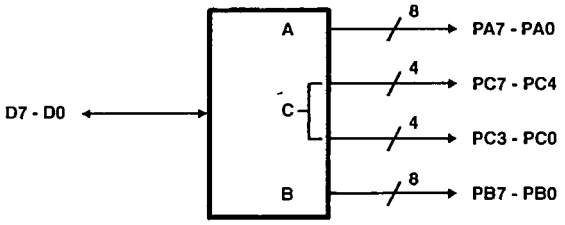
D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	0	0	1	1



Mode 0 Configurations (Continued)

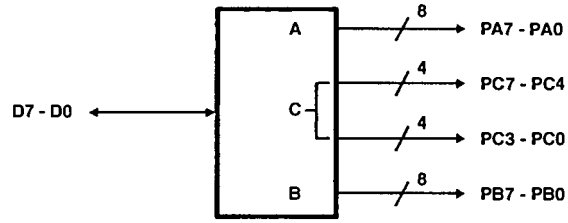
CONTROL WORD #12

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	1	0	0	0



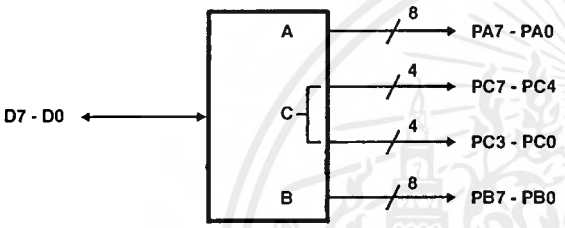
CONTROL WORD #13

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	1	0	0	1



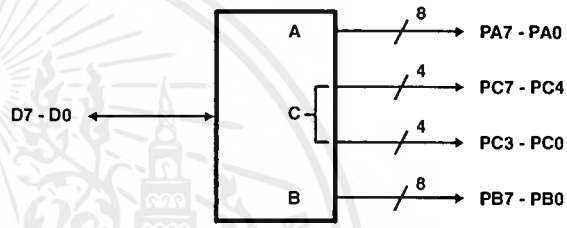
CONTROL WORD #14

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	1	0	1	0



CONTROL WORD #15

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	1	0	1	1



Operating Modes

Mode 1 (Strobed Input/Output)

This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "handshaking" signals. In Mode 1, Port A and Port B use the lines on Port C to generate or accept these "handshaking" signals.

Mode 1 Basic Functional Definitions:

- Two Groups (Group A and Group B)
- Each group contains one 8-bit port and one 4-bit control/data port.
- The 8-bit data port can be either input or output. Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit port.

Input Control Signal Definition

$\overline{STB}$ (Strobe Input)

A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F)

A "high" on this output indicates that the data has been loaded into the input latch; in essence, an acknowledgment. IBF is set by $\overline{STB}$ input being low and is reset by the rising edge of the $\overline{RD}$ input.

INTR (Interrupt Request)

A "high" on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the rising edge of $\overline{STB}$ and reset by the falling edge of $\overline{RD}$. This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

INTE A

Controlled by Bit Set/Reset of PC4.

INTE B

Controlled by Bit Set/Reset of PC2.

