การพัฒนาระบบควบคุมของยานขนส่งแบบนำร่องอัตโนมัติ

A DEVELOPMENT OF CONTROL SYSTEM FOR AUTOMATED GUIDED VEHICLE

นายอนุตรชัย ณ ถลาง

MR. ANUTARACHAI NATHALANG

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์มหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ 2535

ISBN 974-8158-22-5

A DEVELOPMENT OF CONTROL SYSTEM FOR AUTOMATED GUIDED VEHICLE

MR. ANUTARACHAI NATHALANG

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING

GRADUATE SCHOOL

KING MONGKUT 'S INSTITUTE OF TECHNOLOGY LADKRABANG

1992

ISBN 974-8158-22-5

หัวข้อวิทยานิพนธ์	การพัฒนาระบบควบคุมของยานขนส่งแบบนำร่องอัตโนมัติ
นักศึกษา	นายอนุตรชัย ณ ถลาง
อาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ.ดร.โยธิน เปรมปราณีรัชต์
ระดับการศึกษา	วิศวกรรมศาสตร์มหาบัณฑิต สาขาวิศวกรรมไฟฟ้า
ภาควิชา	วิศวกรรมระบบควบคุม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ	2535

บทคัดย่อ

วิทยานิพนธ์นี้เป็นการออกแบบและสร้างระบบควบคุม เพื่อนำไปประยุกต์ใช้กับยานขนส่งแบบนำร่องอัตโนมัติหรือ AGV ระบบควบคุมที่สร้างขึ้นมีหลายส่วน ส่วนแรกเป็นการพัฒนาการสื่อสารข้อมูลระหว่าง AGV กับคอมพิวเตอร์ส่วนกลางทำให้สามารถควบคุมการทำงานของ AGV ตลอดจนรับทราบสภาวะต่างๆ ที่สำคัญของ AGV จากคอมพิวเตอร์ส่วนกลางได้โดยตรง การสื่อสารข้อมูลระหว่างคอมพิวเตอร์กับ AGV ใช้วิธีส่งข้อมูลอนุกรมผ่านคลื่นวิทยุย่านความถี่ 27 MHz ในส่วนที่สองเป็นการพัฒนาระบบการเรียกใช้งาน AGV จากสถานีปฏิบัติงานซึ่งอยู่ในระยะไกล การเรียกจะใช้วิธีกดคีย์โดยสัญญาณเรียกจะถูกส่งมาตามสายเข้าสู่คอมพิวเตอร์ คอมพิวเตอร์จะทำการแปลรหัสสถานีแล้วเก็บไว้ในหน่วยความจำ นอกจากนี้ยังได้พัฒนาในส่วนอื่นอันเป็นส่วนสำคัญของ AGV ทั้งสิ้นอาทิ เช่น การควบคุมตำแหน่งและความเร็วมอเตอร์กระแสตรงโดยไมโครคอนโทรลเลอร์, เทคนิคการนำร่องโดยใช้โฟโต้เซนเซอร์, การตรวจจับวัตถุกีดขวางด้วยอุตร้าโซนิคและการตรวจวัดแรงดันแบตเตอรี่ เป็นต้น งานพัฒนาทั้งหมดนี้จะเป็นรากฐานในการสร้าง AGV ต่อไป

Thesis Title	A Development of Control system for Automated Guided Vehicle
Student	Mr.Anutarachai Nathalang
Thesis Advisor	Assoc.Prof.DR.Yothin Prempraneerach
Level of Study	Master of Engineering in Electrical Engineering
Department	Control Engineering King Mongkut's Institute of Technology Ladgrabang
Year	1992

ABSTRACT

This thesis presents the design and implementation of control system for Automated Guided Vehicle, AGV. The implemented system has many parts. The first is development of data communication between AGV and central computer which enable to control the operation and recognize the status of AGV diretly from supevisory computer. Data communication between computer and AGV is acheived by sending serial data through radio wave at 27 MHz band. The second part is development of AGV calling system. Calling is acheived by pressing a key at remote workstation which signal sending along the wire to the computer. Computer will decode the station code and store it in memory. Moreover , the other parts of AGV are developed such as position and speed control of DC motor by microcontroller , guidance technique using photo sensors, obstacle detection by ultrasonic senser and battery monitoring. These development are the basis to implement AGV in the next state.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยดี ก็เพราะได้รับความเมตตาจากอาจารย์โยธิน เปรมปรานีรัชต์ เป็นอย่างมาก รวมทั้งการให้แนวทางตลอดจนข้อแนะนำต่างๆ ผู้เขียนต้องขอกราบขอบพระคุณอาจารย์เป็นอย่างสูงมา ณ. โอกาสนี้ด้วย

ขอขอบคุณ คุณสุมิตร พนาอุดมทรัพย์ ที่ได้ช่วยแนะนำด้านเทคนิคในจุดที่สำคัญหลายจุด ขอบคุณ คุณมนตรี พรรณรัตน์ ที่ให้การสนับสนุนทางด้านอุปกรณ์หลายชิ้น ขอบคุณ คณชาวดี พิสัยรัตน์ ที่ให้การช่วยเหลือด้านอุปกรณ์การวัด รวมทั้งเพื่อนๆอีกหลายคนที่ไม่ได้กล่าวถึงในที่นี้ซึ่งมีน้ำใจช่วยเหลือผมในเรื่องต่างๆ จนวิทยานิพนธ์นี้สำเร็จลุล่วงไปได้ด้วยดี

นายอนุตรชัย ณ ถลาง

สารบัญ

หน้า

หน้า

สารบัญรูป

สารบัญตาราง

คำอธิบายสัญญลักษณ์/คำย่อ

a	ระยะห่างระหว่างล้อแต่ละข้างบนเพลาล้อเดียวกัน
AGV	ยานขนส่งแบบนำร่องอัตโนมัติ (Automated Guided Vehicle)
b	ระยะฐานล้อของรถยนต์
c	ระยะห่างระหว่างจุดศูนย์กลางของสลักทั้งสองข้าง
C	ตัวแปรเอาท์พุท (output variable)
CB	แถบคลื่นวิทยุสำหรับประชาชน (Citizen Band)
CCD	Charge-Coupled Devices
CNC	การควบคุมเชิงเลขด้วยคอมพิวเตอร์ (Computer Numerical Control)
d	ความยาวของคันชักคันส่ง
DNC	การควบคุมเชิงเลขโดยตรง (Direct Numerical Control)
e	สัญญาณเออเรอร์ (error signal)
FMS	ระบบโรงงานแบบยืดหยุ่น (Flexible Manufacturing System)
$i_a(t)$	กระแสอาร์เมเจอร์ (armature current)
J	ผลรวมของโมเมนต์แรงเฉื่อย (total moment of inertia) ของมอเตอร์และโหลด
K_{a_1}	กำลังขยายของวงจรสวิทชิ่งเซอร์โวแอมป์
K_E	ค่าคงที่ของโวลท์เตจ (voltage constant)
K_a	ค่าคงที่ของทาโคมิเตอร์ (tachometer gain)
K_p	กำลังขยายของตัวควบคุมแบบปรีอปพอร์ชั่นนัล (proportional gain)
K_T	ค่าคงที่แรงบิด (torque constant)
L_a	ความเหนี่ยวนำ (inductance) ของขดอาร์เมเจอร์
m	สัญญาณควบคุม (control signal)
NC	ระบบควบคุมเชิงตัวเลข (numerical control)
PI	ปรีอบพอชั่นแนล-อินติกรัล (Proportional-Integral)
PLC	ตัวควบคุมลอจิกแบบโปรแกรมได้ (programmable logic controller)
q	ค่าผิดพลาดทางตำแหน่งระหว่างตำแหน่งเป้าหมายกับตำแหน่งปัจจุบันมีหน่วยเป็นพัลซ์
R	สัญญาณอ้างอิง (reference signal)
R_a	ความต้านทาน (resistance) ของขดอาร์เมเจอร์
RFI	การรบกวนความถี่วิทยุ (Radio Frequency Interference)

คำอธิบายสัญญลักษณ์/คำย่อ (ต่อ)

Rif	รัศมีวงเลี้ยวของล้อหน้าด้านใน
Rof	รัศมีวงเลี้ยวของล้อหน้าด้านนอก
Rir	รัศมีวงเลี้ยวของล้อหลังด้านใน
Ror	รัศมีวงเลี้ยวของล้อหลังด้านนอก
SSB	การสื่อสารระบบแถบข้างเดียว (Single Side Band)
T_F	แรงบิดเสียดทานภายใน (internal friction torque)
T_G	แรงบิดกำเนิดจากมอเตอร์ (generated torque)
Ti	อินติกรัลไทม์ (integral time)
T_L	แรงบิดโหลด (load torque)
v(t)	โวลท์เตจที่ป้อนให้มอเตอร์
$v_G(t)$	เอาท์พุทโวลท์เตจของทาโคมิเตอร์
x	ระยะจากจุดศูนย์กลางของสลักด้านในโค้งถึงจุดหมุน
β	สัมประสิทธิ์ของวิสคอสแด้มพ์ปิ้ง (viscous damping coefficient)
γ	อัตราลดลงของความเร็ว
θ	มุมล็อคด้านใน (inside lock angle)
$\theta(t)$	ตำแหน่งเชิงมุมของโรเตอร์
Φ	มุมล็อคด้านนอก (outside lock)
$\omega(t)$	ความเร็วเชิงมุมของมอเตอร์
ω_c	ค่าคงที่ของความเร็ว

บทที่ 1
บทนำ

1.1 ความเป็นมาของ AGV

ปัจจุบันอุตสาหกรรมประเภทต่างๆมีการแข่งขันอย่างเสรี ทำให้ผู้ผลิตต้องมีการพัฒนาระบบการผลิตในหลายจุด การจัดการวัสดุ (material handling) ก็เป็นจุดหนึ่งที่ผู้ผลิตต้องพัฒนาให้ดีขึ้นทั้งนี้เพื่อช่วยให้ต้นทุนการผลิตลดลง ตลอดจนพัฒนาระบบการผลิตให้มีคุณภาพและมีประสิทธิภาพสูงยิ่งขึ้น

ระบบจัดการวัสดุนั้นประกอบด้วยระบบการขนส่งวัสดุ (transport system), ระบบการเก็บวัสดุ (warehouse system) และการควบคุมระบบ (system control) การพัฒนาระบบจำเป็นต้องมีการนำเอาระบบอัตโนมัติต่างๆ มาใช้ อาทิเช่น PLC, CNC, DNC, หุ่นยนต์, AGV, สายพานลำเลียง (conveyor) , เครนอัตโนมัติ (automated crane) เป็นต้น

ยานขนส่งแบบนำร่องอัตโนมัติ (Automatic Guided Vehicle) หรือ AGV จัดเป็นหุ่นยนต์เคลื่อนที่ (mobile robot) ชนิดหนึ่งซึ่งมีใช้อย่างแพร่หลายในอุตสาหกรรมมากว่า 30 ปี AGV เป็นยานพาหนะที่ไม่ต้องใช้คนขับที่สามารถวิ่งไปตามเส้นทาง (route) ที่กำหนดได้เองโดยอัตโนมัติ AGV สามารถใช้แทนยานพาหนะอาทิเช่น รถโฟร์คลิฟท์(fork-lift) ,รถบรรทุก ,รถแทรกเตอร์ เป็นต้น หรืออาจใช้แทนระบบสายพานลำเลียง AGV มีหน้าที่ขนส่งวัตถุดิบ,ชิ้นส่วน,ผลิตภัณฑ์หรือสินค้าคงคลังจากสถานที่หนึ่งไปยังอีกสถานที่หนึ่ง AGV มีใช้มากในโรงงานแบบอัตโนมัติ (automated factory)

AGV มักใช้ในระบบโรงงานแบบยืดหยุ่น (Flexible Manufacturing System) หรือ FMS เพื่อป้อนงานแก่เครื่องจักรประเภท CNC , DNC AGV เป็นส่วนหนึ่งในระบบ AMDC (Automated Materials Distribution Center) หรือในโรงเก็บวัสดุ ปัจจุบันนี้เนื่องจากระบบอัตโนมัติเข้ามามีบทบาทในโรงงานอย่างแพร่หลายจึงมักจะพบว่ามีการใช้ AGV อยู่มาก AGV จึงกลายเป็นองค์ประกอบหนึ่งที่สำคัญในระบบโรงงาน FMS การขนส่งวัสดุมีตั้งแต่การนำวัสดุจากโรงเก็บของไปยังพื้นที่ปฏิบัติงาน (work areas) หรือขนส่งระหว่างพื้นที่ปฏิบัติงานจากจุดหนึ่งไปยังอีกจุดหนึ่ง รวมไปถึงการนำผลผลิตไปเก็บยังโกดังเก็บสินค้า

ในโรงงานที่ใช้ระบบอัตโนมัติทั้งหมดนั้น ระบบขนส่งจะถูกควบคุมการทำงานจากคอมพิวเตอร์ส่วนกลาง (central computer) โดยต้องสามารถขนส่งวัสดุ ตามคำสั่งลักษณะต่างๆ ไปยังตำแหน่งใดๆก็ได้ ไม่ว่าจะเป็นหน่วยเครื่องจักร (machining cell) , สถานีประกอบชิ้นส่วน (assembly station) หรือโรงเก็บของในโรงงานประเภทประกอบชิ้นส่วน ระบบขนส่ง อาทิเช่น AGV, สายพานลำเลียงนั้นถือได้ว่าเครื่องมือจัดการขั้นปฐมภูมิ (primary handling device) สำหรับ

นำชิ้นงานจากโรงเก็บของไปยังอุปกรณ์จัดการขั้นทุติยภูมิ (secondary handling device) เช่น แขนหุ่นยนต์ เป็นต้น เพื่อให้แขนหุ่นยนต์หยิบขึ้นงานป้อนเครื่องจักรไปต่อไป AGV ในปัจจุบันได้รับการออกแบบเพื่อให้สามารถทำงานภายใต้การควบคุมจากคอมพิวเตอร์ส่วนกลาง โดยสามารถโปรแกรมให้ AGV วิ่งไปตามเส้นทางและหยุดยังตำแหน่งเป้าหมายได้อย่างถูกต้องแม่นยำ ในปัจจุบัน AGV ส่วนใหญ่จะถูกกำหนดให้เคลื่อนที่ไปตามทางเดินนำร่อง (guide path) ซึ่งติดตั้งอยู่ตามพื้นโรงงาน ทางเดินนำร่องที่นิยมใช้อาทิเช่น การฝังสายตัวนำใต้พื้น,การใช้แถบสีติดบนพื้น เป็นต้น โครงสร้างภายนอกสำหรับ AGV ส่วนมากจะทำเป็นแท่นไว้วางภาชนะที่ใส่วัสดุอาทิเช่น ถาด (trays) , กล่อง (boxes) หรือชั้นวางของ (racks) เป็นต้น ในการติดตั้งระบบ AGV นั้นสิ่งที่ควรคำนึงคือต้องไม่ไปกีดขวางทางเข้าสู่เครื่องจักรเพื่อซ่อมแซมหรือบำรุงรักษา ในทางตรงกันข้ามควรเป็นการเพิ่มความสะดวกให้กับงาน ปัจจุบันนี้เทคโนโลยีทาง AGV ก้าวหน้าไปมาก มีการวิจัยและพัฒนา AGV แบบเคลื่อนที่อิสระ (free ranging) กันอย่างกว้างขวาง AGV แบบนี้ไม่ต้องติดตั้งทางเดินนำร่องบนพื้นแต่จะใช้ระบบนำทาง (navigation system) ซึ่งอยู่บนตัว AGV ควบคุมตำแหน่งการเคลื่อนที่แทน เทคนิคสำหรับ AGV แบบเคลื่อนที่อิสระนั้นมีหลายอย่าง บ่อยครั้งที่มีการนำเทคนิคต่างๆมาผสมผสานกัน อย่างไรก็ตามโรงงานอัตโนมัติในปัจจุบันส่วนมากก็ยังคงใช้ AGV แบบมีทางเดินนำร่องอยู่ โรงงานหลายแห่งมีการติดตั้งแขนหุ่นยนต์บน AGV ทำให้ AGV มีลักษณะเป็นหุ่นยนต์เคลื่อนที่ (mobile robot) สมบูรณ์แบบ วิธีนี้ทำให้สามารถใช้หุ่นยนต์เพียงตัวเดียวกับเครื่องจักรหลายตัว เป็นการเพิ่มประสิทธิภาพในการใช้งานแขนหุ่นยนต์และลดค่าใช้จ่าย

ตามปกติด้านหน้าและด้านหลังของ AGV จะติดบัมเปอร์ (bumper) เพื่อความปลอดภัย AGV ส่วนมากจะใช้แบตเตอรี่ชนิดตะกั่ว-กรด (lead-acid) และขับเคลื่อนด้วยมอเตอร์กระแสตรง

ในกรณีที่ทิศทางการขนส่งวัสดุมีความแน่นอน , ปริมาณการขนส่งมีจำนวนมาก ตลอดจนการขนส่งเป็นไปอย่างต่อเนื่องตลอดเวลา ระบบ AGV จะเสียเปรียบระบบสายพานลำเลียง แต่ข้อดีของ AGV เหนือสายพานลำเลียงก็คือไม่เกะกะ , การติดตั้งทางเดินนำร่องมีความยืดหยุ่น (flexibility) กว่าระบบสายพานลำเลียงเพราะสามารถดัดแปลงแก้ไขได้ง่ายกว่า

นอกจากในโรงงานอุตสาหกรรมแล้ว AGV ยังสามารถนำไปประยุกต์ใช้ในงานด้านอื่นๆ อีกมากอาทิ เช่น การขนสัมภาระไปยังห้องต่างๆของแขกที่มาพักในโรงแรม , การรับส่งเอกสารในสำนักงานโดย AGV ทำหน้าที่เป็นรถส่งสาร (mail mobile) แทนการใช้พนักงาน , การบริการอาหารตามโรงแรมและภัตคารหรือ การขนส่งวัสดุในสถานที่ที่เป็นความลับทางราชการ และอื่นๆ

1.2 <u>เทคโนโลยีทางด้าน AGV ในต่างประเทศ</u> [1],[2]

ระบบ AGV ที่นิยมใช้ในโรงงานอุตสาหกรรมแบบอัตโนมัติในปัจจุบัน ส่วนใหญ่เป็นระบบที่ใช้ทางเดินนำร่องได้แก่ สายตัวนำฝังใต้พื้น, แถบสีติดบนพื้น สำหรับการฝังสายตัวนำใต้พื้นนั้นทำให้ระบบไม่ยืดหยุ่น การแก้ไขดัดแปลงกระทำได้ยาก นอกจากนั้นในระบบ AGV ที่ซับซ้อน การติดตั้งรวมทั้งการสื่อสารด้วยวิธีเหนี่ยวนำผ่านสายตัวนำจะเสียค่าใช้จ่ายสูง

มีหลายกรณีสำหรับระบบ AGV ที่ทางโรงงานมักต้องการดัดแปลงแก้ไข ตัวอย่างการดัดแปลงแก้ไขระบบ AGV อาทิเช่น การต่อเติมทางเดินนำร่อง , การเพิ่มจำนวนสถานีปฏิบัติงาน , การเพิ่มจุดตัด (Junction) , การเปลี่ยนแปลงจำนวน AGV ในระบบ , การเปลี่ยนแปลงลำดับในการขนส่งเช่น เปลี่ยนเส้นทางวิ่งจากสถานีหนึ่งไปอีกสถานีหนึ่ง , การเปลี่ยนแปลงพื้นผิวทำงาน , การเปลี่ยนระบบเซนเซอร์ , การเปลี่ยนวิธีขนถ่ายวัสดุ , การเปลี่ยนวิธีประจุแบตเตอรี่ เป็นต้น ด้วยเหตุนี้ต่อมาผู้ผลิต AGV จึงมีการเสนอ AGV ที่อาศัยทางเดินนำร่องน้อยลง จุดประสงค์สำคัญคือเพื่อให้ระบบมีความยืดหยุ่นมากกว่าที่เป็นอยู่ ทำให้ง่ายต่อการแก้ไขดัดแปลง

ปัจจุบันทิศทางการพัฒนา AGV จึงมุ่งไปที่การพยายามทำให้ AGV สามารถเคลื่อนที่ได้อย่างอิสระ (free ranging) โดยอาศัยระบบการนำทาง การพัฒนา AGV มีการวิจัยทั้งจากบริษัทผู้ผลิตและจากสถาบันวิจัยต่างๆ เพื่อหาเทคโนโลยีใหม่ๆ สำหรับ AGV เพื่อให้ AGV พึ่งพาทางเดินนำร่องน้อยลงโดยได้มีการวิจัยและพัฒนาระบบเซนเซอร์ , เทคนิคการนำทาง , เทคนิคการสื่อสารข้อมูล , ระบบควบคุม รวมทั้งเทคนิคโปรแกรมควบคุมการทำงาน สำหรับแนวโน้มการพัฒนา AGV ในอนาคตคือทำให้ AGV ไม่ต้องใช้ทางเดินนำร่องเลย แนวทางพัฒนาเทคโนโลยี AGV ให้มีความยืดหยุ่นมากขึ้นนั้น สามารถแบ่งได้เป็น 3 แนวหลักได้แก่

ก) พัฒนาการนำร่องแบบไร้สาย (wireless guidance)

เป็นการพัฒนา AGV ให้สามารถเคลื่อนที่ออกจากทางเดินนำร่องชั่วคราว โดย AGV สามารถที่จะเคลื่อนที่กลับไปยังทางเดินนำร่องได้ดังเดิม เทคนิคนี้พบในผู้ผลิตบางราย เป็นลดค่าใช้จ่ายในการดัดแปลงระบบนำร่องโดยไม่จำเป็น นอกจากนี้ในปัจจุบันก็ได้มีการวิจัยระบบนำร่องแบบไร้สายที่สมบูรณ์โดยไม่ต้องใช้ทางเดินนำร่องเลย แต่ยังมีข้อจำกัดหลายอย่างทำให้ยังไม่เป็นที่นิยมนักในอุตสาหกรรม

ข) พัฒนาวิธีการส่งข้อมูล (method of data transmission) วิธีการส่งข้อมูลระหว่าง AGV กับคอมพิวเตอร์ส่วนกลางกระทำได้ หลายวิธี เช่น การส่งข้อมูลผ่านแสงอินฟราเรด , การส่งข้อมูลผ่านคลื่นวิทยุ เป็นต้น

ค) พัฒนาวิธีเก็บตารางการเคลื่อนที่

การเปลี่ยนแปลงจำนวนสถานีหนึ่งไม่พ้นที่จะต้องมีการแก้ไขข้อมูลในตารางการเคลื่อนที่ ตารางนี้จะเก็บข้อมูลเกี่ยวกับการเคลื่อนที่จากสถานีหนึ่งไปยังสถานีหนึ่ง วิธีการเก็บตารางการเคลื่อนที่มีหลาย

วิธีได้แก่

1) เก็บข้อมูลไว้ที่คอมพิวเตอร์ส่วนกลาง ข้อดีคือเปลี่ยนแปลงแก้ไขข้อมูลในตารางได้ง่าย ส่วนข้อเสียคือทำให้ AGV ไม่สามารถทำงานได้ตามลำพัง ต้องพึ่งข้อมูลจากคอมพิวเตอร์ส่วนกลาง

2) เก็บข้อมูลไว้ที่คอมพิวเตอร์ซึ่งอยู่บน AGV ข้อดีของวิธีนี้คือ AGV สามารถทำงานได้โดยไม่ต้องพึ่งคอมพิวเตอร์จากส่วนกลาง และถ้าใช้หน่วยความจำเป็น E^2PROM หรือ RAM PACKED การแก้ไขข้อมูลก็ทำได้ง่าย ส่วนข้อเสียวิธีนี้คือ หากใช้ EPROM การแก้ไขจะไม่ค่อยสะดวก

เทคโนโลยีการสื่อสารข้อมูล

เดิมการส่งข้อมูลระหว่าง AGV กับคอมพิวเตอร์ส่วนกลางจะกระทำสายตัวนำซึ่งฝังใต้พื้น ต่อมาได้มีการเสนอวิธีใหม่คือส่งข้อมูลผ่านแสงอินฟราเรดทำให้สามารถส่งข้อมูลด้วยอัตราบอด (baudrate) ที่สูงขึ้น นอกจากนี้ก็ยังมีการพัฒนาการสื่อสารข้อมูลโดยใช้คลื่นวิทยุ แม้ว่าคลื่นวิทยุมีโอกาสถูกรบกวน (interference) รวมทั้งมีข้อจำกัดทางด้านกฎหมาย แต่ก็ยังคงมีการพัฒนาต่อไป

การใช้อินฟราเรดนั้นมีข้อจำกัดตรงที่การส่งต้องเป็นแนวตรงเท่านั้น ส่วนวิธีการส่งข้อมูลผ่านคลื่นวิทยุจะมีความยืดหยุ่นกว่า ตามปกติลักษณะข้อมูลที่คอมพิวเตอร์ส่วนกลางและ AGV ติดต่อสื่อสารกันนั้นประกอบด้วย

- คำสั่งการขับเคลื่อน (driving order)
- ข้อมูลทางตำแหน่ง (position data)
- การซิงโครไนเซชั่นทางเวลา (synchronization of timers) เพื่อให้คอมพิวเตอร์ส่วนกลาง ทราบตำแหน่งปัจจุบันของ AGV

เทคโนโลยีการนำร่อง

เทคโนโลยี AGV มีการพัฒนาเทคนิคการนำร่องขึ้นมาหลายวิธี ซึ่งสามารถสรุปได้ดังนี้

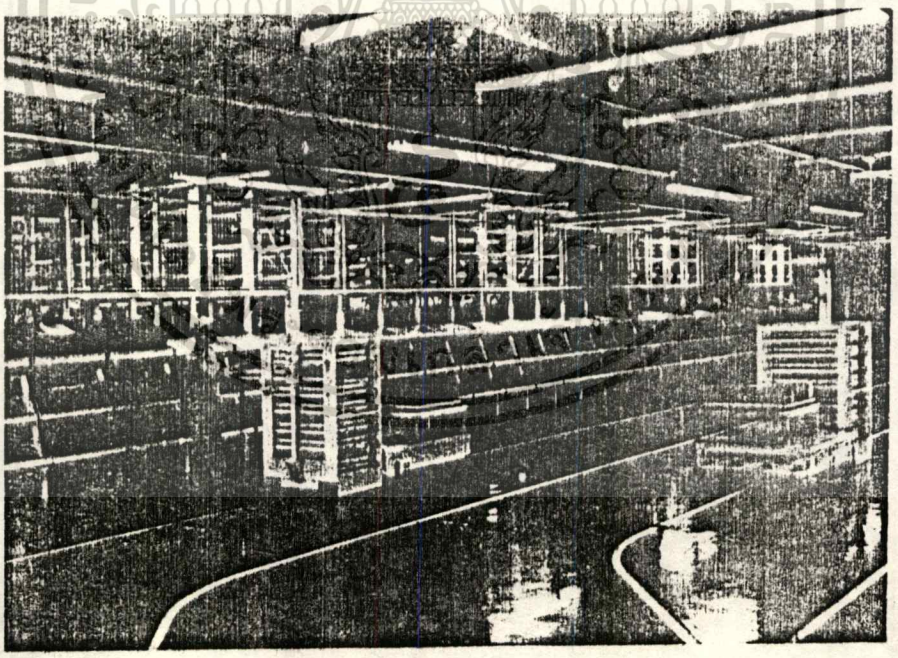
1) <u>การนำร่องโดยใช้ทางเดินนำร่อง</u>

1.1) การฝังสายตัวนำใต้พื้น เป็นวิธีมาตราฐานที่นิยมกันมาก ในการติดตั้งต้องมีการเซาะร่องตามพื้นแล้วฝังสายตัวนำลึกประมาณ 0.5 นิ้ว แล้วปูกระเบื้องยางพร้อมลงอีพอกซี (epoxy resin) การใช้งานจะเริ่มโดยป้อนไฟสลับความถี่สูงแก่ตัวนำเพื่อให้เกิดการเหนี่ยวนำสนามแม่เหล็ก แรงดันที่ใช้ประมาณ 40 โวลท์ กระแสประมาณ 400 มิลลิแอมป์ และความถี่ที่ใช้อยู่ในช่วง 1-15 กิโลเฮิรตซ์ AGV จะมีขดลวด 2 ขดคอยตรวจจับสนามแม่เหล็กที่เหนี่ยวนำจากสายตัวนำ กรณีที่ AGV อยู่วิ่งอยู่ตรงแนวฝังสายตัวนำพอดี ค่าสนามแม่เหล็กที่วัดได้จากขดทั้งสองจะมีค่าเท่ากัน AGV จะวิ่งต่อไปตามปกติ แต่ถ้า AGV วิ่งเบี่ยงเบนไปจากแนวฝังสายตัวนำ สนามแม่เหล็กที่เกิดขึ้นที่ขดลวดทั้งสองจะมี

ค่าไม่เท่ากัน ระบบควบคุมก็จะส่งสัญญาณไปปรับแต่งทิศทางการวิ่งของ AGV ให้ตรงแนวเช่นเดิม ข้อดีของวิธีนี้คือมีความเชื่อถือได้สูง , ทนทาน ส่วนข้อเสียคือระบบไม่ยืดหยุ่น การดัดแปลงแก้ไขทำได้ยาก และค่าใช้จ่ายในการติดตั้งสูงโดยเฉพาะถ้าต้องขนส่งในระยะไกล

1.2) ใช้แถบสีหรือเทปสะท้อนแสงติดบนพื้น AGV จะคอยตรวจจับหาแนวเส้นทางของแถบสีนั้น ข้อดีของวิธีนี้คือระบบการนำร่องไม่ซับซ้อน การเปลี่ยนแปลงแก้ไข , การซ่อมบำรุงกระทำได้ง่าย ตลอดจนค่าใช้จ่ายในการติดตั้งถูก สำหรับเทคนิคในการตรวจจับแถบสีมี 2 วิธีได้แก่

ก) ใช้โฟโต้เซนเซอร์ (photo sensor) ตรวจจับความเข้มแสงที่สะท้อนกลับขึ้นมา ข้อดีของการใช้โฟโต้เซนเซอร์คือสร้างง่ายและราคาถูก ส่วนข้อเสียคือการใช้โฟโต้เซนเซอร์นั้นอ่อนไหวต่อคุณภาพของแถบสีมากอีกทั้งแถบสีนั้น เลอะเลือนและชำรุดเสียหายได้ง่าย นอกจากนี้ความสะอาดของพื้นก็มีผลต่อการตรวจจับเช่นกัน การใช้โฟโต้เซนเซอร์จึงเหมาะสำหรับใน clean room ของโรงงานเท่านั้น ตัวอย่าง AGV ที่ใช้วิธีนี้แสดงไว้ในรูปที่ 1.1



รูปที่ 1.1 ตัวอย่าง AGV ในอุตสาหกรรมซึ่งนำร่องโดยใช้แถบสีติดตามพื้น

ข) ใช้กล้องทีวี (TV camera) โดยนำสัญญาณภาพที่ได้รับ มาทำการประมวลผลเพื่อหาทางเดินของแถบสีบนพื้น การใช้กล้องทีวีมีข้อดีคือทำให้สามารถหาตำแหน่งแถบสีได้ แม้ว่าแถบสีนั้นจะมีการเลอะเลือนหรือชำรุดเสียหายไปบ้างก็ตาม ทำให้สามารถนำไปใช้ขนส่งภายนอกอาคาร (outdoor) ได้โดยเฉพาะการขนถ่ายวัสดุจากอาคารหนึ่งไปสู่อีกอาคารหนึ่ง แต่ข้อเสียก็คือความเร็วในการประมวลผลภาพช้าเกินไปไม่สอดคล้องกับ real time แต่ทั้งนี้ก็ขึ้นกับประสิทธิภาพของซีพียูที่ใช้ประมวลผล

1.3) ใช้แถบโลหะ (metal stripe) วางเป็นแนวแบบเดียวกับแถบสี แล้วใช้พรอคซิมิตี้เซนเซอร์ (proximity sensor) ตรวจหาตำแหน่งแถบโลหะนั้น

1.4) ใช้แถบแม่เหล็ก (magnetic tape) ฝังลงในพื้นเป็นลักษณะตารางทั่วๆพื้นที่ AGV จะทำการตรวจจับแถบแม่เหล็กด้วยแมกเนติคเซนเซอร์ (magnetic sensor)โดย AGV จะเคลื่อนไปตามแนวขอบตาราง จุดตัดของเส้นตารางจะเป็นตัวนับตำแหน่งในการเคลื่อนที่ ลักษณะแมกเนติคเซนเซอร์นั้นประกอบด้วย exciting coil 1 ชุด , detecting coil 2 ชุด exciting coil จะผลิตสนามแม่เหล็กจากไฟสลับโดยมี detecting coil คอยตรวจจับสนามแม่เหล็ก กรณีที่ AGV อยู่ตรงแนวแถบแม่เหล็กพอดี สนามแม่เหล็กที่ตรวจจับได้จะมีค่ามากที่สุด

2) <u>การนำร่องแบบไร้สาย</u> (wireless guidance)

เป็นการนำร่องโดยไม่ต้องติดตั้งทางเดินนำร่องตามพื้น AGV สามารถเคลื่อนที่ได้อย่างอิสระ ในปัจจุบันมีการวิจัยและพัฒนากันมาก สิ่งที่สำคัญที่สุดของการนำร่องแบบไร้สายคือเทคนิคการนำทาง (navigation techniques) ซึ่งมีหลายวิธีอาทิเช่น

2.1) การนำทางด้วยคลื่นวิทยุ (radio navigation) ตามปกติวิธีนี้สามารถหาตำแหน่งของวัตถุที่เคลื่อนที่ในห้องโดยประมาณ โดยมีขอบเขตของการตรวจจับตั้งแต่หลายร้อยเมตรจนถึงหลายกิโลเมตร แต่สำหรับระบบนำร่องแบบไร้สายที่ไม่ใช้ทางเดินนำร่องเลยนั้น วิธีนี้จะไม่เหมาะสมเนื่องเหล็ก (iron) จะทำให้คลื่นวิทยุเกิดการสะท้อน (reflect) และเบี่ยงเบนไป (divert) ทำให้การหาตำแหน่งขาดความถูกต้องแม่นยำ (inaccuracy) โดยเฉพาะหากขอบเขตการตรวจจับต่ำกว่า 100 เมตร

2.2) การนำทางโดยอาศัยระบบดาวเทียม (satellite navigation) หรือรู้จักกันในชื่อ GPS (Global positioning system) สามารถหาตำแหน่งบนพื้นโลกได้โดยการวัดรันนิ่งไทม์ (running time) ของสัญญาณดาวเทียม แต่ความละเอียดสูงสุดของของเทคนิคนี้คือ 10 เมตรเท่านั้นไม่สามารถเพิ่มได้มากกว่านี้

2.3) การใช้อุตร้าโซนิควัดระยะทาง (ultrasound distance measurement) โดย AGV จะเก็บแผนที่เส้นทางทุกเส้นไว้ในหน่วยความจำ วิธีการเคลื่อนที่จะเริ่มที่การกำหนดเป้าหมายปลายทางที่จะไป AGV จะคำนวณหาเส้นทางที่เหมาะสมที่สุดแล้วเคลื่อนที่ไปโดยใช้อุตร้าโซนิคทำหน้า

ที่คอยตรวจจับวัตถุเพื่อหาแนวทางในการเคลื่อนที่

ข้อดีของเทคนิคนี้คือมีความถูกต้องอยู่ในระดับมิลลิเมตร ส่วนข้อเสียคือเทคนิคนี้จำเป็นต้องมีกำแพงตลอดจนวัตถุที่เพียงพอสำหรับใช้ในการตรวจหาตำแหน่ง ส่วนความถูกต้องในการวัดนั้นขึ้นกับอุณหภูมิห้อง , การถ่ายเทของอากาศและอาจถูกรบกวนได้จากแหล่งกำเนิดเสียง (acoustic sources) ที่มีความถี่สูงอาทิ เช่น เครื่องจักรที่กำลังหมุน เป็นต้น

2.4) การใช้เลเซอร์วัดระยะทาง (laser distance measurement) ใช้หลักการวัดรันนิ่งไทม์ (running time) ของสัญญาณ และวัดการรบกวน (interference) ผลลัพธ์และความละเอียดของการวัดเวลาเดินทางของสัญญาณจะอยู่ในหน่วยของเซ็นติเมตร ส่วนการวัดการรบกวนจะอยู่ในหน่วยมิลลิเมตร ข้อเสียของวิธีคือใช้ได้ในบางกรณีเท่านั้นเพราะระบบการขนส่งหลายแห่งเป็นบริเวณเปิดโล่งหรือมีกำแพงซึ่งอยู่ไกลเกินกว่าจะตรวจจับได้ นอกจากนี้กรณีใช้เลเซอร์จะใช้ไม่ได้กับบริเวณที่มีผู้คนและยานพาหนะผ่านหนาแน่น

2.5) การใช้เลเซอร์หาระยะด้วยวิธีสร้างรูปสามเหลี่ยม (laser triangulation) ในวิธีการนี้จะหาตำแหน่งได้โดยการวัดมุมจากจุดสองจุด การสแกนวัตถุที่กำลังเคลื่อนที่นั้นเป็นเรื่องที่ทำได้ยากและค่าใช้จ่ายสูง ความถูกต้องของตำแหน่งจะขึ้นกับระยะทางและมุมที่ทำกับเส้นฐาน (base line) ซึ่งเชื่อมระหว่างจุดสองจุด ระยะทางที่วัดได้อย่างถูกต้องแม่นยำจะอยู่ในช่วง 0-10 เมตร

2.6) การประมวลผลภาพ (image processing) หาตำแหน่ง AGV โดยใช้กล้องติดบนเพดาน (ceiling) ของห้องโถง กล้องที่ใช้อาจจะเป็นแบบ tube หรือ CCD ในทางการค้ากล้อง CCD มีความละเอียดประมาณ 500 pixel สัญญาณภาพที่ได้จะถูกนำไปประมวลผล

2.7) การใช้เครื่องวัดระยะทาง (odometry) เป็นวิธีหาตำแหน่ง AGV เทียบกับตำแหน่งเริ่มต้นโดยใช้โรตารี่ เอนโคดเดอร์ (rotary encoder) ติดตั้งที่ล้อทำหน้าที่คอยวัดตำแหน่ง เริ่มต้นจะต้องมีการกำนดตำแหน่งเริ่มต้น (home position) เพื่อใช้เป็นจุดอ้างอิง วิธีนี้ค่อนข้างง่ายแต่มีข้อเสียคือมักเกิดค่าคลาดเคลื่อนทางตำแหน่งอันเนื่องมาจากการลื่นไถล (slipping) ของล้อ จึงต้องมีการตั้งตำแหน่งใหม่เป็นระยะๆเพื่อลดค่าคลาดเคลื่อนนั้น

2.8) การใช้เข็มทิศไจโรสโคป (gyroscopic compass) เป็นการนำทางโดยอาศัยเทคนิคทางโมเมนตัม ไจโรสโคปจะถูกตั้งขนานกับทิศทางที่ต้องการเคลื่อนที่ไป เมื่อตำแหน่ง AGV เบี่ยงเบนไปจากแนวที่ต้องการ จะก่อให้เกิดความเร่งในทิศตั้งฉากกับทิศทางการเคลื่อนที่ซึ่งค่าความเร่งนี้ไจโรสโคปสามารถตรวจจับได้ ค่าความเร่งนี้จะนำไปใช้คำนวณหาการเบี่ยงเบนของตำแหน่ง จากนั้นจะนำผลการคำนวณป้อนให้ระบบควบคุมตำแหน่ง เพื่อแก้ไขตำแหน่ง AGV ต่อไป

2.9) วิธีคอร์รีเลชั่น (Correlation) โดยใช้ออปติค เซนเซอร์แบบหัวอ่านคู่ (two-tandem joined heads) ทำการสแกนไปบนพื้นซึ่งขรุขระ (roughness) และการวัดค่าของ

หัวอ่านทั้งสองจะเป็นไปอย่างสัมพันธ์กัน หลักของการวัดมีลักษณะเป็นการเลื่อนอย่างอิสระ (slip-ping-independent) การวัดของหัวอ่านทั้งสองนี้จะสามารถประเมินลักษณะการเคลื่อนที่ในระนาบ (plane) ที่ไม่ทราบมาก่อนได้

3) <u>การนำร่องโดยใช้ทางเดินนำร่องร่วมกับการนำร่องแบบไร้สาย</u>

เป็นการนำสองเทคนิคข้างต้นมาผสมผสานกัน ทั้งนี้เพื่อให้ระบบมีประสิทธิภาพสูงขึ้น อาทิเช่น

3.1) ใช้วิธีนำร่องด้วยการฝังสายตัวนำใต้พื้นแบบมาตราฐานเป็นหลัก แล้วเสริมด้วยการใช้เครื่องวัดระยะทาง (odometry) ในบางจุด เช่น บริเวณที่ AGV จะเคลื่อนเข้าไปเทียบท่ายังอุปกรณ์ถ่ายโหลดซึ่งอยู่ที่สถานี เป็นต้น วิธีนี้ทำให้ระบบมีความยืดหยุ่นมากขึ้น ไม่ต้องแก้ไขต่อเติมเส้นทางวิ่งใหม่หากต้องการเพิ่มจำนวนสถานี

3.2) ใช้วิธีนำร่องด้วยการวางกริด (grid) เป็นการนำร่องโดยให้ AGV วิ่งไปตามแนวกริดแล้วใช้เครื่องวัดระยะทางเช่น โรตารี่ เอนโคดเดอร์ คอยวัดตำแหน่งการเคลื่อนที่ กริดมีลักษณะเป็นเส้นที่ต่อกันเป็นรูปตาราง การวัดระยะทางจะมีการอ้างอิงทุกครั้งที่ AGV วิ่งผ่านเส้นกริด ทำให้ค่าคลาดเคลื่อนทางตำแหน่งถูกกำจัดออกไป วิธีนี้มีข้อดีคือมีความยืดหยุ่นสูง สามารถเปลี่ยนแปลงวิธีการวิ่งได้ง่าย , มีความถูกต้องแม่นยำ , ต้นทุนต่ำ ตลอดจนสามารถใช้ได้ทั้งภายในอาคารและนอกอาคาร สำหรับตัวเซนเซอร์ที่ใช้ในการตรวจจับนั้นหากกริดทำจากแถบเหล็กก็ใช้แมกนิติคเซนเซอร์ แต่ถ้าหากกริดทำจากแถบสีก็ใช้กล้องทีวีหรือโฟโต้เซนเซอร์ตรวจจับ กริดที่ใช้กันมีหลายแบบได้แก่

ก) ใช้แถบโลหะติดตามขอบแผ่นคอนกรีต ทำให้ละดวกในการติดตั้งมากขึ้น การตรวจจับแถบโลหะจะใช้พรอคซิมิตี้เซนเซอร์

ข) ใช้แถบสีติดเป็นเส้นตาราง แล้วใช้กล้อง CCD หรือโฟโต้เซนเซอร์ตรวจจับ แบบนี้มีข้อเสียคือ ไวต่อสิ่งสกปรก ทำให้ใช้ได้แต่ใน clean room

ค) ใช้แผ่นปูพื้นแบบเป็นรูปหมากรุก คือมี 2 สีสลับกัน การตรวจจับอาจใช้ใช้กล้อง CCD หรือโฟโต้เซนเซอร์ สำหรับข้อเสียจะเหมือนในข้อ ข)

ลักษณะของกริดนอกจากจะทำเป็นเส้นแล้วยังสามารถทำเป็นจุดได้ด้วย การทำเป็นจุดทำให้ค่าใช้จ่ายถูกลงอีก สำหรับการเลี้ยวจะเกิดขึ้นจากการหมุนล้อซ้ายและล้อขวาด้วยความเร็วต่างกัน งานวิจัยบางแห่งได้เพิ่มเติมในส่วนที่ทำให้ AGV สามารถหลบหลีกวัตถุกีดขวางได้อีกด้วย ทำให้ระบบมีความยืดหยุ่นสูงขึ้นไปอีก

4) <u>การนำร่องด้วยเทคนิคพิเศษอื่นๆ</u>

ในสถาบันวิจัยต่างๆได้มีการพัฒนากันมาก แต่ยังไม่ได้นำมาพัฒนาในอุตสาหกรรมอย่างจริงจัง จึงเป็นเพียงการทดลองที่ได้ผลเท่านั้น ตัวอย่างของการนำร่องด้วยเทคนิคพิเศษอื่นๆ มีดังนี้

4.1) การนำร่องโดยใช้เทคนิคการวัดมุมด้วยเลเซอร์ เป็นระบบการนำทางด้วยเลเซอร์

โดยวิธีการวัดมุมสะท้อนกลับของแสงเลเซอร์ที่ยิงไปตกกระทบแผ่นสะท้อนซึ่งติดอยู่ตามผนังห้อง ระบบนี้ประกอบด้วยเครื่องกำเนิดเลเซอร์พลังงานต่ำ, กระจกหมุน (rotating mirror) , แผ่นสะท้อน (reflector) ติดตามผนังกำแพงในแนวระดับเดียวกันกับลำเลเซอร์และตัวรับออบติค เช่น กระจกสะท้อน , เลนส์ เป็นต้น การทำงานของระบบเริ่มด้วยการยิงเลเซอร์ผ่านกระจกหมุน ซึ่งจะทำให้ได้ลำเลเซอร์กวาดไปรอบๆห้องโดยที่ AGV สามารถรู้มุมลำเลเซอร์ได้จากมุมการหมุนของกระจก ลำเลเซอร์จะไปตกกระทบแผ่นสะท้อนซึ่งติดอยู่ที่ผนังในแนวระดับเดียวกันแล้วสะท้อนกลับมายังตัวรับออบติค ซึ่งใช้แกนร่วมอยู่กับลำกล้องยิงเลเซอร์ มุมที่เลเซอร์สะท้อนกับมาจะเป็นมุมของแผ่นสะท้อนนั้นเทียบกับจุดอ้างอิง เมื่อเลเซอร์กวาดไปครบ 1 รอบ ก็จะได้ค่ามุมของแผ่นสะท้อนทุกๆ แผ่นที่ติดตามผนัง ค่ามุมทุกมุมของแผ่นสะท้อนจะถูกนำไปประมวลผลเพื่อหาตำแหน่งและมุมที่ถูกต้องของตัว AGV ต่อไป ทุกรอบที่เลเซอร์กวาดไป ตำแหน่งและส่วนหัวของตัว AGV จะมีการปรับเปลี่ยนตาม อย่างไรก็ตามวิธีนี้เป็นเพียงต้นแบบในงานวิจัยเท่านั้น ระบบนี้เรียกอีกอย่างว่าเครื่องวัดมุม (angle meter) ข้อดีของวิธีนี้คือเป็นการนำร่อง AGV โดยไม่ต้องใช้ทางเดินนำร่องแต่ก็มีข้อจำกัดกรณีมีวัตถุกีดขวางระหว่างแผ่นสะท้อนกับ AGV ซึ่งจะทำให้การวัดผิดพลาด นอกจากนี้ค่าใช้จ่ายในการสร้างก็สูงและมีข้อจำกัดตรงที่ระบบนี้ต้องการบริเวณที่มีผนังเกือบรอบด้าน

4.2) การนำร่องโดยใช้กล้องอินฟราเรดอ่านตำแหน่ง AGV จากเพดาน วิธีการก็คือจะติดกล้องอินฟราเรดบนเพดานไว้คอยตรวจจับความเคลื่อนไหวของ AGV โดยบน AGV จะติดตั้งตัวส่งแสงอินฟราเรด (infrared light emitter) ข้อมูลภาพจากกล้องจะถูกส่งไปประมวลผลยังคอมพิวเตอร์ส่วนกลางเพื่อหาตำแหน่งการเคลื่อนที่ จากนั้นคอมพิวเตอร์ส่วนกลางจะส่งคำสั่งควบคุมพร้อมด้วยข้อมูลการเคลื่อนที่ไปยัง AGV โดยผ่านคลื่นวิทยุ

เทคโนโลยีการขับเคลื่อนและการเลี้ยว [3]