MODE 1 (PORT A) CONTROL WORD

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	1	1/0	X	X	X

PC6, 7
1 = INPUT
0 = OUTPUT

MODE 1 (PORT B) CONTROL WORD

D7	D6	D5	D4	D3	D2	D1	D0
1	X	X	X	X	1	1	X

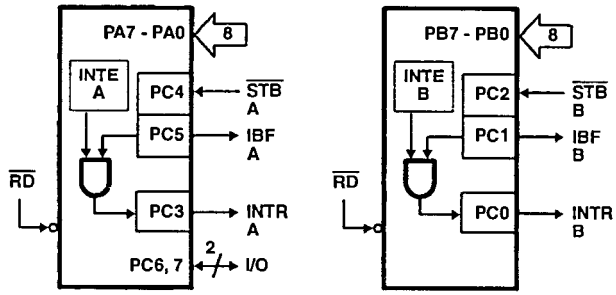


FIGURE 15. MODE 1 INPUT

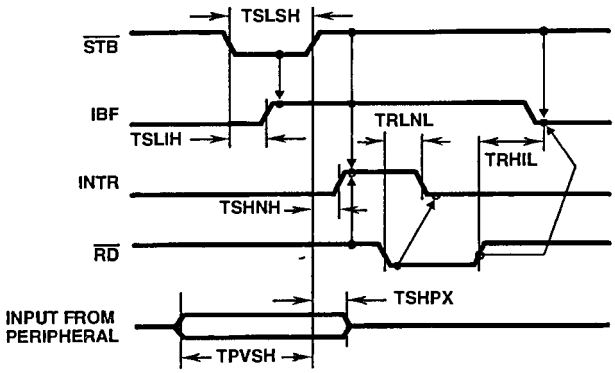


FIGURE 16. MODE 1 (STROBED INPUT)

Output Control Signal Definition

OBF (Output Buffer Full F/F)

The OBF output will go "low" to indicate that the CPU has written data out to the specified port. This does not mean valid data is sent out of the port at this time since OBF can go true before data is available. Data is guaranteed valid at the rising edge of OBF. See Note 1. The OBF F/F will be set by the rising edge of the WR input and reset by ACK input being low.

ACK (Acknowledge Input)

A "low" on this input informs the HS-82C55ARH that the data from Port A or Port B is ready to be accepted. In essence, a response from the peripheral device indicating that it is ready tc accept data. See Note 1.

INTR (Interrupt Request)

A "high" on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU. INTR is set by the rising edge of ACK and reset by the falling edge of WR.

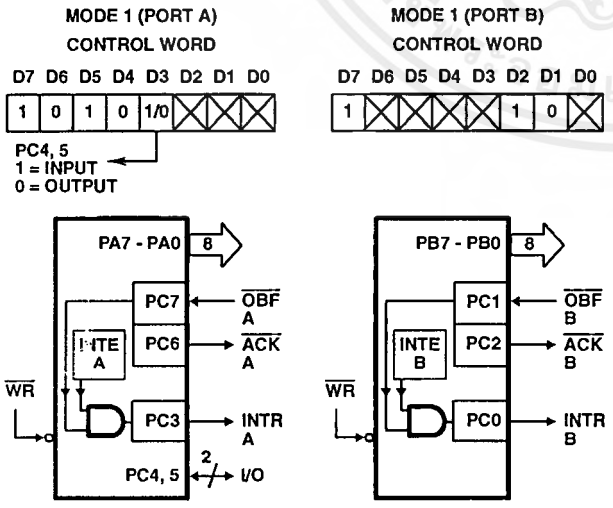


FIGURE 17. MODE 1 OUTPUT

INTE A

Controlled by Bit Set/Reset of PC6.

INTE B

Controlled by Bit Set/Reset of PC2.

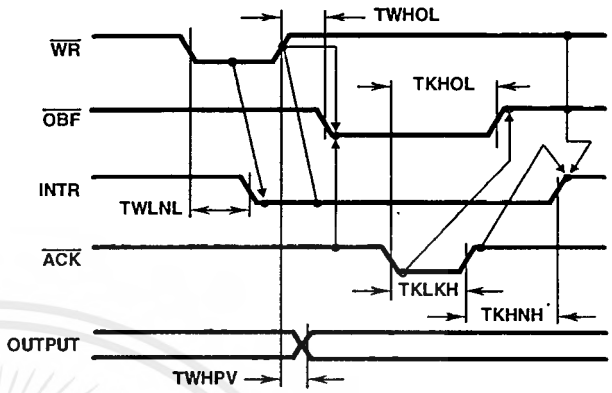


FIGURE 18. MODE 1 (STROBED OUTPUT)

NOTE:

1. To strobe data into the peripheral device, the user must operate the strobe line in a hand shaking mode. The user needs to send $\overline{OBF}$ to the peripheral device, generate an $\overline{ACK}$ from the peripheral device and then latch data into the peripheral device on the rising edge of $\overline{OBF}$.

Combinations of Mode 1: Port A and Port B can be individually defined as input or output in Mode 1 to support a wide variety of strobed I/O applications.

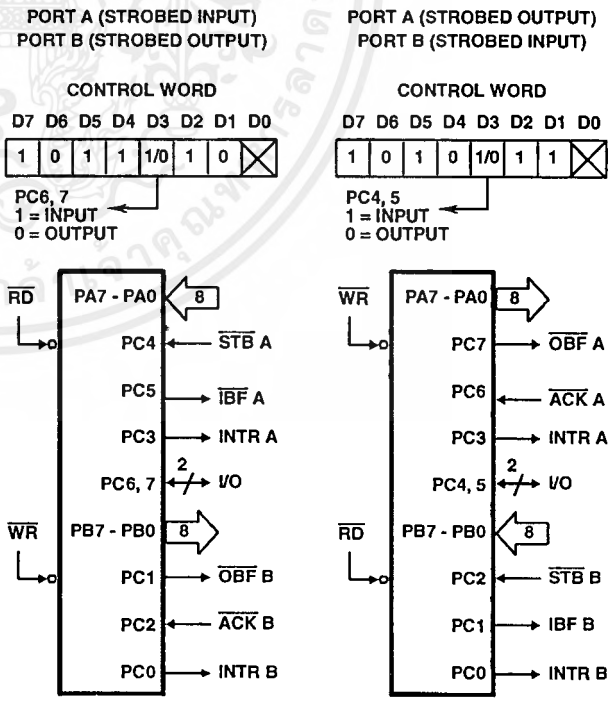


FIGURE 19. COMBINATIONS OF MODE 1

Operating Modes

MODE 2 (Strobed Bidirectional Bus I/O)

The functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O). "Handshaking" signals are provided to maintain proper bus flow discipline similar to MODE 1. Interrupt generation and enable/disable functions are also available.

Mode 2 Basic Functional Definitions:

- Used in Group A only.
- One 8-bit, bidirectional bus port (Port A) and a 5-bit control port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bidirectional bus port (Port A).

Bidirectional Bus I/O Control Signal Definition

INTR (Interrupt Request)

A high on this output can be used to interrupt the CPU for both input or output operations. INTR will be set either by the rising edge of $\overline{ACK}$ (INTE1 = 1) or the rising edge of $\overline{STB}$ (INTE2 = 1). INTR will be reset by the falling edge of $\overline{WR}$ (if previously set by the rising edge or $\overline{ACK}$), the falling edge of $\overline{RD}$ (if previously set by the rising edge of $\overline{STB}$), or the falling edge of $\overline{WR}$ when immediately following a low $\overline{RD}$ pulse or the falling edge of $\overline{RD}$ when immediately following a low $\overline{WR}$ pulse (if previously set by the rising edges of both $\overline{ACK}$ and $\overline{STB}$).

Output Operations

$\overline{OBF}$ (Output Buffer Full)

The $\overline{OBF}$ output will go "low" to indicate that the CPU has written data out to Port A.

$\overline{ACK}$ (Acknowledge)

A "low" on this input enables the tri-state output buffer of Port A to send out the data. Otherwise, the output buffer will be in the high impedance state.

INTE 1 (The INTE Flip-Flop Associated with $\overline{OBF}$)

Controlled by Bit Set/Reset of PC6.

Input Operations

$\overline{STB}$ (Strobe Input)

A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F)

A "high" on this output indicates that data has been loaded into the input latch.

INTE 2 (The INTE Flip-Flop Associated with IBF)

Controlled by Bit Set/Reset of PC4.