AGV ส่วนใหญ่ใช้พลังงานในการขับเคลื่อนจากแบตเตอรี่ชนิดตะกั่ว-กรด (lead-acid) และขับเคลื่อนโดยใช้มอเตอร์กระแสตรงซึ่งมีการควบคุมทิศทางการหมุน , ความเร็วและ ตำแหน่ง โดยใช้คอมพิวเตอร์ (onboard computer) ซึ่งติดตั้งอยู่บน AGV สำหรับเทคนิคการเลี้ยวของ AGV นั้นจะขึ้นกับโครงสร้างระบบล้อ ระบบล้อตามปกติที่ใช้กันอาจเป็นระบบ 3 ล้อหรือระบบ 4 ล้อก็ได้ AGV บางตัวอาจใช้ถึง 6 ล้อทั้งนี้ขึ้นกับผู้ผลิต เมื่อเปรียบเทียบระบบ 4 ล้อกับระบบ 3 ล้อแล้วระบบ 4 ล้อจะมีเสถียรภาพดีกว่า , กำลังลากจูงดีกว่า (more traction) , ช่องว่างทางกลน้อยกว่า แต่ระบบ 3 ล้อก็มีข้อดีตรงที่ระบบเลี้ยวไม่ยุ่งยากซับซ้อน , โครงรถมีน้ำหนักเบากว่าซึ่งทำให้ประหยัดพลังงานจากแบตเตอรี่ได้มาก สำหรับโครงสร้างระบบล้อของ AGV นั้นมีหลายแบบอาทิเช่น

- ระบบ 3 ล้อ เลี้ยวโดยล้อหน้า 1 ล้อโดยให้ล้อหลัง 2 ล้อเป็นล้อขับ
- ระบบ 3 ล้อ เลี้ยวด้วย 2 ล้อหลัง ส่วนล้อหน้าเป็นล้อขับ
- ระบบ 3 ล้อ ล้อหน้าเป็นทั้งล้อขับและเลี้ยว ส่วน 2 ล้อหลังล้อช่วย (castor)
- ระบบ 3 ล้อ เลี้ยวด้วยล้อหน้า 2 ล้อ โดยมีทั้งแบบเลี้ยวพร้อมกัน หรือแต่ละล้อเลี้ยวได้โดยอิสระ ส่วนล้อหลังเป็นล้อขับ
- ระบบ 3 ล้อ เลี้ยวโดยใช้หลักการหมุนในทิศตรงข้ามกัน (differntial steering) ของล้อหลัง 2 ล้อ ส่วนล้อหน้าก็สามารถเลี้ยวได้ด้วย
- ระบบ 3 ล้อ เลี้ยวโดยใช้หลักการหมุนในทิศตรงข้ามกันของล้อหลัง 2 ล้อ ส่วนล้อหน้าเป็นช่วย
- ระบบ 3 ล้อ ล้อทุกล้อเป็นล้อขับที่สามารถเลี้ยวได้โดยอิสระ เทคนิคนี้มีความยืดหยุ่นสูงแต่โครงสร้างทางกลซับซ้อน
- ระบบ 4 ล้อ เลี้ยวโดยใช้หลักการหมุนในทิศตรงข้ามกันของ 2 ล้อขับ อีก 2 ล้อเป็นล้อช่วย
- ระบบ 4 ล้อ ขับเคลื่อนได้ทั้ง 4 ล้อ และล้อแต่ละคู่ก็สามารถเลี้ยวได้ด้วย
- ระบบ 4 ล้อ 2 ล้อหน้าเป็นทั้งล้อขับและล้อสำหรับเลี้ยว ส่วน 2 ล้อหลังเป็นล้อช่วย
- ระบบ 4 ล้อ 2 ล้อหน้าสำหรับเลี้ยวส่วน 2 ล้อหลังเป็นล้อขับ หลักการเดียวกับรถยนต์ทั่วไปที่ขับเคลื่อนล้อหลัง สำหรับในวิทยานิพนธ์นี้เลือกใช้ระบบนี้เป็นกรณีศึกษาสำหรับระบบ AGV
- ระบบ 5 ล้อ ล้อที่ 5 ซึ่งเป็นทั้งล้อขับและล้อสำหรับเลี้ยวจะติดตั้งอยู่ตรงกลางแนวแกนของ 2 ล้อหน้าซึ่งเป็นล้อช่วย ส่วน 2 ล้อหลังก็เป็นล้อช่วยธรรมดาเช่นกัน
- ระบบ 6 ล้อ ล้อที่ 5 และล้อที่ 6 ซึ่งเป็นทั้งล้อขับและล้อสำหรับเลี้ยวจะติดตั้งอยู่ตรงกลางแนวแกน 2 ล้อหน้าซึ่งเป็นล้อช่วย และ 2 ล้อหลังซึ่งเป็นล้อช่วยเช่นกันตามลำดับ

เทคโนโลยีอื่นๆ

ในการใช้ AGV ตามโรงงานหรือสถานที่ต่างๆ ปัญหาหนึ่งที่ต้องพบคือวัตถุกีดขวาง จึงได้มีการพัฒนาเทคโนโลยีในการตรวจจับวัตถุกีดขวาง การตรวจจับวัตถุกีดขวาง (obstacle) มีวิธีการหลายวิธีอาทิเช่น การใช้คลื่นอุตร้าโซนิค,การใช้แสงอินฟราเรด และการใช้กล้องทีวี เป็นต้น สถาบันวิจัยบางแห่งได้พัฒนาให้ AGV มีความฉลาดยิ่งขึ้นโดยการทำให้ AGV สามารถหลบหลีกวัตถุกีดขวางนั้นได้

นอกจากนี้เนื่องจาก AGV ใช้พลังงานจากแบตเตอรี่ จึงต้องมีระบบตรวจจับแรงดันแบตเตอรี่เพื่อรายงานสภาวะกลับไปยังคอมพิวเตอร์ส่วนกลาง

1.3 วัตถุประสงค์ของวิทยานิพนธ์

1) ศึกษาเทคโนโลยีทางด้าน AGV ในต่างประเทศ

2) วางหลักการเกี่ยวกับระบบควบคุม AGV เพื่อใช้เป็นแนวทางในการสร้าง AGV ต่อไป

3) เพื่อศึกษาเทคนิคการสื่อสารข้อมูลแบบอะซิงโครนัสอาล์ฟดูเพล็กซ์ผ่านคลื่นวิทยุ

4) เพื่อศึกษาเทคนิคการส่งข้อมูลตามสายแบบกระแสวนลูป

5) เพื่อศึกษาเทคนิคการควบคุมตำแหน่งและความเร็วมอเตอร์โดยใช้ไมโครโปรเซสเซอร์

6) เพื่อศึกษาเทคนิคการทำแฮนด์เชคกิ้งระหว่างซีพียู 2 ตัว

7) เพื่อประยุกต์ใช้อุตร้าโซนิคเซนเซอร์ในการตรวจจับวัตถุกีดขวาง

8) เพื่อศึกษาเทคนิคทางด้านโปรแกรม อาทิเช่น เทคนิคการเขียนโปรแกรมแบบฝังตัว (resident file) , เทคนิคการเขียนโปรแกรมสื่อสารข้อมูลผ่านพอร์ท RS-232-C , เทคนิคการเขียนโปรแกรมเชื่อมต่อระหว่างภาษาระดับสูงกับภาษาแอสเซมบลี้โดยไม่ต้องทำการลิงค์ (link)

1.4 ขอบเขตของวิทยานิพนธ์

ลักษณะ AGV ที่ใช้ในปัจจุบันตามโรงงานอุตสาหกรรม มีเทคโนโลยีหลายอย่างที่สำคัญได้แก่

ก) การติดต่อกันระหว่างคอมพิวเตอร์ส่วนกลางกับ AGV ใช้วิธีส่งข้อมูลผ่านคลื่นวิทยุ

ข) ขับเคลื่อนด้วยมอเตอร์กระแสตรง โดยใช้ไมโครโปรเซสเซอร์ซึ่งอยู่บน AGV ทำการควบคุมความเร็วและตำแหน่ง

ค) มีการติดตั้งระบบเซนเซอร์ต่างๆ อาทิเช่น ระบบตรวจจับแรงดันแบตเตอรี่ , ระบบนำร่อง, ระบบตรวจจับวัตถุกีดขวาง เป็นต้น

ในโรงงานแบบกึ่งอัตโนมัติ ซึ่งพบมากในประเทศกำลังพัฒนาหลายประเทศยังคงใช้แรงงานคนแทนหุ่นยนต์ในการป้อนชิ้นงานให้เครื่องจักรประเภท NC ทำให้เวลาในป้อนชิ้นงานแก่เครื่องจักรมีความยืดหยุ่น ไม่ตายตัว การนำ AGV มาใช้ในโรงงานแบบนี้จึงควรมีระบบเรียกใช้งานซึ่งสามารถเรียกใช้ AGV จากตำแหน่งเครื่องจักรนั้นๆ ได้โดยตรง การเรียกใช้ทำได้หลายวิธีแต่วิธีหนึ่งที่ค่อนข้างง่ายคือการใช้ปุ่มกดเรียก สัญญาณจากปุ่มกดเรียกจะถูกส่งเข้าคอมพิวเตอร์ส่วนกลางเพื่อประมวลผล ในวิทยานิพนธ์นี้จึงได้มีการเพิ่มส่วนนี้เข้าไปด้วย วิทยานิพนธ์นี้เป็นการสร้างระบบควบคุมการทำงานต่างๆ ซึ่งสามารถนำไปประยุกต์ใช้ในการสร้างAGV ต่อไป ซึ่งมีขอบเขตวิทยานิพนธ์ดังต่อไปนี้

1) สร้างระบบสื่อสารข้อมูลระหว่างไมโครคอมพิวเตอร์กับไมโครคอนโทรลเลอร์

2) สร้างระบบเรียกใช้งาน AGV จากพื้นที่ปฏิบัติงาน

3) สร้างระบบควบคุมความเร็วและตำแหน่งมอเตอร์กระแสตรง

4) สร้างระบบเซนเซอร์ต่างๆได้แก่ โฟโต้เซนเซอร์เพื่อใช้ในการนำร่อง ,ระบบตรวจจับวัตถุ

กีดขวางด้วยคลื่นอุตร้าโซนิคและระบบตรวจจับแรงดันแบตเตอรี่

5) พัฒนาโปรแกรมควบคุมการทำงานในส่วนไมโครคอมพิวเตอร์อันประกอบด้วย โปรแกรมสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์ , โปรแกรมรับการเรียกจากสถานีปฏิบัติงาน

6) พัฒนาโปรแกรมควบคุมการทำงานในส่วนไมโครคอนโทรลเลอร์อันประกอบด้วย โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์ , โปรแกรมควบคุมการเคลื่อนที่และโปรแกรมปฏิบัติตามคำสั่งต่างๆ

1.5 <u>เนื้อหาของวิทยานิพนธ์</u>

บทที่ 1 กล่าวถึงความเป็นมาของ AGV ,เทคโนโลยีทางด้าน AGV ในต่างประเทศ ,วัตถุประสงค์ ,ขอบเขตและเนื้อหาของวิทยานิพนธ์นี้

บทที่ 2 กล่าวถึงรายละเอียดและทฤษฎีต่างๆที่เกี่ยวข้องกับระบบควบคุม AGV ในวิทยานิพนธ์นี้ ได้แก่ การสื่อสารข้อมูลระหว่างไมโครคอมพิวเตอร์กับไมโครคอนโทรลเลอร์ , การรับส่งข้อมูลแบบอะซิงโครนัส, ทิศทางการส่งผ่านข้อมูล, พอร์ทสื่อสารข้อมูลอนุกรม, โครงสร้างของพอร์ทสื่อสารข้อมูลอนุกรม, การสื่อสารด้วยคลื่นวิทยุ, ประเภทการสื่อสารด้วยคลื่นวิทยุ, วิธีการสื่อสารทางวิทยุ, โครงสร้างวงจรสื่อสารด้วยคลื่นวิทยุ, สาเหตุและปัญหาการรบกวน, การแก้ไขปัญหาสัญญาณรบกวน, สัญญาณรบกวน AGV, การเรียกใช้ AGV จากสถานีปฏิบัติงาน, การอินเตอร์เฟสแบบกระแสวนลูป,การควบคุมความเร็วมอเตอร์กระแสตรง , คุณสมบัติ PI คอนโทรลเลอร์, ข้อดีข้อเสีย , สมการทางไฟฟ้า และสมการไดนามิกส์ของมอเตอร์กระแสตรง,ทาโคมิเตอร์, การควบคุมตำแหน่งมอเตอร์กระแสตรง, หลักการทั่วไป, กรอบความเร็วสำหรับควบคุมตำแหน่ง, อินคริเมนทัลโรตารี่เอนโคดเดอร์, การคำนวณระยะทางการเคลื่อนที่ของ AGV, การวิเคราะห์การเลี้ยว, เทคโนโลยีการตรวจจับวัตถุกีดขวาง, คลื่นอุตร้าโซนิค , ชนิดของคลื่นอุตร้าโซนิค, อุตร้าโซนิคทรานสดิวเตอร์, แบตเตอรี่, ข้อกำหนดของแบตเตอรี่

บทที่ 3 กล่าวถึงการออกแบบและสร้างระบบควบคุม AGV ได้แก่ ระบบสื่อสารข้อมูลระหว่างไมโครคอมพิวเตอร์กับไมโครคอนโทรลเลอร์ , ระบบเรียกใช้งานจากสถานีปฏิบัติงาน , ระบบควบคุมการเคลื่อนที่ และระบบเซนเซอร์

บทที่ 4 กล่าวถึงโปรแกรมควบคุมการทำงาน ประกอบด้วยโปรแกรมควบคุมการทำงานในส่วนไมโครคอมพิวเตอร์ และในส่วนไมโครคอนโทรลเลอร์

บทที่ 5 เป็นการทดลอบการทำงานของระบบต่างๆ รวมทั้งสรุปผลการทดสอบ

บทที่ 6 เป็นการสรุปผลงานวิจัยทั้งหมดรวมทั้งปัญหาและข้อเสนอแนะ

ภาคผนวก ก แสดงรายละเอียดของโปรแกรมโดยสมบูรณ์

ภาคผนวก ข กล่าวถึงแนวทางในการพัฒนา AGV ในขั้นต่อไป

บทที่ 2
ทฤษฎีที่เกี่ยวข้องกับระบบควบคุม AGV

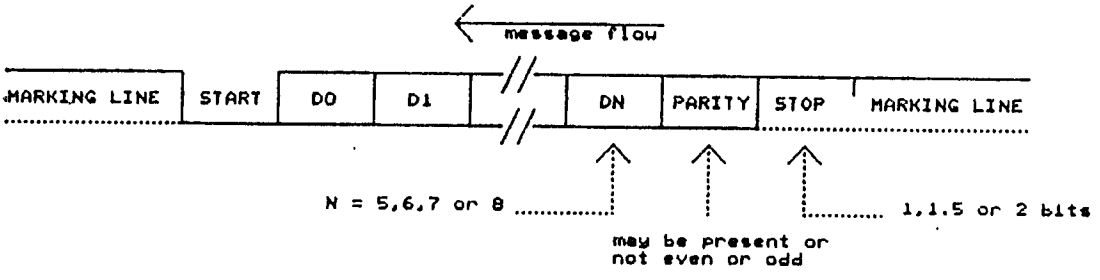
การสื่อสารข้อมูลระหว่างไมโครคอมพิวเตอร์กับไมโครคอนโทรลเลอร์

การสื่อสารข้อมูลระหว่างไมโครคอมพิวเตอร์กับไมโครคอนโทรลเลอร์สำหรับวิทยานิพนธ์นี้ใช้วิธีส่งข้อมูลอนุกรมแบบอะซิงโครนัส ฮาลฟ์ดูเพล็กซ์ (asynchronous half duplex) ผ่านทางคลื่นวิทยุ

ข้อมูลอนุกรมที่ออกจากไมโครคอมพิวเตอร์จะถูกส่งไปยังภาคเข้ารหัส (encoder) เพื่อทำการแปลงข้อมูลเป็นขบวนพัลซ์ จากนั้นจึงนำไปผสมกับสัญญาณความถี่วิทยุ (modulator) แบบอัมปลิจูดมอดูเลชั่น (amplitude modulation) เพื่อส่งออกสายอากาศ ด้านภาครับเมื่อได้รับสัญญาณก็จะถอดรหัส (decode) ออกมาเป็นขบวนพัลซ์เหมือนเดิม แล้วจึงแปลงกลับเป็นข้อมูลอนุกรมอีกครั้งเพื่อส่งเข้าพอร์ทอนุกรมของไมโครคอนโทรลเลอร์ สำหรับการส่งข้อมูลจากไมโครคอนโทรลเลอร์กลับไปยังไมโครคอมพิวเตอร์ก็ใช้วิธีการเดียวกัน

การรับส่งข้อมูลแบบอะซิงโครนัส [4]

คือระบบการรับส่งข้อมูลที่แต่ละคำ ถูกส่งออกไปอย่างไม่มีกำหนดเวลาแน่นอน นั่นคือระยะเวลาระหว่างข้อมูลแต่ละคำที่ถูกส่งออกไปมีค่าไม่แน่นอน ตัวอย่างเช่น การส่งข้อมูลจากคอมพิวเตอร์ A ไปยังคอมพิวเตอร์ B ในการป้อนข้อมูลจากคีย์บอร์ดนั้น ผู้พิมพ์ดีดไม่สามารถพิมพ์ด้วยอัตราเร็วคงที่ได้ โดยเวลาแต่ละครั้งที่กดคีย์จะห่างไม่เท่ากัน ดังนั้นสิ่งที่กำหนดเวลาในการส่งข้อมูลคือ ความพร้อมเพรียงของเครื่องรับและเครื่องส่ง ในการส่งข้อมูลอนุกรมแบบอะซิงโครนัสนั้นโครงสร้างของข้อมูลที่จะส่งจะมีลักษณะเป็นเฟรม (frame) แต่ละเฟรมประกอบด้วยบิทเริ่มต้น (start bit), บิทข้อมูล (data bit) และบิทสุดท้ายคือบิทสิ้นสุด (stop bit) โดยบิทเริ่มต้นจะแสดงถึงการเริ่มต้นของกลุ่มข้อมูล แล้วตามด้วยส่วนของกลุ่มข้อมูล และบางกรณีอาจจะมีการเพิ่มบิทพาริตี้ (parity bit) เพื่อใช้ตรวจสอบความถูกต้องของข้อมูล บิทสิ้นสุดของข้อมูลจะเป็นบอกว่าข้อมูลในเฟรมนี้หมดลงเพียงแค่นี้ ลักษณะการส่งข้อมูลอนุกรมแบบอะซิงโครนัส แสดงดังรูปที่ 2.1



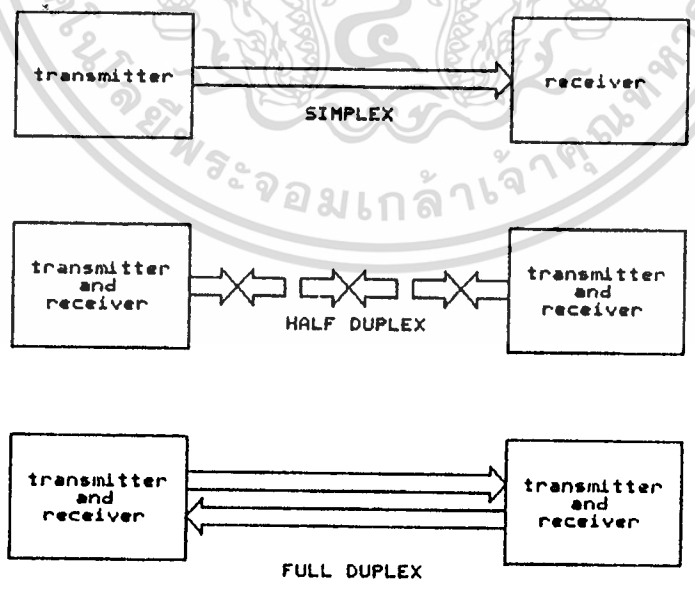
รูปที่ 2.1 การส่งข้อมูลอนุกรมแบบอะซิงโครนัส

ทิศทางการส่งผ่านข้อมูล

ทิศทางการส่งข้อมูลมี 3 แบบ คือแบบทิศทางเดียวหรือซิมเพล็กซ์ (simplex) , แบบสองทางแต่โต้ตอบกันไม่ได้หรือฮาล์ฟดูเพล็กซ์ (half duplex) และแบบที่สามเป็นแบบส่งข้อมูลได้สองทิศทางและโต้ตอบกันได้ในเวลาเดียวกัน หรือแบบฟูลดูเพล็กซ์ (full duplex) ลักษณะการส่งข้อมูลแสดงตามรูปที่ 2.2

วิธีที่ง่ายที่สุดคือ การส่งข้อมูลแบบไปในทางเดียว เช่น ส่งข้อมูลจากเทอร์มินัล A ไปยังเทอร์มินัล B ในกรณีนี้เทอร์มินัล A จะต้องเป็นเครื่องส่งและ B จะต้องเป็นเครื่องรับเท่านั้น อาทิเช่น การกระจายเสียงวิทยุหรือโทรทัศน์ , การส่งข้อมูลลจากคอมพิวเตอร์ไปยังเครื่องพิมพ์ เป็นต้น

ในแบบที่สองจะแตกต่างจากกรณีแรกคือ เป็นการส่งในลักษณะที่เทอร์มินัล A และ B สามารถทำหน้าที่เป็นได้ทั้งเครื่องรับและเครื่องส่ง เช่น เทอร์มินัล A ส่งข้อมูลไปให้ B ได้และ B ก็ส่งข้อมูลตอบกลับมาให้ A ได้เช่นกัน แต่ตัวส่งอยู่คนละช่วงเวลากัน ตัวอย่างเช่น ระบบวิทยุติดต่อ , ระบบ ATM เป็นต้น สำหรับแบบที่สามนั้นเทอร์มินัล A และ B สามารถจะโต้ตอบกันได้คือ เป็นทั้งเครื่องรับ และเครื่องส่งในเวลาเดียวกันได้คือเทอร์มินัล A ส่งข้อมูลไปเทอร์มินัล B ซึ่งเป็นเวลาเดียวกับที่เทอร์มินัล B ส่งข้อมูลไปให้เทอร์มินัล A เทอร์มินัล A และ B จะมีการทำงานที่เป็นอิสระต่อกัน ตัวอย่างการส่งแบบนี้ เช่น การสื่อสารทางโทรศัพท์ และการสื่อสารในระบบคอมพิวเตอร์ เป็นต้น



รูปที่ 2.2 การส่งข้อมูลลักษณะต่างๆ

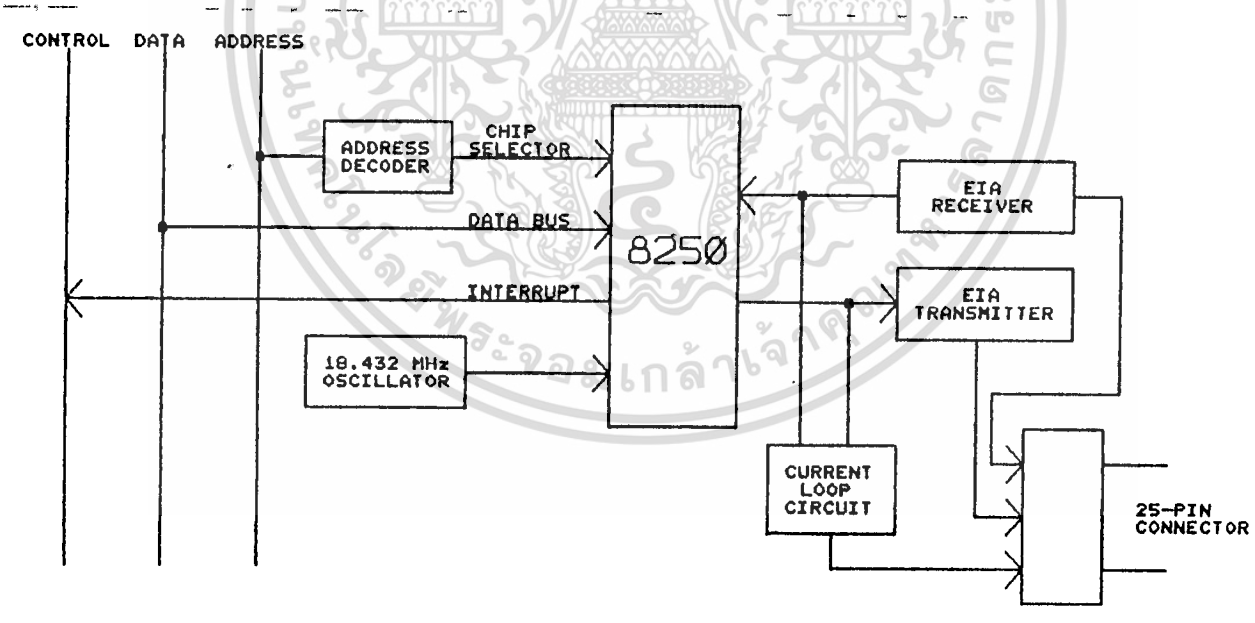
พอร์ทสื่อสารข้อมูลอนุกรม

พอร์ทสื่อสารข้อมูลอนุกรมเป็นส่วนสำคัญส่วนหนึ่งที่มีอยู่บนไมโครคอมพิวเตอร์ พอร์ทนี้เป็นทางออกของข้อมูลที่ผู้ใช้สามารถส่งหรือรับกับระบบอื่นได้ การอินเตอร์เฟสกับเทอร์มินัลภายนอกใช้มาตราฐาน RS-232-C อย่างไรก็ตามผู้ออกแบบได้ให้ทางเลือกในการสื่อสารด้วยกระแสวนลูป (current loop) หรือแบบแรงดันมาตราฐาน RS-232-C โดยใช้จัมเปอร์ (jumper) เพื่อเลือกระบบ

วงจรพอร์ทสื่อสารนี้ใช้ชิพ 8250 ทำหน้าที่ควบคุมเกี่ยวกับการสื่อสารแบบอนุกรมทั้งหมด ซึ่งก่อนจะทำงานต้องได้รับการโปรแกรมก่อน หลังจากนั้นจึงจะทำงานตามรูปแบบที่ได้โปรแกรมไว้จนกว่าจะมีการโปรแกรมใหม่ การกำหนดอัตราบอด (baud rate) ในการส่งสามารถกำหนดได้ตั้งแต่ 50 ถึง 9600 บิท/วินาที พอร์ทอนุกรมบนไมโครคอมพิวเตอร์มีอยู่สองพอร์ทคือ COM1 และ COM2

โครงสร้างของพอร์ทสื่อสารข้อมูลอนุกรม [5]

พอร์ทสื่อสารของไมโครคอมพิวเตอร์ที่จะกล่าวถึงนี้เป็นระบบมาตราฐานตามแบบเครื่อง IBM PC/AT โครงสร้างบล็อคไดอะแกรมแสดงดังรูปที่ 2.3



รูปที่ 2.3 โครงสร้างพอร์ทสื่อสารข้อมูลอนุกรมที่ใช้ชิพ 8250

พิจารณาได้ว่า 8250 เป็นตัวรับข้อมูลจากบัสข้อมูลของระบบ ซีพียูจะติดต่อกับ 8250 ในลักษณะเป็นพอร์ทอินพุทเอาท์พุท

การสื่อสารด้วยคลื่นวิทยุ

พาหะของสารสื่อหรือช่องทาง (media of channels) ที่ได้รับการพัฒนามาจนถึงปัจจุบันนี้ และมีความเจริญก้าวหน้าไปเป็นอันมากนั้นคือการสื่อสารโดยอาศัยคลื่นวิทยุ ซึ่งช่วยให้ติดต่อสื่อสารกันได้เป็นระยะทางไกลมากขึ้น

ประเภทการสื่อสารด้วยคลื่นวิทยุ [6]

ประเภทของการสื่อสารด้วยคลื่นวิทยุจะแบ่งตามย่านความถี่ออกเป็น 3 ประเภทดังนี้

1) การสื่อสารย่านความถี่ HF (high frequency) คือ ย่านความถี่สูง เริ่มตั้งแต่ความถี่ 3 - 30 MHz เครื่องรับ-ส่งวิทยุย่าน HF ส่วนใหญ่ออกแบบให้ใช้รับส่งสัญญาณในระบบ SSB และ CB การสื่อสารในย่านความถี่ HF จะเป็นการสื่อสารระยะไกลและเป็นการเชื่อมโยงระหว่างจุดต่อจุด

2) การสื่อสารย่านความถี่ VHF (very high frequency) คือ ย่านความถี่สูงมาก เริ่มตั้งแต่ความถี่ 30 - 300 MHz ส่วนใหญ่จะใช้รับส่งสัญญาณ FM การสื่อสารในย่านความถี่นี้มีทั้งแบบเชื่อมโยงระหว่างจุดต่อจุดและการสื่อสารแบบเคลื่อนที่ ระยะทางที่ติดต่อสื่อสารกันมักไม่เกิน 50 กิโลเมตร

3) การสื่อสารย่านความถี่ UHF (ultrahight frequency) คือย่านความถี่สูงยิ่ง เริ่มตั้งแต่ความถี่ 300 - 3000 MHz ส่วนใหญ่จะใช้รับส่งสัญญาณ FM การสื่อสารในย่านความถี่นี้มีทั้งแบบเชื่อมโยงระหว่างจุดต่อจุดและการสื่อสารแบบเคลื่อนที่ ระยะทางที่ติดต่อสื่อสารกันมักไม่เกิน 50 กิโลเมตร

วิธีการสื่อสารทางวิทยุ

การสื่อสารทางวิทยุจำแนกออกเป็น

1) การสื่อสารทางเดียว (one-way radio communications) สถานีต้นทางเป็นสถานีส่งฝ่ายเดียว ส่วนสถานีปลายทางเป็นสถานีรับเท่านั้น เช่น วิทยุกระจายเสียง , โทรทัศน์ และวิทยุติดตามตัว (paging radio)

2) การสื่อสารสองทาง (two-way radio communications) มีสถานีเป็นโครงข่าย (network) ตั้งแต่สองสถานีขึ้นไป แต่ละคู่สถานีสามารถติดต่อโต้ตอบกันได้ด้วยวิธีดังนี้

- ซิมเพล็กซ์ (simplex) แต่ละสถานีจะต้องผลัดกันส่ง ผลัดกันรับ จะโต้ตอบสวนทางกันไม่ได้ เช่น ในการติดต่อทางวิทยุโทรศัพท์ เครื่องวิทยุที่ใช้จะมีทั้งเครื่องรับและเครื่องส่งรวมอยู่ด้วยกัน (radio transceiver)

- ดูเพล็กซ์ (duplex) คู่สถานีสามารถโต้ตอบในเวลาเดียวกัน เช่นเดียวกับการสนทนาทางโทรศัพท์ธรรมดา วิธีนี้จะแยกภาคเครื่องรับออกจากเครื่องส่งโดยใช้ความถี่ในการรับและส่งคน

ละความถี่ที่ไม่รบกวนกัน เครื่องวิทยุจะเปิดไว้ให้รับและส่งอยู่ตลอดเวลาทั้งสองสถานีก็ได้ หรือจะเปิดแต่เครื่องรับเพียงอย่างเดียวแล้วเปิดเครื่องส่งโดยการใช้ระบบสัญญาณ (signalling) จากคู่สถานีไปบังคับก็ได้

- เซมิดูเพล็กซ์ (semi-duplex) สถานีส่งทำงานแบบดูเพล็กซ์ ส่วนคู่สถานีทำงานแบบซิมเพล็กซ์ โดยใช้สองความถี่ การติดต่อสื่อสารทางวิทยุซึ่งมีศูนย์วิทยุควบคุมข่าย (network control) ส่วนใหญ่เป็นแบบซิมเพล็กซ์ ใช้ความถี่เดียวหรือสองความถี่

โครงสร้างวงจรสื่อสารด้วยคลื่นวิทยุ

โครงสร้างวงจรสื่อสารด้วยคลื่นวิทยุ สามารถแบ่งได้เป็น 2 ส่วนๆใหญ่ๆคือ

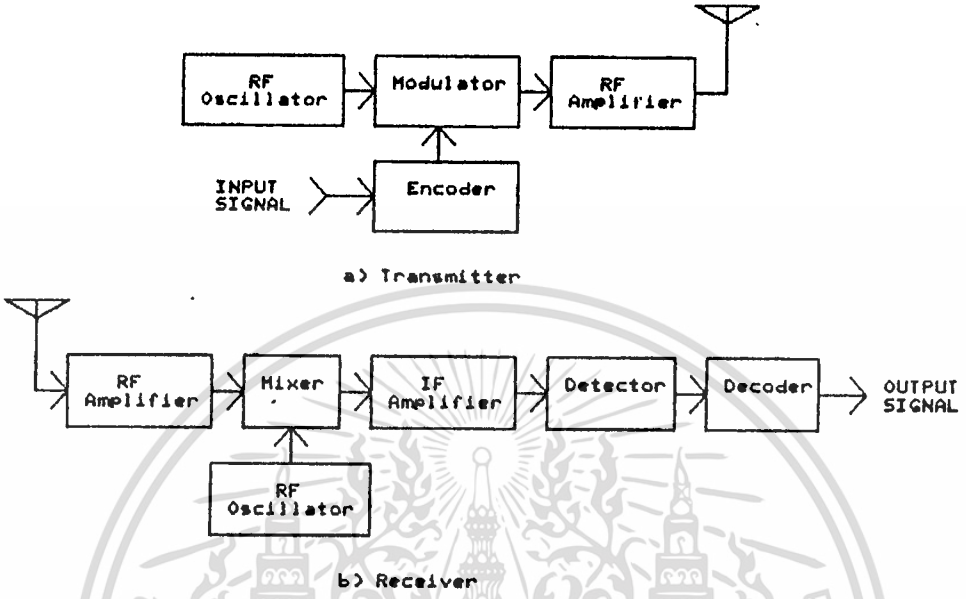
1) ภาคส่งและเข้ารหัส (transmitter and encoder) ข้อมูลที่ต้องการส่งจะถูกนำไปผสมกับสัญญาณความถี่วิทยุแล้วส่งออกอากาศ โดยเครื่องส่งคลื่นวิทยุนี้ประกอบด้วย

- ส่วนเข้ารหัส (encoder)
- ส่วนกำเนิดสัญญาณความถี่วิทยุ (RF oscillator)
- ส่วนขยายความถี่วิทยุและสายอากาศ (RF amplifier and antenna)
- ส่วนผสมคลื่น (modulator)

2) ภาครับ,ส่วนเชื่อมต่อและถอดรหัส (receiver , interface and decoder)

ทำหน้าที่รับคลื่นวิทยุที่ส่งมาจากเครื่องส่งแล้วนำมาแปลรหัสเพื่อนำไปใช้งานต่อไปดังรูปที่ 2.4 ซึ่งอธิบายได้ดังนี้คือ เมื่อสัญญาณวิทยุจากเครื่องส่งมาเข้าสายอากาศของเครื่องรับแล้วจะผ่านภาคขยายของสัญญาณ เพื่อให้มีขนาดสูงขึ้น และมาผสมกับความถี่จากวงจรออสซิลเลเตอร์ จะได้เป็นความถี่ IF และผ่านไปยังภาคขยาย IF แล้วไปยังภาคดีเทคเตอร์ เพื่อแยกสัญญาณความถี่ตัวที่ต้องการออกจากสัญญาณความถี่พาห์ที่ถูกมอดูเลทแล้ว จากนั้นจึงถูกส่งไปถอดรหัส เพื่อนำไปใช้งานต่อไป โดยสรุปแล้วเครื่องรับคลื่นวิทยุประกอบด้วย

- ส่วนกำเนิดความถี่วิทยุ (RF oscillator)
- ส่วนรวมคลื่น (mixer)
- ส่วนขยายสัญญาณความถี่วิทยุ (RF amplifier)
- ส่วนขยายความถี่ไอเอฟ (IF amplifier)
- ส่วนดีเทคเตอร์ (detector)
- ส่วนเชื่อมต่อ (interface)
- ส่วนถอดรหัส (decoder)



รูปที่ 2.4 บล็อคไดอะแกรมวงจรสื่อสารด้วยคลื่นวิทยุ

สาเหตุและปัญหาการรบกวน

สัญญาณที่ไม่พึงปรารถนาในย่านความถี่วิทยุที่เข้ามารบกวนในระบบเราเรียกว่าสัญญาณ RFI ซึ่งเกิดขึ้นได้ง่ายและก็แพร่กระจายออกไปได้ง่ายเช่นเดียวกัน

RFI เป็นคำย่อมาจาก Radio Frequency Interference หรือสัญญาณรบกวนย่านความถี่วิทยุ สัญญาณนี้เกิดขึ้นได้ทุกหนทุกแห่งและสามารถแพร่ออกไปในอากาศ , อวกาศหรืออื่นๆ ได้โดยง่าย RFI จัดเป็นมลภาวะทางอิเลคทรอนิคส์ (electronics pollution) อย่างหนึ่ง ย่านความถี่ของ RFI กว้างมาก ไม่มีขอบเขตจำกัด ดังนั้นจึงสามารถแทรกตัวเข้าไปได้เสมอ RFI อาจเป็นสัญญาณที่ใช้งานของระบบหนึ่ง แต่มีผลเข้าไปกวนอีกระบบหนึ่ง เช่น สัญญาณจากเครื่องรับ-ส่งวิทยุเข้าไปกวนระบบการรับภาพทางทีวี เป็นต้น สาเหตุที่เกิด RFI อาทิเช่น เกิดจากการอาร์ค (arc), เกิดจากเครื่องส่ง , เกิดจากวงจรออสซิลเลเตอร์ เป็นต้น

อุปกรณ์ที่มักผลิตสัญญาณ RFI ได้แก่

1) เครื่องมือสื่อสาร เช่น วิทยุสมัครเล่น , เครื่องส่ง AM-TV , วิทยุสื่อสารของตำรวจ-ทหาร และหน่วยราชการอื่นๆ , ระบบการนำร่องการเดินเรือหรือเครื่องบิน ฯลฯ

2) เครื่องใช้ทางอิเลคทรอนิคส์ เช่น เครื่องหรี่ไฟ , ระบบจุดระเบิดในรถยนต์ , เครื่องใช้ไฟฟ้าภายในบ้าน , หลอดฟลูออเรสเซนต์ , หม้อแปลง ฯลฯ

3) อุปกรณ์ในโรงงานอุตสาหกรรม เช่น มอเตอร์ , ระบบควบคุมระยะทางไกล , ระบบสื่อสารภายใน , เครื่องควบคุมความร้อน เป็นต้น

4) ระบบการส่งจ่ายพลังงานของการไฟฟ้า

5) เครื่องคอมพิวเตอร์

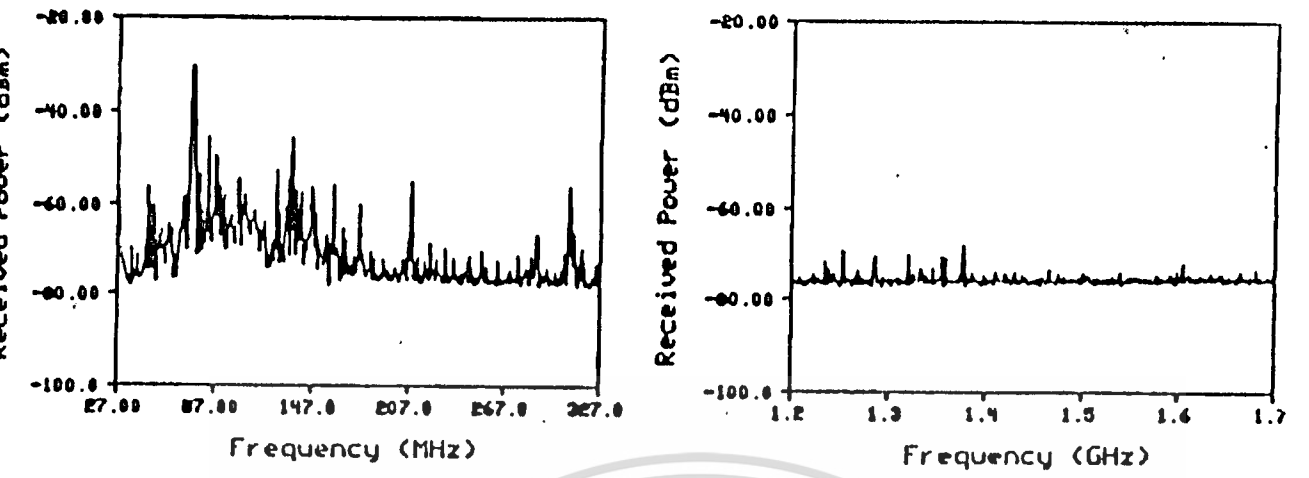
การแก้ไขปัญหาสัญญาณวิทยุรบกวน

ก่อนอื่นต้องทราบว่า ความถี่สัญญาณ RFI ย่านใดเข้ามารบกวน ถ้ารู้ทิศทางให้พยายามหันสายอากาศหนีไปทางอื่นหรือหันให้รับสัญญาณ RFI นั้นให้น้อยที่สุด หรือใช้สายอากาศที่มีอัตราส่วนลูปส่วนหน้าต่อลูปส่วนหลังสูงๆ หรือใช้ฟิลเตอร์ทำการตัดสัญญาณที่รบกวนออก อย่าใช้สายนำสัญญาณที่มีอิมพิแดนซ์ต่างกันมาต่อกันโดยไม่มีระบบการแมตชิ่ง (matching) พยายามอย่าสร้างลูปโดยไม่จำเป็นเพราะถ้าสร้างสายให้เป็นลูปก็จะเสมือนเป็นสายอากาศขึ้นมา อันจะเป็นผลทำให้เกิดการเหนี่ยวนำสัญญาณรบกวนเข้ามาได้

สัญญาณรบกวนสำหรับ AGV [2]

ในการนำ AGV ไปใช้งานยังบริเวณที่มีสัญญาณรบกวน อาทิเช่น โรงงาน นั้น การสื่อสารข้อมูลด้วยคลื่นวิทยุมีโอกาสถูกสัญญาณรบกวนได้ง่าย จากการตรวจวัดปรากฏว่ามีอุปกรณ์อุตสาหกรรมบางชนิดที่สามารถปล่อย RFI ออกมาทำให้เกิดการรบกวนคลื่นวิทยุ RFI ที่เกิดขึ้นส่วนใหญ่อยู่ในย่านความถี่ HF , VHF อุปกรณ์บางอย่าง เช่น เครื่องควบคุมความร้อน บางขณะสามารถผลิตสัญญาณ RFI ในย่านความถี่ที่สูงกว่า 1 GHz

ในรูปที่ 2.5 เป็นตัวอย่างแสดงให้เห็นระดับสัญญาณรบกวนย่านความถี่วิทยุ VHF และ UHF ซึ่งจะเห็นได้ว่าย่านความถี่ UHF นั้นมี RFI ต่ำกว่าความถี่ย่าน VHF ดังนั้นการใช้คลื่นวิทยุในสื่อสารข้อมูลสำหรับ AGV นั้นหากต้องการให้ใช้ได้อย่างกว้างขวาง ก็ควรใช้คลื่นวิทยุในย่านความถี่สูงยิ่ง ซึ่งจะทำให้เราสามารถนำ AGV ไปใช้ในบริเวณที่มีสัญญาณรบกวนได้เป็นอย่างดี นอกจากนี้ในทางปฏิบัติจริงจะต้องมีการระบุข้อกำหนด (specification) สำหรับ AGV เกี่ยวกับความแรงของสัญญาณรบกวนว่าจะต้องไม่เกินกี่ dB uV/m ณ.ความถี่ย่านที่ใช้งานอีกด้วย



a) 27 MHz to 327 MHz (VHF) b) 1200 MHz to 1700 MHz (UHF)

รูปที่ 2.5 แสดงสเปกตรัมของสัญญาณรบกวนซึ่งวัดห่างจากกระบอกสูบของเครื่องจักร 4 เมตร

<u>การเรียกใช้ AGV จากสถานีปฏิบัติงาน</u>

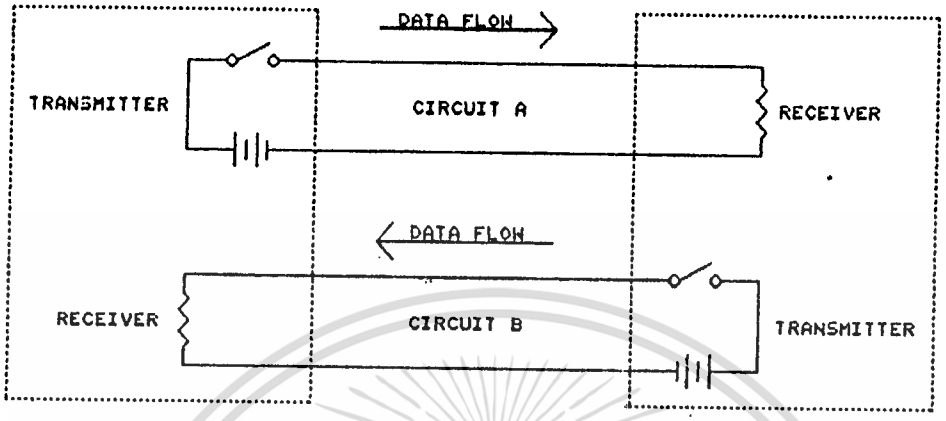
การขนถ่ายวัสดุหรือชิ้นส่วนการผลิตในโรงงานหลายประเภทนั้น ไม่สามารถกำหนดเวลาได้แน่นอน โดยเฉพาะโรงงานประเภทกึ่งอัตโนมัติ ในการนำ AGV มาใช้กับโรงงาน จึงต้องมีวิธีที่จะเรียกใช้ได้โดยตรงและมีประสิทธิภาพ ในโรงงานแบบเดิมที่ใช้แรงคน การขนถ่ายมีทั้งการใช้รถเข็น , รถลากและรถโฟร์คลิฟท์ ฯลฯ การขนถ่ายจึงล่าช้าขาดประสิทธิภาพและสิ้นเปลืองค่าแรงมากกว่า การเรียกใช้ก็กระทำไม่สะดวก วิทยานิพนธ์นี้ได้ใช้เทคนิคในการเรียกใช้ AGV โดยการกดคีย์เรียกจากสถานีปฏิบัติงาน (work station) โดยตรง สัญญาณการกดคีย์จะถูกส่งมาตามสายส่งแบบกระแสวนลูป (current loop) เข้าสู่คอมพิวเตอร์ ซึ่งจะต้องมีวงจรคอยรับข้อมูล ดังจะได้กล่าวต่อไป

การอินเตอร์เฟสแบบกระแสวนลูป 20 มิลลิแอมป์ [4]

พอร์ทอินพุท/เอาท์พุท แบบอนุกรมของคอมพิวเตอร์และอุปกรณ์เพอร์ริเฟอรัล (peripheral) จำนวนมากใช้การอินเตอร์เฟสแบบนี้

วงจรแบบกระแสวนลูปจะส่งข้อมูลเป็นบิทๆเรียงต่อกันแบบอนุกรม โดยระดับลอจิกถูกแทนด้วยการเปิดหรือปิดวงจร โดยทั่วไปนั้น วงจรที่ใช้ในการอินเตอร์เฟสกับคอมพิวเตอร์หรือเทอร์มินัลแบบกระแสวนลูปจะประกอบด้วยสายลองเส้น โดยกระแสจะไหลผ่านแต่ละเส้นคนละทิศทางกัน (กระแสจะใช้แทนลอจิก) ดังรูปที่ 2.6 ตามทฤษฎีแล้วจำนวนอุปกรณ์ที่อยู่ในลูปมีจำนวนเท่าใดก็ได้ขึ้นอยู่กับตัวจ่ายไฟที่ต้องจ่ายกระแสได้เพียงพอตามมาตราฐาน อุปกรณ์แต่ละตัวในลูปจะเหมือนตัวต้านทาน ดังนั้นถ้าเราเพิ่มอุปกรณ์ลงในลูปหรือเพิ่มขนาดของลูป (ความยาวสาย) ความต้านทานจะเพิ่มขึ้นด้วย จึงต้อง