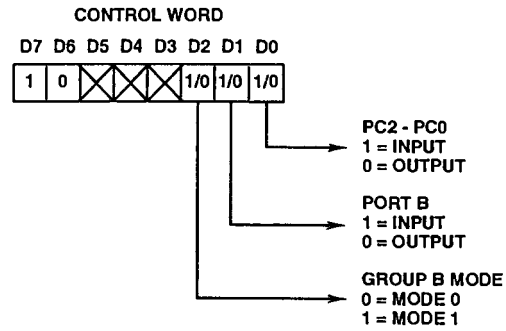


FIGURE 20. MODE CONTROL WORD

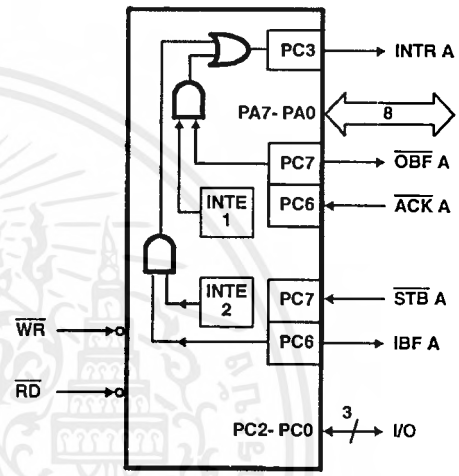
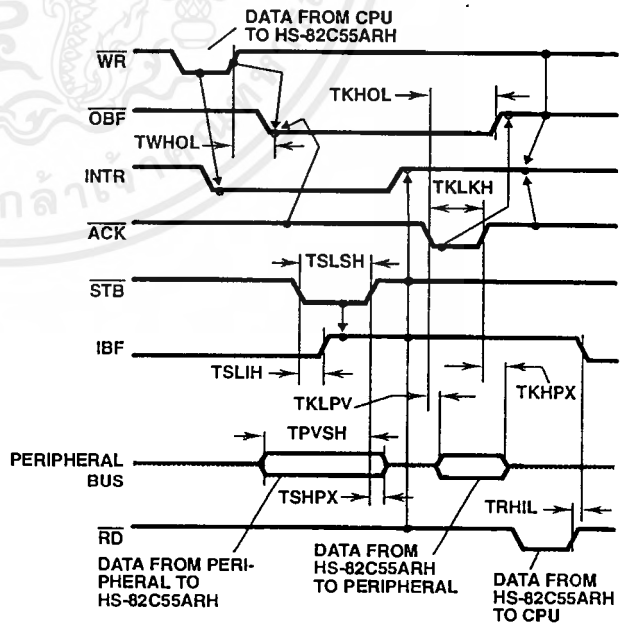


FIGURE 21. MODE 2 (BIDIRECTIONAL)



NOTE: Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$ and $\overline{STB}$ occurs before $\overline{RD}$ is permissible.

FIGURE 22. MODE 2 (BIDIRECTIONAL)

MODE DEFINITION SUMMARY

	MODE 0		MODE 1		MODE 2	
	IN	OUT	IN	OUT	GROUP A ONLY	
PA0	In	Out	In	Out	⟷	
AP1	In	Out	In	Out	⟷	
PA2	In	Out	In	Out	⟷	
PA3	In	Out	In	Out	⟷	
PA4	In	Out	In	Out	⟷	
PA5	In	Out	In	Out	⟷	
PA6	In	Out	In	Out	⟷	
PA7	In	Out	In	Out	⟷	
PB0	In	Out	In	Out	-	Mode 0 or Mode 1 Only
PB1	In	Out	In	Out	-	
PB2	In	Out	In	Out	-	
PB3	In	Out	In	Out	-	
PB4	In	Out	In	Out	-	
PB5	In	Out	In	Out	-	
PB6	In	Out	In	Out	-	
PB7	In	Out	In	Out	-	
PC0	In	Out	INTR B	INTR B	I/O	
PC1	In	Out	IBF B	$\overline{OBF}$ B	I/O	
PC2	In	Out	$\overline{STB}$ B	$\overline{ACK}$ B	I/O	
PC3	In	Out	INTR A	INTR A	INTR A	
PC4	In	Out	$\overline{STB}$ A	I/O	$\overline{STB}$ A	
PC5	In	Out	IBF A	I/O	IBF A	
PC6	In	Out	I/O	$\overline{ACK}$ A	$\overline{ACK}$ A	
PC7	In	Out	I/O	$\overline{OBF}$ A	$\overline{OBF}$ A	

Special Mode Combination Considerations

There are several combinations of modes possible. For any combination, some or all of Port C lines are used for control or status. The remaining bits are either inputs or outputs as defined by a "Set Mode" command.