เพิ่มแรงดันขึ้นเพื่อรักษากระแสไว้ที่ 20 มิลลิแอมป์



รูปที่ 2.6 ลักษณะการอินเตอร์เฟสเทอร์มินัลของคอมพิวเตอร์แบบกระแสวนลูป 20 มิลลิแอมป์

การอินเตอร์เฟสแบบกระแสวนลูปประกอบด้วยแหล่งจ่ายกระแส (current source) ,สวิทช์ตัดต่อกระแสหรือตัวส่งข้อมูล (transmitter) และวงจรตรวจจับกระแสหรือตัวรับข้อมูล (receiver) กรณีที่แหล่งจ่ายกระแสอยู่ที่ตัวส่ง ตัวส่งจะถูกเรียกว่า active transmitter แต่ถ้าแหล่งจ่ายกระแสอยู่ที่ตัวรับ ตัวรับจะถูกเรียกว่า active receiver

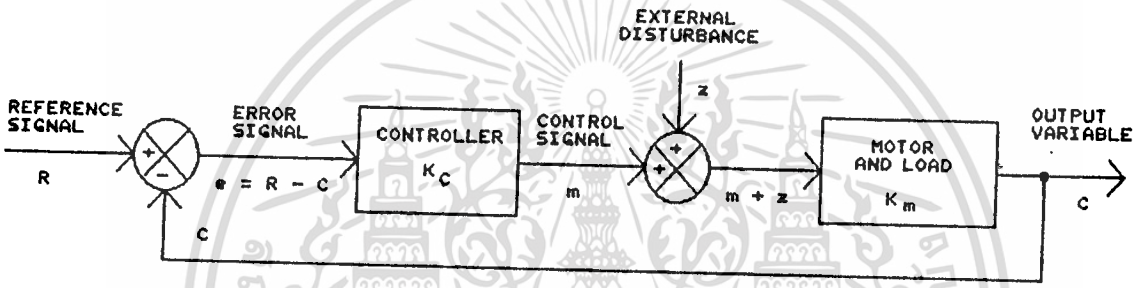
ในการส่งข้อมูลในระยะทางสั้นๆ (< 2000 ฟุต) และอัตราบอดต่ำกว่า 9600 บิท/วินาที การอินเตอร์เฟสแบบกระแสวนลูป 20 มิลลิแอมป์ สามารถทำงานได้ดีกว่าการอินเตอร์เฟสแบบ RS-232-C และมักใช้การอินเตอร์เฟสแบบนี้ในกรณีที่ต้องเดินสายผ่านบริเวณที่มีสัญญาณรบกวนมากๆ การอินเตอร์เฟสแบบนี้ยังใช้สำหรับแยกกราวน์ของอุปกรณ์ 2 ตัวออกจากกัน นอกจากนี้ยังสร้างได้ง่ายรวมทั้งราคาก็ไม่แพง

การอินเตอร์เฟสแบบกระแสวนลูปจะใช้สาย 4 เส้นได้แก่ สายส่งข้อมูล (transmit data), สายนำสัญญาณกลับ (transmit return),สายรับข้อมูล (receive data) และสายนำสัญญาณกลับ (receive return) ปัญหาหลักที่เกิดกับการอินเตอร์เฟสแบบนี้คือการขาดมาตราฐานทางกลและทางไฟฟ้า เช่น กรณีตัวส่งแบบ active มีความต้านทานภายใน 15 กิโลโอห์ม เพื่อให้ได้กระแส 20 มิลลิแอมป์จึงต้องใช้แหล่งจ่ายไฟที่มีขนาดถึง 300 โวลท์ ส่วนตัวรับแบบ passive มีความต้านทานภายใน 1 กิโลโอห์ม ถ้าต่ออุปกรณ์ 2 ตัวเข้าด้วยกันจะทำให้ตัวรับเสียหายได้เนื่องจากจากแหล่งจ่ายไฟมีขนาดใหญ่เกินไป ในทางปฏิบัติค่าความต้านทานในลูปไม่ควรมีค่าเกิน 30 โอห์ม

การควบคุมความเร็วมอเตอร์กระแสตรง [7]

ในระบบการควบคุมมอเตอร์นั้นจุดมุ่งหมายของตัวคอนโทรลเลอร์คือ การควบคุมตัวแปรเอาท์พุทให้เข้าสู่ค่าที่กำหนดไว้และสามารถรักษาค่าตัวแปรดังกล่าวให้อยู่ที่ค่ากำหนดนั้นได้ เพื่อให้เป็นตามเป้าหมายนี้ ตัวคอนโทรลเลอร์จะต้องแก้ไขผลกระทบของสัญญาณรบกวนจากภายนอกในวิถีทางที่เหมาะสม

ระบบควบคุมมอเตอร์พื้นฐานแสดงในรูปที่ 2.7 ตัวคอนโทรลเลอร์จะปรับค่าตัวแปรเอาท์พุท C โดยใช้สัญญาณควบคุม m เพื่อไปทำให้สัญญาณ R-C มีค่าน้อยที่สุดเท่าที่จะทำได้ สัญญาณจากภายนอกที่กระทำต่อระบบจะเข้ามารวมกับตัวแปรได้ผลรวมเป็น (m + z)



รูปที่ 2.7 บล็อคไดอะแกรมลูปการควบคุมมอเตอร์

ถ้าระบบตามรูปที่ 2.7 เป็นเชิงเส้นเราจะได้ว่า

$$m = K_c (R - C) \text{ และ } C = K_m (m + z) \quad \ldots\ldots(2.1)$$

เมื่อ K_m เป็นมอเตอร์และโหลด ดังนั้นเราจะหาตัวแปรเอาท์พุทได้เป็น

$$C = K_c K_m R/(1 + K_c K_m) + K_m z/(1 + K_c K_m) \quad \ldots\ldots(2.2)$$

คุณสมบัติของ PI คอนโทรลเลอร์

การควบคุมแบบนี้เราจะหาสัญญาณควบคุม (m) ได้ตามสมการคือ

$$m = K_p e + (1/T_i) \int_0^T e\,dt + m(0) \quad \ldots\ldots(2.3)$$

โดยที่

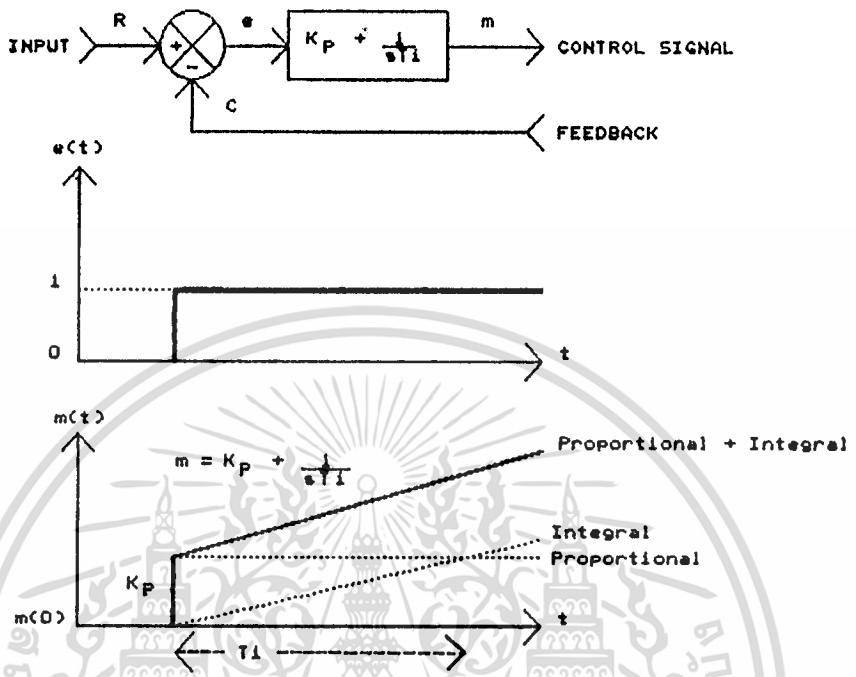
K_p = อัตราขยายของ P คอนโทรลเลอร์ (Proportional gain)

e = สัญญาณเออเรอร์ (error signal)

Ti = อินติกรัลไทม์ (Integral time)

m(0) = เอาท์พุทของตัวคอนโทรลเลอร์เมื่อเออร์เรอร์เป็นศูนย์

ลักษณะของสัญญาณควบคุม m ที่ได้จาก PI คอนโทรลเลอร์แสดงไว้ในรูปที่ 2.8



รูปที่ 2.8 แสดงลักษณะของสัญญาณควบคุม m ที่ได้จาก PI คอนโทรลเลอร์

ข้อดีของ PI คอนโทรลเลอร์

การควบคุมแบบนี้จะให้คุณสมบัติเป็นทั้งแบบ P คอนโทรลเลอร์ และแบบ I คอนโทรลเลอร์รวมกัน ซึ่งทำให้ค่าออฟเซ็ท (offset) ที่เกิดขึ้นกรณีมีการเปลี่ยนแปลงโหลดของ P คอนโทรลเลอร์หมดไป

ข้อเสียของ PI คอนโทรลเลอร์

ระบบอาจจะไม่มีเสถียรภาพกรณีที่ค่า Ti มีค่าน้อยไม่เหมาะสมกับกระบวนการ (process) ที่มีไทม์แล็ก (time lag) มากๆ เพราะการตอบสนองของตัวแปรกระบวนการ (C) จะช้ามาก ไม่สามารถแก้ไขค่าเออร์เรอร์ได้ทันเวลา

ระบบควบคุมแบบ PI จะใช้ได้กับระบบที่มีการเปลี่ยนแปลงโหลดมากๆ แต่โหลดควรเปลี่ยนอย่างช้าๆ เมื่อเทียบกับ Ti เพื่อป้องกันไม่ให้เกิดการออสซิเลท (oscillate) อันเนื่องจากโอเวอร์ชูท (overshoot) ของการอินติกรัล และระบบมักจะให้โอเวอร์ชูทสูงก่อนที่จะเข้าสู่ค่าคงที่

สมการทางไฟฟ้าของมอเตอร์กระแสตรง

$$v(t) = L_a di_a(t)/dt + R_a i_a(t) + K_\varepsilon \omega(t) \qquad \ldots.(2.4)$$

โดยที่

$v(t)$ = โวลท์เตจที่ป้อนให้มอเตอร์

$i_a(t)$ = กระแสอาร์เมเจอร์ (armature current)

$\omega(t)$ = ความเร็วเชิงมุม (angular velocity) ของมอเตอร์

K_E = ค่าคงที่ของโวลท์เตจ (voltage constant)

L_a = ค่าความเหนี่ยวนำ (inductance) ของขดลวดอาร์เมเจอร์

R_a = ความต้านทาน (resistance) ของขดลวดอาร์เมเจอร์

สมการไดนามิคของมอเตอร์กระแสตรง

$$T_G(t) = Jd\omega(t)/dt + \beta\omega(t) + T_F(t) + T_L(t) \quad ....(2.5)$$

โดยที่

$T_G(t)$ = แรงบิดกำเนิดจากมอเตอร์ (generated torque)

$T_F(t)$ = แรงบิดเสียดทานภายใน (internal friction torque)

$T_L(t)$ = แรงบิดโหลด (load torque)

J = โมเมนต์แรงเฉื่อย (moment of inertia) รวมระหว่างโมเมนต์แรงเฉื่อยของมอเตอร์ (J_M) และโมเมนต์แรงเฉื่อยของโหลด (J_L)

β = สัมประสิทธิ์ของวิสคอสแด้มพ์ปิ้ง (viscous damping coefficient)

สำหรับแรงบิดที่กำเนิดโดยมอเตอร์ T_G นั้นจะเป็นสัดส่วนกับกระแสอาร์เมเจอร์ i_a คือ

$$T_G(t) = K_T i_a(t) \quad ....(2.6)$$

โดยที่

K_T = ค่าคงที่แรงบิด (torque constant) ของมอเตอร์

สมการ (2.4),(2.5) และ (2.6) สามารถเขียนในรูปสมการลาปลาซ (laplace) ได้เป็น

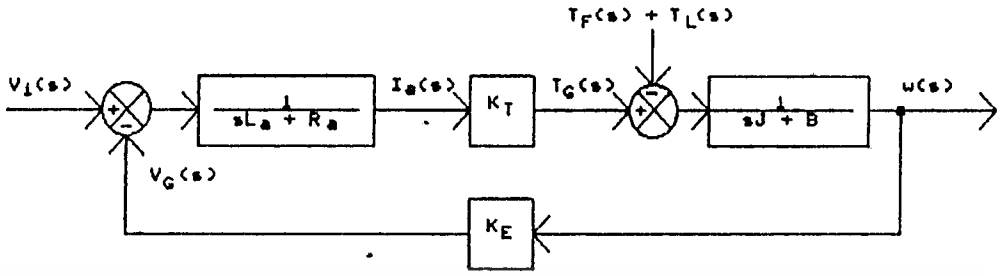
$$V(s) = (sL_a + R)I_a(s) + K_E\omega(s) \quad ....(2.7)$$

$$T_G(s) = (Js + \beta)\omega(s) + T_F(s) + T_L(t) \quad ....(2.8)$$

$$T_G(s) = K_T I_a(s) \quad ....(2.9)$$

บล็อคไดอะแกรมโมเดลของมอเตอร์กระแสตรง

จากสมการ (2.7) , (2.8) และ (2.9) สามารถเขียนบล็อคไดอะแกรมโมเดลของมอเตอร์กระแสตรงได้ดังรูปที่ 2.9



รูปที่ 2.9 บล็อคไดอะแกรมโมเดลมอเตอร์กระแสตรง

จากบล็อคไดอะแกรมตามรูปที่ 2.9 เป็นระบบที่มี 2 อินพุท 1 เอาท์พุท ซึ่งความเร็วเอาท์พุท $\omega(s)$ ของระบบนี้สามารถเขียนได้เป็น

$$\omega(s) = G1(s)V(s) + G2(s)[\, T_F(s) + T_L(s) \,] \qquad ....(2.10)$$

โดยที่

$G1(s)$ = ทรานสเฟอร์ฟังก์ชั่นของความเร็วเชิงมุมกับโวลท์อินพุท ($T_F(s) + T_L(s) = 0$)

$$G1(s) = \omega(s)/V(s)$$

$$G1(s) = K_T/[\,(sL_a + R_a)(sJ + \beta) + K_E K_T\,] \qquad ....(2.11)$$

$G2(s)$ = ทรานสเฟอร์ฟังก์ชั่นของความเร็วเชิงมุมกับแรงบิดโหลดเมื่อกำหนดให้ $V(s) = 0$

$$G2(s) = \omega(s)/[T_F(s) + T_L(s)]$$

$$G2(s) = -\,[1/(Js + \beta)]/[1 + \{ K_T K_E/((Js + \beta)(L_a s + R_a))\}] \qquad (2.12)$$

ทาโคมิเตอร์ (Tachometer)

ทาโคมิเตอร์เป็นเครื่องมือที่สามารถแปลงพลังงานกลไปเป็นพลังงานไฟฟ้า และให้กำเนิดเอาท์พุทโวลท์เตจที่เป็นสัดส่วนกับขนาดของความเร็วเชิงมุม ทาโคมิเตอร์สามารถใช้เป็นตัวแสดงความเร็วของเพลามอเตอร์ (shaft) หรือใช้เป็นตัวป้อนกลับสำหรับการควบคุมความเร็วซึ่งจะทำให้เสถียรภาพของระบบดีขึ้น

ลักษณะการทำงานของทาโคมิเตอร์กับดีซีเยนเนอเรเตอร์จะเหมือนกันเพียงแต่ต่างกันที่การใช้งาน กล่าวคือทาโคมิเตอร์มีขอบเขตการใช้งานเพียงเป็นตัววัดความเร็วของเพลา แล้วส่งผลออกมาเป็นสัญญาณเอาท์พุทที่อ่านค่าได้ง่ายเท่านั้น สัญญาณนี้จะถูกนำไปป้อนกลับเพื่อเปรียบเทียบกับสัญญาณอินพุท ซึ่งจะได้สัญญาณเออเรอร์ออกมา จากนั้นสัญญาณเออเรอร์จะถูกส่งไปเข้าตัวคอนโทรลเลอร์เพื่อทำการ

ไปควบคุมความเร็วต่อไป

ลักษณะพื้นฐานของทาโคมิเตอร์คือจะให้เอาท์พุทโวลท์เตจเป็นสัดส่วนกับความเร็วของโรเตอร์ ดังนั้นคุณลักษณะทางไดนามิคส์ของทาโคมิเตอร์สามารถแสดงได้ด้วยสมการ

$$v_G(t) = K_G \omega(t) \qquad \ldots.(2.13)$$

โดยที่

$v_G(t)$ = เอาท์พุทโวลท์เตจของทาโคมิเตอร์

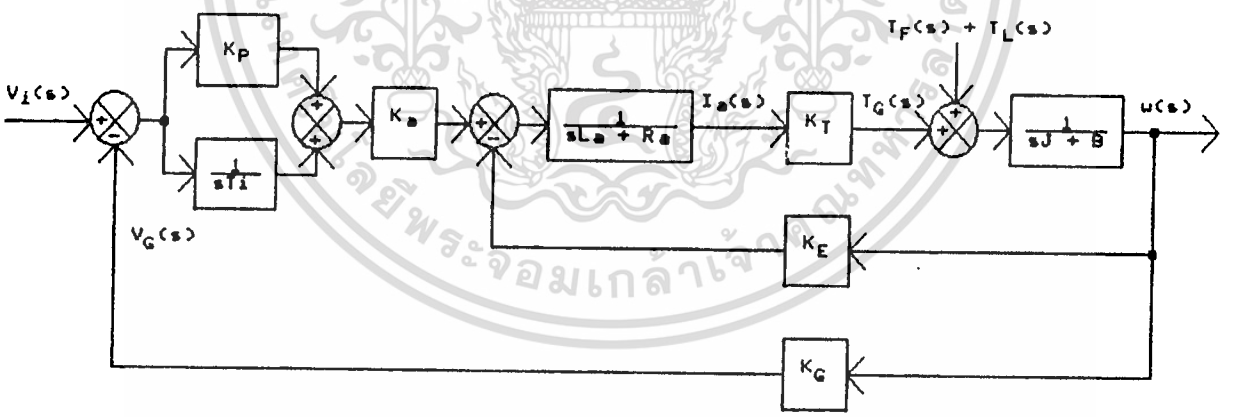
K_G = ค่าคงที่ของทาโคมิเตอร์

ทรานสเฟอร์ฟังก์ชั่นของทาโคมิเตอร์จะหาได้โดยการแปลงสมการ (2.13) เป็นสมการลาปลาซ (laplace) ซึ่งจะได้ว่า

$$V_G(s)/\omega(s) = K_G \qquad \ldots.(2.14)$$

บล็อคไดอะแกรมการควบคุมความเร็วมอเตอร์กระแสตรงด้วย PI คอนโทรลเลอร์

เราสามารถสรุปรูปแบบในการควบคุมความเร็วได้ดังบล็อคไดอะแกรมตามรูปที่ 2.10



รูปที่ 2.10 บล็อคไดอะแกรมการควบคุมความเร็วมอเตอร์กระแสตรงด้วย PI คอนโทรลเลอร์

โดยที่

K_a = อัตราขยายของวงจรสวิทชิ่งเซอร์โวแอมป์

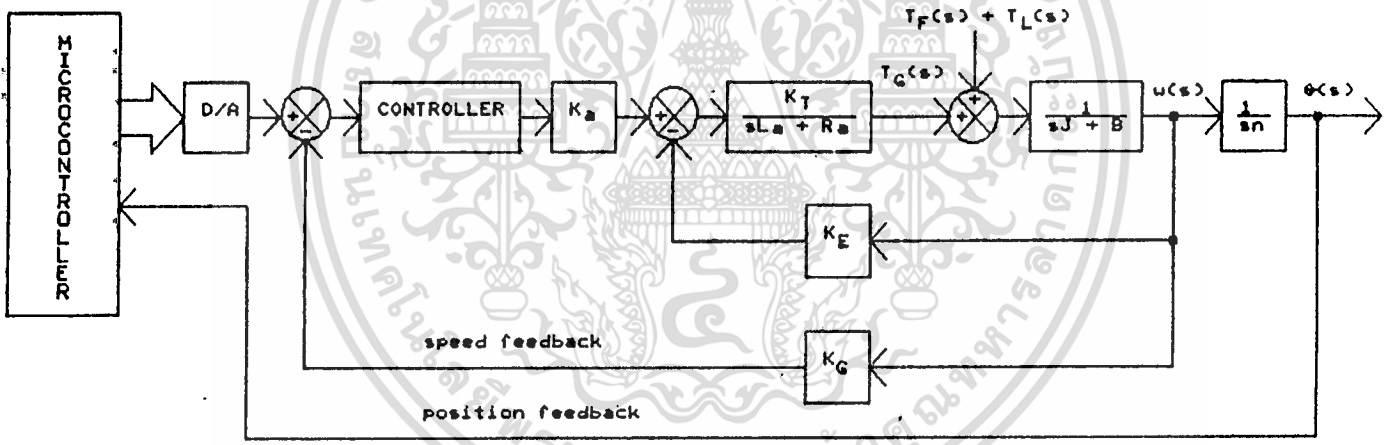
การควบคุมตำแหน่งมอเตอร์กระแสตรง

ตามปกติ AGV จะขับเคลื่อนโดยใช้มอเตอร์กระแสตรง ซึ่งจะต้องมีการควบคุมตำแหน่งในการเคลื่อนที่ ทั้งนี้ก็เพื่อให้ AGV สามารถหยุดยังตำแหน่งเป้าหมายที่ต้องการได้อย่างถูกต้องแม่นยำ

หลักการโดยทั่วไป [8]

ในรูปที่ 2.11 แสดงการควบคุมตำแหน่งมอเตอร์กระแสตรงโดยใช้ไมโครคอนโทรลเลอร์ทำหน้าที่นับค่าพัลซ์ทางตำแหน่งเพื่อนำไปเปรียบเทียบกับตำแหน่งเป้าหมาย (position setpoint) ซึ่งถูกกำหนดไว้แล้วในหน่วยความจำ

ไมโครคอนโทรลเลอร์จะนำค่าผลต่างทางตำแหน่ง (position error) ระหว่างตำแหน่งปัจจุบันกับตำแหน่งเป้าหมายไปเปิดตารางความเร็ว เพื่อนำค่าความเร็วของมอเตอร์ที่เหมาะสมออกไปใช้ ทั้งนี้ก็เพื่อให้มอเตอร์สามารถหยุดยังตำแหน่งเป้าหมายได้อย่างถูกต้องแม่นยำ



รูปที่ 2.11 บล็อคไดอะแกรมการควบคุมตำแหน่งมอเตอร์กระแสตรงโดยใช้ไมโครคอนโทรลเลอร์

ในการควบคุมตำแหน่งของมอเตอร์กระแสตรงซึ่งมีระบบการป้อนกลับนั้น ความเร็วมอเตอร์ในช่วงที่มีการเพิ่มความเร็ว (acceleration), ลดความเร็ว (deceleration) และช่วงความเร็วคงที่ (slewing) นั้นมีความสำคัญมาก

ความสัมพันธ์ระหว่างตำแหน่งโรเตอร์เชิงมุม $\theta(t)$ กับความเร็วเชิงมุม $\omega(t)$

$$\theta(t) = \int \omega(t)dt \qquad \ldots\ldots(2.15)$$

จะเห็นว่าถ้าเราจะทำการควบคุมตำแหน่งเชิงมุม $\theta(t)$ ได้ เราก็จำเป็นต้องควบคุมความเร็วเชิงมุม $\omega(t)$ ด้วย

จากสมการ (2.5) ซึ่งเป็นสมการไดนามิคส์ของมอเตอร์กระแสตรง ทำการพิจารณาการควบคุมความเร็วในช่วงเพิ่มความเร็ว (acceleration) และลดความเร็ว (deceleration) นั้น เราจำเป็นต้องมีการควบคุมแรงบิด T_g กล่าวคือ

ก) การเพิ่มความเร็ว (acceleration)

ตามปกติมอเตอร์จะถูกเพิ่มความเร็วจากการป้อนแรงบิด T_g จากสมการ (2.5) หาคำตอบทั่วไปของสมการได้ว่า

$$\omega(t) = [(T_g - T_F - T_L)/\beta][1 - \exp(-\beta/J)t] \quad ....(2.16)$$

ข) ขณะความเร็วคงที่ (slewing)

จากเงื่อนไขคือ $d\omega(t)/dt = 0$ จากสมการ (2.5) จะได้คำตอบคือ

$$\omega(t) = T_g/\beta \quad ....(2.17)$$

ค) การลดความเร็ว (deceleration)

ในช่วงนี้จะไม่คิดเทอม $\beta\omega$ กล่าวคือสมการ (2.5) จะกลายเป็น

$$T_g = Jd\omega(t)/dt \quad ....(2.18)$$

ซึ่งจะได้คำตอบของสมการคือ

$$\omega(t) = \omega_c - (T_g/J)t \quad ....(2.19)$$

โดยที่

ω_c = ค่าคงที่ของความเร็ว

กรอบความเร็วสำหรับควบคุมตำแหน่ง (Speed profile for position control)

ลักษณะความเร็วตั้งแต่มอเตอร์เริ่มหมุนจนถึงหยุดที่ตำแหน่งเป้าหมายแสดงในรูปที่ 2.12 จากรูปจะแบ่งช่วงความเร็วออกได้เป็น 5 ช่วงดังนี้คือ

1) ช่วงเร่งความเร็ว (acceleration)

ในช่วงนี้ความเร็วของมอเตอร์จะเปลี่ยนแปลงตามสมการที่ (2.16) แรงบิด T_g สามารถหาได้จากสมการ (2.6)

2) ช่วงความเร็วคงที่ (slewing)

เมื่อความเร็วของโรเตอร์เพิ่มถึงค่าสูงสุด จะเริ่มเห็นผลของการควบคุมความเร็วแบบป้อนกลับโดยความเร็วจะเข้าสู่ค่าคงที่

3) ช่วงลดความเร็ว (deceleration)

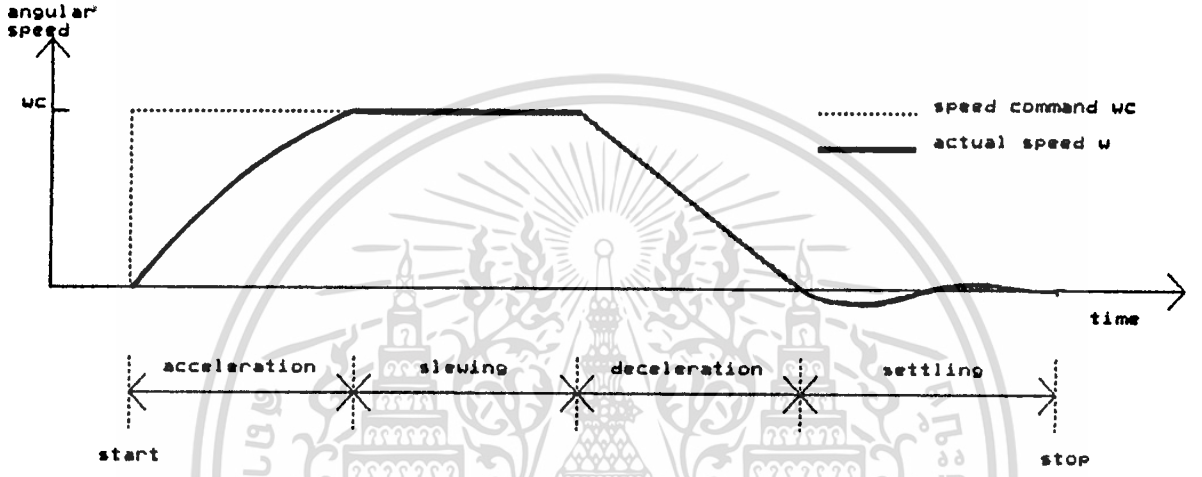
ในช่วงนี้ความเร็วเอาท์พุทจะเป็นไปตามสมการ

$$\omega_c = \sqrt{2 \cdot q \cdot \ddot{r}} \quad ....(2.20)$$

โดยที่

q = ค่าผลต่างทางตำแหน่งระหว่างตำแหน่งที่อยู่ปัจจุบันกับตำแหน่งเป้าหมาย มีหน่วยเป็นจำนวนพัลซ์ (ตำแหน่งการเคลื่อนที่จะวัดโดยใช้อินคริเมนทัลโรตารี่เอนโคดเดอร์)

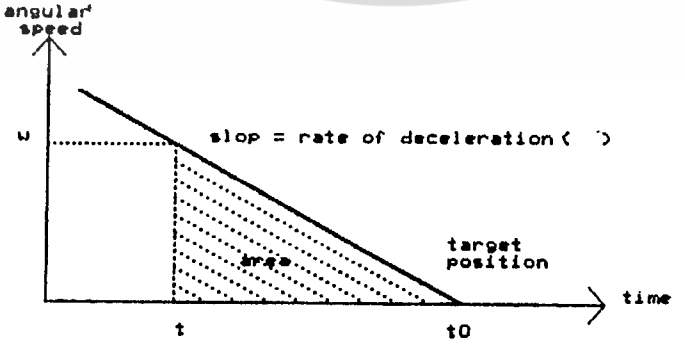
α = อัตราการลดลงของความเร็ว (rate of deceleration)



รูปที่ 2.12 กรอบความเร็วในการควบคุมตำแหน่ง

จากสมการ (2.6) และสมการ (2.19) จะได้ว่า

$$\alpha = K_T i_a(t)/J \qquad \ldots.(2.21)$$



รูปที่ 2.13 แสดงความสัมพันธ์ระหว่าง ω , α และ q

ในรูปที่ 2.13 แสดงความสัมพันธ์ระหว่างความเร็วเชิงมุม ω ,อัตราการลดความเร็ว α และค่าผลต่างทางตำแหน่ง q โดยจากรูปเมื่อมอเตอร์ลดความเร็วลง ความเร็วเชิงมุม ω ที่เวลา t จะมีค่าเท่ากับ

$$\omega(t) = (t_0 - t) \alpha \qquad \ldots.(2.22)$$

เมื่อ t_0 เป็นเวลาขณะที่ถึงมอเตอร์หมุนไปถึงตำแหน่งเป้าหมายพอดี

ตามรูปที่ 2.13 ระยะทางในการเคลื่อนที่จากตำแหน่งที่เวลา t ถึงตำแหน่งเป้าหมายที่เวลา t_0 จะมีค่าเท่ากับพื้นที่สามเหลี่ยมใต้เส้นกราฟ โดยจะมีค่าเท่ากับ

$$q = 1/2 (t_0 - t) \alpha (t_0-t) \qquad \ldots.(2.23)$$

แทนค่า (t0 - t) จากสมการ (2.22) ลงในสมการ (2.23) จะได้ว่า

$$\omega(t) = \sqrt{2 \cdot q \cdot \alpha} \qquad \ldots.(2.24)$$

ซึ่งจะเห็นได้ว่าเป็นสมการเดียวกันกับสมการ (2.20)

ในการควบคุมตำแหน่งการเคลื่อนที่ของมอเตอร์ให้หยุดยังตำแหน่งเป้าหมายอย่างนิ่มนวลนั้น ตามปกติเราจะทำการลดความเร็วของมอเตอร์ลงก่อนที่มอเตอร์จะหมุนไปถึงตำแหน่งเป้าหมาย สมการ (2.24) เป็นการลดความเร็วของมอเตอร์ที่สอดคล้องกับคุณลักษณะของระบบ ในการควบคุมตำแหน่งมอเตอร์โดยใช้ไมโครคอนโทรลเลอร์ จะนำสมการนี้ไปสร้างเป็นตารางความเร็ว เพื่อใช้ในการกำหนดค่าความเร็วที่เหมาะสมกับระยะทาง การใช้วิธีเปิดตาราง (lookup table) มีข้อดีคือทำให้ไม่ต้องเสียเวลาในการคำนวณ

4) ช่วงก่อนเข้าสู่ค่าคงที่ (settling)

ตามอุดมคติแล้วถ้าพิจารณาสมการ (2.24) มอเตอร์จะหยุดที่ตำแหน่งเป้าหมายพอดี โดยไม่เกิดการแกว่งแต่อย่างใด แต่ในทางปฏิบัตินั้นเป็นไปได้มากที่มอเตอร์จะหมุนออกนอกตำแหน่งเป้าหมายทั้งนี้เพราะข้อมูลความเร็วที่ป้อนมอเตอร์นั้นมิได้เป็นฟังก์ชั่นเชิงเส้นดังสมการ (2.24) เนื่องจากข้อมูลความเร็วถูกเก็บไว้ในหน่วยความจำของไมโครคอนโทรลเลอร์ เวลาในช่วงนี้จะเริ่มนับตั้งแต่จุดที่มอเตอร์หมุนถึงตำแหน่งเป้าหมายและเกิดการแกว่งไปจนกว่าจะเข้าสู่ค่าขอบเขตที่ยอมรับได้ของตำแหน่งเป้าหมาย

5) หยุด (stop)

อินคริเมนทัลโรตารี่เอนโคดเดอร์ (Incremental rotary encoder) [9]

ในการควบคุมตำแหน่งของมอเตอร์จำเป็นต้องมีตัวเซนเซอร์ทำหน้าที่วัดตำแหน่งเพื่อนำไปป้อนกลับ สำหรับตัวเซนเซอร์ที่นิยมใช้กันก็คืออินคริเมนทัลโรตารี่เอนโคดเดอร์ซึ่งจัดเป็นออปติคอล เอนโคดเดอร์ (optical encoder) แบบหนึ่ง เอนโคดเดอร์นี้จะสร้างสัญญาณพัลซ์ที่แปรผันโดยตรงกับการหมุน

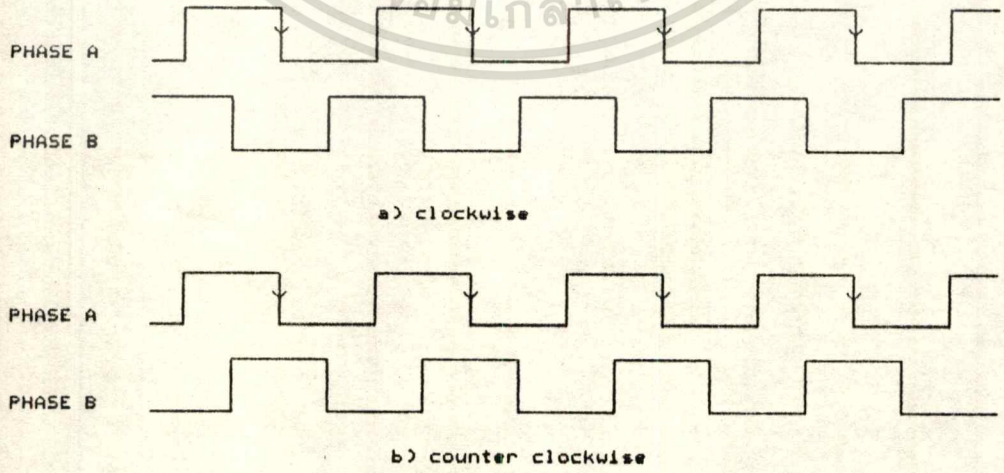
ของเพลา

อินคริเมนทัลโรตารี่เอนโคดเดอร์ตามรูปที่ 2.14 ประกอบด้วยส่วนที่สำคัญคือ ตัวกำเนิดแสง (light source) , จานหมุน (disc) , ตัวเซนเซอร์แสง (light sensor) และวงจรอิเลคทรอนิคส์ บนแผ่นจานหมุนจะเจาะเป็นช่องๆ ตัวกำเนิดแสงอาจจะเป็นหลอดไฟหรือ LED ก็ได้

ความละเอียดของเอนโคดเดอร์คือจำนวนสัญญาณพัลซ์ต่อการหมุนเพลา 1 รอบโดยทั่วไปมีค่าความละเอียดตั้งแต่ 15 ถึง 10000 พัลซ์ จำนวนพัลซ์ต่อ 1 รอบของสัญญาณที่เอนโคดเดอร์สร้างออกมาจะเท่ากับจำนวนช่องบนแผ่นจานหมุน

รูปที่ 2.14 อินคริเมนทัลโรตารี่เอนโคดเดอร์

อินคริเมนทัลโรตารี่เอนโคดเดอร์ จะให้เอาท์พุทพัลซ์ออกมา 2 ช่อง (channel) คือ A และ B ดังแสดงในรูปที่ 2.15



รูปที่ 2.15 แสดงสัญญาณเอาท์พุทของเอนโคดเดอร์ 2 ช่องมีมุมเฟสต่างกัน 90 องศาไฟฟ้า

จากรูปเฟสของสัญญาณ 2 ช่องนี้จะต่างกัน 90 องศาทางไฟฟ้า เราเรียกสัญญาณ 2 ช่องนี้ว่าเป็นควอดราเจอร์ (quadrature) กันซึ่งเหมาะที่จะใช้ในการรับรู้ทิศทางการหมุนของเพลา

การคำนวณระยะทางการเคลื่อนที่ของ AGV

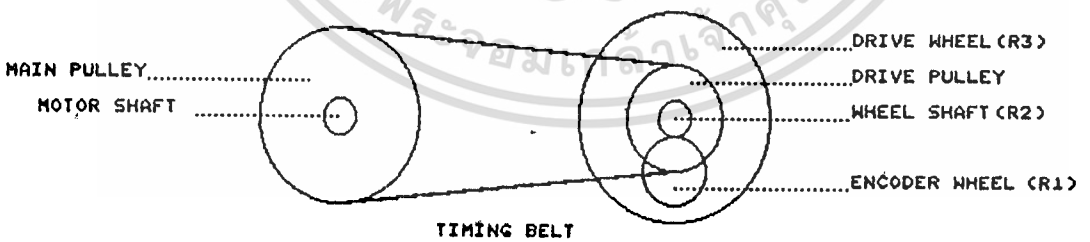
สำหรับเอนโคดเดอร์ที่ใช้มีความละเอียดทางตำแหน่งเท่ากับ 450 พัลซ์/รอบ พิจารณาระบบสายพานส่งกำลังตามรูปที่ 2.16

- รัศมีของล้อเอนโคดเดอร์ (encoder wheel) เท่ากับ R1 เซ็นติเมตร
- รัศมีเพลาล้อขับ (drive wheel shaft) เท่ากับ R2 เซ็นติเมตร
- รัศมีล้อขับ (drive wheel) เท่ากับ R3 เซ็นติเมตร

<u>คำนวณ</u>

เอนโคดเดอร์หมุน 1 รอบ จำนวนพัลซ์ที่สร้างขึ้น เท่ากับ 450 พัลซ์
ล้อเอนโคดเดอร์หมุน 1 รอบ เพลาล้อขับหมุน เท่ากับ R1/R2 รอบ
เพลาล้อขับหมุน 1 รอบ ล้อขับจะหมุนไป เท่ากับ 1 รอบ
เพลาล้อขับหมุน R1/R2 รอบ ล้อขับจะหมุนไป เท่ากับ R1/R2 รอบ
ล้อขับหมุน R1/R2 รอบ คิดเป็นระยะทาง เท่ากับ 2πR3(R1/R2) เซ็นติเมตร

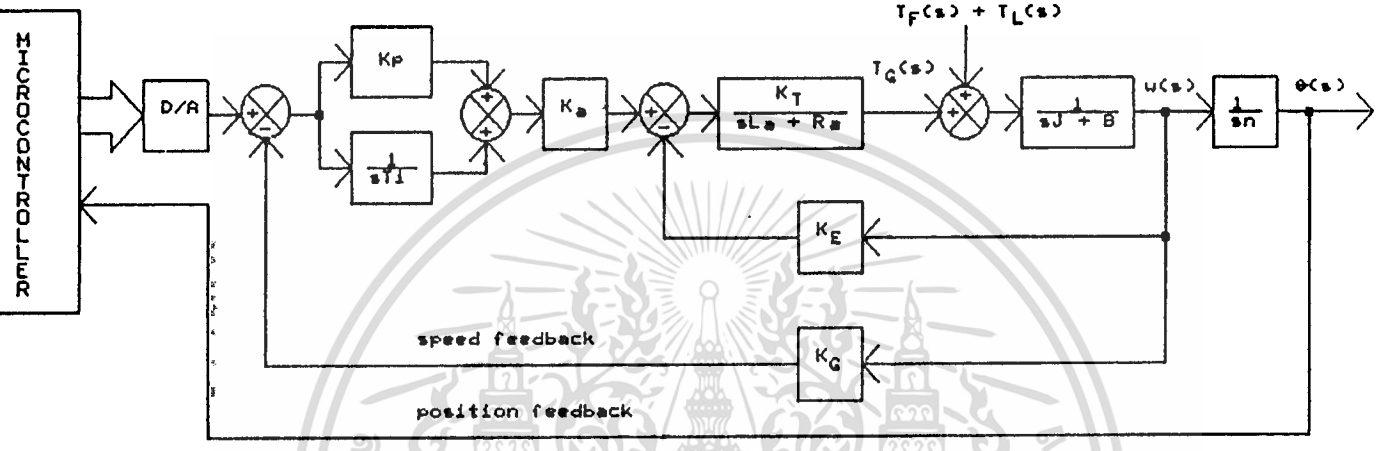
สรุปได้ว่า AGV เคลื่อนที่ไป 2πR3(R1/R2) เซ็นติเมตร จะเท่ากับจำนวนพัลซ์ = 450 พัลซ์



รูปที่ 2.16 แสดงระบบสายพานส่งกำลังสำหรับขับเคลื่อน AGV

บล็อคไดอะแกรมลูปการควบคุมทั้งตำแหน่งและความเร็วของมอเตอร์กระแสตรง

จากรูปที่ 2.10 และ 2.11 เราสามารถสรุปรูปแบบในการควบคุมทั้งตำแหน่งและความเร็วได้ดังบล็อคไดอะแกรมตามรูปที่ 2.17



รูปที่ 2.17 บล็อคไดอะแกรมลูปการควบคุมตำแหน่งและความเร็วมอเตอร์กระแสตรง

<u>การวิเคราะห์การเลี้ยว</u> [10]

หน้าที่ของระบบเลี้ยวคือเปลี่ยนการหมุนของตัวส่งกำลังไปเป็นการหมุนของล้อ โดยหลักการแล้วระบบเลี้ยวควรมีการทดแรงให้มีการได้เปรียบเชิงกลเพื่อให้ AGV สามารถเลี้ยวได้ง่ายโดยใช้แรงน้อยที่สุด ไม่ว่าจะต้องการเลี้ยวไปในทิศทางใดๆก็ตาม

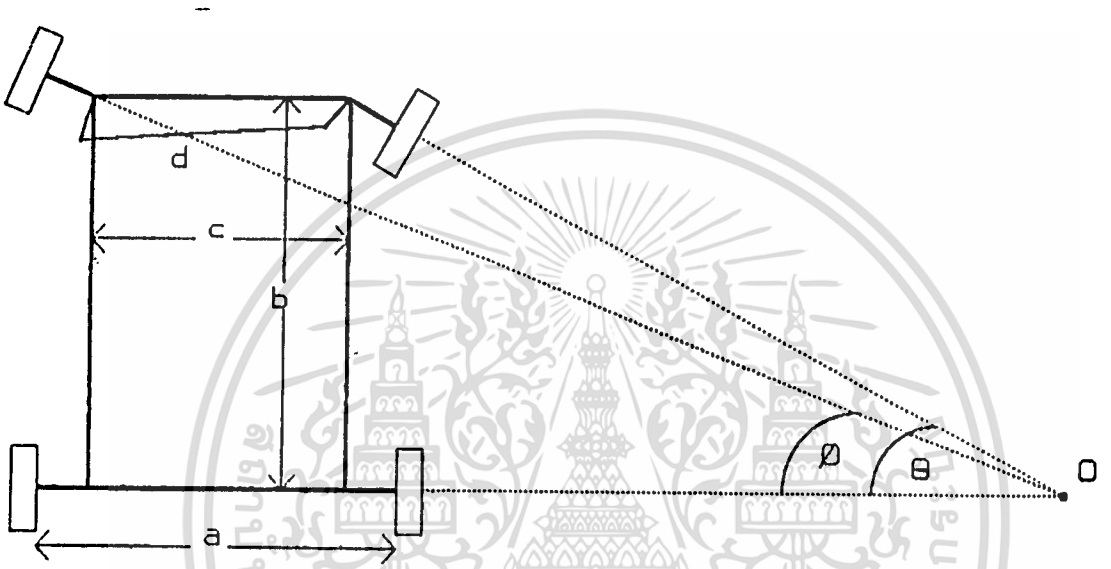
การเคลื่อนที่และการเลี้ยวนั้นจะต้องสัมพันธ์กัน ในหลายรูปแบบล้อเดียวกันอาจมีทำหน้าที่เป็นทั้งล้อขับ (drive wheel) และล้อสำหรับเลี้ยว (steering wheel) ไปในตัว

ในการพิจารณาขนาดล้อนั้น ต้องคำนึงถึงจุด C.G ด้วย สำหรับขนาดล้อที่ใหญ่เกินไปนั้นจุด C.G ของ AGV จะอยู่สูง ทำให้การเคลื่อนที่มีเสถียรภาพน้อยลง นอกจากนี้ล้อใหญ่ยังมีผลให้การวัดตำแหน่งมีความถูกต้องแม่นยำ (accuracy) น้อยลง

ในระบบรถ 4 ล้อ การเลี้ยวโค้งโดยไม่เกิดการลื่นไถลไปทางด้านข้างนั้น ล้อทุกล้อจะต้องหมุนโดยมีจุดศูนย์กลางร่วมจุดหนึ่ง แต่มีรัศมีวงเลี้ยวต่างกัน ดังแสดงตามรูปที่ 2.18

ในขณะที่รถเลี้ยว 2 ล้อหน้าจะต้องหมุนในลักษณะที่แน่นอนทั้งคู่และสัมพันธ์ซึ่งกันและกัน นอกจากนี้แนวแกนล้อหน้าทั้งคู่จะต้องไปตัดกับแนวแกนล้อหลังที่จุดเดียวกัน (จุด o) เพื่อขจัดการลื่นไถลทาง

ด้านข้างและทำให้ล้อทั้งหมดเกิดการกลิ้งไปอย่างแท้จริง จุดศูนย์กลางนี้เป็นจุดศูนย์กลางการหมุนชั่วขณะ (instantaneous center) เนื่องจากแนวแกนล้อหลังอยู่คงที่ ดังนั้นจะเห็นได้ว่าจุดศูนย์กลางร่วมจุดนี้จะต้องยื่นออกไปอยู่นอกตัวรถ



รูปที่ 2.18 การวิเคราะห์การเลี้ยวของรถ 4 ล้อ

โดยที่

o = จุดศูนย์กลางการหมุนชั่วขณะในขณะที่รถเลี้ยว

θ = มุม inside lock

ϕ = มุม outside lock

c = ระยะห่างระหว่างจุดศูนย์กลางของสลักทั้งสองข้าง

d = ความยาวของคันชักคันส่ง

a = ระยะห่างระหว่างล้อแต่ละข้างบนเพลาล้อเดียวกัน

b = ระยะฐานล้อของรถยนต์

x = ระยะจากจุดศูนย์กลางของสลักด้านในโค้งถึงจุด o

จากทฤษฎีตรีโกณมิติเราจะได้ว่า

$$\cot \phi = (c + x)/b$$

$$= c/b + x/b$$

$$= c/b + \cot \theta$$

นั่นคือ

$$\cot \phi - \cot \theta = c/b \qquad \ldots.(2.25)$$

สมการนี้ให้เงื่อนไขพื้นฐานซึ่งกลไกในระบบเลี้ยวทุกแบบต้องทำได้ ถ้าต้องการให้ล้อทุกล้อกลิ้งอย่างแท้จริงโดยไม่เกิดการไถลไปทางด้านข้าง และจากรูปจะได้เพิ่มเติมคือ

รัศมีวงเลี้ยวของล้อหน้าด้านนอก

$$Rof = b/\sin \phi + (a - c)/2 \qquad \ldots.(2.26)$$

รัศมีวงเลี้ยวของล้อหน้าด้านใน

$$Rif = b/\sin \phi - (a - c)/2 \qquad \ldots.(2.27)$$

รัศมีวงเลี้ยวของล้อหลังด้านนอก

$$Ror = b \cot \phi + (a - c)/2 \qquad \ldots.(2.28)$$

รัศมีวงเลี้ยวของล้อหลังด้านใน

$$Rir = b \cot \phi - (a - c)/2 \qquad \ldots.(2.29)$$

จะเห็นได้ว่ารัศมีวงเลี้ยวสามารถระบุได้หลายแบบ เพื่อหลีกเลี่ยงความสับสน SAE จึงกำหนดคำนิยามไว้ว่า "รัศมีวงเลี้ยวคือรัศมีส่วนโค้งที่เกิดจากการเคลื่อนที่ของล้อหน้าด้านนอกในขณะเลี้ยววงแคบที่สุด"

รัศมีวงเลี้ยวขึ้นอยู่กับระยะ a,b,c และมุม θ สูงสุดที่ล้อด้านในจะสามารถบิดไปได้ จากตำแหน่งที่ AGV วิ่งในแนวเส้นตรง ดังนั้น Rof เมื่อแสดงในเทอมของพารามิเตอร์เหล่านี้จะกลายเป็น

$$Rof = [(b/\sin \theta)^2 + c^2 + 2bc/ \tan \theta]^2 + (a - c)/2 \qquad \ldots.(2.30)$$

เทคโนโลยีการตรวจจับวัตถุกีดขวาง

สำหรับ AGV ที่ใช้กันโดยทั่วไปนั้น จะต้องมีการป้องกันการชนกับสิ่งต่างๆ ในระหว่างการเคลื่อนที่ ทั้งนี้ก็เพื่อความปลอดภัยในการทำงานตลอดจนในการขนถ่ายวัสดุ เทคนิคที่ใช้ในการตรวจจับวัตถุกีดขวางมีหลายวิธี อาทิเช่น การใช้กล้องวิดีโอ , การใช้อุตร้าโซนิค เป็นต้น สำหรับวิทยานิพนธ์นี้ได้เลือกใช้อุตร้าโซนิคเนื่องจากมีราคาถูกกว่า , มีขนาดเล็กและน้ำหนักเบาตลอดจนวงจรที่ใช้ก็ไม่ยุ่งยากซับซ้อน การนำอุตร้าโซนิคทรานสดิวเซอร์มาใช้ในการตรวจจับวัตถุกีดขวางจะใช้หลักการสะท้อนกลับของคลื่นอุตร้าโซนิค โดยจะส่งคลื่นอุตร้าโซนิคออกไปเป็นชุดๆ ชุดละประมาณ 10 ลูก ความถี่ในการส่งประมาณ 64 Hz แต่ละครั้งที่ส่งจะมีการรอการสะท้อนกลับ โดยในระหว่างรอจะมีการนับจำนวนลูกคลื่นความถี่ด้วยตัวนับ (counter) สำหรับความถี่ที่นับนั้นจะใช้ความถี่เดียวกันกับความถี่ของคลื่นอุตร้าโซนิค จำนวนลูกคลื่นความถี่ที่นับได้นี้จะสัมพันธ์กับระยะห่างระหว่าง AGV กับวัตถุกีดขวางกล่าวคือถ้าวัตถุกีดขวางอยู่ใกล้ จำนวนลูกคลื่นความถี่ที่นับได้ก็จะน้อย แต่ถ้าวัตถุกีดขวางอยู่ไกลจำนวนลูกคลื่นความถี่ที่นับได้ก็จะมาก จำนวนที่นับได้นี้จะถูกแปลงเป็นปริมาณอนาล็อคด้วย D/A เพื่อนำไปใช้ประโยชน์ในภาคถัดไป สำหรับความถี่ของคลื่นอุตร้าโซนิคจะใช้ที่ 40 KHz เนื่องจากเป็นค่าที่เหมาะสมที่สุด สำหรับทฤษฎีที่เกี่ยวข้องกับอุตร้าโซนิคทรานสดิวเซอร์นั้นมีดังนี้

คลื่นอุตร้าโซนิค

โดยทั่วไปหูมนุษย์จะได้ยินความถี่เสียงสูงสุดประมาณ 15 - 16 KHz ปกติแล้วคำว่าอุตราโซนิคมักจะหมายถึงเสียงที่มีความถี่สูงกว่า 20 KHz ขึ้นไปเช่น ย่าน 20 KHz , 25 KHz , 31.5 KHz , 40 KHz , 45 KHz เป็นต้น คลื่นอุตราโซนิคมีคุณสมบัติแตกต่างจากคลื่นเสียงธรรมดาตรงที่คลื่นเสียงธรรมดามนุษย์สามารถได้ยินและมีลักษณะการเคลื่อนที่กระจายไปทุกทิศทุกทาง ส่วนคลื่นอุตร้าโซนิคนั้นจะมีลักษณะการเคลื่อนที่เป็นแนวเส้นตรงคล้ายกับการเคลื่อนที่ของแสง คุณสมบัติอย่างหนึ่งของคลื่นก็คือถ้าคลื่นมีความถี่สูงขึ้นความยาวคลื่นก็จะสั้นลง ถ้าความยาวคลื่นยาวกว่าช่องเปิดที่ให้เสียงนั้นออกมาของตัวที่กำเนิดเสียงความถี่นั้น คลื่นจะเกิดหักเหที่ขอบด้านนอกของตัวกำเนิดคลื่นเสียงทำให้เกิดการกระจายคลื่นแต่ถ้าความถี่สูงขึ้นมาอยู่ในย่านอุตร้าโซนิคเช่น 40 KHz ความยาวคลื่นจะสั้นลง• คลื่นเสียงก็จะมีลักษณะพุ่งออกมาเป็นลำเรียกว่ามีทิศทางนั่นเอง

การมีทิศทางของคลื่นอุตร้าโซนิค ทำให้เราสามารถนำเอาไปใช้งานได้หลายอย่าง เช่น นำไปใช้ในเครื่องควบคุมระยะไกล (ultrasonic remote control) , เครื่องล้างอุปกรณ์ (ultrasonic cleaner) โดยทำน้ำสั่นที่ความถี่สูง, เครื่องวัดความหนาของวัตถุ , เครื่องวัดความลึกและทำแผนที่ใต้ท้องทะเล , เครื่องทดสอบการรั่วของท่อและการใช้งานทางด้านการแพทย์ เป็นต้น ความถี่ที่ใช้ก็มักจำกัดอยู่เพียงไม่เกิน 50 KHz เพราะความถี่สูงกว่านี้อากาศจะดูดกลืนคลื่นเสียงได้มากขึ้น

ทำให้ระดับความแรงของคลื่นเสียงที่ระยะต่างออกไปลดลงอย่างรวดเร็ว คลื่นอุตร้าโซนิค 40 KHz จะเป็นค่าเหมาะที่สุดที่จะนำมาใช้ในการทำงาน ส่วนระยะทางสูงสุดก็ประมาณ 20 เมตร คลื่นจะเคลื่อนที่ไปได้มากน้อยเท่าใดขึ้นอยู่กับอุปกรณ์ที่ใช้สร้างและสิ่งแวดล้อมต่างๆ ด้วย

ชนิดของคลื่นอุตร้าโซนิค

คลื่นอุตร้าโซนิคที่เดินผ่านตัวกลางต่างๆมีหลายชนิดด้วยกัน แต่ละชนิดจะแตกต่างกันตามการเคลื่อนที่ของอนุภาคในตัวกลางนั้น ชนิดของคลื่นอุตร้าโซนิคมีดังนี้

ก) คลื่นตามยาว (Longitudinal wave)

ทุกๆจุดบนคลื่นมีการเคลื่อนที่ในทิศทางเดียวกับทิศทางการเคลื่อนที่ คลื่นตามยาวนี้สามารถเดินทางผ่านของแข็ง , ของเหลว และก๊าซ โดยมากถ้ากล่าวถึงความเร็วของคลื่นเสียงแล้วจะหมายถึงความเร็วของคลื่นตามยาว ในการพิจารณาคลื่นตามยาวที่เดินผ่านตัวกลางต่างๆได้นั้นตัวกลางจะต้องมีขนาดใหญ่พอเมื่อเทียบกับความยาวคลื่น

ข) คลื่นตามขวาง (Transverse wave)

ทุกๆจุดบนคลื่นมีการเคลื่อนที่ในทิศทางที่ตั้งฉากกับทิศทางการเคลื่อนที่ของคลื่น คลื่นชนิดนี้จะเดินทางผ่านตัวกลางที่มีขนาดของตัวกลางใหญ่กว่าขนาดของความยาวคลื่น สามารถเดินทางผ่านตัวกลางที่เป็นของแข็ง ไม่สามารถเดินทางผ่านตัวกลางที่เป็นของเหลวและก๊าซได้ ความเร็วของคลื่นตามขวางจะน้อยกว่าคลื่นตามยาวในขณะที่ผ่านตัวกลางชนิดเดียวกัน

ค) คลื่นผิวหน้า (Surface wave or rayleigh)

คล้ายคลื่นตามขวาง แต่จะต่างกันตรงที่ว่าการเปลี่ยนแปลงตำแหน่งของอนุภาคไม่เป็นเพียงในทิศทางที่ตั้งฉากกับทิศทางการเคลื่อนที่เพียงอย่างเดียว แต่มีการเปลี่ยนแปลงในทิศทางเดียวกับทิศทางของการเคลื่อนที่ด้วยจึงทำให้คลื่นเคลื่อนที่ไปตามระนาบในแนวนอน ด้วยเหตุนี้คลื่นจึงเดินทางผ่านไปเฉพาะบนผิวของตัวกลางเท่านั้น

อุตร้าโซนิคทรานสดิวเซอร์ (Ultrasonic transducer)

เป็นอุปกรณ์ที่สามารถแปลงพลังงานในรูปอื่นให้มาเป็นพลังงานทางกลโดยการสั่นไปมา ทำให้เกิดคลื่นเสียงย่านอุตร้าโซนิคกระจายไปในอากาศได้ หรือแปลงพลังงานกลให้มาเป็นพลังงานรูปอื่นได้

ในปัจจุบันอุตร้าโซนิค ทรานสดิวเซอร์มีหลายแบบขึ้นกับหลักการที่ใช้ แบบที่นิยมใช้กันมากได้แก่ แบบเปียโซอิเลคทริค (piezoelectric) ซึ่งแปลงไปมาระหว่างพลังงานไฟฟ้าและพลังงานกลโดยมีความถี่เรโซแนนซ์ (resonance) ค่าหนึ่ง แบบแมกนีโตสตริคทีฟ (magnetostrictive) ซึ่งแปลงไปมาระหว่างพลังงานไฟฟ้ากับขดลวดกับตำแหน่งความยาวของแกนเหล็กที่สวมขดลวดนั้นอยู่และ

แบบอิเลคโทรสตริคทีฟ (electrostrictive) ซึ่งแปลงไปมาระหว่างพลังงานไฟฟ้ากับพลังงานกล

สำหรับอุตร้าโซนิคทรานสดิวเซอร์ที่ใช้ในงานวิจัยนี้เป็นแบบเปียโซอิเลคทริคซึ่งจะมีอยู่ 2 อย่างคือ ตัวส่ง (transmitter) และตัวรับ (receiver) ตัวส่งเป็นอุตร้าโซนิคทรานสดิวเซอร์ที่ออกแบบมาให้แปลงสัญญาณไฟฟ้าที่ให้ตัวมันเป็นคลื่นเสียงย่านอุตร้าโซนิค หน้าที่ของตัวส่งจะคล้ายลำโพง ส่วนตัวรับเป็นอุตร้าโซนิคทรานสดิวเซอร์ที่ออกแบบมาให้แปลงคลื่นเสียงย่านอุตร้าโซนิคที่มาตกกระทบตัวมัน ให้ออกมาเป็นสัญญาณไฟฟ้า หน้าที่ของตัวรับจึงคล้ายไมโครโฟน

<u>แบตเตอรี่</u> [11]

แบตเตอรี่มี 5 ชนิดแต่ละชนิดมีรูปร่าง , โครงสร้าง และขนาดแตกต่างกัน

1) ซิงค์ (zinc) ไม่นิยมใช้ในหุ่นยนต์เพราะกำลังงานหมดเร็ว ทั้งยังจ่ายพลังงานได้ต่ำ

2) อัลคาไลน์ (alkaline) ราคาแพงกว่าซิงค์ แต่ให้กำลังงานตลอดจนอายุการใช้งานนานกว่า แต่ข้อเสียของแบตเตอรี่แบบนี้ก็คือ ไม่สามารถประจุใหม่ (recharge) ได้

3) นิเกิล-แคดเมียม (nickel-cadmium) หรือที่นิยมเรียก "Ni-Cad" สามารถประจุใหม่ได้ , หาง่าย ราคาไม่แพงนัก แต่ให้กำลังงานน้อย เหมาะใช้ในหุ่นยนต์ขนาดเล็ก ใช้งานได้ไม่นานเหมือนซิงค์และอัลค่าไลน์ ในปัจจุบันมีถ่านนิเกิล-แคดเมียมความจุสูงออกมาใหม่โดยสูงกว่าเดิมประมาณ 2-3 เท่าซึ่งเหมาะกับงานด้านนี้ แต่ราคาแพง

4) ตะกั่ว-กรด (lead-acid) มีเห็นใช้ทั่วไปในรถยนต์,รถมอเตอร์ไซค์ สามารถประจุใหม่ได้ มีทั้งขนาดเล็กและขนาดใหญ่ มีกำลังงานสูงแต่น้ำหนักมาก ทำให้ AGV ต้องแบกน้ำหนักมากขึ้น นอกจากนี้เราจะเห็นว่ามีการใช้แบตเตอรี่ตะกั่ว-กรดกับระบบจ่ายไฟไม่ขาดตอน (UPS) สำหรับคอมพิวเตอร์หรือระบบโทรศัพท์ สำหรับ AGV ที่ใช้กันอยู่ โดยมากมักจะใช้แบตเตอรี่แบบนี้

5) เจล-เซล (gel-cell) ประจุใหม่ได้ คล้ายตะกั่ว-กรดลิดแอสิดแต่จ่ายกระแสสูงกว่า

ข้อกำหนดของแบตเตอรี่

1) <u>แรงดัน</u> สำหรับแบตเตอรี่ที่ประจุใหม่ได้นั้น ขณะประจุเต็มจะมีระดับแรงดันสูงกว่าระดับปกติที่กำหนดประมาณ 20-30 % เช่น แบตเตอรี่ 12 โวลท์ ชนิดตะกั่ว-กรด เมื่อประจุจนเต็มจะมีค่าประมาณ 13.8 โวลท์ แบตเตอรี่ทั้งหมดจะถือว่าหมดสภาพเมื่อแรงดันลดต่ำลงเหลือประมาณ 80 % จากระดับปกติ เช่นแบตเตอรี่ 6 โวลท์จะถือว่าหมดสภาพ ถ้าแรงดันตกต่ำกว่า 4.8 โวลท์ ขณะที่แบตเตอรี่หมดสภาพ มันจะไม่สามารถจ่ายกระแสได้ตามอัตราปกติอีกต่อไป การทดสอบระดับแรงดันแบตเตอรี่จะต้องทำขณะที่แบตเตอรี่ใช้งานอยู่ การทดสอบจะคลาดเคลื่อนถ้าขณะทดสอบไม่มีโหลด

2) <u>กระแส-เวลา</u> (ampere - hour) แสดงถึงปริมาณความจุของแบตเตอรี่

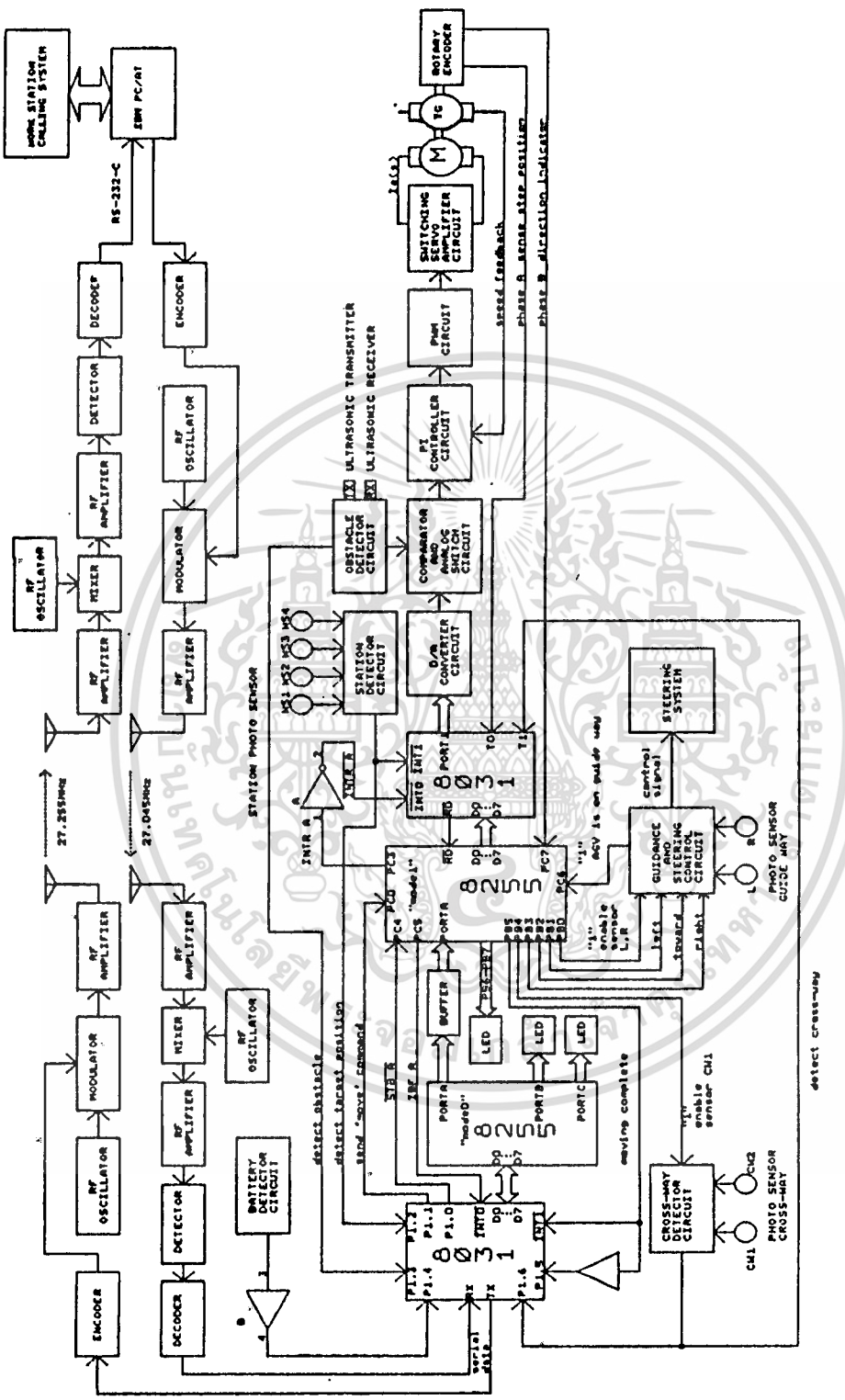
บทที่ 3

การออกแบบและการสร้างระบบควบคุม AGV

ระบบ AGV ที่สมบูรณ์แสดงได้ดังบล็อคไดอะแกรมในรูปที่ 3.1 พิจารณาระบบจากบล็อคไดอะแกรม AGV ขับเคลื่อนด้วยมอเตอร์กระแสตรงซึ่งควบคุมความเร็วด้วยอนาล็อค PI คอนโทรลเลอร์และควบคุมตำแหน่งด้วยไมโครคอนโทรลเลอร์ 8031-2 การวัดตำแหน่งการเคลื่อนที่ของ AGV จะใช้โรตารี่เอนโคดเดอร์ AGV สามารถติดต่อสื่อสารกับคอมพิวเตอร์ส่วนกลางได้โดยการส่งข้อมูลอนุกรมแบบอะซิงโครนัสอาล์ฟดูเพล็กซ์ด้วยอัตราบอด 150 บิท/วินาที ผ่านคลื่นวิทยุแบบ AM ย่านความถี่ 27 MHz สำหรับคอมพิวเตอร์ส่วนกลางก็สามารถส่งข้อมูลให้ AGV ได้ในลักษณะเดียวกัน ข้อมูลที่ใช้ติดต่อสื่อสารกันระหว่าง AGV กับคอมพิวเตอร์ส่วนกลางประกอบด้วย คำสั่งควบคุมต่างๆ , ข้อมูลในการเคลื่อนที่สำหรับ AGV ในเคลื่อนที่ไปตามเส้นทางที่กำหนด , และข้อมูลแสดงสถานภาพ (status) ที่สำคัญของ AGV เทคนิคการนำร่องของ AGV จะใช้การติดแถบสีตามพื้นโรงงาน แล้วใช้โฟโต้เซนเซอร์ L,R ตรวจหาตำแหน่งแถบสี สัญญาณจากเซนเซอร์ทั้งสองจะถูกนำไปเข้าวงจรควบคุมการนำร่องและการเลี้ยว เพื่อควบคุมให้ AGV วิ่งไปตามแนวแถบสี ในการสั่งให้ AGV เลี้ยวสามารถทำได้โดยส่งสัญญาณควบคุมจากไมโครคอนโทรลเอร์ 8031-2 ไปยังวงจรควบคุมการนำร่องและการเลี้ยว AGV มีวงจรตรวจจับตำแหน่งทางแยก โดยเมื่อ AGV วิ่งไปถึงทางแยก สัญญาณจากโฟโต้เซนเซอร์ CW1 , CW2 จะถูกส่งไปเข้าไมโครคอนโทรลเลอร์ทั้งสองตัว เพื่อบอกให้ AGV ทราบว่าขณะนี้ถึงทางแยกแล้ว สำหรับตำแหน่งสถานีปฏิบัติงาน (work station) จะมีลักษณะเป็นจุดแสดงตำแหน่ง (marker) ที่ทำจากแถบสีเช่นกัน โดยเมื่อ AGV วิ่งมาถึงสถานี โฟโต้เซนเซอร์ WS1 หรือ WS2 หรือ WS3 หรือ WS4 จะส่งสัญญาณไปยังไมโครคอนโทรลเลอร์ทั้งสองตัว เพื่อบอกให้ AGV ทราบว่าถึงสถานีเป้าหมายแล้ว นอกจากนี้ AGV ยังมีระบบเซนเซอร์อื่นๆอีก ได้แก่ ระบบตรวจจับวัตถุกีดขวางด้วยอุตร้าโซนิค โดยเมื่อ AGV ตรวจพบวัตถุกีดขวางบนทางเดินนำร่อง AGV จะลดความเร็วลงในลักษณะแปรโดยตรงกับระยะทางระหว่าง AGV กับวัตถุกีดขวางนั้น และยังมีระบบตรวจวัดแรงดันแบตเตอรี่อีกด้วย

ในวิทยานิพนธ์นี้ได้ทำการออกแบบและสร้างระบบควบคุมการทำงานของ AGV ในส่วนที่สำคัญที่สอดคล้องกับบล็อคไดอะแกรมตามรูปที่ 3.1 ระบบควบคุมการทำงาน AGV ที่ได้ออกแบบและสร้างได้แก่

1) ระบบสื่อสารข้อมูลระหว่างไมโครคอมพิวเตอร์กับไมโครคอนโทรเลอร์
2) ระบบเรียกใช้งานจากสถานีปฏิบัติงาน
3) ระบบควบคุมการเคลื่อนที่
4) ระบบเซนเซอร์ต่างๆ ประกอบด้วย ระบบนำร่องด้วยโฟโต้เซนเซอร์ , ระบบตรวจจับวัตถุกีดขวางด้วยอุตร้าโซนิค , และระบบตรวจวัดแรงดันแบบตเตอรี่



รูปที่ 3.1 บล็อคไดอะแกรมระบบควบคุม AGV ในงานวิจัยนี้

3.1 ระบบสื่อสารข้อมูลระหว่างไมโครคอมพิวเตอร์กับไมโครคอนโทรลเลอร์

การสื่อสารข้อมูลระหว่างไมโครคอมพิวเตอร์กับไมโครคอนโทรเลอร์ในวิทยานิพนธ์นี้ เลือกใช้วิธีสื่อสารข้อมูลแบบอะซิงโครนัส ฮาลฟ์ดูเพล็กซ์ (asynchronous half duplex) ผ่านคลื่นวิทยุแบบ AM ย่านความถี่ 27 MHz โดยได้ทำการออกแบบวงจรเข้ารหัส (encoder) ทางภาคส่งและวงจรถอดรหัส (decoder) ทางภาครับ เพื่อให้สามารถส่งข้อมูลแบบอะซิงโครนัสจากไมโครคอมพิวเตอร์ไปยังไมโครคอนโทรลเลอร์ หรือส่งข้อมูลจากไมโครคอนโทรลเลอร์กลับมายังไมโครคอมพิวเตอร์ได้

หลักการทำงาน

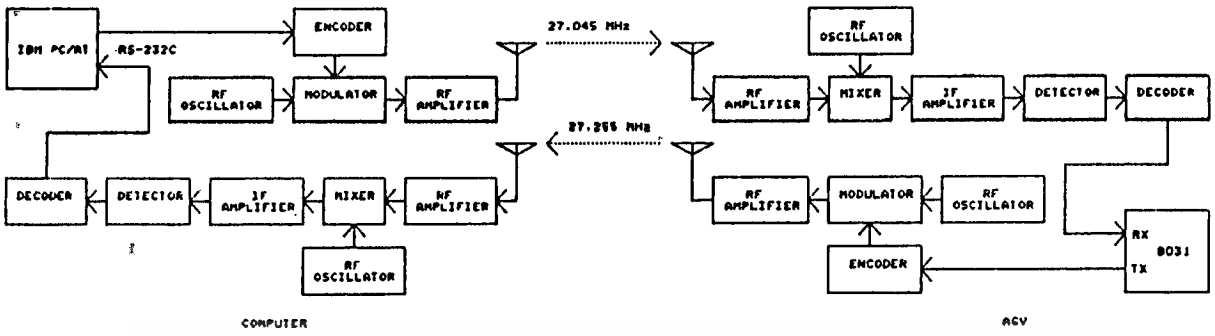
ตามรูปที่ 3.2 เป็นบล็อคไดอะแกรมแสดงการรับส่งข้อมูลระหว่างไมโครคอมพิวเตอร์กับไมโครคอนโทรลเลอร์ ข้อมูลจากไมโครคอมพิวเตอร์จะส่งผ่านพอร์ทอนุกรมไปยังวงจรเข้ารหัสซึ่งทำหน้าที่แปลงข้อมูลอนุกรมมาตราฐาน RS-232-C ซึ่งมีระดับแรงดันตามลอจิก "0" และ "1" คือ +10 โวลท์ และ -10 โวลท์ตามลำดับ ไปเป็นสัญญาณพัลซ์รูปสี่เหลี่ยมซึ่งมีความกว้างของพัลซ์อยู่ 2 สถานะกล่าวคือ ลอจิก "0" พัลซ์จะมีขนาดแคบ ส่วนลอจิก "1" พัลซ์จะมีขนาดกว้าง ดังนั้นขบวนข้อมูลอนุกรมที่ถูกส่งออกมาก็จะถูกแปลงไปเป็นขบวนของพัลซ์ซึ่งมีความกว้างและแคบแตกต่างกันไปแล้วแต่รหัสของข้อมูลนั้นๆ

ขบวนพัลซ์จะถูกส่งไปมอดูเลทกับสัญญาณ RF แล้วผ่านเข้าสู่ภาคขยายสัญญาณ RF เพื่อส่งออกสายอากาศ ทางด้านภาครับด้านไมโครคอนโทรลเลอร์ สัญญาณที่ได้รับจะถูกขยายแล้วส่งต่อไปยังภาคมิกเซอร์ได้เป็นสัญญาณ IF ทำการขยายสัญญาณ IF แล้วส่งต่อไปภาคดีเทคเตอร์ หลังจากภาคดีเทคตอร์เราจะได้ขบวนพัลซ์กลับคืนมา จากนั้นจึงส่งต่อไปภาคถอดรหัสเพื่อแปลงขบวนพัลซ์กลับไปเป็นข้อมูลอนุกรมเพื่อส่งไปเข้าพอร์ทอนุกรมของไมโครคอนโทรลเลอร์ต่อไป สำหรับการส่งข้อมูลอนุกรมจากไมโครคอนโทรลเลอร์กลับไปยังไมโครคอมพิวเตอร์ก็ใช้หลักการในทำนองเดียวกัน

ข้อมูลอนุกรม

ลักษณะข้อมูลอนุกรมที่ใช้ส่งมีขนาด 10 บิท ประกอบด้วยบิทเริ่มต้น (start bit) , บิทข้อมูล (data bit) 8 บิท , และบิทสิ้นสุด (stop bit) โดยไม่มีบิทพาริตี้ (parity bit) ทั้งนี้เพื่อให้สอดคล้องกับข้อจำกัดทางฮาร์ดแวร์ที่ตรงกันระหว่าง ไมโครคอมพิวเตอร์ 286-AT กับไมโครคอนโทรลเลอร์ 8031

สำหรับอัตราบอด (baud rate) ในการรับส่งนั้น จะขึ้นกับความยาวของโปรแกรมตลอดจนความเหมาะสมในแง่อื่นๆ สำหรับในวิทยานิพนธ์นี้ใช้อัตราบอดเท่ากับ 150 บิท/วินาที



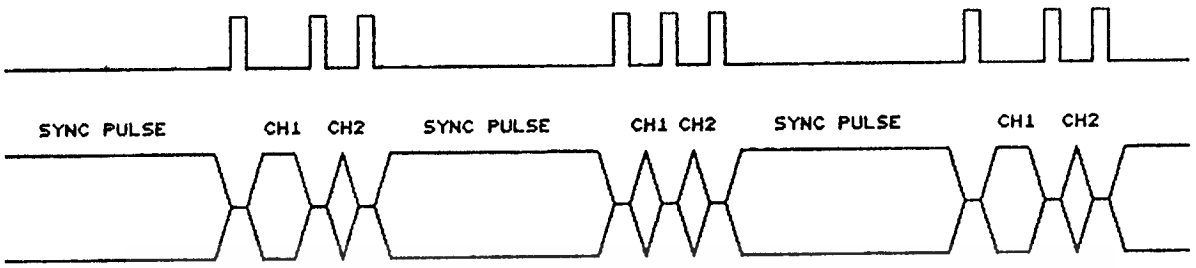
รูปที่ 3.2 บล็อคไดอะแกรมการสื่อสารรข้อมูลระหว่างไมโครคอมพิวเตอร์กับไมโครคอนโทรลเลอร์

รูปที่ 3.3 ลักษณะข้อมูลอนุกรมแบบ 10 บิท ไม่มีบิทพาริตี้

ขบวนพัลซ์ (pulse train)

ลักษณะขบวนพัลซ์แสดงไว้ในรูปที่ 3.4 ขบวนพัลซ์จะประกอบด้วยซิงโครนัสพัลซ์ (synchronous pulse) ตามด้วยสัญญาณพัลซ์ช่อง (channel) ที่ 1 และช่องที่ 2 ตามลำดับ แต่ละพัลซ์จะมีช่องว่างกั้นไว้เพื่อให้ภาครับสามารถแยกได้ ดังนั้นสัญญาณทั้ง 2 จึงช่องเป็นอิสระไม่ขึ้นแก่กัน สัญญาณซิงโครนัสพัลซ์มีไว้สำหรับเป็นหลักในด้านการส่งและด้านการแยกสัญญาณทางเครื่องรับเพื่อไม่ให้รูปคลื่นสัญญาณทั้ง 2 ช่องสับสนปะปนกัน สัญญาณช่องที่ 1 เป็นสัญญาณที่แปลมาจากข้อมูลอนุกรม 1 บิทโดยจะเก็บรหัสบิทนั้นในรูปพัลซ์ซึ่งมีความกว้าง 2 ขนาด กล่าวคือพัลซ์แคบกำหนดให้ใช้แทนลอจิก "0" และพัลซ์กว้างใช้แทนลอจิก "1" สัญญาณช่องที่ 2 มีไว้เผื่อเพิ่มเติมในอนาคตหากต้องการขยายสมรรถนะของ AGV เช่นต้องการให้ไมโครคอมพิวเตอร์สามารถติดต่อกับ AGV มากกว่าหนึ่งตัว

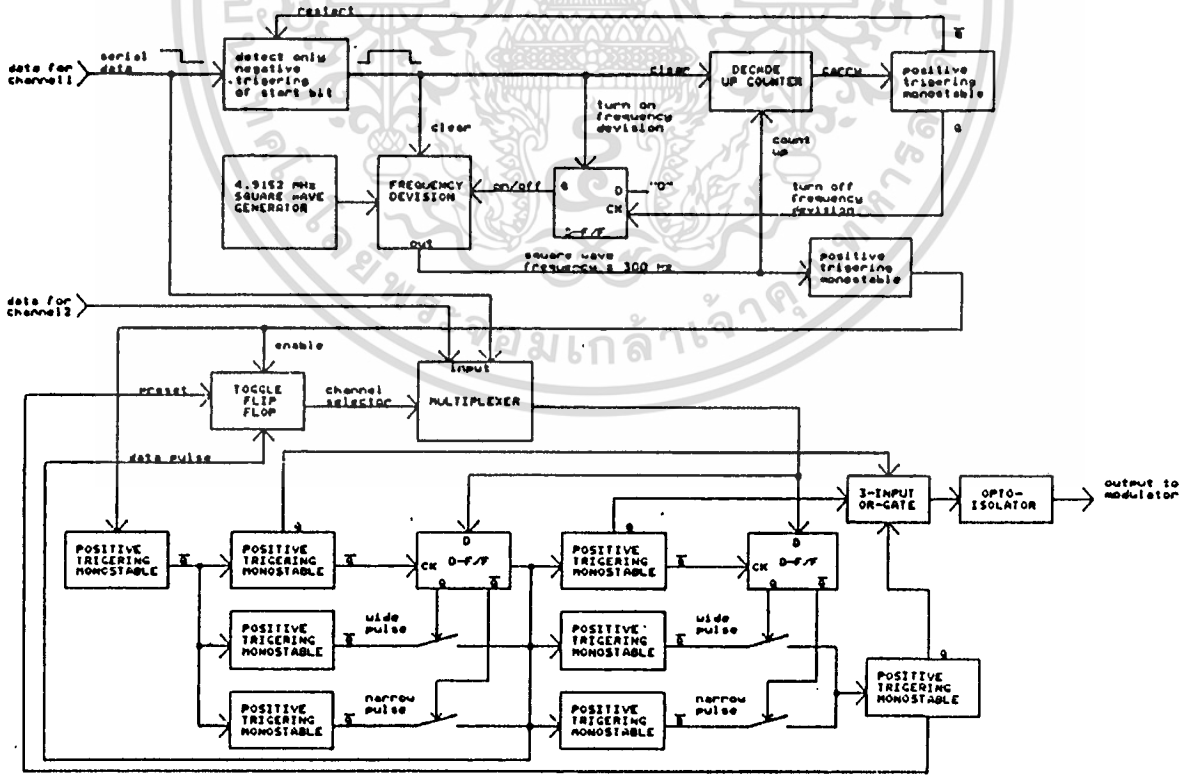
ขณะที่ไม่มีการส่งข้อมูลจะมีแต่คลื่นพาห์ที่ถูกส่งออกอากาศอย่างต่อเนื่อง แต่ทันทีที่มีการส่งข้อมูลอนุกรม ขบวนพัลซ์ก็จะเกิดขึ้นทันทีโดยมีองค์ประกอบต่างๆ ดังกล่าว ข้อมูลอนุกรมขนาด 1 เฟรม (10 บิท) ก็จะมีองค์ประกอบดังกล่าว 10 ชุด ขบวนพัลซ์ที่ได้จะถูกส่งต่อไปยังภาคมอดูเลทเพื่อรวมกับคลื่นพาห์แล้วส่งออกอากาศไปยังเครื่องรับต่อไป



รูปที่ 3.4 ลักษณะขบวนพัลซ์ที่ได้หลังจากผ่านวงจรเข้ารหัส

ภาคเข้ารหัส (Encoder)

หน้าที่ของภาคเข้ารหัสคือแปลงข้อมูลอนุกรมที่ถูกส่งออกมาจากไมโครคอมพิวเตอร์หรือจากไมโครคอนโทรลเลอร์ให้เป็นขบวนพัลซ์เพื่อส่งไปภาคมอดดูเลทต่อไป บล็อคไดอะแกรมการแปลงข้อมูลอนุกรมเป็นขบวนพัลซ์แสดงไว้ในรูปที่ 3.5



รูปที่ 3.5 บล็อคไดอะแกรมการแปลงข้อมูลอนุกรมเป็นขบวนพัลซ์

หลักการทำงาน

ที่สภาวะปกติขณะไม่มีข้อมูลอนุกรมถูกส่งออกมานั้น ระดับแรงดันที่ขา TXD ของพอร์ทอนุกรมจะคงสภาวะลอจิกที่เรียกว่า marking หรือลอจิก "1" วงจรแปลงข้อมูลอนุกรมเป็นขบวนพัลซ์จะไม่ให้เอาท์พุทใดๆออกมานั่นคือเอาท์พุทเป็นศูนย์โวลท์ ณ.เวลานี้ภาคส่งจะมีแต่คลื่นพาห์ล้วนๆ ออกอากาศอย่างต่อเนื่อง แต่ทันทีที่พอร์ทอนุกรมส่งบิทเริ่มต้นซึ่งเป็นลอจิก "0" ออกมา วงจรจะทำการสร้างพัลซ์สี่เหลี่ยมขึ้นมา 10 ลูกที่มีความถี่เท่ากับอัตราบอดของข้อมูลอนุกรม ขอบขาขึ้นของพัลซ์ 10 ลูกจะไปกระตุ้นโมโนสเตเบิ้ล 3 ตัว ตัวที่หนึ่งจะสร้างพัลซ์กว้าง (wide pulse) สำหรับใช้แทนลอจิก "1" ตัวที่สองจะสร้างพัลซ์แคบ (narrow pulse) สำหรับใช้แทนลอจิก "0" พัลซ์ทั้งสองนี้ถูกสร้างขึ้นพร้อมกัน แต่พัลซ์ใดจะถูกเลือกไปใช้จะอยู่ที่ว่าลอจิกของบิทที่เข้ามานั้นเป็นอะไร เช่นถ้าบิทที่เข้ามาเป็นบิทเริ่มต้น (ลอจิก "0") พัลซ์แคบจะถูกคัดเลือก การคัดเลือกจะกระทำโดยใช้อนาล็อคสวิทช์ โมโนสเตเบิ้ลตัวที่สามจะทำหน้าที่สร้างช่องว่างกั้นระหว่างซิงค์โครนัสพัลซ์กับสัญญาณช่องที่ 1 และยังทำหน้าที่เป็นตัวกระตุ้นดีฟลิบฟลอบ ดังจะได้กล่าวต่อไป

ในเวลาเดียวกันข้อมูลบิทนั้นก็จะถูกส่งไปเข้าตัวมัลติเพล็กเซอร์ด้วย เอาท์พุทของมัลติเพล็กเซอร์จะเป็นสัญญาณช่องที่ 1 ซึ่งเป็นข้อมูลอนุกรมสลับกันกับสัญญาณช่องที่ 2

นอกจากนี้พัลซ์ 10 ลูกที่เกิดขึ้นจะไปกระตุ้นท็อคเกิ้ลฟลิบฟลอบ (toggle flip-flop) โดยเอาท์พุทของท็อคเกิ้ลฟลิบฟลอบจะเป็นตัวตัดสินใจเลือกอินพุทให้แก่มัลติเพล็กเซอร์

ในกรณีบิทที่เข้ามาเป็นบิทเริ่มต้น มัลติเพล็กเซอร์จะเลือกสัญญาณช่องที่ 1 เสมอ เอาท์พุทของมัลติเพล็กเซอร์จะเป็นข้อมูลสัญญาณช่องที่ 1 (ข้อมูลอนุกรม) สลับกับสัญญาณช่องที่ 2 ไปเรื่อยๆ จนกว่าบิทสิ้นสุด (บิทที่ 10) ของสัญญาณช่องที่ 1 จะมาถึง

เอาท์พุทจากมัลติเพล็กเซอร์จะถูกส่งไปรอที่ขาข้อมูลของดีฟลิบฟลอบ 2 ตัว ดีฟลิบฟลอบตัวแรกจะทำหน้าที่ตัดสินใจให้สัญญาณช่องที่ 1 ว่าจะเลือกใช้พัลซ์กว้างหรือพัลซ์แคบ ส่วนดีฟลิบฟลอบตัวหลังจะทำหน้าที่ตัดสินใจให้สัญญาณช่องที่ 2 ว่าจะเลือกใช้พัลซ์กว้างหรือพัลซ์แคบเช่นกัน เอาท์พุทของดีฟลิบฟลอบจะไปควบคุมการทำงานของอนาล็อคสวิทช์ว่าจะให้อนาล็อคสวิทช์ตัวใดทำงาน

ในขณะเดียวกันขอบขาขึ้นของพัลซ์กว้างหรือพัลซ์แคบที่ถูกเลือกแล้วจากอนาล็อคสวิทช์สำหรับสัญญาณช่องที่ 1 จะถูกส่งไปกระตุ้นท็อคเกิ้ลฟลิบฟลอบตัวเดิมให้เลือกอินพุทจากสัญญาณช่องที่ 2 เข้าสู่มัลติเพล็กเซอร์แทนบ้าง

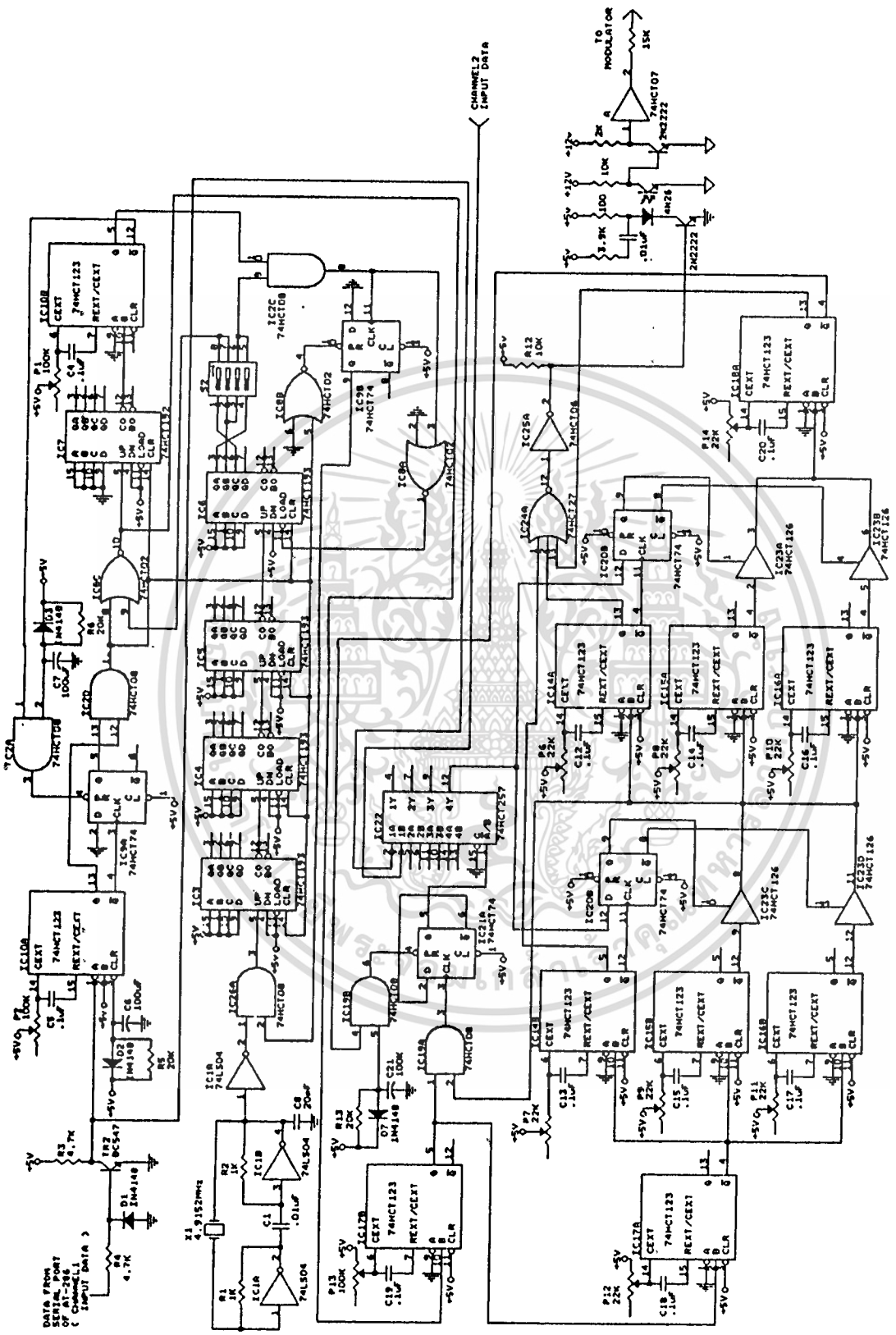
ขณะเดียวกันขอบขาขึ้นดังกล่าวก็จะไปกระตุ้นโมโนสเตเบิ้ลอีก 3 ตัว ตัวที่หนึ่งจะสร้างพัลซ์กว้างสำหรับใช้แทนลอจิก "1" ตัวที่สองจะสร้างพัลซ์แคบสำหรับใช้แทนลอจิก "0" พัลซ์ทั้งสองนี้ถูกสร้างขึ้นพร้อมกัน แต่พัลซ์ใดจะถูกเลือกไปใช้จะอยู่ที่ว่าลอจิกของสัญญาณช่องที่ 2 ที่เข้ามาขณะนั้นว่าเป็นอะไร เช่น ถ้าสัญญาณที่เข้ามาเป็นลอจิก "0" พัลซ์แคบก็จะถูกคัดเลือก การคัดเลือกจะใช้อนาล็อคสวิทช์เช่น

เดียวกันกับกรณีสัญญาณช่องที่ 1 ส่วนโมโนสเตเบิ้ลตัวที่สามจะทำหน้าที่สร้างช่องว่างกั้นระหว่างสัญญาณช่องที่ 1 กับสัญญาณช่องที่ 2 ขอบขาขึ้นของลอจิก "0" ของโมโนสเตเบิ้ลตัวที่สามนี้จะไปกระตุ้นดีฟลิบฟลอบตัวหลังต่อไป