During a read of Port C, the state of all the Port C lines, except the $\overline{ACK}$ and $\overline{STB}$ lines, will be placed on the data bus. In place of the ACK and STB line states, flag status will appear on the data bus in the PC2, PC4, and PC6 bit positions as illustrated by Figure 25.

Through a "Write Port C" command, only the Port C pins programmed as outputs in a Mode 0 group can be written. No other pins can be affected by a "Write Port C" command, nor can the interrupt enable flags be accessed. To write to any Port C output programmed as an output in a Mode 1 group or to change an interrupt enable flag, the "Set/Reset Port C Bit" command must be used.

With a "Set/Reset Port C Bit" command, any Port C line programmed as an output (including IBF and $\overline{OBF}$) can be written, or an interrupt enable flag can be either set or reset. Port C lines programmed as inputs, including $\overline{ACK}$ and $\overline{STB}$ lines, associated with Port C fare not affected by a "Set/Reset Port C Bit" command. Writing to the corresponding Port C bit positions of the $\overline{ACK}$ and $\overline{STB}$ lines with the "Set/Reset Port C Bit" command will affect the Group A and Group B interrupt enable flags, as illustrated in Figure 25.

INPUT CONFIGURATION

D7	D6	D5	D4	D3	D2	D1	D0
I/O	I/O	IBFA	INTEA	INTRA	INTEB	IBFB	INTRB
GROUP A					GROUP B		

OUTPUT CONFIGURATION

D7	D6	D5	D4	D3	D2	D1	D0
$\overline{OBFA}$	INTEA	I/O	I/O	INTRA	INTEB	$\overline{OBFB}$	INTRB
GROUP A					GROUP B		

FIGURE 23. MODE 1 STATUS WORD FORMAT

D7	D6	D5	D4	D3	D2	D1	D0
$\overline{OBFA}$	INTE1	IBFA	INTE2	INTRA	X	X	X
GROUP A					GROUP B		

NOTE: (Defined by Mode 0 or Mode 1 Selection)

FIGURE 24. MODE 2 STATUS WORD FORMAT

Current Drive Capability

Any output on Port A, B or C can sink or source 2.5mA. This feature allows the 82C55A to directly drive Darlington type drivers and high-voltage displays that require such sink or source current.

Reading Port C Status (Figures 23 and 24)

In Mode 0, Port C transfers data to or from the peripheral device. When the 82C55A is programmed to function in Modes 1 or 2, Port C generates or accepts "hand shaking" signals with the peripheral device. Reading the contents of Port C allows the programmer to test or verify the "status" of each peripheral device and change the program flow accordingly.

There is no special instruction to read the status information from Port C. A normal read operation of Port C is executed to perform this function.

INTERRUPT ENABLE FLAG*	POSITION	ALTERNATE PORT C PIN SIGNAL (MODE)
INTE B	PC2	$\overline{ACK}$B (Output Mode 1) or STBB (Input Mode 1)
INTE A2	PC4	$\overline{STB}$A (Input Mode 1 or Mode 2)
INTE A1	PC6	$\overline{ACK}$A (Output Mode 1 or Mode 2)

FIGURE 25. INTERRUPT ENABLE FLAGS IN MODES 1 AND 2

Metallization Topology

DIE DIMENSIONS:
3420µm x 4350µm x 485µm ± 25µm

METALLIZATION:
Type: Al/Si
Thickness: 11kÅ ± 2kÅ

GLASSIVATION:
Type: SiO2
Thickness: 8kÅ ± 1kÅ

WORST CASE CURRENT DENSITY:
7.7×10^4 A/cm^2

Metallization Mask Layout

HS-82C55ARH