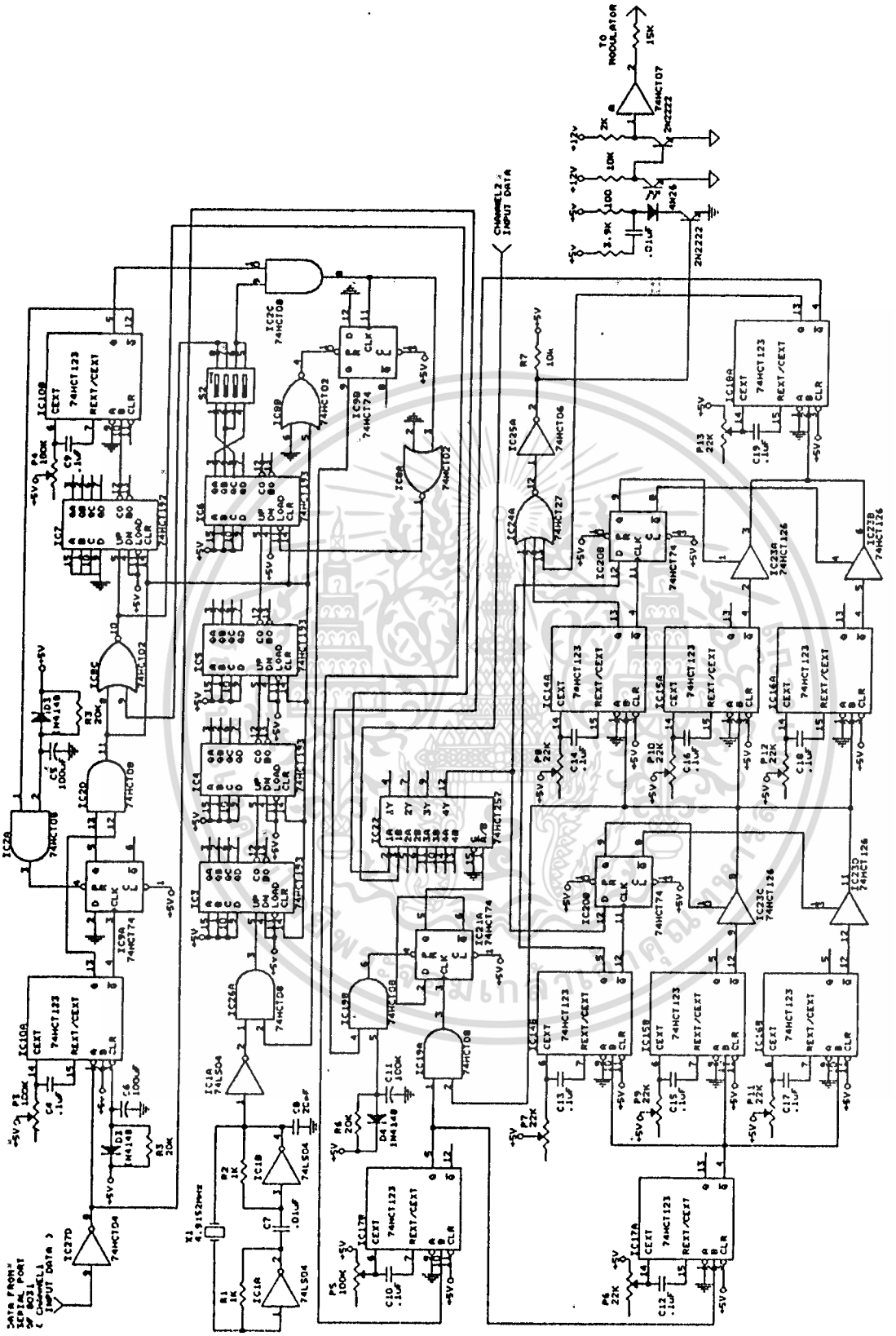
ขอบขาขึ้นของพัลซ์กว้างหรือแคบที่ได้รับคัดเลือกจากอนาล็อคสวิทช์สำหรับสัญญาณช่องที่ 2 จะไปกระตุ้นโมโนสเตเบิ้ลตัวสุดท้ายให้สร้างช่องว่างกั้นระหว่างสัญญาณช่องที่ 2 กับซิงค์โครนัสพัลซ์ อันถือเป็นสิ้นสุดการแปลงบิทนั้นๆโดยสมบูรณ์ จากนั้นวงจรจะทำการแปลงบิทใหม่ต่อไปจนครบ 10 บิท เมื่อครบแล้วเอาท์พุทของภาคเข้ารหัสจะเป็น 0 โวลท์เช่นเดิม

รูปที่ 3.6 และ รูปที่ 3.7 เป็นวงจรแปลงข้อมูลอนุกรมเป็นขบวนพัลซ์สำหรับไมโครคอมพิวเตอร์ 286-AT และสำหรับ 8031-1 ตามลำดับ ระหว่างภาคเข้ารหัสและภาคมอดูเลทของเครื่องส่งจะมีแยกภาคทั้งสองออกจากกันโดยใช้ออบโต้ไอโซเลเตอร์ (opto-isolator) ทั้งนี้เพื่อป้องกันไม่ให้สัญญาณรบกวนซึ่งเกิดจากคอมพิวเตอร์หรือวงจรส่วนอื่นๆ เข้าไปรบกวนภาคส่ง

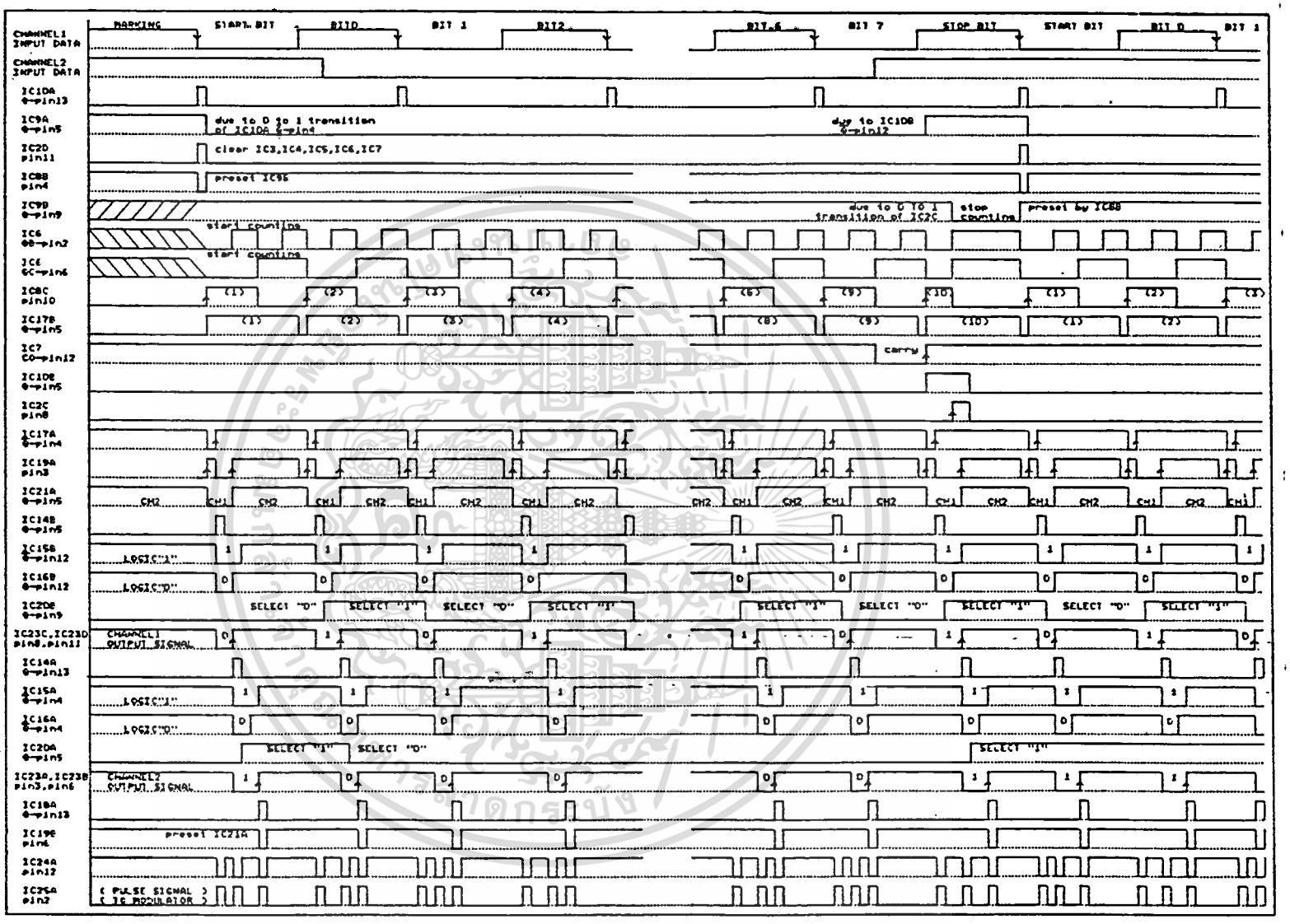
ส่วนในรูปที่ 3.8 แสดงแผนผังเวลาของวงจรแปลงข้อมูลอนุกรมเป็นขบวนพัลซ์



รูปที่ 3.6 วงจรแปลงข้อมูลอนุกรมที่ส่งจากคอมพิวเตอร์ 286-AT เป็นขนานพัลส์



รูปที่ 3.7 วงจรแปลงข้อมูลอนุกรมที่ส่งจาก 8031-1 เป็นขนานพัลซ์



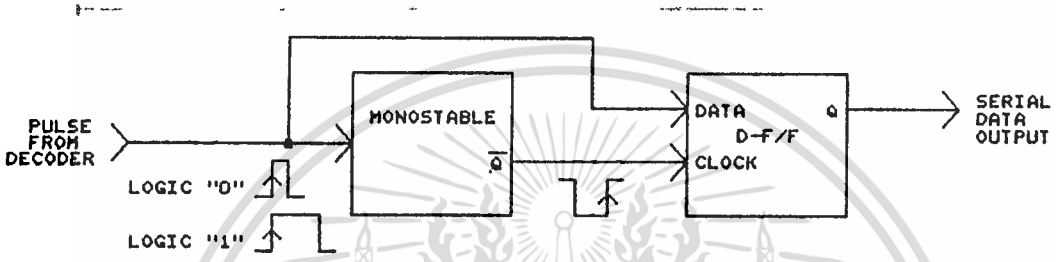
รูปที่ 3.8 แผนผังเวลาของวงจรแปลงข้อมูลอนุกรมเป็นขบวนพัลซ์

ภาคถอดรหัส (Decoder)

หน้าที่ของภาคถอดรหัสมี 2 ส่วนคือ

1) แยกสัญญาณ (demultiplex) ช่องที่ 1 กับช่องที่ 2 ที่รวมกันมาในขบวนพัลซ์ออกจากกัน

2) แปลงสัญญาณช่องที่ 1 ที่ได้กลับเป็นข้อมูลอนุกรมดังเดิม



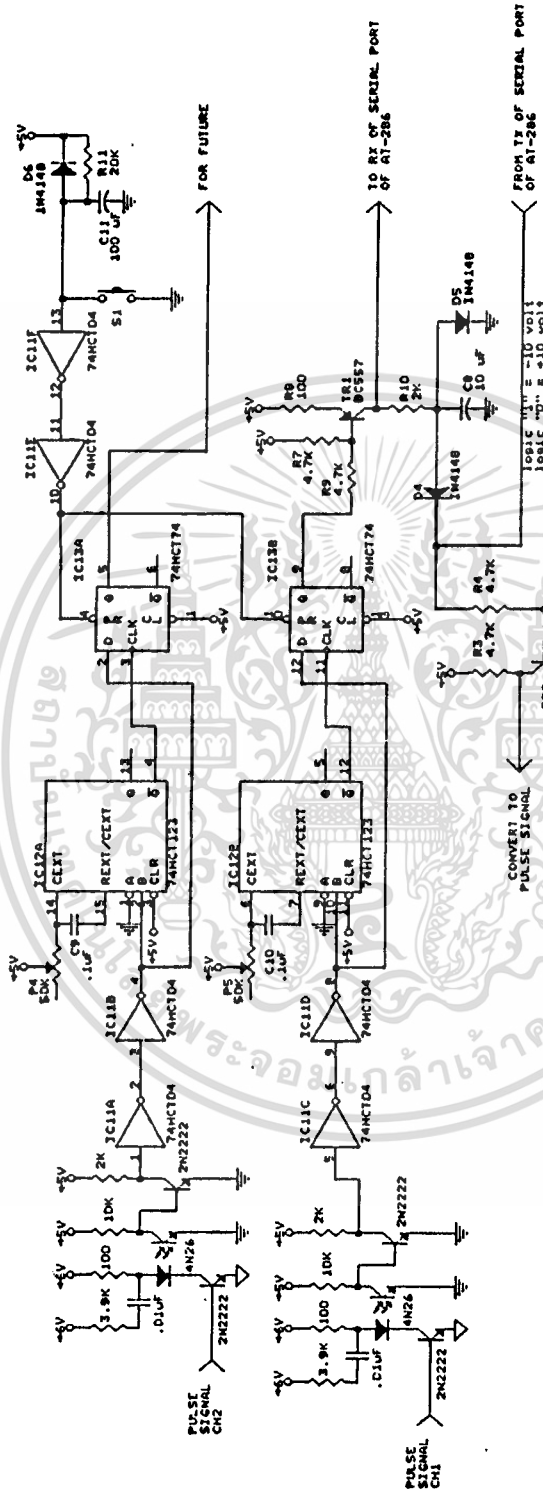
รูปที่ 3.9 บล็อคไดอะแกรมวงจรแปลงขบวนพัลซ์เป็นข้อมูลอนุกรม

หลักการทำงาน

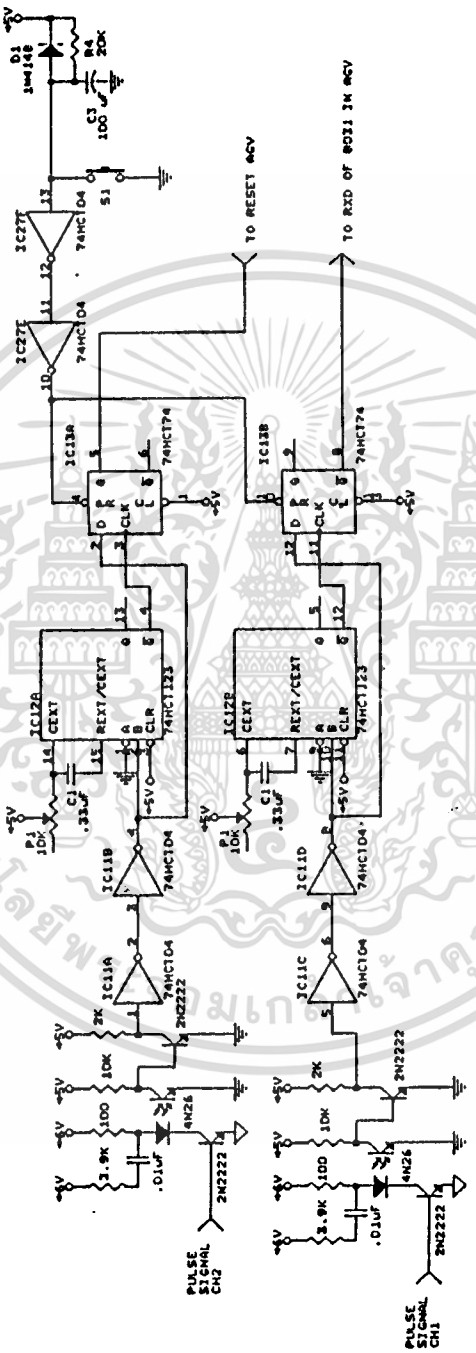
จากรูปที่ 3.9 พัลซ์ข้อมูลของสัญญาณช่องที่ 1 จะถูกส่งไปรอที่ขาข้อมูลของดีฟลิบฟลอบ ขณะเดียวกันขอบขาขึ้นของพัลซ์ข้อมูลก็จะไปกระตุ้นโมโนสเตเบิ้ลให้เอาท์พุทลอจิกศูนย์ขึ้นมาลูกหนึ่ง ลอจิกศูนย์ลูกนี้จะมีความกว้างของพัลซ์ (pulse width) อยู่ระหว่างพัลซ์แคบกับพัลซ์กว้างซึ่งได้กล่าวมาแล้ว

ลอจิกศูนย์ลูกนี้จะทำหน้าที่ตรวจสอบพัลซ์ข้อมูลว่าเป็นพัลซ์แคบหรือพัลซ์กว้าง กล่าวคือขอบขาขึ้นของลอจิกศูนย์จะไปกระตุ้นขาคล็อค (clock) ของดีฟลิบฟลอบ ทำให้ดีฟลิบฟลอบปล่อยข้อมูลออกไปยังเอาท์พุทขา Q ถ้าข้อมูลขณะกระตุ้นเป็นพัลซ์แคบ เอาท์พุทที่ขา Q จะเป็น "0" แต่ถ้าข้อมูลขณะกระตุ้นเป็นพัลซ์กว้าง เอาท์พุทที่ขา Q จะเป็น "1" ด้วยวิธีนี้ทำให้เราสามารถแปลงขบวนพัลซ์ข้อมูลให้กลับเป็นข้อมูลอนุกรมดังเดิม

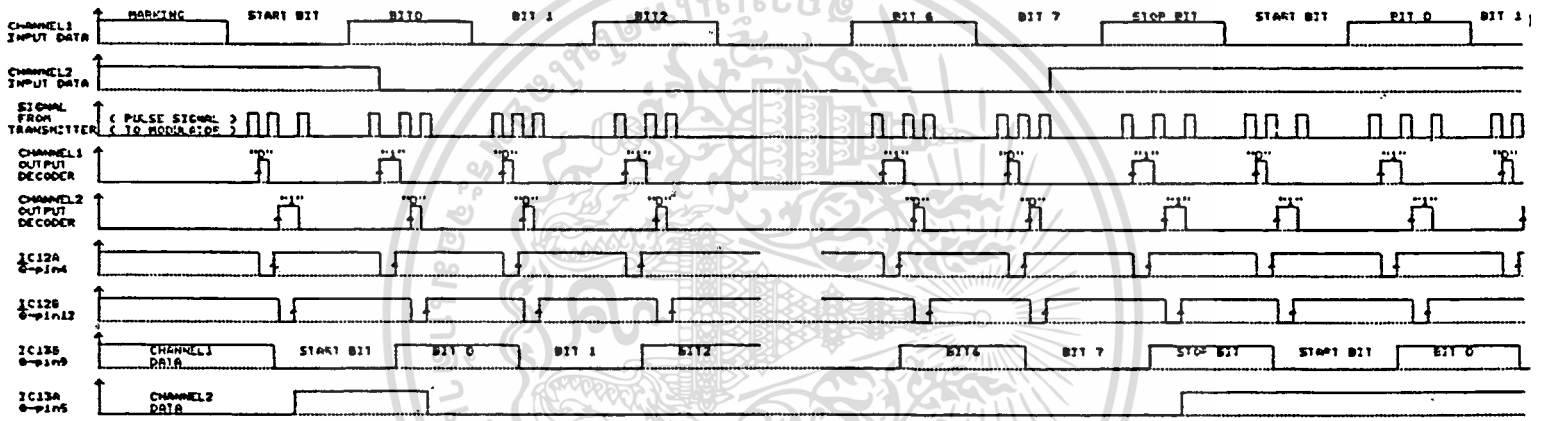
ในรูปที่ 3.10 และ 3.11 เป็นวงจรแปลงสัญญาณพัลซ์เป็นข้อมูลอนุกรมสำหรับไมโครคอมพิวเตอร์ 286-AT และสำหรับไมโครคอนโทรลเลอร์ 8031-1 ตามลำดับ ส่วนรูปที่ 3.12 แสดงแผนผังเวลาของวงจรแปลงขบวนพัลซ์เป็นข้อมูลอนุกรม ระหว่างเครื่องรับกับภาคถอดรหัสจะถูกแยกโดยออบโต้ไอโซเลเตอร์เช่นเดียวกับกรณีระหว่างภาคเข้ารหัสและภาคส่ง ทั้งนี้เก็เพื่อป้องกันปัญหาสัญญาณรบกวนที่ผ่านมาทางสายกราวน์มิให้เข้าไปกวนภาครับ



รูปที่ 3.10 วงจรแปลงแบบพัลส์เป็นข้อมูลอนุกรมสำหรับคอมพิวเตอร์ 286-AT



รูปที่ 3.11 วงจรแปลงขบวนพัลซ์เป็นข้อมูลอนุกรมสำหรับ 8031-1



รูปที่ 3.12 แผนผังเวลาของวงจรแปลงขบวนพัลซ์เป็นข้อมูลอนุกรม

3.2 ระบบเรียกใช้งานจากสถานีปฏิบัติงาน

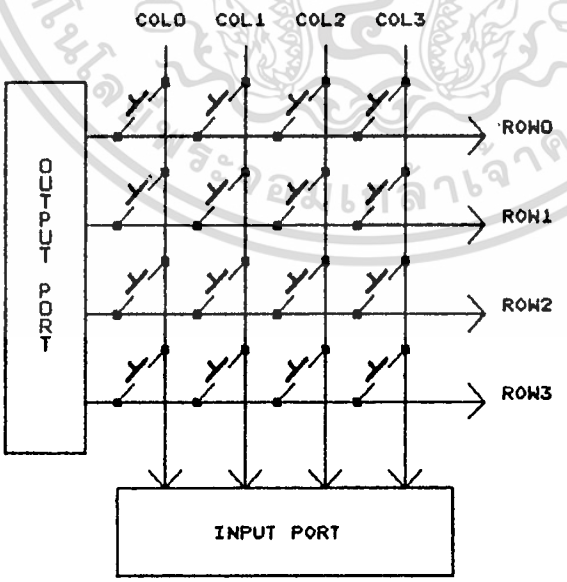
การเรียก AGV จะใช้การกดคีย์เรียกจากสถานีปฏิบัติงานนั้นๆโดยตรง สัญญาณเรียกจะถูกส่งเข้าไปยังไมโครคอมพิวเตอร์พร้อมกับรหัสหมายเลขสถานีนั้น การเรียกจะกระทำเวลาใดก็ได้โดยไม่ต้องคำนึงว่าขณะนั้นเครื่องคอมพิวเตอร์ทำอะไรอยู่ กรณีที่เครื่องคอมพิวเตอร์ไม่ว่างจะมีการเก็บข้อมูลนั้นไว้ชั่วคราว โดยรอจนกว่าเครื่องคอมพิวเตอร์ว่าง จึงจะมีการส่งข้อมูล

จำนวนสถานีปฏิบัติงานในโรงงานต่างๆนั้นเราไม่สามารถระบุแน่นอนลงไปได้ ทั้งนี้ขึ้นกับขนาดของพื้นที่ปฏิบัติงาน , ขั้นตอนการผลิต ตลอดจนปริมาณการผลิต อย่างไรก็ตามที่สำคัญที่สุดคือเราจะต้องทราบว่าสถานีใดส่งสัญญาณเรียกเข้ามายังเครื่องคอมพิวเตอร์

การกดคีย์เรียกจากสถานีต่างๆจะเลียนแบบการกดคีย์บอร์ดในเครื่องคอมพิวเตอร์ [12] วิธีการของเครื่องคอมพิวเตอร์คือจะจัดวางคีย์บอร์ดแบบเมตริกซ์ดังตัวอย่างรูปที่ 3.13

การทำงานของระบบจะมีซีพียู 1 ตัวทำหน้าที่สแกนหาการกดคีย์ วิธีตรวจสอบการกดคีย์คือจะส่ง 00h ไปยังเอาท์พุทพอร์ทแล้วอ่านค่าจากอินพุทพอร์ทหนึ่งเข้ามา ทำการตรวจสอบดูว่ามีบิทไหนบ้างที่เป็นศูนย์ หากไม่มีก็แสดงว่ายังไม่มีการกดคีย์ ซีพียูก็จะรอต่อไป หากมีแสดงว่าขณะนั้นมีการกดคีย์แล้ว ซีพียูก็จะทำการอ่านรหัสของคีย์นั้น โดยการส่ง 0 ไปยังเอาท์พุทพอร์ททีละแถว แล้วอ่านค่าจากอินพุทพอร์ทเข้ามาตรวจสอบดูว่ามีบิทใดบ้างที่เป็นศูนย์ หากยังไม่พบก็จะตรวจสอบแถวถัดไปจนกว่าจะพบ

ด้วยวิธีการนี้ประกอบกับเทคนิคทางด้านซอฟท์แวร์อีกเล็กน้อย เราก็จะได้รหัสของคีย์ที่กด

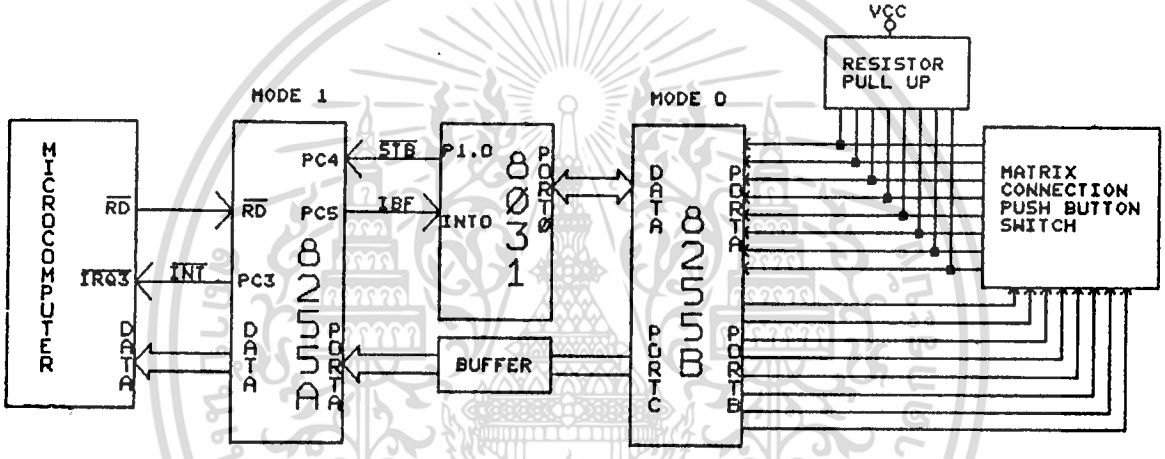


รูปที่ 3.13 ตัวอย่างคีย์บอร์ดเมตริกซ์

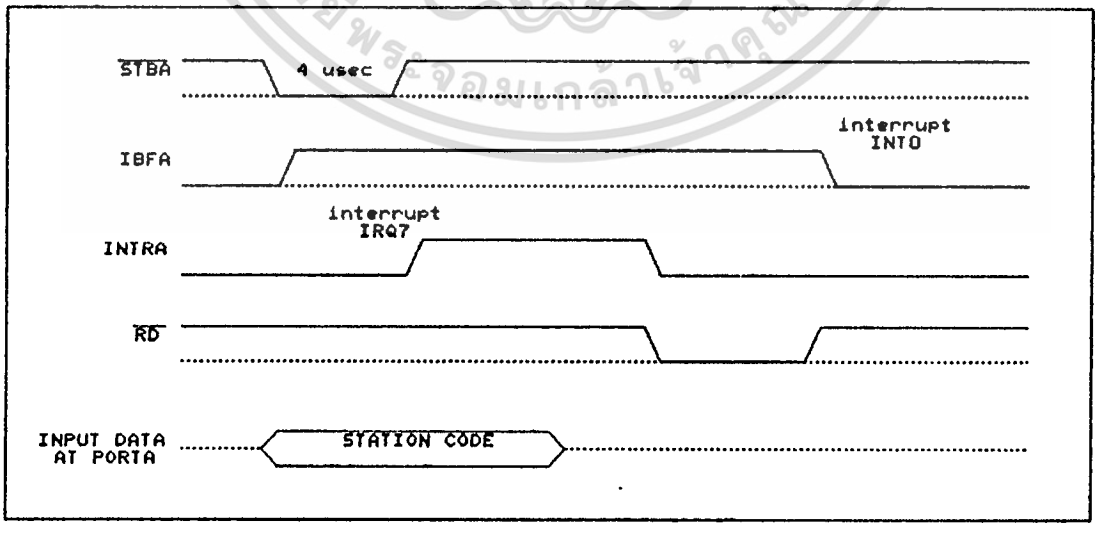
จากหลักการนี้ นำมาประยุกต์ใช้กับการกดปุ่มเรียกจากสถานีต่างๆ สำหรับรหัสสถานีที่ได้ก็จะส่งไปยังไมโครคอมพิวเตอร์ต่อไป

การติดต่อกับเครื่องคอมพิวเตอร์

เพื่อให้การส่งรหัสสถานีแก่ไมโครคอมพิวเตอร์มีประสิทธิภาพ จึงได้นำวิธี handshaking มาใช้ ในรูปที่ 3.14 เป็นบล็อคไดอะแกรมวงจรรับรหัสสถานีจาก 8031-A ด้วยวิธี handshaking โดยการตั้งโหมด 8255-A ที่โหมด 1 แบบสโตรบอินพุท (strobe input) 8255-A ตัวนี้จะจัดการเกี่ยวกับการทำ handshaking ให้ทั้งหมด



รูปที่ 3.14 บล็อคไดอะแกรมการรับรหัสสถานีจาก 8031-A ด้วยวิธี handshaking



รูปที่ 3.15 แผนผังเวลาการทำงานของ 8255-A ในโหมด 1 สโตรบอินพุท

พิจารณารูปที่ 3.14 8031-A จะทำหน้าที่สแกนตรวจการกดคีย์ปุ่มไปเรื่อยๆ ในทันทีที่มีการกดคีย์เรียกใช้จากสถานี ซีพียู 8031-A จะทำการอ่านรหัสของสถานีที่เรียกมา จากนั้นก็จะส่งรหัสสถานีนั้นออกไปยังพอร์ท C ของ 8255-B พร้อมด้วยสัญญาณ STBA แบบเดียวกับรูปที่ 3.15 ไปยัง 8255-A

8255-A จะส่งสัญญาณ IBF="1"เป็นการตอบรับเพื่อบอกให้ 8031-A ทราบว่าขณะนี้มีข้อมูลอยู่ในบัฟเฟอร์อย่าเพิ่งส่งข้อมูลใหม่เข้ามา ในขณะเดียวกัน 8255-A ก็จะส่งสัญญาณ INTRA ไปอินเตอรัพท์ไมโครคอมพิวเตอร์เพื่อให้ไมโครคอมพิวเตอร์มารับข้อมูลรหัสสถานีซึ่งรออยู่ที่พอร์ท A

แต่ถ้าส่งสัญญาณ INTRA ไปแล้ว แต่ไมโครคอมพิวเตอร์ยังไม่ว่างเพราะกำลังทำอินเตอรัพท์ที่มีลำดับความสำคัญสูงกว่าอยู่ 8255-A ก็จะคงลอจิก IBF = "1" ไปเรื่อยๆ ซึ่งหากขณะนั้นเกิดมีการกดคีย์จากสถานีอื่นซ้อนเข้ามาอีก 8031-A จะทำการเก็บรหัสสถานีใหม่นั้นในหน่วยความจำเป็นการชั่วคราว

เมื่อไมโครคอมพิวเตอร์มาอ่านข้อมูลที่ค้างไปแล้ว สัญญาณ IBF ก็จะกลับเป็น "0" ตามเดิมเป็นการบอกให้ 8031-A ทราบว่า 8255-A บัฟเฟอร์ว่างและพร้อมที่จะรับข้อมูลใหม่ การเปลี่ยนลอจิกจาก "1"เป็น "0" ของ IBF จะเป็นสัญญาณอินเตอรัพท์ INTO ให้ 8031-A โปรแกรมอินเตอรัพท์ INTO ของ 8031-A จะไปดึงข้อมูลรหัสสถานีที่เก็บไว้ในหน่วยความจำ (ถ้ามี) แล้วส่งออกไปพอร์ท C ของ 8255-B พร้อมกับส่งสัญญาณ STB ไปยัง 8255-A เป็นการเริ่มการทำ handshaking ครั้งใหม่

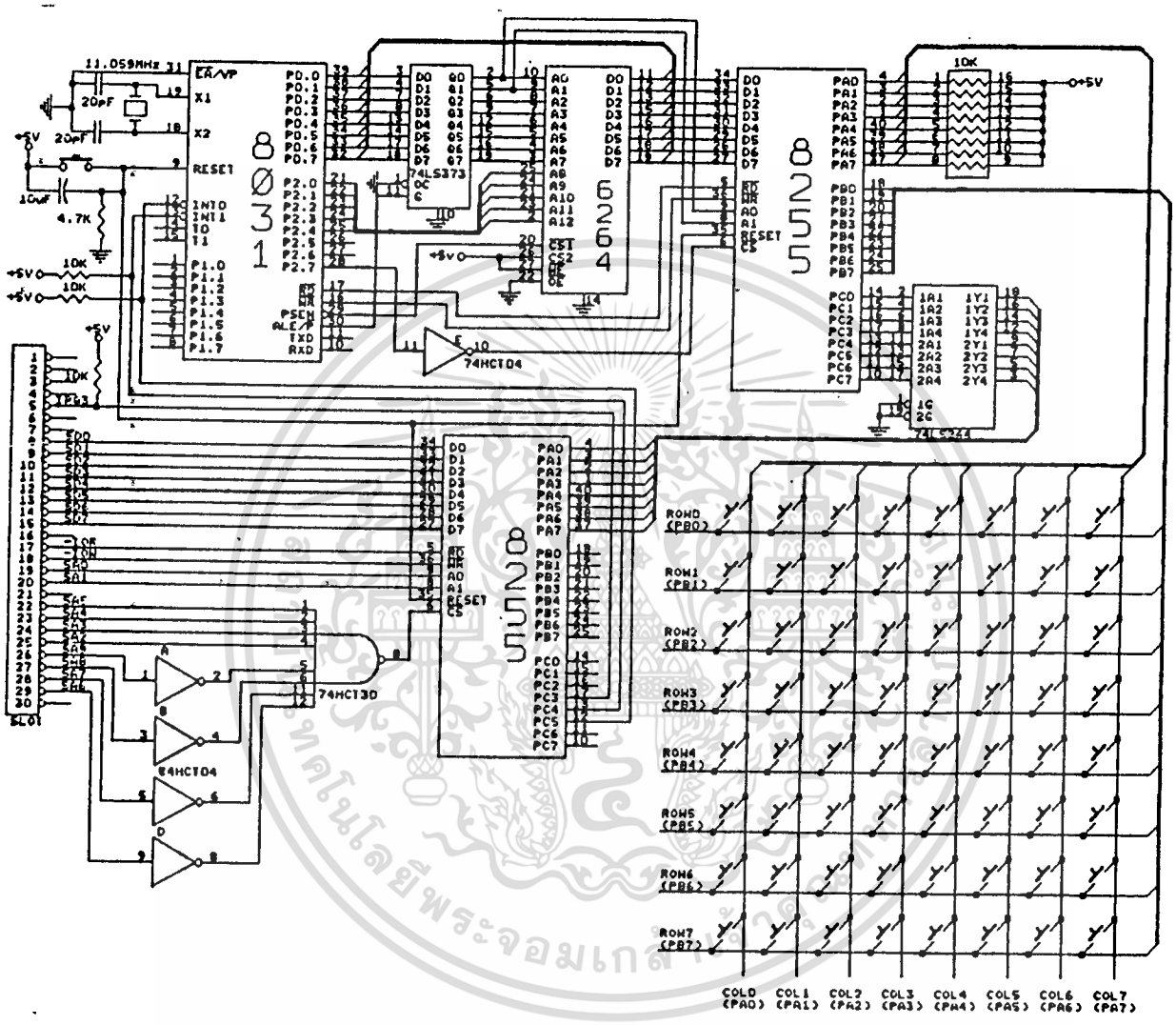
สำหรับการเก็บข้อมูลรหัสสถานีในหน่วยความจำและดึงออกไปใช้นั้นจะเป็นแบบเข้าก่อน-ออกก่อน (first in-first out) ดังนั้นสถานีใดเรียกเข้ามาก่อนก็จะถูกส่งไปเข้าไมโครคอมพิวเตอร์ก่อน สถานีใดเรียกทีหลังก็จะถูกส่งไปทีหลัง

การส่งรหัสสถานีให้ไมโครคอมพิวเตอร์ด้วยวิธี handshaking นี้ทำให้เราสามารถที่จะเรียกใช้ AGV จากสถานีปฏิบัติงานในขณะใดก็ได้โดยไม่ต้องคำนึงว่าเครื่องคอมพิวเตอร์จะทำงานอื่นอยู่หรือไม่ ทำให้ระบบการเรียกใช้ AGV มีประสิทธิภาพสูง

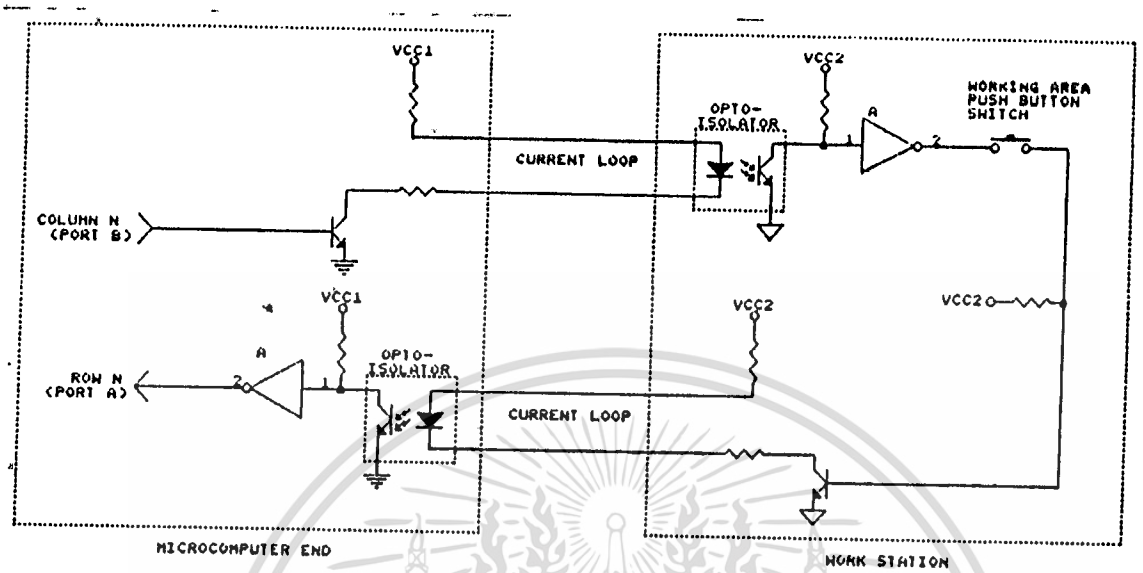
สำหรับกรณีที่มีการเรียกพร้อมกันจากหลายสถานี 8031-A จะไม่สนใจและจะรอการกดปุ่มครั้งต่อไป ทั้งนี้เพื่อป้องกันปัญหาการแปลรหัสสถานีผิดพลาด สำหรับวงจรที่ใช้ทดลองแสดงไว้ดังรูปที่ 3.16

การเรียกใช้ AGV จากระยะไกล

สำหรับการเรียกระยะไกลจากพื้นที่ปฏิบัติงานไปยังเครื่องคอมพิวเตอร์นั้น เพื่อป้องกันปัญหาสัญญาณรบกวน จึงเลือกใช้วิธีส่งข้อมูลแบบกระแสวนลูป (current loop) ลักษณะของวงจรกระแสวนลูปเป็นไปตามรูปที่ 3.17



รูปที่ 3.16 วงจรรับข้อมูลการกดคีย์เรียกและเชื่อมต่อกับคอมพิวเตอร์ 286-AT



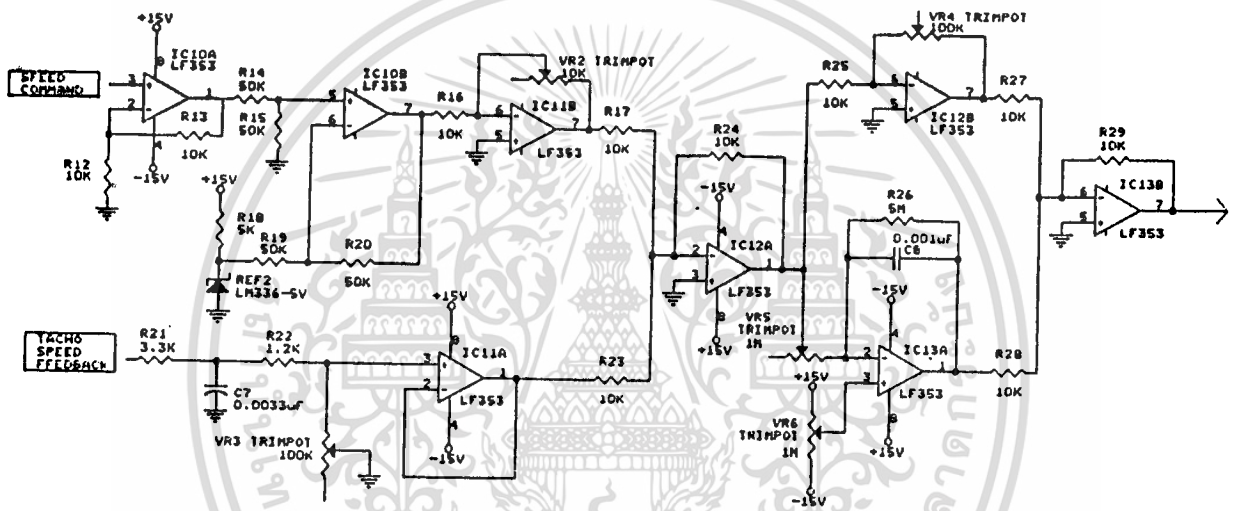
รูปที่ 3.17 วงจรกระแสวนลูป

3.3 ระบบควบคุมการเคลื่อนที่

จากทฤษฎีระบบควบคุมความเร็วและตำแหน่งในบทที่ 2 นำมาสร้างระบบควบคุมความเร็วและตำแหน่งของมอเตอร์กระแสตรง ได้ดังนี้

วงจรอนาล็อค: PI คอนโทรลเลอร์

วงจร PI คอนโทรลเลอร์สามารถสร้างขึ้นได้จากออปแอมป์ โดยพิจารณาการสร้างจากสมการ (2.3) ดังรูปที่ 3.18



รูปที่ 3.18 วงจร PI คอนโทรลเลอร์

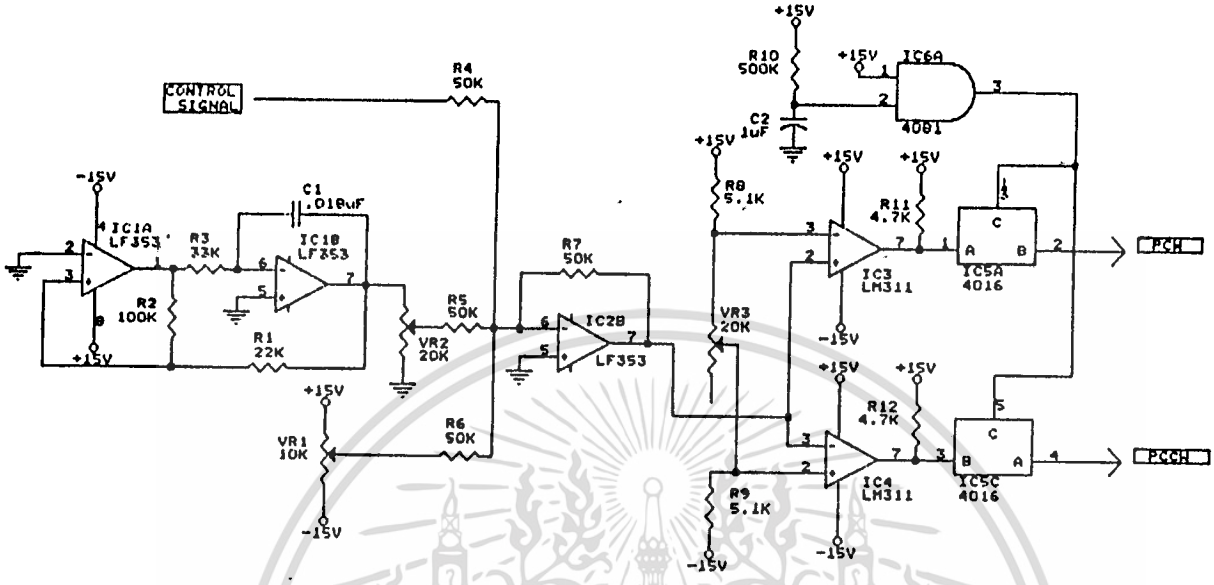
จากวงจรจะได้ว่า

$$K_p = VR4/R25 \qquad \ldots.(3.1)$$

$$Ti = (C8)(VR5) \qquad \ldots.(3.2)$$

วงจรพัลซ์วิดท์มอดูเลเตอร์ (Pulse width modulator)

วงจรพัลซ์วิดท์มอดูเลชั่นหรือ PWM เป็นวงจรสำหรับสร้างพัลซ์ ที่สามารถเปลี่ยนแปลงความกว้างของพัลซ์ตามระดับโวลท์เตจอินพุท เพื่อนำไปขับวงจรสวิทชิ่งเซอร์โวแอมป์ (SSA) วงจรนี้ประกอบด้วยส่วนกำเนิดสัญญาณสามเหลี่ยมและส่วนสร้างพัลซ์ดังในรูปที่ 3.19



รูปที่ 3.19 วงจรพัลซ์วิดท์มอดดูเลชั่น

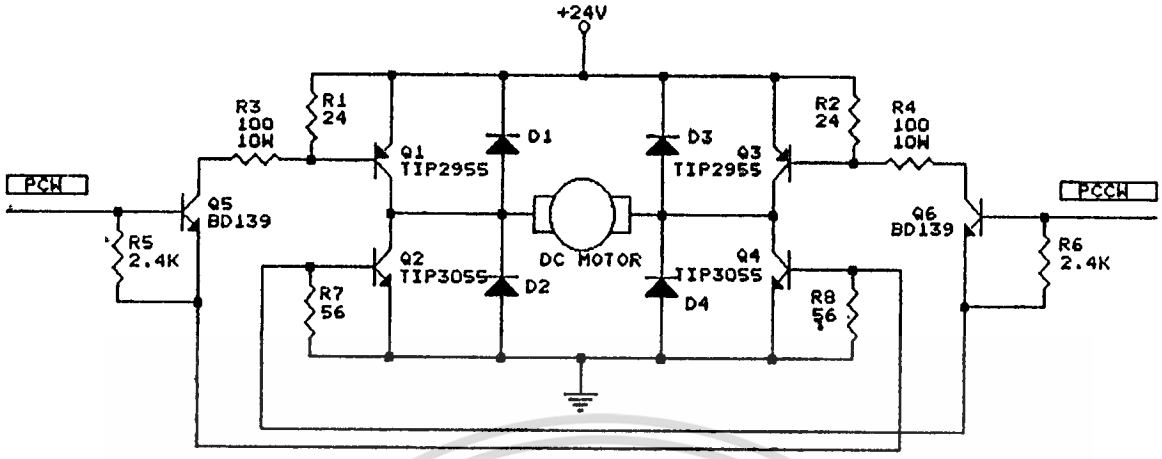
จากรูปที่ 3.19 ในส่วน IC1 เป็นวงจรกำเนิดสัญญาณรูปฟันเลื่อยซึ่งมีสูตรในการคำนวณคือ

$$f_o = R_2/4RCR_1 \qquad \ldots.(3.3)$$

ในส่วน IC2 ทำหน้าที่รวมสัญญาณอินพุทกับสัญญาณสามเหลี่ยม LM311 ต่อในลักษณะเป็นวงจร window comparator ซึ่งมีหน้าที่สร้างพัลซ์กล่าวคือเมื่อโวลท์เตจอินพุทมีค่าบวก จะได้พัลซ์ออกมาทาง pcw และเมื่อโวลท์เตจอินพุทมีค่าลบจะได้พัลซ์ออกมาทาง pccw โดยความกว้างของพัลซ์ทั้งสองเป็นสัดส่วนโดยตรงกับค่าสัมบูรณ์ของโวลท์เตจอินพุท

วงจรสวิทชิ่งเซอร์โวแอมป์ (Switching servo amplifier)

วงจรสวิทชิ่งเซอร์โวแอมป์มีหน้าที่ขยายสัญญาณอินพุทเพื่อไปขับมอเตอร์กระแสตรง การออกแบบวงจรสวิทชิ่งเซอร์โวแอมป์หรือ SSA จำเป็นต้องคำนึงถึงการไหลของกระแสไฟฟ้าในมอเตอร์ สัญญาณที่ใช้ขับวงจร SSA มีลักษณะเป็นพัลซ์จึงมีผลทำให้กระแสไหลในตัวมอเตอร์ไม่ต่อเนื่อง รายละเอียดของวงจรดังแสดงไว้ในรูปที่ 3.20



รูปที่ 3.20 วงจรสวิทชิ่งเซอร์โวแอมป์

จากรูปที่ 3.20 Q1,Q2,Q3 และ Q4 ทำหน้าที่หมือนสวิทช์ 4 ตัวของวงจร Q5 และ Q6 มีหน้าที่ขยายกระแสให้กับ Q1 และ Q3 ตามลำดับ ในการทำงานจะทำเป็นคู่ๆกล่าวคือ ถ้ามีสัญญาณส่งมาจาก pcw Q1 และ Q4 จะทำงาน แต่ถ้ามีสัญญาณส่งมาจาก pccw Q2 และ Q3 ก็จะทำงาน ส่วน Q1 และ Q4 จะหยุดทำงานไป

วงจรระบบควบคุมความเร็วและตำแหน่งมอเตอร์กระแสตรง

จากบล็อคไดอะแกรมลูปการควบคุมตำแหน่งและความเร็วของมอเตอร์กระแสตรง ดังรูปที่ 2.14 นำมาสร้างเป็นวงจรได้ดังรูปที่ 3.21

วงจรประกอบด้วยไมโครคอนโทรลเลอร์ 8031-2 ทำหน้าที่นับค่าพัลซ์ของตำแหน่ง ซึ่งส่งมาจากโรตารี่เอนโคดเดอร์ พร้อมทั้งตรวจสอบทิศทางการหมุนของมอเตอร์ด้วยว่าถูกต้องหรือไม่ จากนั้นจะนำค่าที่นับได้ไปเปรียบเทียบกับค่าตั้งทางตำแหน่ง (position setpoint) เพื่อประมวลผลหาค่าความเร็วของมอเตอร์ที่เหมาะสมจากตารางความเร็วที่ 4.1 ซึ่งเก็บอยู่ในหน่วยความจำ แล้วจะส่งค่าความเร็วที่ได้ออกไปยังพอร์ท 1 เพื่อเข้าวงจร D/A เอาท์พุทจาก D/A จะเป็นสัญญาณคำสั่งทางความเร็ว (speed command) สำหรับส่งเข้าวงจร PI คอนโทรลเลอร์ต่อไป

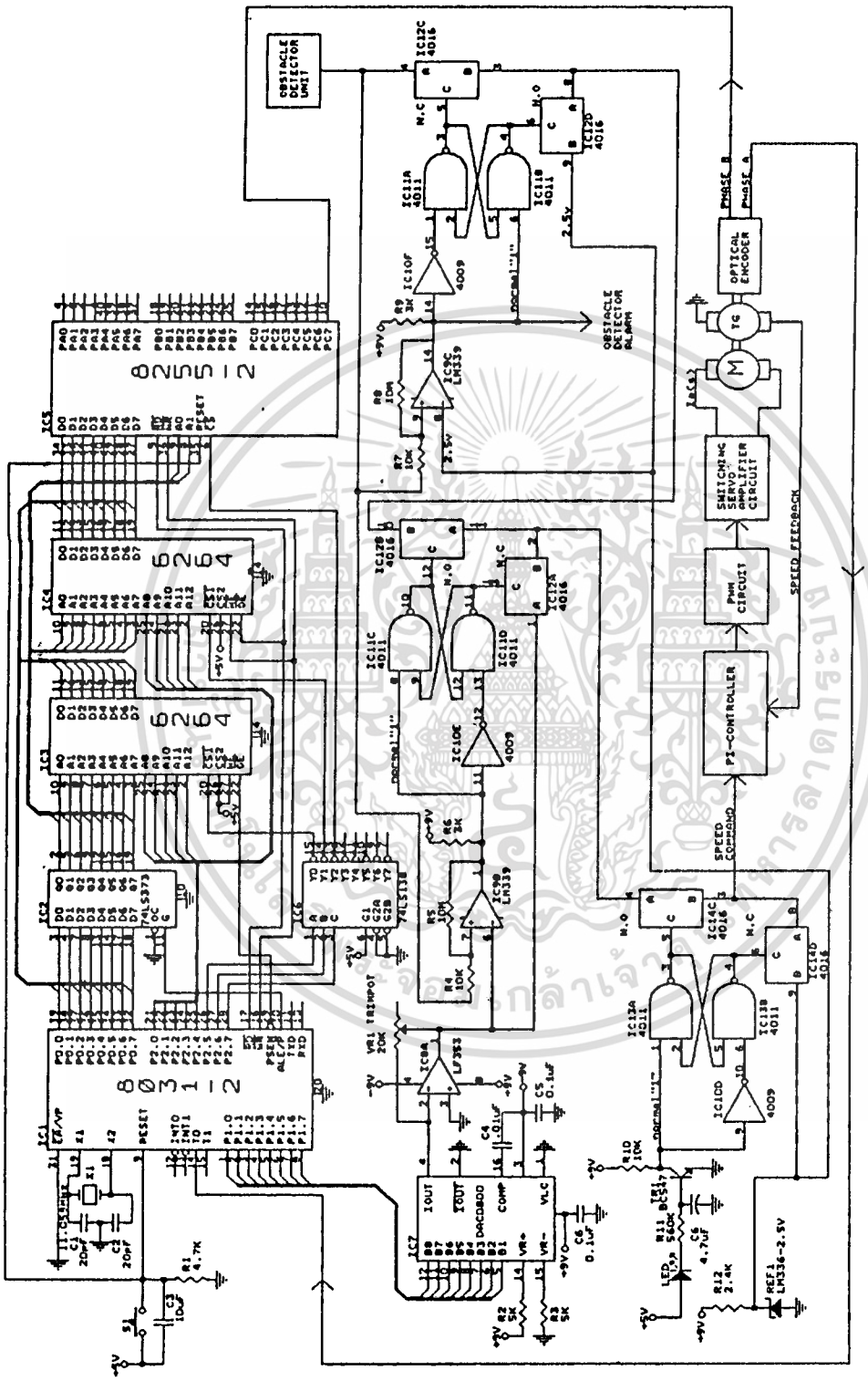
สัญญาณพัลซ์ของตำแหน่งที่ได้จากโรตารี่เอนโคดเดอร์ประกอบด้วยสัญญาณจากช่องสัญญาณ A ซึ่งจะส่งไปเข้าขาอินเตอรัพท์ T0 ของไมโครคอนโทรลเลอร์ และสัญญาณจากช่องสัญญาณ B ซึ่งจะส่งไปเข้าขา PC7 ของ 8255-2

ในส่วนชุดวงจรเปรียบเทียบและอนาล็อคสวิทช์ (IC9B, IC10E, IC11C-D, IC12A-B) ตามรูปมีหน้าที่เปรียบเทียบ ค่าความเร็วที่ส่งออกมาจาก D/A กับค่าแรงดันเอาท์พุทจากวงจรตรวจจับวัตถุกีด

ขวาง กรณีที่พบวัตถุกีดขวาง แรงดันจากวงจรตรวจจับวัตถุกีดขวางจะมีค่าต่ำกว่าค่าความเร็วจาก D/A วงจรเปรียบเทียบและอนาล็อคสวิทช์จะเปลี่ยนไปรับค่าแรงดันจากวงจรตรวจจับวัตถุกีดขวางแทน ด้วยวิธีนี้เราจะได้ค่าความเร็วของมอเตอร์ที่แปรผันโดยตรงกับระยะระหว่างวัตถุกีดขวางกับตัวอุตร้าโซนิค ทำให้การชะลอความเร็วในขณะที่ AGV ตรวจพบวัตถุกีดขวางมีความนุ่มนวลยิ่งขึ้น

จากรูปที่ 3.21 นี้ ค่าความเร็วเป็นศูนย์จะตรงกับระดับแรงดันคือ 2.5 โวลท์ กรณีที่ AGV วิ่งเข้าไปใกล้วัตถุมากขึ้นจนกระทั่งแรงดันเอาท์พุทจากวงจรตรวจจับมีค่าต่ำกว่า 2.5 โวลท์ วงจรเปรียบเทียบและอนาล็อคสวิทช์ (IC9C, IC10F, IC11A-B, IC12C-D) จะทำการตัดค่าแรงดันเอาท์พุทของวงจรตรวจจับที่ส่งเข้าวงจร PI คอนโทรลเลอร์ออก ทำให้มอเตอร์หยุดหมุน โดยจะเปลี่ยนไปรับค่าแรงดันคงที่ 2.5 โวลท์จาก reference diode แทน

สำหรับวงจรอนาล็อค (IC10D, IC13A-B, IC14C-D) นั้นทำหน้าที่จ่ายแรงดันคงที่ 2.5 โวลท์ในช่วงที่ยังไม่มีการสวิทช์ "on" วงจรไมโครคอนโทรลเลอร์ ทั้งนี้ก็เพื่อป้องกันมิให้มอเตอร์หมุนตลอดเวลาในขณะที่วงจรไมโครคอนโทรลเลอร์ยัง "off" อยู่ แต่ทันทีที่มีการสวิทช์ "on" วงจรไมโครคอนโทรลเลอร์ วงจรอนาล็อคสวิทช์นี้จะเปลี่ยนไปรับค่าความเร็ว +2.5 โวลท์จาก D/A แทน



รูปที่ 3.21 วงจรระบบควบคุมความเร็วและตำแหน่งมอเตอร์กระแสตรง

3.4 ระบบเซนเซอร์

ระบบเซนเซอร์ที่ศึกษาในวิทยานิพนธ์นี้ประกอบด้วยส่วนต่างๆดังนี้

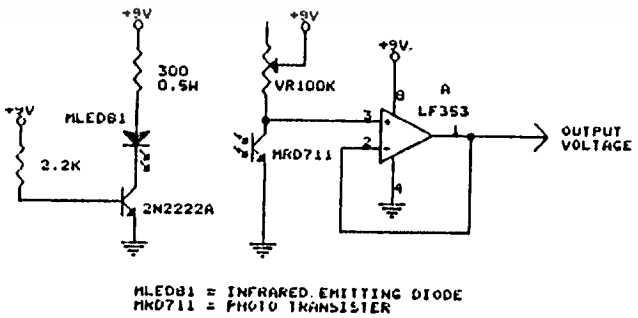
1) ระบบเซนเซอร์สำหรับใช้ในการนำร่อง
2) ระบบเซนเซอร์ที่ใช้สำหรับตรวจจับวัตถุกีดขวาง
3) ระบบเซนเซอร์ที่ใช้ตรวจวัดระดับแรงดันของแบตเตอรี่

3.4.1 ระบบนำร่องด้วยโฟโต้เซนเซอร์

สำหรับ AGV ตามแนวคิดในวิทยานิพนธ์นี้ เป็นระบบ AGV ที่นำร่องโดยใช้โฟโต้เซนเซอร์ ทางเดินนำร่อง (guide path) ใช้เทปสีติดตามพื้น โฟโต้เซนเซอร์จะทำการตรวจหาตำแหน่งของเทปสีแล้วส่งสัญญาณไปเข้าวงจรควบคุมการนำร่อง วงจรควบคุมการนำร่องจะทำหน้าที่ควบคุมให้ AGV วิ่งไปตามแนวเส้นทางที่กำหนดโดยไม่ออกนอกทาง สำหรับตัวอย่างวงจรควบคุมการนำร่องนั้นแสดงไว้ในรูปที่ ข.9 ตามภาคผนวก ข ซึ่งเป็นแนวทางในการพัฒนา AGV ในขั้นต่อไป

การพิจารณาเลือกสีของเทปนั้นต้องเปรียบเทียบกับสีของพื้นโรงงานเป็นหลัก สีของเทปที่เลือกใช้ต้องให้ผลแตกต่างของความเข้มแสงสะท้อนอย่างเด่นชัด เช่น กรณีที่พื้นโรงงานเป็นสีอ่อนก็ควรเลือกใช้เทปสีเข้มเช่น สีดำเป็นต้น ลักษณะของวงจรโฟโต้เซนเซอร์แสดงดังรูปที่ 3.22 วงจรที่ออกแบบขึ้นนี้ใช้ตรวจหาตำแหน่งเทปสีดำ อุปกรณ์สำคัญได้แก่ไดโอดเปล่งแสงอินฟราเรด MLED81 และโฟโต้ดีเทคเตอร์ MRD711

วงจรโฟโต้เซนเซอร์



รูปที่ 3.22 วงจรโฟโต้เซนเซอร์

หลักกการทำงาน

จากรูปที่ 3.22 MLED81 ทำหน้าที่ปล่อยแสงอินฟราเรดไปตกกระทบพื้นหรือเทปสีแล้วสะท้อนกลับขึ้นไปยัง MRD711 ปริมาณความเข้มของแสงสะท้อนที่ MRD711 รับได้จะขึ้นกับโทนสีของตำแหน่งที่แสงไปตกกระทบเป็นหลัก ถ้าตำแหน่งนั้นมีสีอ่อน (พื้น) การดูดกลืนแสงจะน้อยกว่าสีเข้ม แสงส่วนใหญ่จะสะท้อนกลับไปเข้า MRD711 ได้มากกว่าส่งผลให้ MRD711 "on" แต่ถ้าตำแหน่งนั้นมีสีเข้ม (เทป-สีดำ) การดูดกลืนแสงจะมีมากกว่า แสงที่สะท้อนกลับไปเข้า MRD711 จะมีปริมาณน้อยกว่า ส่งผลให้ MRD711 "off" แรงดัน VCE ของ MRD711 จะขึ้นกับปริมาณแสงที่ตกกระทบเป็นสำคัญ ความสัมพันธ์จะเป็นเชิงเส้นหรือไม่จะขึ้นกับการปรับ VR แรงดันเอาท์พุทที่ได้จะส่งต่อไปเข้าวงจรควบคุมการนำร่องต่อไป

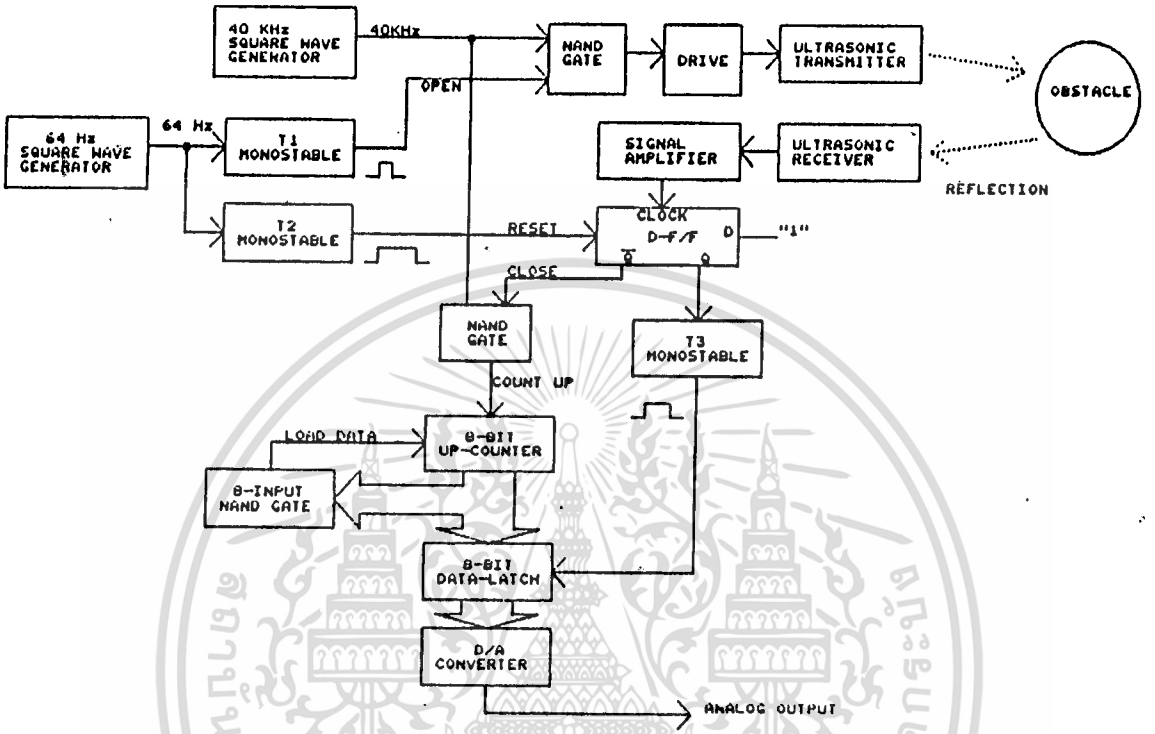
3.4.2 ระบบตรวจจับวัตถุกีดขวางด้วยอุตร้าโซนิค

การตรวจจับสิ่งกีดขวางมักกระทำขณะที่ AGV กำลังวิ่งไปตามเส้นทาง โดยใช้อุตร้าโซนิคตรวจหาวัตถุ เมื่ออุตร้าโซนิคตรวจพบแล้ว จะส่งสัญญาณไปยังวงจรควบคุมความเร็วมอเตอร์กระแสตรง สัญญาณนี้จะกลายเป็นค่าตั้งทางความเร็ว (speed setpoint) ให้กับมอเตอร์แทนค่าเดิมที่ส่งออกมาจากไมโครคอนโทรลเลอร์ 8031-2 ด้วยวิธีนี้จะทำให้ความเร็วของมอเตอร์แปรผันโดยตรงกับระยะห่างระหว่างวัตถุกีดขวางกับตัวอุตร้าโซนิค ซึ่งใน AGV จริงมักจะติดตั้งอยู่ตรงส่วนหน้าสุดของตัวรถ ยิ่งวัตถุนั้นเคลื่อนที่เข้ามาใกล้ตัวอุตร้าโซนิคมากเท่าไร ความเร็วของมอเตอร์ก็จะยิ่งช้าลงมากเท่านั้น และมอเตอร์จะหยุดหมุนทันทีที่ระยะห่างระหว่างวัตถุกับตัวอุตร้าโซนิคมีค่าต่ำกว่าค่าระยะต่ำสุดที่ตั้งไว้ ซึ่งเป็นระยะปลอดภัย

เมื่อวัตถุนั้นเคลื่อนที่ห่างออกจากอุตร้าโซนิค มอเตอร์ก็จะเริ่มหมุนอีกครั้งด้วยความเร็วที่เป็นสัดส่วนโดยตรงกับระยะห่างที่เพิ่มขึ้น และมอเตอร์จะกลับไปรับค่าตั้งทางความเร็วที่ส่งมาจากไมโครคอนโทรลเลอร์ 8031-2 ตามปกติอีกครั้ง เมื่อระยะห่างระหว่างวัตถุกีดขวางกับตัวอุตร้าโซนิคมีค่าเกินค่าระยะสูงสุดที่ตั้งไว้

ลักษณะการทำงานที่สัมพันธ์กันของการตรวจจับวัตถุกีดขวางกับไมโครคอนโทรลเลอร์ 8031-2 ดังกล่าวนี้ ทำได้โดยการต่อวงจรเปรียบเทียบและอนาล็อคสวิทช์ดังแสดงในรูปที่ 3.21

สำหรับหลักการทำงานในการตรวจจับวัตถุกีดขวางด้วยอุตร้าโซนิค แสดงไว้ในรูปที่ 3.23



รูปที่ 3.23 บล็อคไดอะแกรมแสดงการตรวจจับวัตถุกีดขวาง

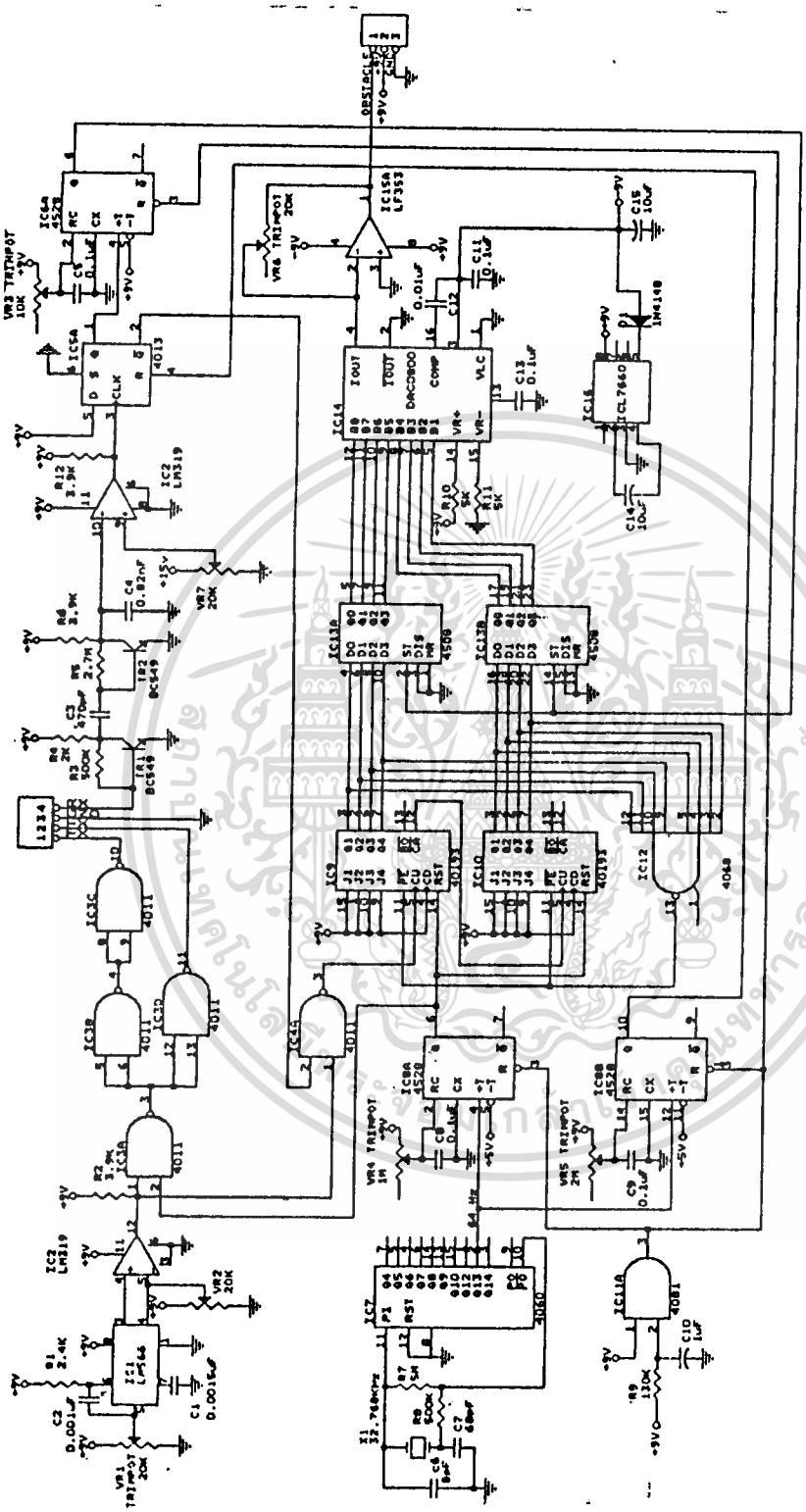
หลักการทำงาน

วงจรตรวจจับวัตถุกีดขวางจะให้สัญญาณแรงดันเอาท์พุท 2 - 5 โวล์ท ที่สัมพันธ์กับระยะ 0.3 - 1.2 เมตร ซึ่งเป็นระยะห่างระหว่างวัตถุกีดขวางกับตัวอุตร้าโซนิค สำหรับกรณีระยะมากกว่า 1.2 เมตรขึ้นไปนั้น วงจรจะให้แรงดันเอาท์พุท 5 โวลท์คงที่

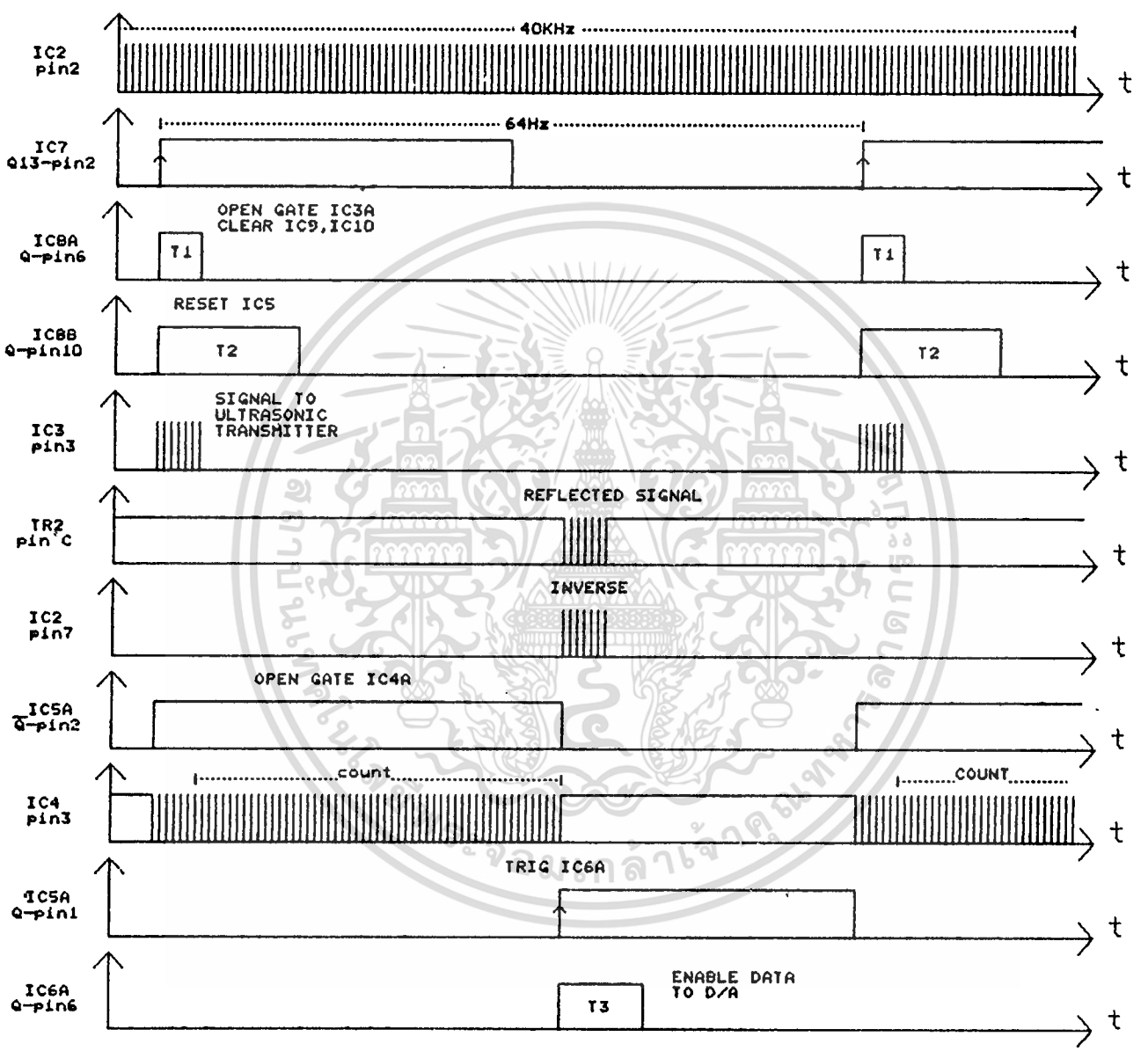
แรงดันเอาท์พุทนี้จะถูกใช้เป็นค่าตั้งทางความเร็ว (speed setpoint) สำหรับวงจรควบคุมความเร็วแบบ PI คอนโทรลเลอร์ เฉพาะกรณีที่วัตถุกีดขวางอยู่ห่างจากตัวอุตร้าโซนิคในช่วงที่กำหนด คือ 0.3 - 1.2 เมตร เท่านั้น แต่ในสภาวะปกติที่ไม่มีวัตถุกีดขวางวงจรควบคุมความเร็วจะรับค่าตั้งทางความเร็วจากไมโครคอนโทรลเลอร์ 8031-2 เท่านั้น

ตามรูปที่ 3.23 การตรวจจับวัตถุสามารถทำได้โดยการส่งคลื่นอุตร้าโซนิคความถี่ 40 KHz ออกจากตัวส่ง (transmitter) ทีละประมาณ 10-20 พัลซ์ โดยใช้ความถี่ในการส่งเท่ากับ 64 Hz หรือส่งประมาณทุกๆ 15 msec คลื่นอุตร้าโซนิคที่ส่งออกไปมีความเร็วประมาณ 330 m/sec จะไปกระทบวัตถุกีดขวาง (obstacle) แล้วสะท้อนกลับมายังตัวรับ (receiver) จากนั้นสัญญาณที่รับได้จะผ่านเข้าภาคขยายสัญญาณ เพื่อส่งไปกระตุ้นดีฟลิบฟลอบต่อไป

ขอบขาขึ้นของพัลซ์สี่เหลี่ยมที่ผลิตจากภาคกำเนิดความถี่ 64 Hz จะไปกระตุ้นโมโนสเตเบิ้ล 2 ตัวพร้อมกัน โมโนสเนเบิ้ลตัวแรกจะสร้างพัลซ์มีความกว้างพัลซ์ T1 ส่วนโมโนสเตเบิ้ลตัวที่สองจะสร้างพัลซ์มีความกว้างพัลซ์ T2 พัลซ์ T1 มีหน้าที่เปิดเกทเพื่อปล่อยให้พัลซ์สี่เหลี่ยมที่ผลิตจากภาคกำเนิดความถี่ 40 KHz ออกไปสู่ตัวส่งอุตร้าโซนิค ด้วยวิธีนี้เราจะได้คลื่นอุตร้าโซนิคออกไปทีละประมาณ 10 ถึง 20 พัลซ์ ในทุกๆ ประมาณ 15 msec ส่วนพัลซ์ T2 มีหน้าที่คอยรีเซท (reset) ดีฟลิบฟลอบซึ่งต่อขาข้อมูลรอไว้ที่ลอจิก "1" สำหรับพัลซ์สี่เหลี่ยมความถี่ 40 KHz นั้น ส่วนหนึ่งจะถูกส่งไปเข้าตัวนับความถี่ (counter) ซึ่งก่อนถึงจะมีเกทกั้นอยู่ 1 ตัว เกทตัวนี้เปิดปิดโดยเอาท์พุทขา Q ของดีฟลิบฟลอบ การนับความถี่จะเริ่มขึ้นทันทีที่มีการรีเซทดีฟลิบฟลอบด้วยพัลซ์ T2 และจะนับไปเรื่อยๆจนกว่าจะมีสัญญาณอุตร้าโซนิคสะท้อนกลับมา ขอบขาขึ้นของสัญญาณอุตร้าโซนิคที่สะท้อนกลับมาจะกระตุ้นดีฟลิบฟลอบให้ปล่อยลอจิก "1" ออกไปเอาท์พุทเป็นผลให้เกทที่กั้นตัวนับปิดตัว การนับจึงสิ้นสุดลง อย่างไรก็ตามปริมาณความถี่ที่นับได้จะถูกกักไว้ด้วยตัวกักข้อมูล (data lacth) สำหรับขา Q ของดีฟลิบฟลอบจะต่ออยู่กับโมโนสเตเบิ้ลตัวที่สามซึ่งจะสร้างพัลซ์ความกว้าง T3 ทุกครั้งที่ขา Q เปลี่ยนค่าจาก 0 เป็น 1 เมื่อมีสัญญาณอุตร้าโซนิคสะท้อนกลับมา พัลซ์ T3 มีหน้าที่อีนาเบิ้ลตัวกักข้อมูลให้ปล่อยปริมาณความถี่ที่นับได้ไปเข้าตัว D/A เพื่อแปลงเป็นสัญญาณอนาล็อคต่อไป ด้วยวิธีนี้ปริมาณความถี่ที่นับได้จะถูกส่งไปเข้าตัว D/A เฉพาะเมื่อมีอุตร้าโซนิคสะท้อนกลับมาเท่านั้น แรงดันเอาท์พุทที่ได้จึงสัมพันธ์กับระยะห่างระหว่าง AGV กับวัตถุ เนื่องจากค่าที่ตัวนับนับได้สูงสุดคือ 255 ซึ่งจะเกิดขึ้นเมื่อวัตถุห่างจาก AGV โดยประมาณตั้งแต่ 1.2 เมตรขึ้นไป ดังนั้นจึงได้กำหนดให้ตัวนับคงค่า 255 ไว้กรณีที่ระยะห่างระหว่าง AGV กับวัตถุเกิน 1.2 เมตร ซึ่งตรงกับแรงดันเอาท์พุทของ D/A คือ 5 โวลท์ นั่นคือกรณีอุตร้าโซนิคไม่สะท้อนกลับ หรือไม่สามารถตรวจจับสัญญาณได้ เพราะสัญญาณอ่อนมากเนื่องจากระยะวัตถุห่างจาก AGV มาก ปริมาณความถี่ที่นับได้ก็จะคงเดิมคือ 255 เป็นผลให้แรงดันเอาท์พุทจาก D/A คงที่ที่ 5 โวลท์ ในรูปที่ 3.24 แสดงวงจรจับวัตถุกีดขวางที่สมบูรณ์ ส่วนในรูปที่ 3.25 แสดงแผนผังเวลาแสดงการทำงานของวงจรตรวจจับวัตถุกีดขวาง



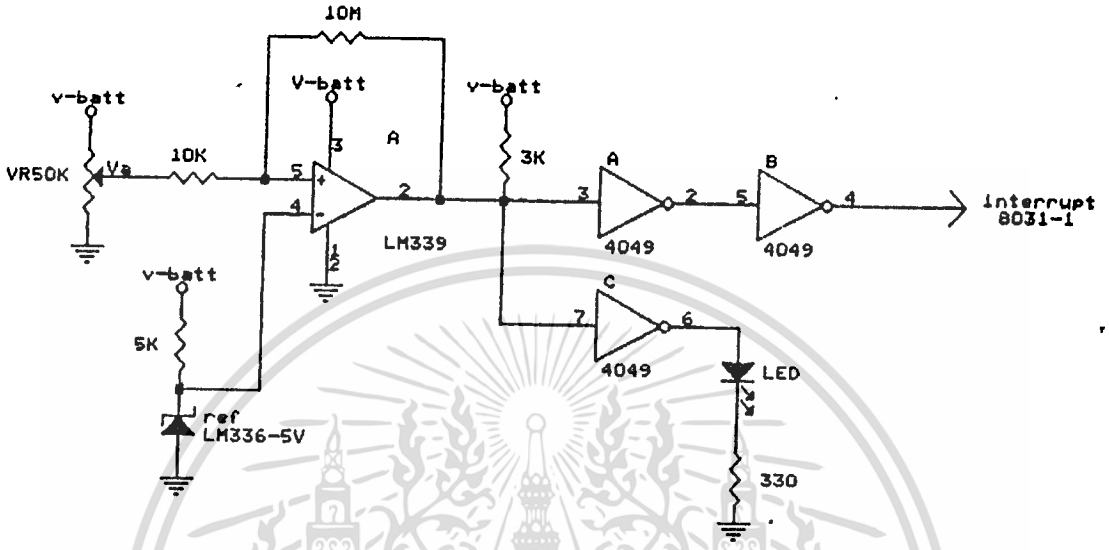
รูปที่ 3.24 วงจรตรวจจับวัตถุกีดขวางที่สมบูรณ์



รูปที่ 3.25 แผนผังเวลาแสดงการทำงานของวงจรตรวจจับวัตถุกีดขวาง

3.4.2 ระบบตรวจวัดระดับแรงดันแบตเตอรี่

วงจรตรวจจับระดับแรงดันแบตเตอรี่แสดงดังรูปที่ 3.26



รูปที่ 3.26 วงจรตรวจวัดแรงดันแบตเตอรี่

วงจรประกอบด้วย LM339 ทำหน้าที่เปรียบเทียบระดับแรงดัน Va กับระดับแรงดันอ้างอิง Vref แรงดัน Va นั้นจะแปรผันโดยตรงกับแรงดัน V-batt หาก V-batt มีค่าต่ำลงย่อมทำให้ Va มีค่าต่ำลงไปด้วย ส่วน Vref มีค่าคงที่ที่ 5 โวลท์เนื่องจาก reference diode เมื่อ Va มีค่าต่ำกว่า Vref วงจรจะส่งสัญญาณเอาท์พุทไปยังขาอินเตอรัพท์ของ 8031-1

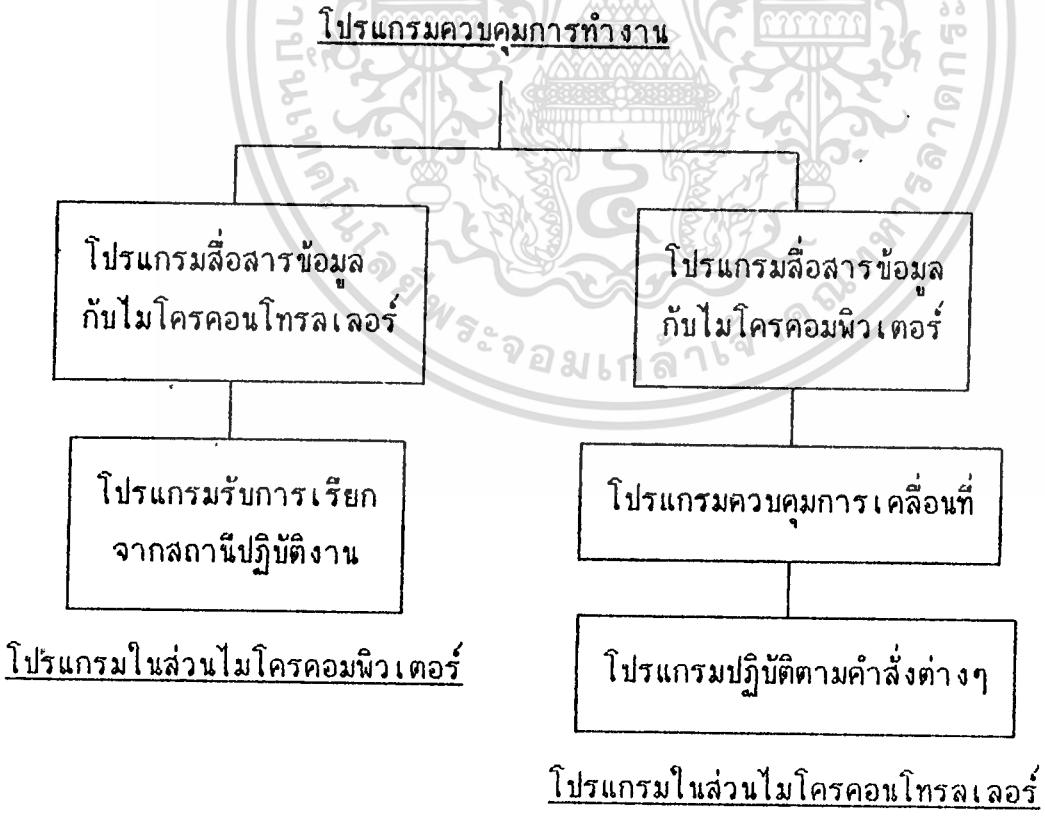
เพื่อให้วงจรทำงานทันทีที่แบตเตอรี่มีแรงดันต่ำกว่า 80 % ของแรงดันปกติ จึงตั้งค่า Va ไว้ที่ 6.25 โวลท์ แบตเตอรี่ขณะปกติมีแรงดัน 12 โวลท์ เมื่อแรงดันตกต่ำกว่า 80 % คือ 9.6 โวลท์ จะทำให้ Va ก็จะต่ำกว่า 5 โวลท์ มีผลให้วงจรส่งสัญญาณเอาท์พุทไปแสดงผลที่ LED และไปยังขาอินเตอรัพท์ของ 8031-1

บทที่ 4
โปรแกรมควบคุมการทำงาน

เป็นหัวใจสำคัญของระบบควบคุม AGV แบ่งเป็น 2 ส่วนใหญ่ๆได้แก่

4.1) โปรแกรมควบคุมการทำงานในส่วนไมโครคอมพิวเตอร์
ประกอบด้วยโปรแกรมที่สำคัญๆหลายโปรแกรมได้แก่
- 4.1.1) โปรแกรมสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์
- 4.1.2) โปรแกรมรับการเรียกจากสถานีปฏิบัติงาน

4.2) โปรแกรมควบคุมการทำงานในส่วนไมโครคอนโทรลเลอร์
ประกอบด้วยโปรแกรมที่สำคัญๆหลายโปรแกรมได้แก่
- 4.2.1) โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์
- 4.2.2) โปรแกรมควบคุมการเคลื่อนที่
- 4.2.3) โปรแกรมปฏิบัติตามคำสั่งต่างๆ



รูปที่ 4.1 บล็อคไดอะแกรมโปรแกรมควบคุมการทำงาน

4.1 โปรแกรมควบคุมการทำงานในส่วนไมโครคอมพิวเตอร์

การควบคุมการทำงานในส่วนไมโครคอมพิวเตอร์ ประกอบด้วยโปรแกรมต่างๆได้แก่

4.1.1) โปรแกรมสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์

ทำหน้าที่ส่งคำสั่งจากไมโครคอมพิวเตอร์ 286-AT ไปยังไมโครคอนโทรลเลอร์ 8031-1 เพื่อให้ทำงานตามต้องการ นอกจากนี้ยังทำหน้าที่คอยรับข้อมูลหรือข้อความต่างๆจากไมโครคอนโทรลเลอร์ 8031-1

การสื่อสารข้อมูลเป็นการส่งข้อมูลอนุกรมแบบอะซิงโครนัส อาร์ฟดูเพล็กซ์ ขนาด 10 บิท ประกอบด้วยบิทเริ่มต้น , บิทข้อมูล 8 บิท และบิทสิ้นสุด 1 บิท อัตราบอดเท่ากับ 150 บิท/วินาที

การโปรแกรมพอร์ทอนุกรม [13]

ทำการติดต่อกับพอร์ทอนุกรม COM1 โดยใช้โปรแกรมอินเตอร์รัพท์ 14h จาก ROM BIOS มาช่วย ภายในอินเตอร์รัพท์หมายเลข 14h มีฟังก์ชั่นให้เลือกได้ 4 ฟังก์ชั่น การกำหนดหมายเลขฟังก์ชั่นกระทำในรีจิสเตอร์ AH ส่วนรายละเอียดในฟังก์ชั่นจะกำหนดในรีจิสเตอร์ AL ฟังก์ชั่นต่างๆมีดังนี้

1) ฟังก์ชั่น 0 : กำหนดพารามิเตอร์สำหรับพอร์ทอนุกรม
ฟังก์ชั่นนี้จะตั้งค่าพารามิเตอร์ตามต้องการ ได้แก่
- อัตราบอด 110 , 150 , 300 , 600 , 1200 , 2400 , 4800 , 9600 บิท/วินาที
- บิทพาริตี้ ไม่มีพาริตี้ (none) , พาริตี้คู่ (even) , พาริตี้คี่ (odd)
- จำนวนบิทสิ้นสุด 1 บิท , 2 บิท
- จำนวนบิทข้อมูล 7 บิท , 8 บิท

2) ฟังก์ชั่น 1 : ส่งข้อมูลอักขระ 1 ตัวไปยังพอร์ทอนุกรม
ฟังก์ชั่นนี้จะส่งอักขระ 1 ตัวที่อยู่ในรีจิสเตอร์ AL ออกไปยังพอร์ทอนุกรม เมื่อส่งออกไปแล้ว รีจิสเตอร์ AH จะรายงานผลของการส่งให้ทราบ โดยถ้าบิทที่ 7 ของ AH = 0 แสดงว่าฟังก์ชั่นนี้ประสบผลสำเร็จ แต่ถ้าบิทที่ 7 ของ AH = 1 แสดงว่ามีความผิดพลาดเกิดขึ้นโดยบิทที่ 0 - 6 จะบอกชนิดของความผิดพลาด

3) ฟังก์ชั่น 2 : รับอักขระ 1 ตัวจากพอร์ทอนุกรม
ฟังก์ชั่นนี้จะรับอักขระ 1 ตัวจากพอร์ทอนุกรมเก็บในรีจิสเตอร์ AL โดยจะแสดงผลการรับในรีจิสเตอร์ AH โดยถ้าบิทที่ 7 ของ AH = 0 แสดงว่าฟังก์ชั่นประสบผลสำเร็จ แต่ถ้าบิทที่ 7 ของ AH = 1 แสดงว่ามีความผิดพลาดในการรับเกิดขึ้น โดยบิทที่ 0 - 6 จะบอกชนิดความผิดพลาดนั้น

4) ฟังก์ชั่น 3 : แสดงสถานภาพปัจจุบันของพอร์ทอนุกรม

ฟังก์ชั่นนี้จะรายงานสถานภาพปัจจุบันของพอร์ทอนุกรมในรีจิสเตอร์ AH และ AL โดยใน AH จะรายงานสถานภาพของพอร์ท ส่วน AL จะรายงานสถานภาพของโมเดม

สำหรับคำสั่งต่างๆที่ส่งไปยังไมโครคอนโทรลเลอร์ 8031-1 นั้นมีดังต่อไปนี้

1) คำสั่ง set\...,...\...,...\...,...\...

เป็นคำสั่งสำหรับส่งข้อมูลการเคลื่อนที่ให้ 8031-1 สำหรับข้อมูลการเคลื่อนนั้นประกอบด้วย

ก) ค่าตั้งทางตำแหน่ง (position command)

ใช้ตัวย่อคือ P หรือ p เป็นการกำหนดระยะเชิงมุมที่ต้องการให้มอเตอร์หมุนไป ค่าตั้งทางตำแหน่งนี้มีหน่วยเป็นจำนวนพัลซ์ หรืออาจเทียบหาระยะทางการเคลื่อนที่ของ AGV จริงได้จากสูตร

จำนวนพัลซ์ n พัลซ์ จะเท่ากับระยะทาง 2 π n R3 R1/(450 R2) เซ็นติเมตร

โดยที่

R1 = รัศมีของล้อเอนโคดเดอร์ หน่วยเป็นเซ็นติเมตร

R2 = รัศมีเพลาล้อขับ หน่วยเป็นเซ็นติเมตร

R3 = รัศมีล้อขับ หน่วยเป็นเซ็นติเมตร

จำนวนพัลซ์ทางตำแหน่งสามารถกำหนดได้ตั้งแต่ 0 ถึง 65,535 พัลซ์ นอกจากนี้ต้องระบุทิศทางการหมุนของมอเตอร์ด้วย โดยในที่นี้กำหนดให้

0 = มอเตอร์หมุนในทิศตามเข็มนาฬิกา (clockwise)

1 = มอเตอร์หมุนในทิศทวนเข็มนาฬิกา (counter clockwise)

ตัวอย่าง \p=10000,0\

ข้อมูลนี้เป็นการกำหนดให้มอเตอร์หมุนตามเข็มนาฬิกาไปเป็นระยะเชิงมุมเทียบเท่าจำนวนพัลซ์ทางตำแหน่ง 10,000 พัลซ์ หรือเทียบได้กับการเคลื่อนที่ของ AGV เป็นระยะทาง 2 π 10000 R3 R1/ (450 R2) เซ็นติเมตร

ข) ทิศทางการเลี้ยว (turn command)

ใช้ตัวย่อคือ T หรือ t เป็นการกำหนดทิศทางการหมุนของแขนเซอร์โว ซึ่งมี 2 ทิศทางคือ

- กำหนดให้แขนเซอร์โวหมุนไปทางซ้าย (turn left) ใช้ตัวย่อว่า L หรือ l
- กำหนดให้แขนเซอร์โวหมุนไปทางขวา (turn right) ใช้ตัวย่อว่า R หรือ r

ตัวอย่าง \t=l\ เป็นการกำหนดให้แขนเซอรโวหมุนไปทางซ้าย

ตัวอย่าง การใช้คำสั่ง get\...,..\...,..\...,..\...

```
Transmit command:get\p=30000,0\t=r\p=10000,0\t=l\p=20000,0\
Receive message:\agv o.k
```

ตัวอย่างนี้เป็นการส่งข้อมูลเพื่อให้มอเตอร์และชุดเซอร์โวทำงานตามลำดับดังนี้

1) กำหนดให้มอเตอร์หมุนไปในทิศตามเข็มนาฬิกา 30,000 พัลซ์
2) กำหนดให้แขนเซอร์โวหมุนไปทางขวา*
3) กำหนดให้มอเตอร์หมุนไปในทิศตามเข็มนาฬิกา 10,000 พัลซ์
4) กำหนดให้แขนเซอร์โวหมุนไปทางซ้าย*
5) กำหนดให้มอเตอร์หมุนไปในทิศตามเข็มนาฬิกา 20,000 พัลซ์

*หมายเหตุ รายละเอียดของวงจรควบคุมแขนเซอร์โวมีในภาคผนวก ข

2) คำสั่ง data

เป็นคำสั่งขอดูข้อมูลการเคลื่อนที่ที่ส่งไปจากการใช้คำสั่ง get\ ว่าถูกต้องหรือไม่

ตัวอย่าง การใช้คำสั่ง data

```
Transmit command:data
Receive message:\press 'y' to continue
```

ไมโครคอนโทรลเลอร์ 8031-1 จะแสดงแอดเดรสไบท์ต่ำ (lower byte) ซึ่งเก็บข้อมูลการเคลื่อนที่ซึ่งอยู่ในคำสั่ง set\ พร้อมข้อมูล โดยแสดงออกทางพอร์ท B และพอร์ท C ตามลำดับ การขอดูข้อมูลในไบท์ถัดไปให้กด "y" ตามด้วยการกด <ENTER> หากไม่ต้องการดูอีกแล้วให้กด <ENTER>

3) คำสั่ง load

เป็นคำสั่งเคลื่อนย้ายข้อมูลจากหน่วยความจำภายนอกของไมโครคอนโทรลเลอร์ 8031-1 ไปเก็บยังหน่วยความจำภายนอกของไมโครคอนโทรลเลอร์ 8031-2

ตัวอย่าง การใช้คำสั่ง load

```
Transmit command:load
 Receive message:\load complete
```

หากการเคลื่อนย้ายข้อมูลเป็นไปโดยสมบูรณ์ 8031-1 จะตอบกลับมาว่า "\load complete"

4) คำสั่ง move

เป็นคำสั่งให้มอเตอร์เริ่มหมุนตามข้อมูลที่มี

ตัวอย่าง การใช้คำสั่ง move

```
Transmit command:move
 Receive message:\moving complete
```

เมื่อมอเตอร์หมุนตามโปรแกรมเรียบร้อยแล้วจะส่งข้อความกลับมาว่า "\moving complete"

5) คำสั่ง read

เป็นคำสั่งให้ไมโครคอนโทรลเลอร์ 8031-1 อ่านสถานะปัจจุบันที่สำคัญๆ ของระบบจากพอร์ท 1 แล้วส่งกลับมายังคอมพิวเตอร์ 286-AT

ตัวอย่าง การใช้คำสั่ง read

```
Transmit command:read
Receive message:\target=not reach\obstacle=no\battery=normal
                \move=doing\cross-way=no\
```

จอภาพจะแสดงตัวอย่างของสถานะปัจจุบันที่เกิดขึ้น

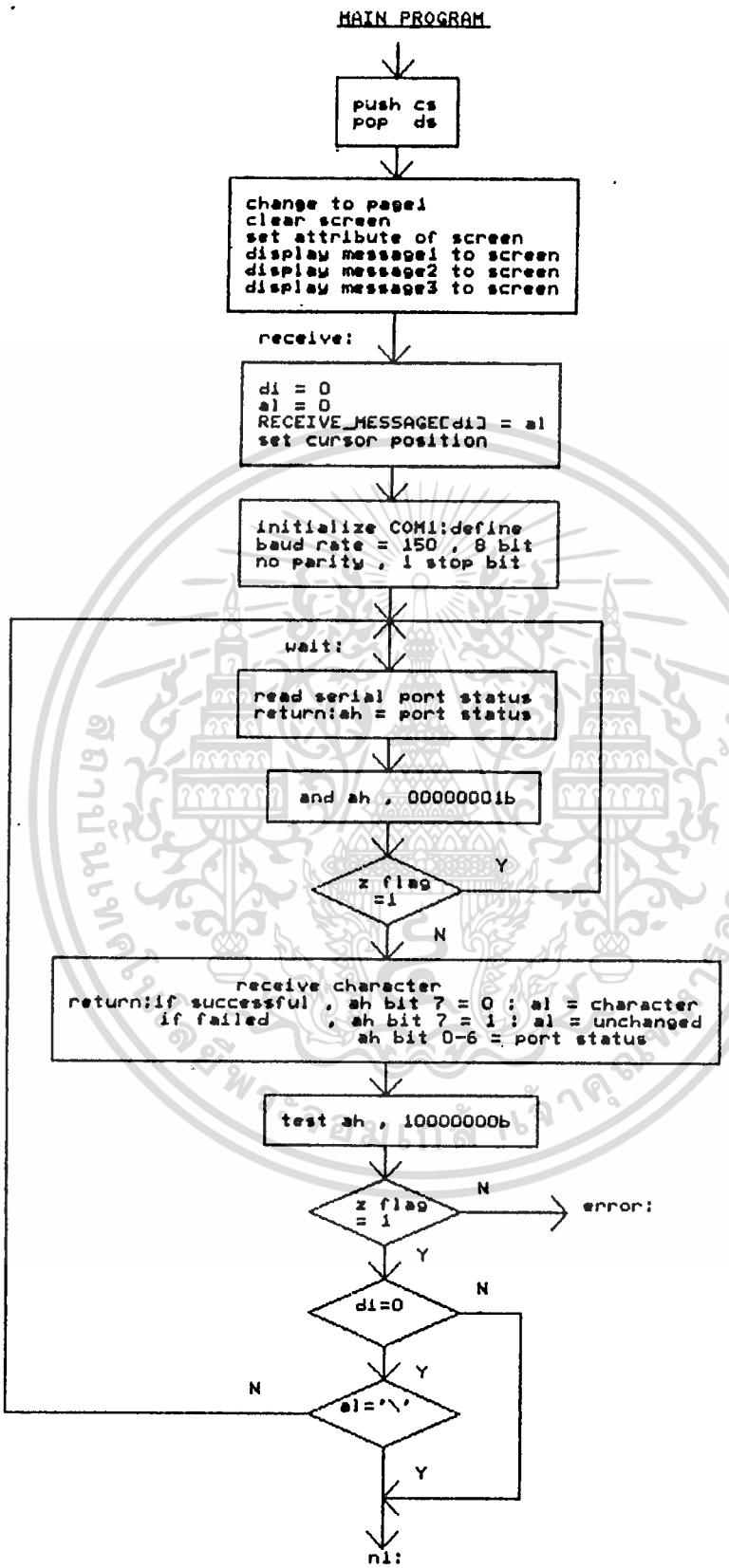
6) คำสั่ง test

เป็นคำสั่งทดสอบการสื่อสารข้อมูลอนุกรม โดยการป้อนข้อความใดๆก็ได้จากคีย์บอร์ด ข้อมูลที่ป้อนจะถูกส่งไปยังไมโครคอนโทรลเลอร์ 8031-1 แล้วถูกส่งกลับมายังคอมพิวเตอร์ 286-AT ในทันที หากข้อมูลที่ตอบกลับมาเหมือนเดิมทุกประการ ก็แสดงว่า การสื่อสารข้อมูลอนุกรมเป็นปกติ

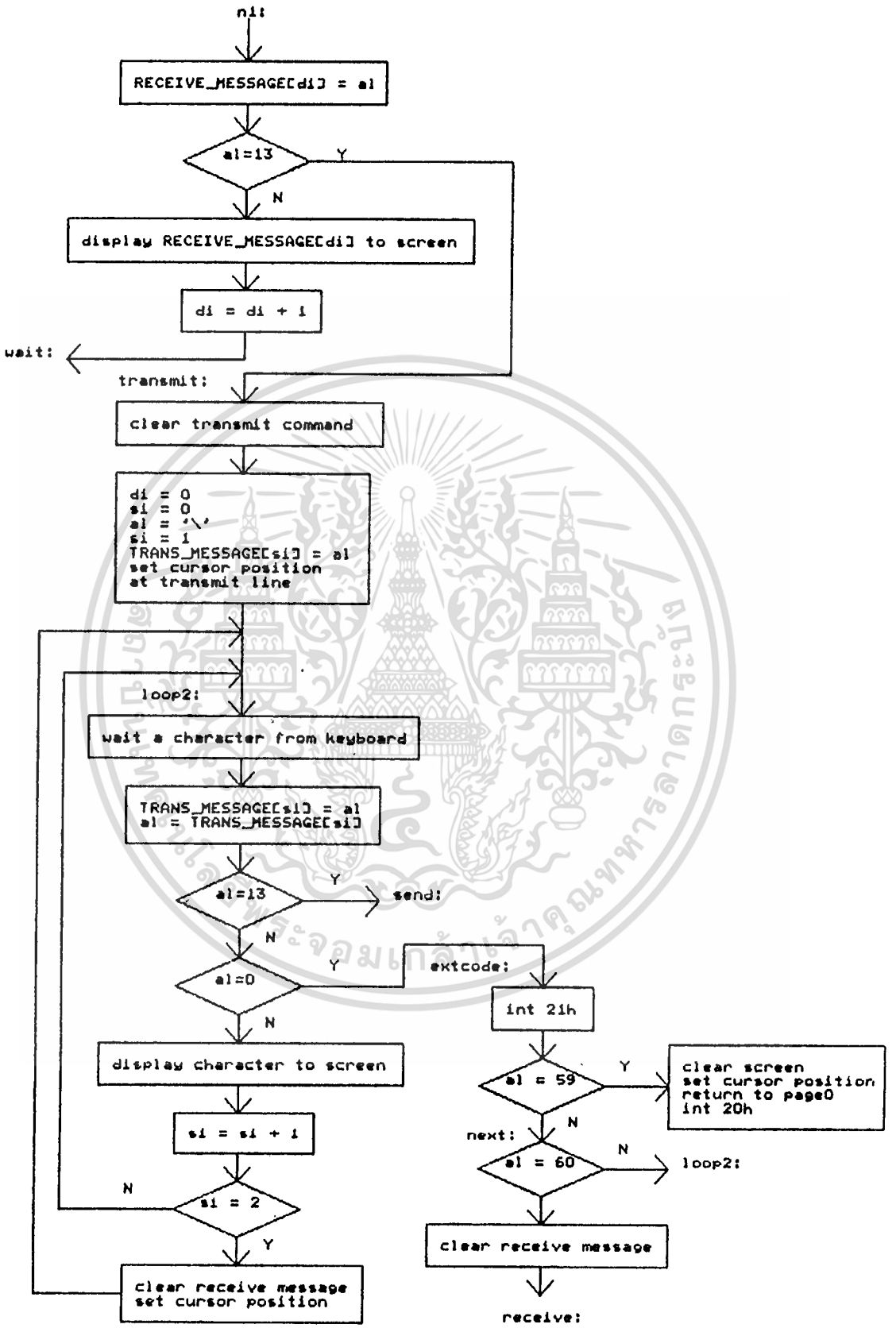
ตัวอย่าง การใช้คำสั่ง test

```
Transmit command:test\this command use to test communication
Receive message:\this command use to test communication
```

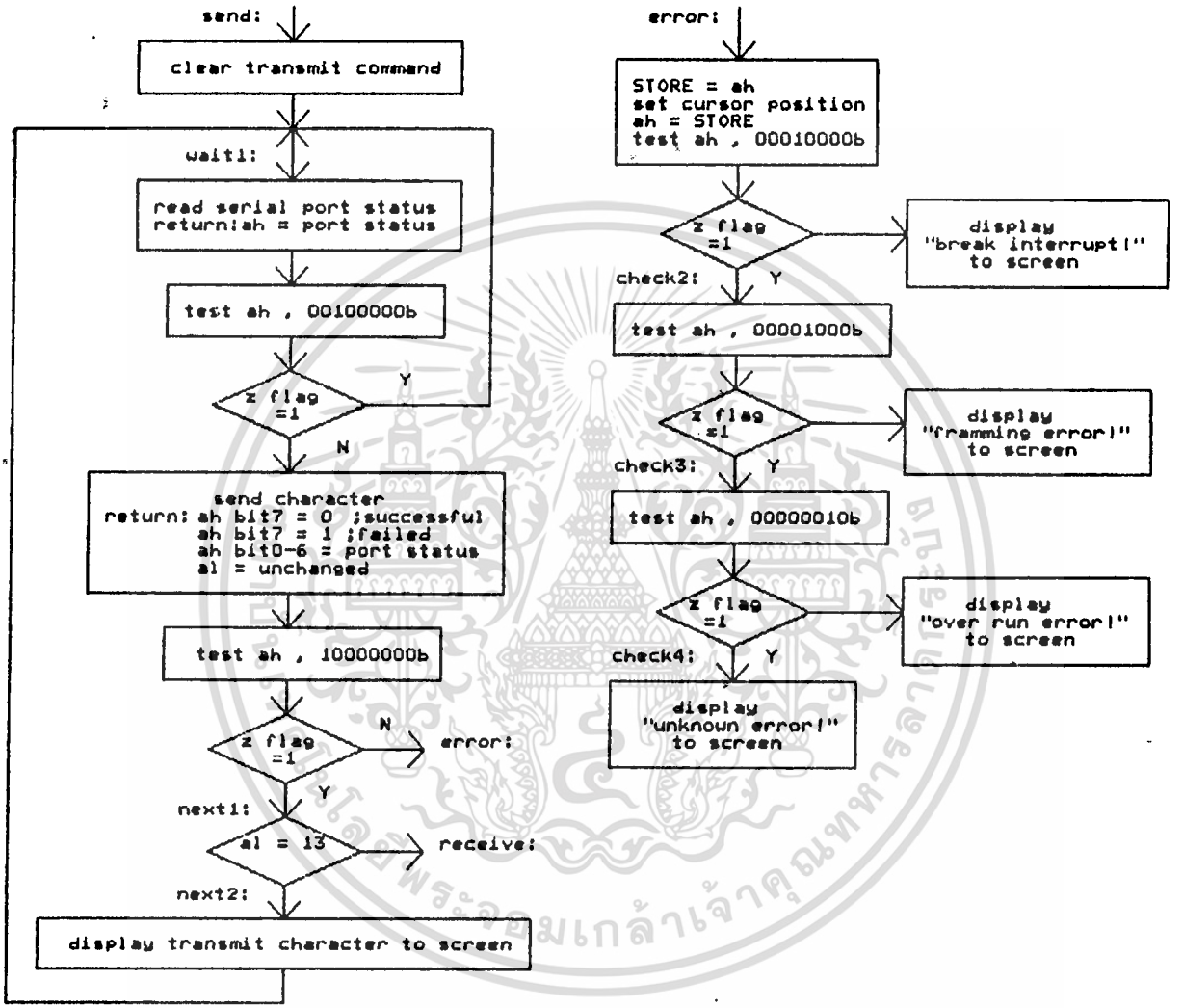
สำหรับแผนผังโปรแกรมการสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์ 8031-1 มีรายละเอียดดังรูปที่ 4.2



รูปที่ 4.2 แผนผังโปรแกรมสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์ 8031-1



รูปที่ 4.2 แผนผังโปรแกรมสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์ 8031-1 (ต่อ)



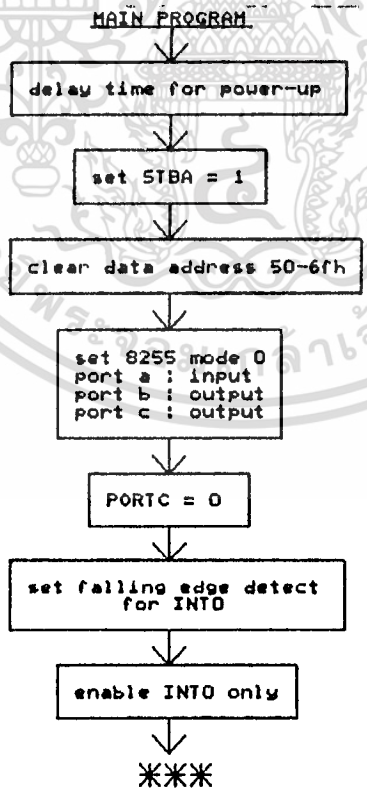
รูปที่ 4.2 แผนผังโปรแกรมสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์ 8031-1 (ต่อ)

4.1.2) โปรแกรมรับการเรียกจากสถานีปฏิบัติงาน

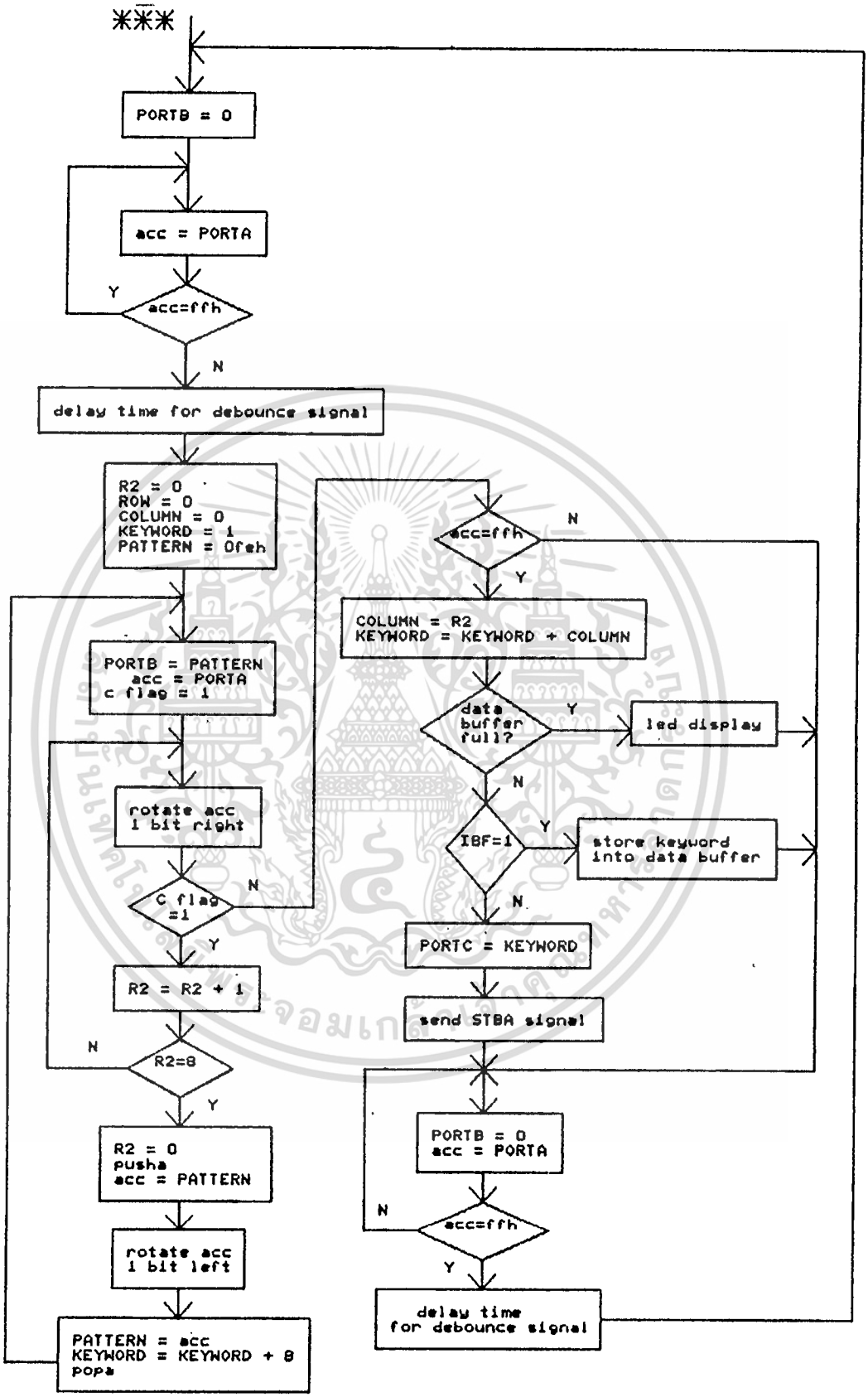
จากหัวข้อ 3.2) เป็นรายละเอียดทางด้านฮาร์ดแวร์สำหรับการเรียกใช้งานจากสถานีปฏิบัติงาน สำหรับโปรแกรมควบคุมการทำงานในส่วนนี้มี 2 โปรแกรมคือ

4.1.2.1) โปรแกรมรับการกดคีย์จากสถานีปฏิบัติงาน

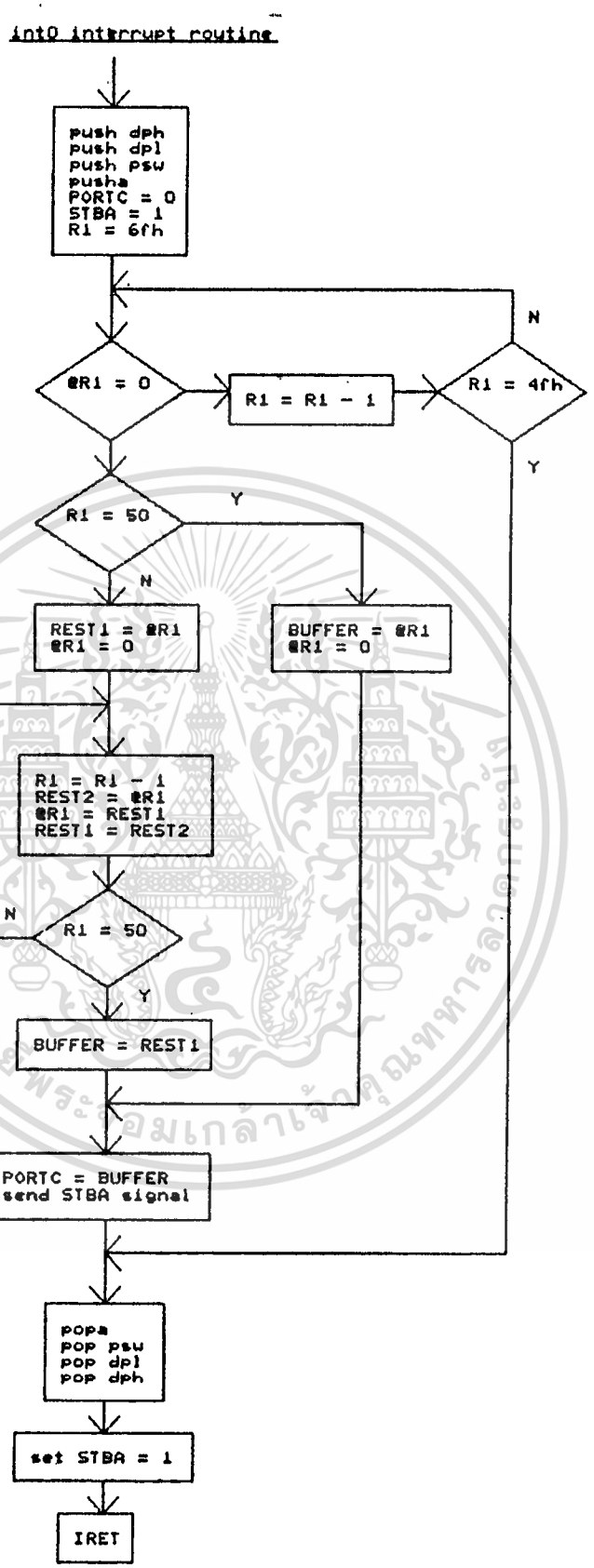
โปรแกรมนี้เขียนด้วยแอสเซมบลี้ 8031 เป็นโปรแกรมควบคุมการทำงานบน 8031-A ซึ่งเชื่อมต่ออยู่กับไมโครคอมพิวเตอร์ 286-AT ดังรูปที่ 3.16 ให้ทำหน้าที่คอยสแกนหาการกดคีย์เรียกจากสถานีปฏิบัติงาน (workstation) ทันทีที่พบว่ามีการกดคีย์ โปรแกรมจะทำการถอดรหัสของสถานีนั้น จากนั้น 8031-A จะติดต่อไปยัง 286-AT ในลักษณะ handshaking โดยเริ่มจากการส่งสัญญาณสโตรป STBA ไปบอก 286-AT ให้มาอ่านรหัสสถานีนั้นไป แต่ถ้าขณะนั้น 286-AT ไม่ว่างเช่น กำลังทำอินเตอรัพท์ที่มีลำดับความสำคัญสูงกว่าอยู่ 8031-A ก็จะเก็บรหัสสถานีนั้นไว้ในหน่วยความจำภายใน (internal memory) เป็นการชั่วคราว และจะส่งให้ 286-AT ทันทีที่การอินเตอรัพท์ที่มีลำดับความสำคัญนั้นสิ้นสุดลง การเก็บและการดึงรหัสสถานีจากหน่วยความจำนี้จะกระทำในลักษณะที่เรียกว่า เข้าก่อน-ออกก่อน (first in - first out) รายละเอียดเกี่ยวกับการทำ handshaking ได้กล่าวมาแล้วในหัวข้อ 3.2) ส่วนแผนผังโปรแกรมนี้แสดงไว้ตามรูปที่ 4.3



รูปที่ 4.3 แผนผังโปรแกรมรับการกดคีย์จากสถานีปฏิบัติงาน



รูปที่ 4.3 แผนผังโปรแกรมรับการกดคีย์จากสถานีปฏิบัติงาน (ต่อ)



รูปที่ 4.3 แผนผังโปรแกรมรับการกดคีย์จากสถานีปฏิบัติงาน (ต่อ)

4.1.2.2) โปรแกรมนำรหัสสถานีจาก 8031-A เก็บยังหน่วยความจำและย้ายลงสแต็ค

เป็นโปรแกรมแบบฝังตัวในหน่วยความจำ (resident file) [14] โปรแกรมนี้เขียนด้วยแอสเซมบลี้ 80286 ประกอบด้วยโปรแกรมอินเตอรัพท์ (interrupt routine) 2 โปรแกรมคือ

ก) โปรแกรมนำรหัสสถานีเก็บยังหน่วยความจำ

เป็นโปรแกรมอินเตอรัพท์หมายเลข 0fh ที่ตอบสนองต่อสัญญาณอินเตอรัพท์ภายนอก กล่าวคือเมื่อมีสัญญาณอินเตอรัพท์จากภายนอกเข้ามาที่ขา IRQ7 ของ 8259 โปรแกรมเคาน์เตอร์จะกระโดดไปยังแอดเดรสของอินเตอรัพท์ 0fh โปรแกรมจะทำการอ่านรหัสสถานีจากพอร์ท A ของ 8255-A เข้าไปเก็บไว้ในตัวแปรอาเรย์ (array variable) พร้อมกันนั้นก็จะแสดงข้อความปรากฏบนจอภาพเพื่อบอกให้ผู้ใช้ทราบว่ามีการเรียกใช้ AGV จากสถานีใด จากนั้นเครื่องจะกลับคืนสู่ dos

หากมีสัญญาณอินเตอรัพท์จากภายนอกเข้ามาอีก 286-AT ก็จะอ่านข้อมูลเก็บในตัวแปรอาเรย์เดิมสะสมไปเรื่อยๆ กรณีที่ตัวแปรเก็บข้อมูลจนเต็มจะมีข้อความปรากฏบนจอภาพบอกให้ทราบว่าข้อมูลเต็มแล้ว สำหรับการดึงข้อมูลออกไปใช้นั้นจะกระทำทีละ 1 ไบท์ โดยใช้คำสั่ง int 6fh

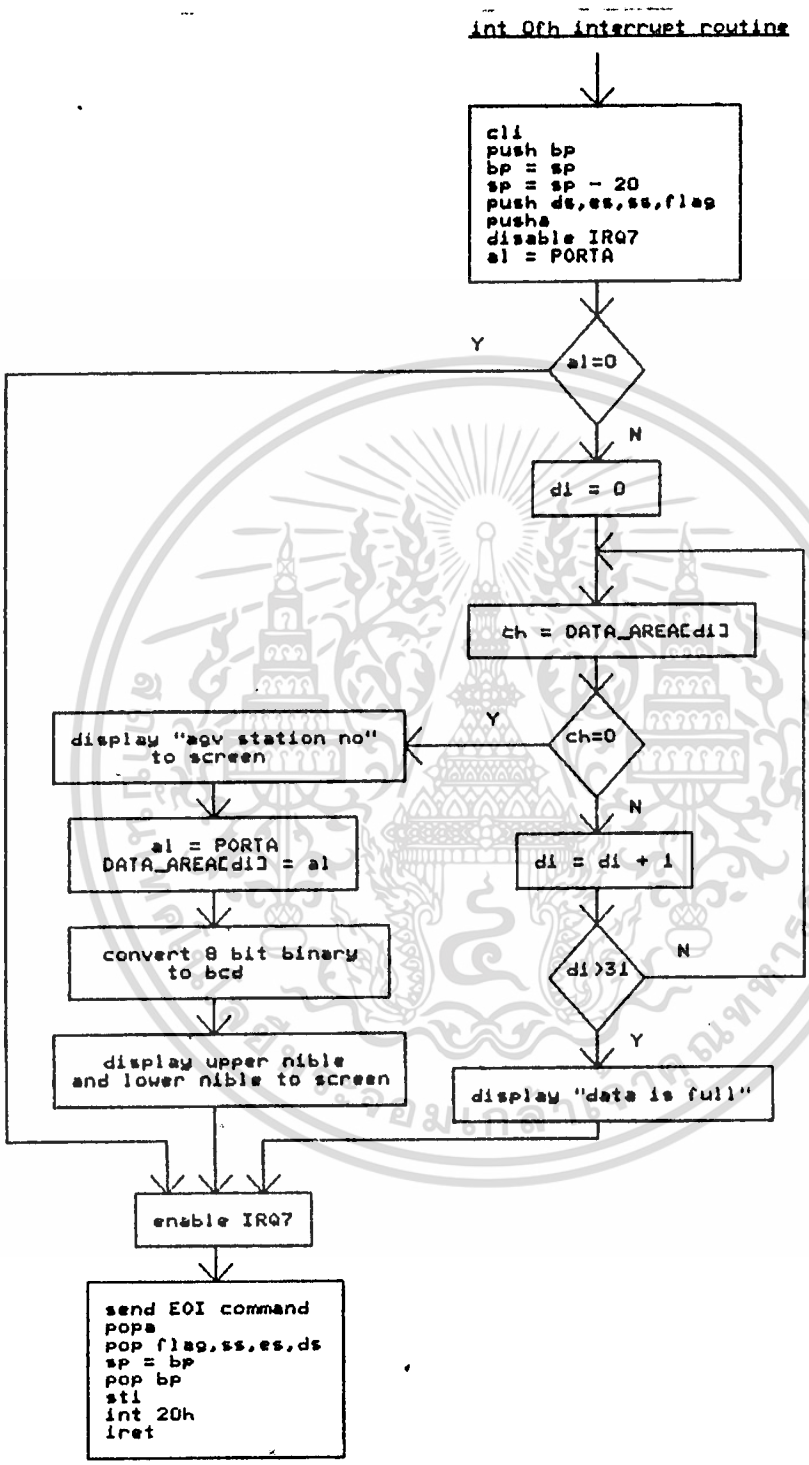
การเก็บข้อมูลและการดึงข้อมูลจากตัวแปรอาเรย์นี้จะเป็นแบบเข้าก่อน-ออกก่อน (first in-first out) แผนผังการทำงานของโปรแกรมแสดงไว้ในรูปที่ 4.4

ข) โปรแกรมย้ายรหัสสถานีไปเก็บยังสแต็ค

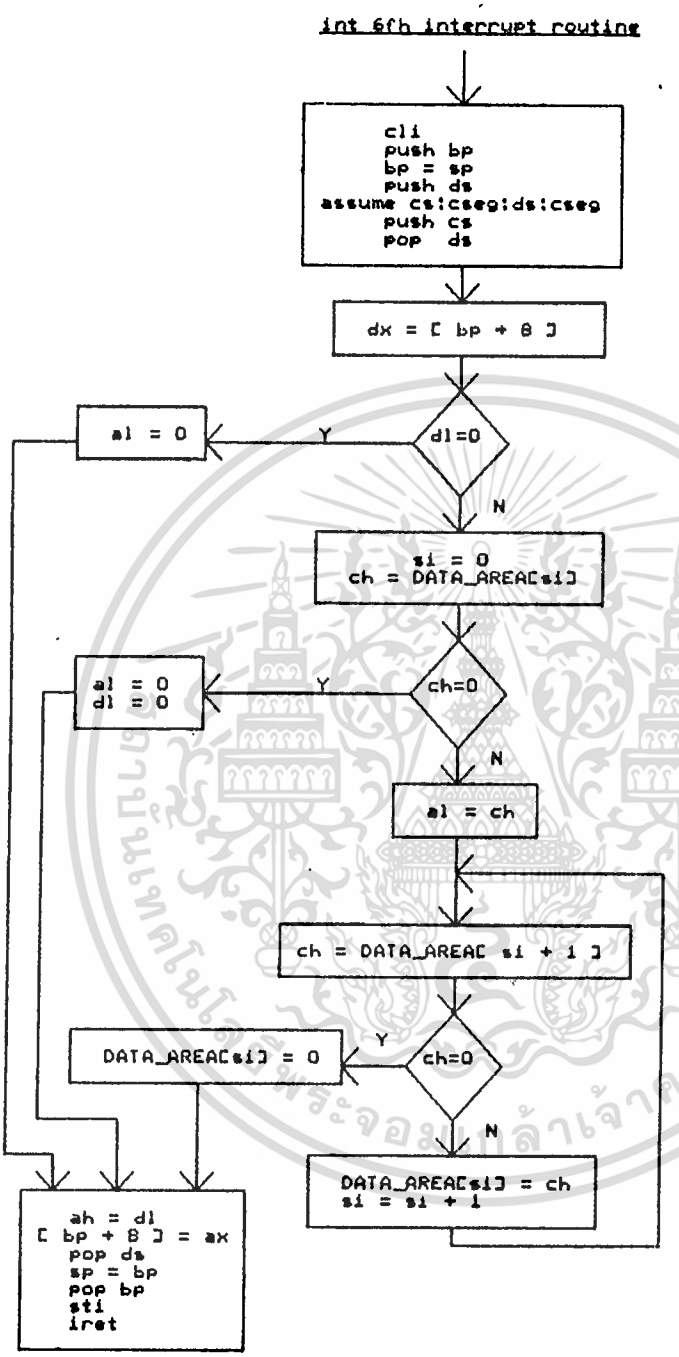
เป็นโปรแกรมอินเตอรัพท์หมายเลข 6fh ที่ตอบสนองต่อการเรียกใช้คำสั่ง int 6fh โปรแกรมอินเตอรัพท์นี้จะทำการย้ายข้อมูลรหัสสถานีขนาด 1 ไบท์ซึ่งเก็บอยู่ในตัวแปรอาเรย์ไปเก็บไว้ในสแต็ค (stack) สำหรับแอดเดรสของสแตคจะใช้ BP เป็นตัวชี้แอดเดรสแทน SP

ประโยชน์ของอินเตอรัพท์นี้ก็คือ ทำให้เราสามารถนำข้อมูลรหัสสถานีที่เก็บในตัวแปรอาเรย์ซึ่งเป็นตัวแปรในภาษาแอสเซมบลี้ไปประมวลผลในโปรแกรมภาษาระดับสูงเช่น Pascal หรือ C ได้เลยโดยไม่ต้องทำการ link ระหว่างโปรแกรมภาษาระดับสูงกับโปรแกรมภาษาแอสเซมบลี้ เพราะทุกครั้งเราจะดึงข้อมูลจากภาษาแอสเซมบลี้ เราสามารถใช้คำสั่ง int 6fh ได้โดยตรง

แผนผังการทำงานของโปรแกรมแสดงไว้ในรูปที่ 4.5



รูปที่ 4.4 แผนผังโปรแกรมนำรหัสสถานีเก็บยังหน่วยความจำ



รูปที่ 4.5 แผนผังโปรแกรมย้ายรหัสสถานีไปเก็บยังสแต็ค

4.2 โปรแกรมควบคุมการทำงานในส่วนไมโครคอนโทรลเลอร์

ในส่วนชุดไมโครคอนโทรลเลอร์สำหรับควบคุมการทำงานของ AGV นั้นจะประกอบด้วยไมโครคอนโทรลเลอร์ 8031 สองตัว ทำหน้าที่แตกต่างกันดังนี้

ก) ไมโครคอนโทรลเลอร์ 8031-1

ทำหน้าที่คอยรับคำสั่งจากไมโครคอมพิวเตอร์ แล้วประมวลผลตามคำสั่ง จากนั้นจะส่งข้อมูลหรือข้อความกลับไปยังไมโครคอมพิวเตอร์ โปรแกรมควบคุมการทำงานประกอบด้วยโปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์ และโปรแกรมปฏิบัติตามคำสั่งต่างๆได้แก่ คำสั่ง get\ , data , load , move , read และ test

ข) ไมโครคอนโทรลเลอร์ 8031-2

ทำหน้าที่ควบคุมตำแหน่งเชิงมุมในการหมุนของมอเตอร์กระแสตรงและควบคุมการทำงานของแขนเซอร์โว โดยจะปฏิบัติตามข้อมูลซึ่งส่งมาจากไมโครคอนโทรลเลอร์ 8031-1 ข้อมูลในการเคลื่อนที่ส่งมาจาก 8031-1 ประกอบด้วย ระยะเชิงมุมสำหรับการหมุนของมอเตอร์ , ทิศทางการหมุน และทิศทางการหมุนของแขนเซอร์โว สำหรับการติดต่อกันระหว่างไมโครคอนโทรลเลอร์ 8031-1 กับไมโครคอนโทรลเลอร์ 8031-2 จะใช้วิธี handshaking ดังได้กล่าวไปแล้วในบทที่ 3

4.2.1 โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์

ทำหน้าที่คอยรับ-ส่งข้อมูลระหว่างไมโครคอนโทรลเลอร์ 8031-1 กับไมโครคอมพิวเตอร์ การทำงานของโปรแกรมจะเริ่มต้นด้วยการส่งข้อมูลผ่านพอร์ทอนุกรม SBUF ไปยังไมโครคอมพิวเตอร์ เพื่อบอกให้ทราบว่า 8031-1 พร้อมที่รับจะคำสั่งแล้ว โดยโปรแกรมจะคอยอยู่จนกว่าจะมีการส่งคำสั่งจากไมโครคอมพิวเตอร์ไปให้ เมื่อไมโครคอมพิวเตอร์ส่งคำสั่งมาแล้ว 8031-1 จะทำการแปลคำสั่งแล้วปฏิบัติตามคำสั่งนั้น จากนั้นจะส่งข้อความตอบรับกลับไปยังไมโครคอมพิวเตอร์อีกครั้งเพื่อแสดงว่าการทำงานสิ้นสุด คำสั่งหรือข้อความที่ใช้ในการรับ-ส่งจะอยู่ในรูปข้อมูลอนุกรมขนาด 10 บิท ไม่มีบิทพาริตี้ อัตราบอดเท่ากับ 150 บิท/วินาที สำหรับแผนผังโปรแกรมนี้แสดงไว้ในรูปที่ 4.6

เทคนิคการรับส่งข้อมูลอนุกรม

ในการรับส่งข้อมูลอนุกรมตามปกติมี 2 วิธี ได้แก่ วิธีโพลลิ่ง (polling) และวิธีอินเตอรัพท์ (interrupt) สำหรับวิทยานิพนธ์นี้จะใช้วิธีโพลลิ่งเนื่องจากเป็นวิธีที่ง่ายและตรงไปตรงมา โดย 8031-1 จะทำหน้าที่รับหรือส่งข้อมูลอย่างเดียวจนกว่าจะเสร็จ อย่างไรก็ตามวิธีนี้มีข้อเสียคือทำให้ซีพียูต้องเสียเวลามาคอยรับส่งข้อมูลอย่างเดียวทำให้ไม่มีเวลาไปทำงานอื่นเลย ผิดกับวิธีอินเตอรัพท์ซึ่งซีพียูไม่ต้องเสียเวลามาคอยรับหรือส่งข้อมูล ซีพียูสามารถใช้เวลาขณะคอยข้อมูลไปทำงานอื่นได้ วิธีอินเตอรัพท์เป็นการเพิ่มประสิทธิภาพในการทำงานให้ซีพียู

การโปรแกรมพอร์ทอนุกรมเพื่อการรับส่งข้อมูล [12]

สำหรับรายละเอียดของการโปรแกรมพอร์ทอนุกรมสำหรับซีพียู 8031-1 ประกอบด้วย

1) <u>การตั้งค่าพารามิเตอร์สำหรับพอร์ทอนุกรม</u> มีรายละเอียดดังนี้

ก) ตั้งค่าในรีจิสเตอร์ TMOD — กำหนดให้ไทม์เมอร์ 1 ทำงานในโหมด 2 และกำหนดให้บิท GATE=0 เพื่อให้ไทม์เมอร์ถูกควบคุมด้วยบิท TR1 เท่านั้น

ข) ตั้งค่าในรีจิสเตอร์ TCON — โดยตั้งค่าบิท TR1=1 เพื่ออินาเบิ้ลให้ไทม์เมอร์ 1 เริ่มทำงานและตั้งค่าบิท IT0 และบิท IT1=1 เพื่อให้อินเตอรัพท์ INT0 และ INT1 ตามลำดับ ทำงานทันทีเมื่อมีสัญญาณขอบขาลงเข้ามา

ค) ตั้งค่าในรีจิสเตอร์ TH1 — กำหนดอัตราบอด = 150 บิท/วินาที

ง) ตั้งค่าในรีจิสเตอร์ SCON — กำหนดให้พอร์ทอนุกรมทำงานในโหมด 1 ซึ่งโหมดนี้เป็นการรับส่งข้อมูลขนาด 10 บิท ประกอบด้วยบิทเริ่มต้น 1 บิท , บิทข้อมูล 8 บิทและบิทสิ้นสุด 1 บิท ไม่มีบิทพาราตี้

2) <u>การส่งข้อมูล</u>

การส่งข้อมูลอนุกรมจะเริ่มขึ้นทันทีที่มีการเขียนข้อมูลขนาด 1 ไบท์ ลงในรีจิสเตอร์บัฟเฟอร์ SBUF รีจิสเตอร์บัฟเฟอร์ SBUF จะทำการแปลงข้อมูลขนานเป็นข้อมูลอนุกรมแล้วส่งออกไปยังขา TXD เมื่อข้อมูลถูกส่งออกไปหมดแล้ว SBUF จะว่างลงทำให้บิท TI ในรีจิสเตอร์ SCON ถูกเซ็ทเป็น 1 โดยอัตโนมัติ เมื่อ SBUF ว่างเราก็สามารถส่งข้อมูลไบท์ต่อไปได้ ขั้นตอนการส่งข้อมูลอนุกรมมีดังนี้

ก) อีนาเบิ้ลการอินเตอรัพท์พอร์ทอนุกรมโดยตั้งค่าบิท ES=1 ในรีจิสเตอร์ IE

ข) เริ่มต้นการส่งข้อมูลโดยตั้งค่าบิท TI=1 ในรีจิสเตอร์ SCON ทำให้เกิดการอินเตอรัพท์ขึ้น

ค) ซีพียูกระโดดไปทำงานที่แอดเดรส 0023h ซึ่งเป็นแอดเดรสของโปรแกรมอินเตอรัพท์พอร์ทอนุกรม ภายในโปรแกรมอินเตอรัพท์จะเริ่มด้วยการเคลียร์บิท TI=0 จากนั้นจึงเขียนข้อมูลที่ต้องการส่งลงในรีจิสเตอร์บัฟเฟอร์ SBUF จากนั้นซีพียูจะกลับไปทำงานที่โปรแกรมหลัก

ง) ตรวจสอบข้อมูลที่จะส่งว่าเป็นข้อมูลสุดท้ายหรือไม่ หากไม่ใช่ให้ทำข้อ จ) ต่อไป แต่ถ้าเป็นข้อมูลสุดท้ายก็ให้ทำการดิสเอเบิ้ลการอินเตอรัพท์พอร์ทอนุกรม โดยให้บิท ES=0 จากนั้นให้รอจนกว่าบิท TI=1 อีกครั้ง เมื่อ TI=1 แล้วให้ทำการเคลียร์ TI=0

จ) เมื่อข้อมูลใน SBUF ถูกส่งออกไปหมดแล้ว บิท TI จะเป็น 1 โดยอัตโนมัติ ทำให้เกิดการอินเตอรัพท์อีกครั้ง ซีพียูกลับไปทำข้อ ค) ใหม่

3) <u>การรับข้อมูล</u>

ก่อนที่จะทำการรับข้อมูลอนุกรมนั้น เราต้องอีนาเบิ้ลการรับโดยการตั้งค่าบิท REN ในรีจิสเตอร์ SCON เท่ากับ 1 ทั้งนี้เป็นการป้องกันมิให้ SBUF รับข้อมูลที่ไม่ต้องการจากขา RXD เข้ามา เมื่อ SBUF รับข้อมูลเข้ามาครบ 8 บิทแล้ว บิท RI ใน SCON จะเท่ากับ 1 อันเป็นการแสดงว่ามีข้อมูล 1 ไบท์ใน SBUF ที่พร้อมจะให้เราอ่านเข้าไปเก็บในหน่วยความจำ สำหรับขั้นตอนการรับข้อมูลมีดังนี้

ก อีนาเบิ้ลการรับโดยตั้งค่าบิท REN=1 ในรีจิสเตอร์ SCON

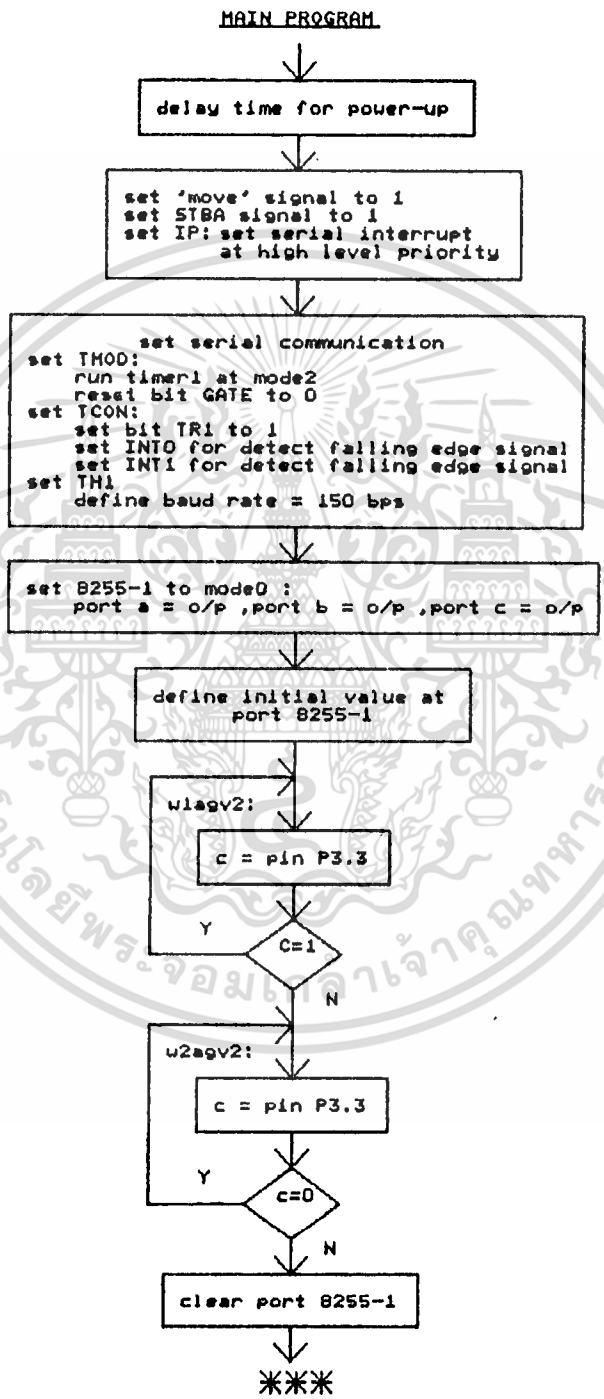
ข) อีนาเบิ้ลการอินเตอรัพท์พอร์ทอนุกรมโดยตั้งค่าบิท ES=1 ในรีจิสเตอร์ IE

ค) เมื่อ SBUF รับข้อมูลจนเต็ม บิท RI ในรีจิสเตอร์ SCON จะเท่ากับ 1 เกิดการอินเตอรัพท์พอร์ทอนุกรมขึ้น

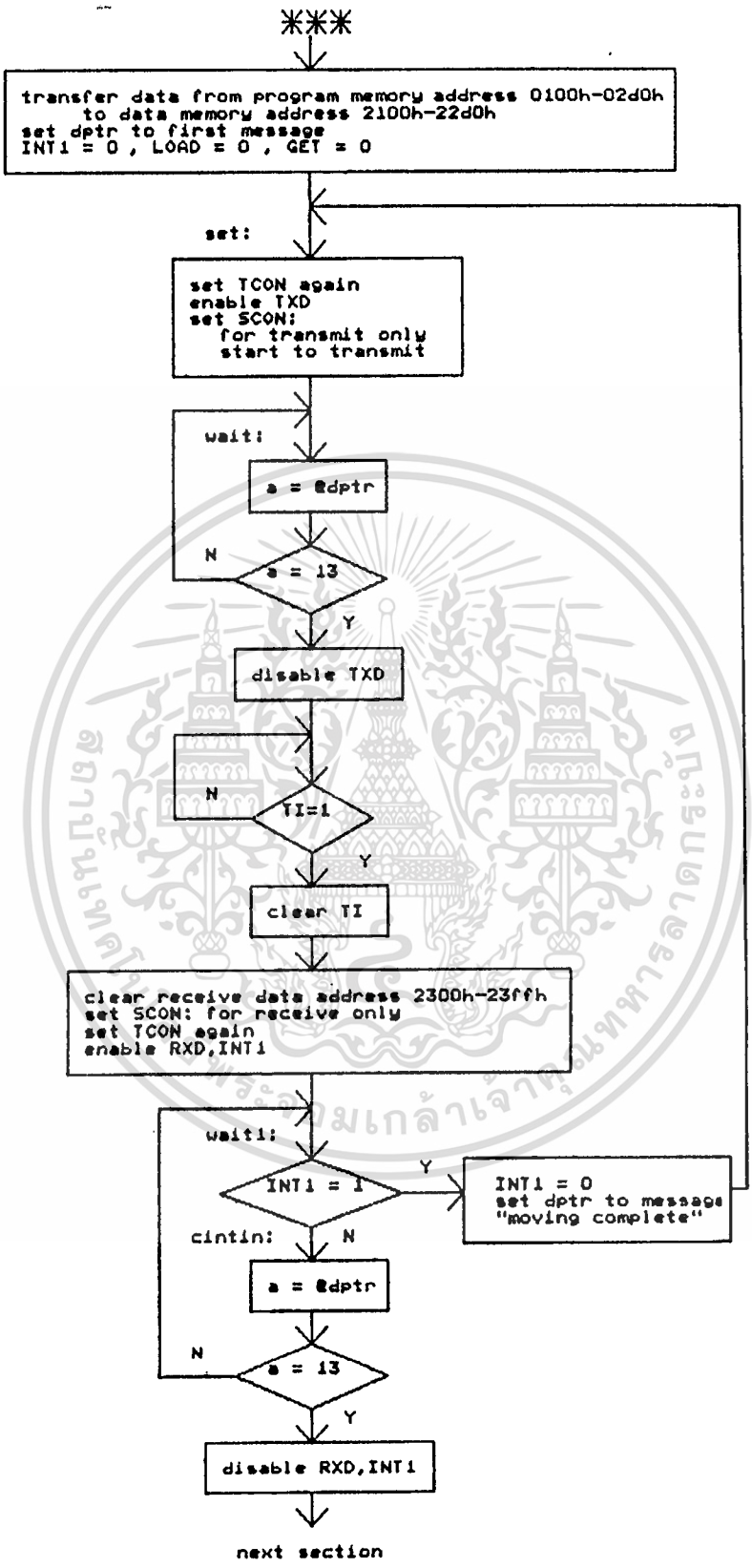
ง) ซีพียูกระโดดไปทำงานที่แอดเดรส 0023h ซึ่งเป็นแอดเดรสของโปรแกรมอินเตอรัพท์พอร์ทอนุกรม ภายในโปรแกรมอินเตอรัพท์จะเริ่มด้วยการเคลียร์บิท RI=0 แล้วทำการอ่านข้อมูลในรีจิสเตอร์บัฟเฟอร์ SBUF เข้ามาเก็บยังแอดเดรส 2300h-23xxh จากนั้นซีพียูจะกลับไปทำงานที่โปรแกรมหลัก

จ) ตรวจสอบข้อมูลที่รับมาว่าเป็นข้อมูลสุดท้ายหรือไม่ หากไม่ใช่ให้ทำข้อ ฉ) ต่อไป แต่ถ้าเป็นข้อมูลสุดท้ายก็ให้ทำการดิสเอเบิ้ลการอินเตอรัพท์พอร์ทอนุกรม โดยให้บิท ES=0

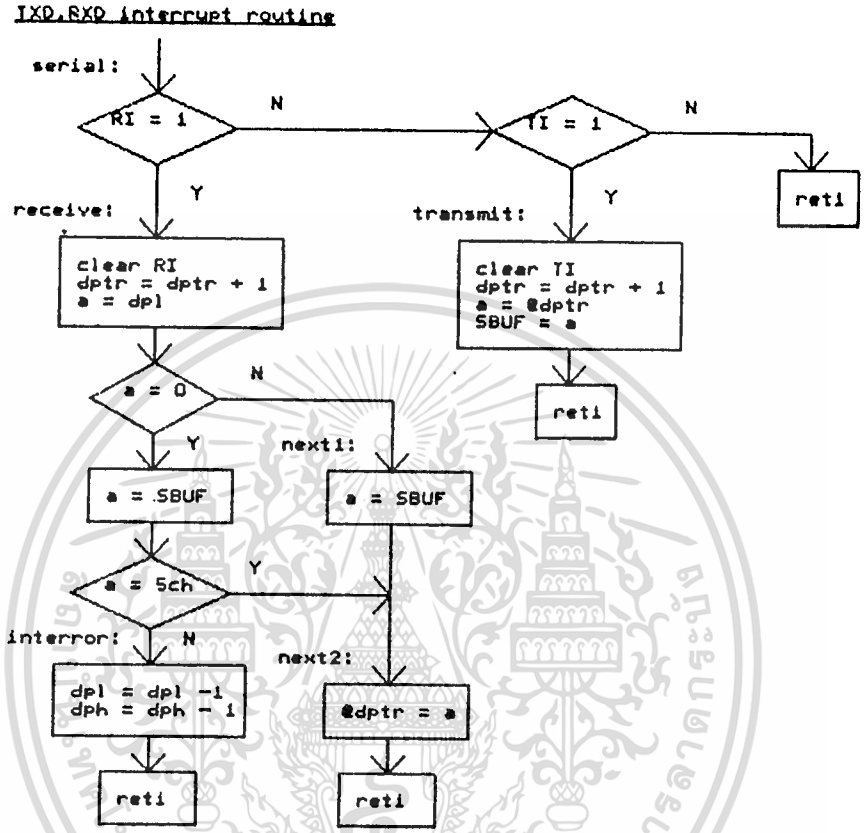
ฉ) เมื่อ SBUF รับข้อมูลจนเต็มอีกครั้ง บิท RI จะเป็น 1 อีก ทำให้เกิดการอินเตอรัพท์อีกครั้ง ซีพียูกลับไปทำข้อ ง) ใหม่



รูปที่ 4.6 แผนผังโปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์



รูปที่ 4.6 แผนผังโปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์ (ต่อ)



รูปที่ 4.6 แผนผังโปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์ (ต่อ)

4.2.2 โปรแกรมควบคุมการเคลื่อนที่

เป็นโปรแกรมควบคุมการทำงานบนซีพียู 8031-2 เพื่อควบคุมตำแหน่งของมอเตอร์กระแสตรงให้หยุดยังตำแหน่งที่ต้องการได้อย่างแม่นยำเที่ยงตรง ข้อมูลทางตำแหน่งสำหรับมอเตอร์รวมทั้งข้อมูลการหมุนของแขนเซอรโวจะถูกส่งมาจากซีพียู 8031-1 ด้วยวิธี handshaking โปรแกรมควบคุมนี้ประกอบด้วยโปรแกรมหลักและโปรแกรมอินเตอรัพท์ซึ่งมีรายละเอียดดังนี้

4.2.2.1 โปรแกรมหลัก (main program)

มีลำดับขั้นตอนการทำงานดังนี้

1) หลังทำการรีเซท โปรแกรมจะทำการตั้งค่าพารามิเตอร์ในรีจิสเตอร์ต่างๆได้แก่

- ตั้งค่าในรีจิสเตอร์ควบคุมใน 8255-2 ให้ทำงานในโหมด 1 แบบสโตรบอินพุท เพื่อให้ 8255-2 ทำหน้าที่ handshaking
- ตั้งค่าบิท INTEA ใน 8255-2 เท่ากับ 1
- ตั้งค่าในรีจิสเตอร์ IP โดยกำหนดให้อินเตอรัพท์ T0 มีลำดับความสำคัญสูงสุด
- ตั้งค่าในรีจิสเตอร์ TMOD โดยกำหนดให้เคาน์เตอร์ 0 และเคาน์เตอร์ 1 ทำงานในโหมด 2 และกำหนดให้บิท GATE เท่ากับ 0
- ตั้งค่าในรีจิสเตอร์ TCON โดยกำหนดให้บิท TR0 และบิท TR1 เท่ากับ 1 และกำหนดให้อินเตอรัพท์ INT0 ทำงานเมื่อมีสัญญาณขอบขาลง (falling edge) เข้ามา
- ตั้งค่าในรีจิสเตอร์ TL0 , TH0 , TL1 และ TH1 เท่ากับ 255

2) ซีพียู 8031-2 ส่งสัญญาณลอจิกศูนย์ไปยังขา P3.3 ของซีพียู 8031-1 เพื่อบอกให้ 8031-1 ทราบว่าขณะนี้พร้อมแล้วที่จะรับข้อมูล ข้อมูลใน 8031-1 นั้นเป็นข้อมูลการเคลื่อนที่ที่ได้รับจากการใช้คำสั่ง get\ บนไมโครคอมพิวเตอร์ 286-AT การส่งข้อมูลจาก 8031-1 มายัง 8031-2 นั้นจะใช้วิธี handshaking โดย 8031-2 จะรอสัญญาณอินเตอรัพท์ INTRA ที่ส่งมายังขา INT0 เมื่อมีการเรียกใช้อินเตอรัพท์ INT0 โปรแกรมอินเตอรัพท์จะทำการอ่านข้อมูลทางตำแหน่งเก็บไว้ในหน่วยความจำสำหรับรายละเอียดของโปรแกรมอินเตอร์รัพท์จะได้กล่าวต่อไป

3) ซีพียู 8031-2 รอรับสัญญาณ 'move' ที่ส่งจาก 8031-1 มายังขา P1.0 สัญญาณ 'move' เป็นสัญญาณลอจิกศูนย์ที่ส่งออกมาจาก 8031-1 หลังใช้คำสั่ง move จากไมโครคอมพิวเตอร์ 286-AT เป็นการสั่งให้ 8031-2 เริ่มทำการควบคุมตำแหน่งมอเตอร์ตามข้อมูลได้รับมา

4) เมื่อได้รับสัญญาณ 'move' แล้ว โปรแกรมจะตรวจสอบรหัสคำสั่งการเคลื่อนที่ , ทิศทางการหมุน จากนั้นจะโหลดค่าระยะทางซึ่งมีขนาด 16 บิท ไปแยกเก็บในตัวแปร HIBYTE และ LOBYTE โดย HIBYTE นั้นเก็บค่าไบท์บน (upper byte) ส่วน LOBYTE เก็บค่าไบท์ล่าง(lower byte) ระยะทางดังกล่าวนี้อยู่ในรูปของจำนวนพัลซ์ จากนั้นโปรแกรมจะตรวจสอบเงื่อนไขของจำนวนพัลซ์ใน

HIBYTE และ LOBYTE เพื่อนำไปเทียบหาความเร็วที่เหมาะสมในตารางที่ 4.1 จากนั้นจะส่งค่าความเร็วที่ได้ออกสู่พอร์ท 1 เพื่อใช้เป็นค่าคำสั่งความเร็ว (speed command) ให้กับวงจร PI คอนโทรลเลอร์ต่อไป

ตารางที่ 4.1 ค่าคำสั่งทางความเร็วที่เก็บในหน่วยความจำ

program memory address	speed command	program memory address	speed command	program memory address	speed command	program memory address	speed command
0100h	080h	0140h	0dah	0180h	000h	01c0h	025h
0101h	08bh	0141h	0dah	0181h	001h	01c1h	026h
0102h	08fh	0142h	0dbh	0182h	001h	01c2h	027h
0103h	093h	0143h	0dch	0183h	002h	01c3h	027h
0104h	096h	0144h	0dch	0184h	002h	01c4h	028h
0105h	099h	0145h	0ddh	0185h	003h	01c5h	029h
0106h	09bh	0146h	0deh	0186h	003h	01c6h	02ah
0107h	09dh	0147h	0deh	0187h	004h	01c7h	02ah
0108h	09fh	0148h	0dfh	0188h	004h	01c8h	02bh
0109h	0a1h	0149h	0e0h	0189h	005h	01c9h	02ch
010ah	0a3h	014ah	0e0h	018ah	005h	01cah	02dh
010bh	0a5h	014bh	0e1h	018bh	006h	01cbh	02dh
010ch	0a7h	014ch	0e2h	018ch	006h	01cch	02eh
010dh	0a8h	014dh	0e2h	018dh	007h	01cdh	02fh
010eh	0aah	014eh	0e3h	018eh	007h	01ceh	030h
010fh	0abh	014fh	0e4h	018fh	008h	01cfh	031h
0110h	0adh	0150h	0e4h	0190h	008h	01d0h	031h
0111h	0aeh	0151h	0e5h	0191h	009h	01d1h	032h
0112h	0afh	0152h	0e6h	0192h	009h	01d2h	033h
0113h	0b1h	0153h	0e6h	0193h	00ah	01d3h	034h
0114h	0b2h	0154h	0e7h	0194h	00ah	01d4h	035h
0115h	0b3h	0155h	0e7h	0195h	00bh	01d5h	036h
0116h	0b4h	0156h	0e8h	0196h	00bh	01d6h	036h
0117h	0b6h	0157h	0e9h	0197h	00ch	01d7h	037h
0118h	0b7h	0158h	0e9h	0198h	00dh	01d8h	038h
0119h	0b8h	0159h	0eah	0199h	00dh	01d9h	039h
011ah	0b9h	015ah	0eah	019ah	00eh	01dah	03ah
011bh	0bah	015bh	0ebh	019bh	00eh	01dbh	03bh
011ch	0bbh	015ch	0ech	019ch	00fh	01dch	03ch
011dh	0bch	015dh	0ech	019dh	00fh	01ddh	03dh
011eh	0bdh	015eh	0edh	019eh	010h	01deh	03eh
011fh	0beh	015fh	0edh	019fh	011h	01dfh	03fh
0120h	0bfh	0160h	0eeh	01a0h	011h	01e0h	040h
0121h	0c0h	0161h	0eeh	01a1h	012h	01e1h	041h
0122h	0c1h	0162h	0efh	01a2h	012h	01e2h	042h
0123h	0c2h	0163h	0f0h	01a3h	013h	01e3h	043h
0124h	0c3h	0164h	0f0h	01a4h	013h	01e4h	044h
0125h	0c4h	0165h	0f1h	01a5h	014h	01e5h	045h
0126h	0c5h	0166h	0f1h	01a6h	015h	01e6h	046h
0127h	0c6h	0167h	0f2h	01a7h	015h	01e7h	047h
0128h	0c7h	0168h	0f2h	01a8h	016h	01e8h	048h
0129h	0c8h	0169h	0f3h	01a9h	016h	01e9h	049h
012ah	0c9h	016ah	0f4h	01aah	017h	01eah	04bh
012bh	0c9h	016bh	0f4h	01abh	018h	01ebh	04ch
012ch	0cah	016ch	0f5h	01ach	018h	01ech	04dh
012dh	0cbh	016dh	0f5h	01adh	019h	01edh	04eh
012eh	0cch	016eh	0f6h	01aeh	019h	01eeh	050h
012fh	0cdh	016fh	0f6h	01afh	01ah	01efh	051h
0130h	0ceh	0170h	0f7h	01b0h	01bh	01f0h	052h
0131h	0ceh	0171h	0f7h	01b1h	01bh	01f1h	054h
0132h	0cfh	0172h	0f8h	01b2h	01ch	01f2h	055h
0133h	0d0h	0173h	0f8h	01b3h	01dh	01f3h	057h
0134h	0d1h	0174h	0f9h	01b4h	01dh	01f4h	058h
0135h	0d2h	0175h	0f9h	01b5h	01eh	01f5h	05ah
0136h	0d2h	0176h	0fah	01b6h	01fh	01f6h	05ch
0137h	0d3h	0177h	0fah	01b7h	01fh	01f7h	05eh
0138h	0d4h	0178h	0fbh	01b8h	020h	01f8h	060h
0139h	0d5h	0179h	0fbh	01b9h	021h	01f9h	062h
013ah	0d5h	017ah	0fch	01bah	021h	01fah	064h
013bh	0d6h	017bh	0fch	01bbh	022h	01fbh	066h
013ch	0d7h	017ch	0fdh	01bch	023h	01fch	069h
013dh	0d8h	017dh	0fdh	01bdh	023h	01fdh	06ch
013eh	0d8h	017eh	0feh	01beh	024h	01feh	070h
013fh	0d9h	017fh	0ffh	01bfh	025h	01ffh	074h

ตามตารางที่ 4.1 เป็นการเทียบหาค่าคำสั่งทางความเร็วซึ่งอยู่ในรูปออฟเซ็ท ไบนารี่ โดยค่าผลต่างทางตำแหน่ง (position error) หรือค่าระยะทาง จะกำหนดให้อยู่ในรูปแอดเดรสไบท์ต่ำ

(lower byte) ส่วนแอดเดรสไบท์สูง (upper byte) กำหนดไว้ที่ 01h

ข้อมูลในตารางที่ 4.1 ได้มาจากการคำนวณตามเงื่อนไขความสัมพันธ์ระหว่างผลต่างทางตำแหน่ง (q) กับความเร็วเชิงมุม (ω) โดยอาศัยสมการ (2.24) ซึ่งมีรายละเอียดดังนี้

ก) ผลต่างทางตำแหน่ง (q) ตั้งแต่ 0 ถึง +127 หรือตั้งแต่ 00000000b ถึง 01111111b (มอเตอร์หมุนตามเข็มนาฬิกา) ค่าความเร็วจะมีค่าเท่ากับ

$$\omega_c = \sqrt{127 \cdot q} \qquad \ldots.(4.1)$$

ข) ผลต่างทางตำแหน่ง (q) ตั้งแต่ 0 ถึง -128 หรือตั้งแต่ 11111111b ถึง 10000000b (มอเตอร์หมุนทวนเข็มนาฬิกา) ค่าความเร็วจะมีค่าเท่ากับ

$$\omega_c = -\sqrt{-128 \cdot q} \qquad \ldots.(4.2)$$

ค) ผลต่างทางตำแหน่ง (q) มากกว่า +127 (มอเตอร์หมุนตามเข็มนาฬิกา) กำหนดให้ค่าความเร็วเท่ากับ +127 คงที่

ง) ผลต่างทางตำแหน่ง (q) น้อยกว่า -128 (มอเตอร์หมุนทวนเข็มนาฬิกา) กำหนดให้ค่าความเร็วเท่ากับ -128 คงที่

จากค่าความเร็วที่คำนวณได้ตามเงื่อนไขนี้ จะต้องนำไปเทียบหาค่าออฟเซ็ท ไบนารี่ (offset binary) ที่สอดคล้องกับระดับแรงดันของวงจร D/A อีกทีหนึ่ง จึงจะได้เป็นค่าความเร็วที่เก็บในตารางที่ 4.1 สำหรับหลักการเทียบหาค่าออฟเซ็ท ไบนารี่ เป็นดังตารางที่ 4.2

ตารางที่ 4.2 ความสัมพันธ์ระหว่างค่าความเร็ว , ค่าออฟเซ็ทไบนารี่และแรงดัน D/A

ค่าความเร็ว	ค่าออฟเซ็ท ไบนารี่	ระดับแรงดัน (โวลท์)
+127	11111111b	+5
0	10000000b	+2.5
-128	00000000b	0

5) โปรแกรมทำการอินาเบิ้ลอินเตอรัพท์ T0 เพื่อเริ่มต้นนับจำนวนพัลซ์ ซึ่งจะสัมพันธ์กับระยะเชิงมุมที่มอเตอร์หมุนไป โปรแกรมจะนับค่าพัลซ์ทางตำแหน่งพร้อมกับลดค่าใน LOBYTE ลงไปเรื่อยๆ เมื่อ LOBYTE มีค่าเป็นศูนย์จะมีการขอยืม (borrow) จาก HIBYTE การควบคุมตำแหน่งจะทำไปเรื่อยๆ จนกว่าค่าใน HIBYTE และ LOBYTE เหลือศูนย์ทั้งคู่ มอเตอร์จึงจะหยุดหมุน จากนั้นโปร

แกรมจะกลับไปตรวจสอบรหัสคำสั่งชุดต่อไป

6) กรณีตรวจพบว่าเป็นคำสั่งให้แขนเซอร์โวหมุน โปรแกรมจะส่งคำสั่งควบคุมไปยังพอร์ท B ของ 8255-2 จากนั้นโปรแกรมจะกลับไปตรวจสอบรหัสคำสั่งชุดต่อไป สำหรับรายละเอียดเกี่ยวกับการควบคุมแขนเซอร์โวดูในภาคผนวก ข

7) เมื่อมอเตอร์หยุดตรงตำแหน่งที่ต้องการแล้ว โปรแกรมจะส่งสัญญาณ 'moving complete' ให้ 8031-1 ทราบ เพื่อให้ 8031-1 ส่งข้อความแสดงการทำงานสิ้นสุดไปยังคอมพิวเตอร์ 286-AT

4.2.2.2 โปรแกรมอินเตอรัพท์ (interrupt routine)

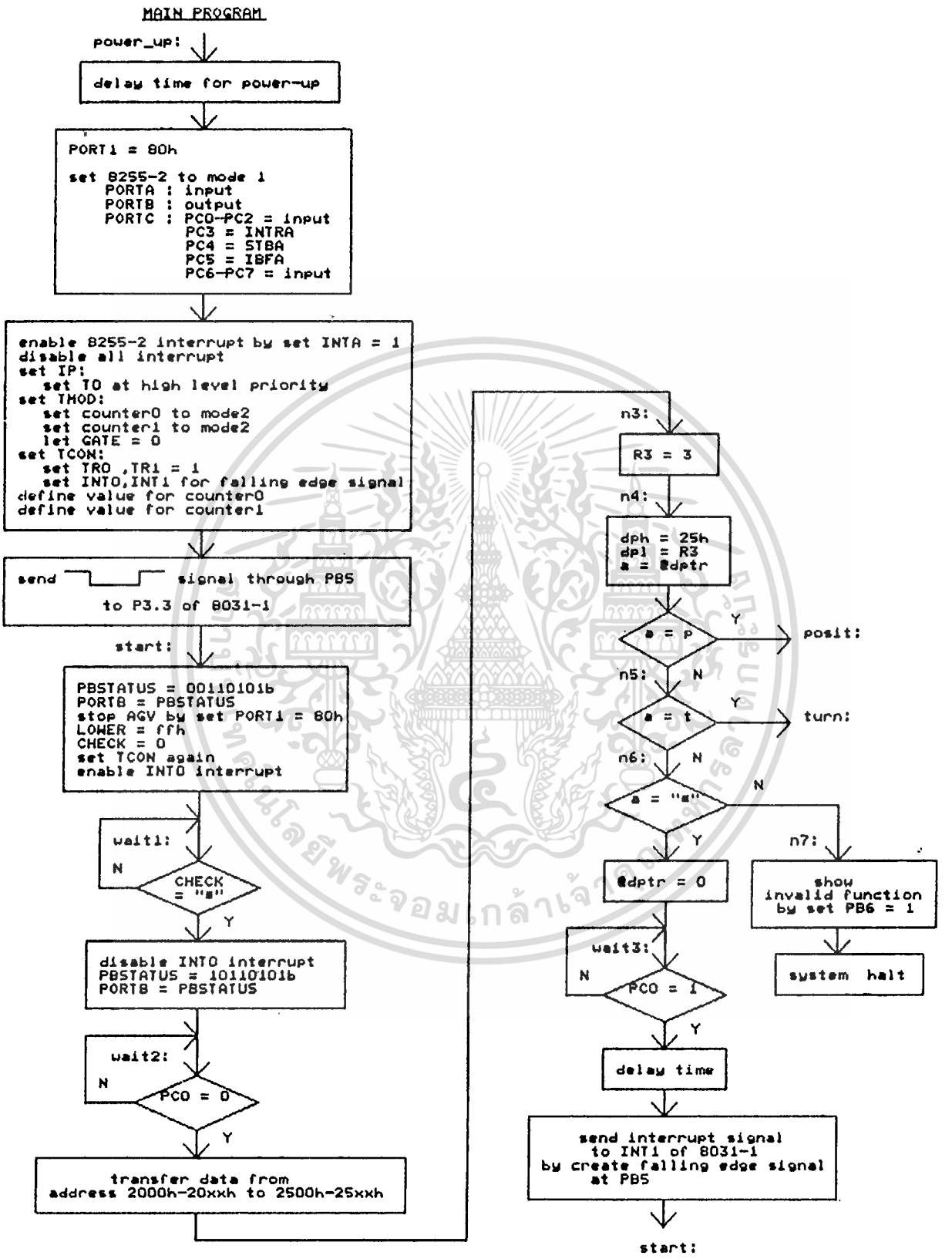
โปรแกรมควบคุมการทำงานบน 8031-2 มีการใช้โปรแกรมอินเตอรัพท์ต่างๆดังนี้

ก) โปรแกรมย่อยอินเตอรัพท์ INTO จะทำงานเมื่อมีสัญญาณ INTRA ที่ส่งมาจากขา PC3 ของ 8255-2 เป็นการสั่งให้ 8031-2 อ่านข้อมูลการเคลื่อนที่จากพอร์ท A ที่ส่งมาจาก 8031-1 ไปเก็บไว้ที่แอดเดรส 2000h-20xxh ลักษณะของข้อมูลที่เก็บจะอยู่ในลักษณะตารางดังรูปที่ 4.7

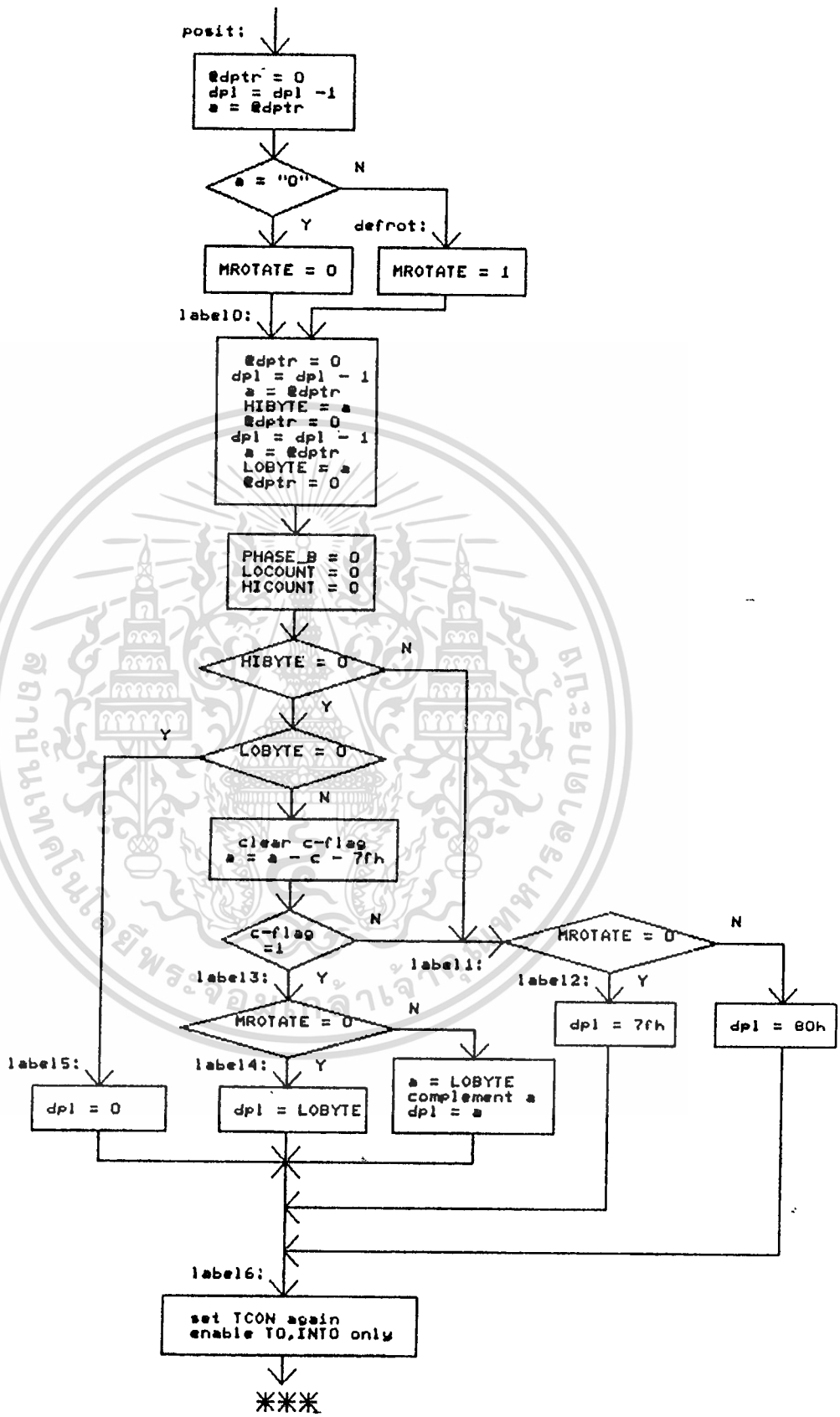
	address 2003h	address 2002h	address 2001h	address 2000h
address 2003h	position command "P" or "p"	rotate cw = "0" ccw = "1"	position pulse high byte	position pulse low byte
address 2007h	turn command "T" or "t"	direction left = "L" right = "R"	space	space
address 200bh	position command "P" or "p"	rotate cw = "0" ccw = "1"	position pulse high byte	position pulse low byte
address 200fh	turn command "T" or "t"	direction left = "L" right = "R"	space	space
address 2013h	position command "P" or "p"	rotate cw = "0" ccw = "1"	position pulse high byte	position pulse low byte
address 2017h	last "g"	space	space	space

รูปที่ 4.7 ลักษณะข้อมูลการเคลื่อนที่ที่เก็บในแอดเดรส 2000h - 20xxh

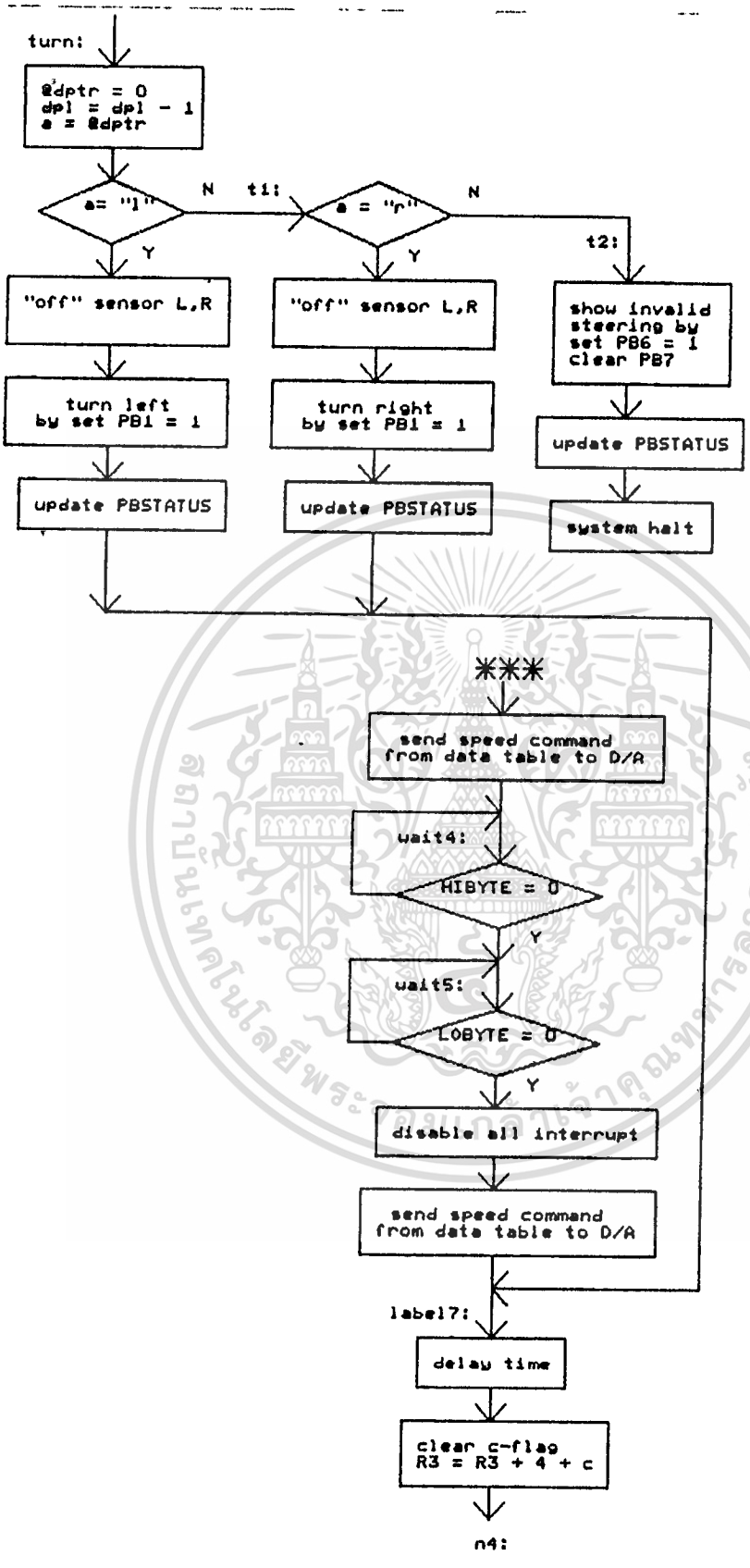
ข) โปรแกรมย่อยอินเตอรัพท์ T0 จะทำงานเมื่อมีสัญญาณขอบขาลงจากสัญญาณช่อง A จากออปติคัลเอ็นโคดเดอร์เข้ามาที่ขาอินเตอรัพท์ T0 โปรแกรมจะทำการตรวจสอบลอจิกของสัญญาณจากช่องสัญญาณ B เพื่อหาทิศทางการหมุนของมอเตอร์ หากทิศทางการหมุนถูกต้องตรงกับคำสั่งก็จะลดค่าพัลซ์ทางตำแหน่งที่เหลือในตัวแปร LOBYTE ลง 1 พัลซ์ กรณีที่ LOBYTE ลดค่าลงเหลือศูนย์ จะมีการขอยืมค่าจาก HIBYTE จากนั้นจะนำค่าล่าสุดใน LOBYTE ไปเป็นแอดเดรสเพื่อชี้หาค่าความเร็วที่เหมาะสมในตารางที่ 4.1 แล้วนำค่าความเร็วนั้นไปใช้งาน



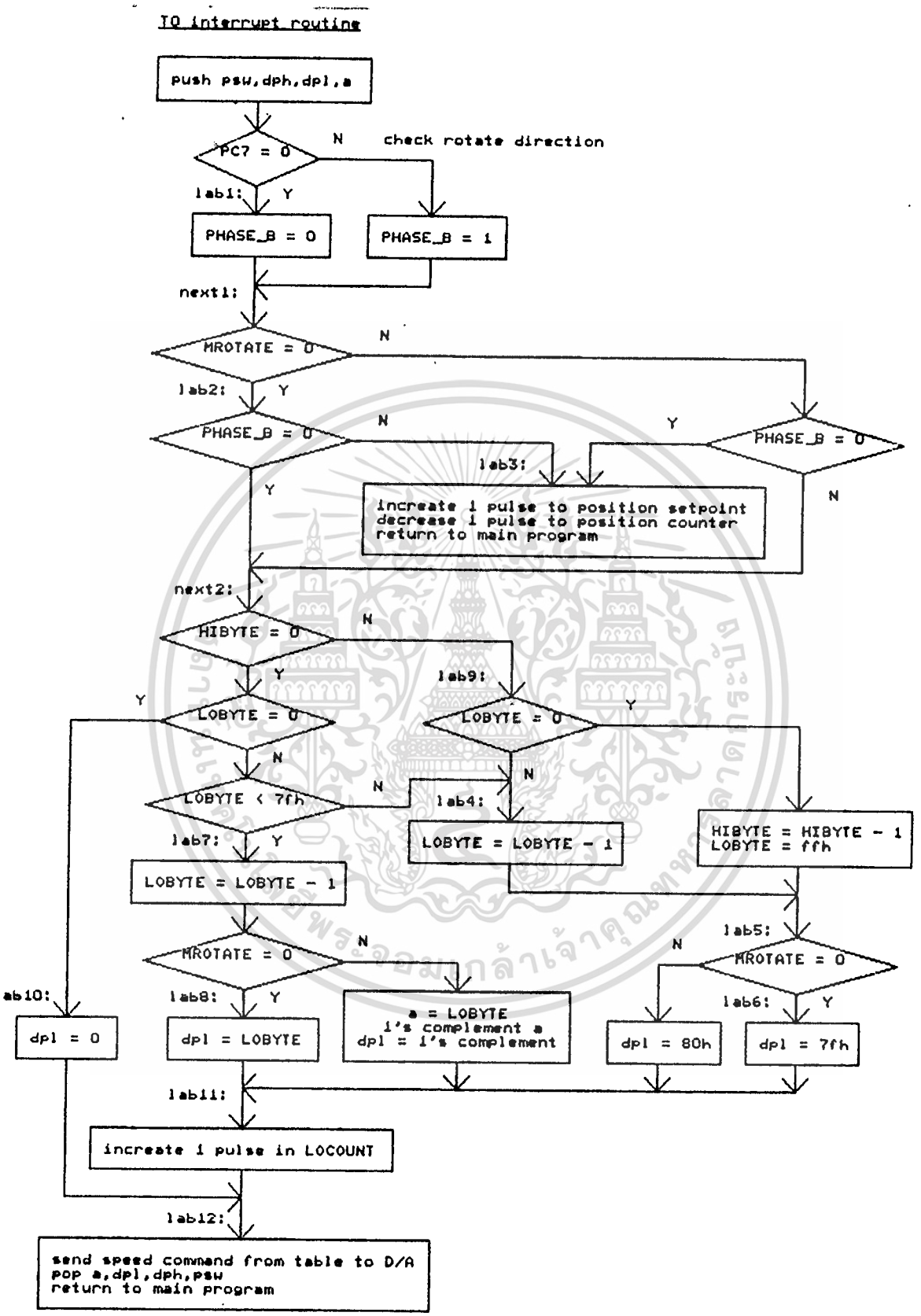
รูปที่ 4.8 แผนผังโปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรงและควบคุมการหมุนแขนเซอร์โว



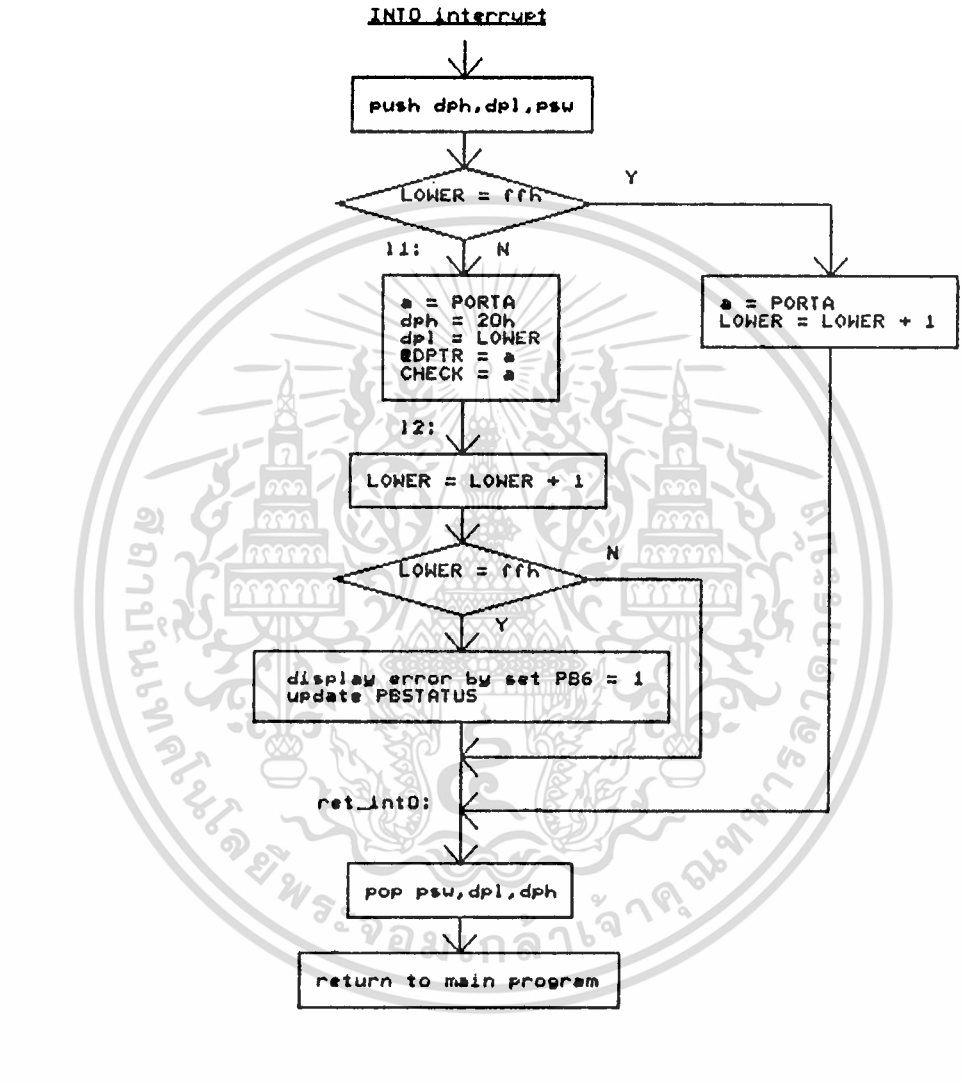
รูปที่ 4.8 แผนผังโปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรง
และควบคุมการหมุนแขนเซอร์โว (ต่อ)



รูปที่ 4.8 แผนผังโปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรง และควบคุมการหมุนแขนเซอร์โว (ต่อ)



รูปที่ 4.8 แผนผังโปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรง
และควบคุมการหมุนแขนเซอร์โว (ต่อ)



รูปที่ 4.8 แผนผังโปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรง
และควบคุมการหมุนแขนเซอร์โว (ต่อ)

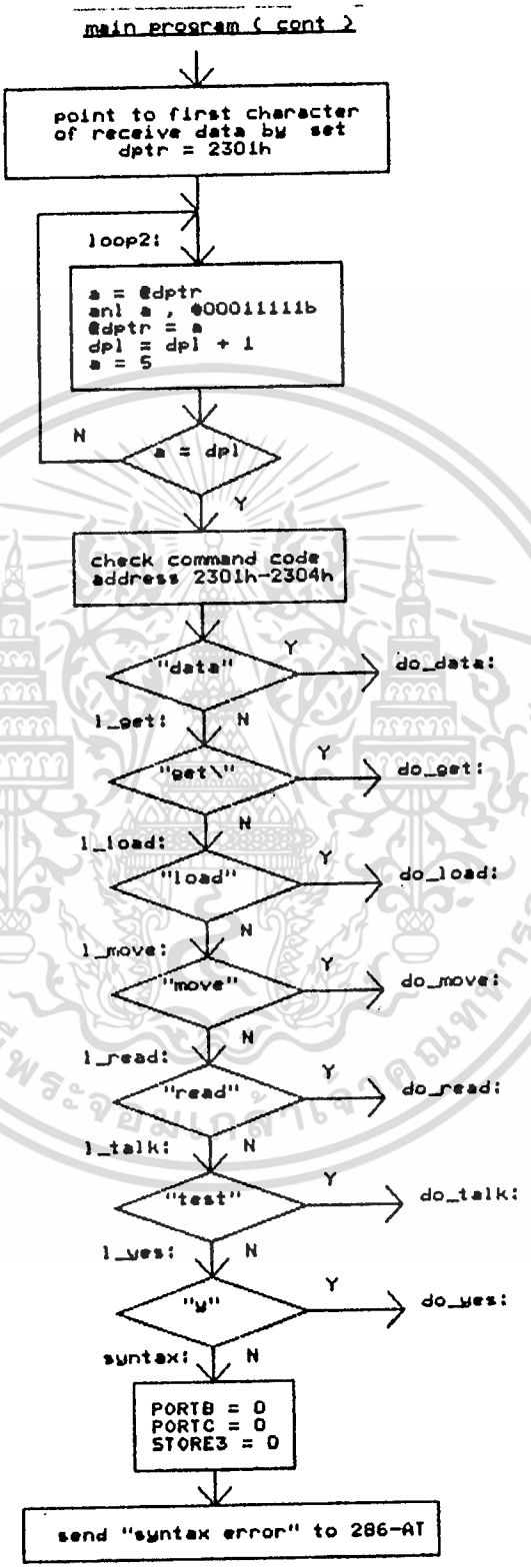
4.2.3 โปรแกรมคำสั่งต่างๆ

เป็นโปรแกรมหลักที่ควบคุมการทำงานบนซีพียู 8031-1 คำสั่งต่างๆที่รับจากไมโครคอมพิวเตอร์ 286-AT จะถูกเก็บไว้ที่แอดเดรส 2300-23xxh ลักษณะคำสั่งประกอบด้วยรหัสแอสกี้ (ASCII code) ของอักขระ (character) 4 ตัวเรียงกันอยู่ สำหรับอักขระนั้นอาจเป็นตัวพิมพ์ใหญ่หรือตัวพิมพ์เล็กก็ได้

โปรแกรมนี้จะเริ่มด้วยการนำรหัสคำสั่งที่เก็บในแอดเดรส 2301h-2304h มาปรับรหัสแอสกี้ของตัวพิมพ์ใหญ่และตัวพิมพ์เล็กให้ตรงกันเพื่อให้ง่ายต่อการแปลคำสั่งดังตารางที่ 4.3 ซึ่งแสดงรายละเอียดของรหัสคำสั่งก่อนและหลังทำการปรับ ส่วนในรูปที่ 4.9 เป็นแผนผังโปรแกรมในส่วนการปรับรหัสและเลือกคำสั่ง

ตารางที่ 4.3 รายละเอียดรหัสคำสั่งต่างๆ

คำสั่ง	รหัสแอสกี้(ASCII code)	รหัสคำสั่งหลังทำการปรับ
get\	67 , 65 , 74 , 5c	07 , 05 , 14 , 1C
GET\	47 , 45 , 54 , 5c	07 , 05 , 14 , 1C
data	64 , 61 , 74 , 61	04 , 01 , 14 , 01
DATA	44 , 41 , 54 , 41	04 , 01 , 14 , 01
move	6d , 6f , 76 , 65	0d , 0f , 16 , 05
MOVE	4d , 4f , 56 , 45	0d , 0f , 16 , 05
read	72 , 65 , 61 , 64	12 , 05 , 01 , 04
READ	52 , 45 , 41 , 44	12 , 05 , 01 , 04
test	74 , 65 , 73 , 74	14 , 05 , 13 , 14
TEST	54 , 45 , 53 , 54	14 , 05 , 13 , 14



รูปที่ 4.9 แผนผังโปรแกรมในส่วนการปรับรหัสคำสั่งและเลือกคำสั่ง

สำหรับรายละเอียดการทำงานของโปรแกรมในส่วนคำสั่งต่างๆมีดังต่อไปนี้

4.2.3.1) คำสั่ง get\

เป็นคำสั่งที่บรรจุคำสั่งย่อยๆที่เกี่ยวกับการเคลื่อนที่เอาไว้ ได้แก่ คำสั่งในการควบคุมตำแหน่งมอเตอร์กระแสตรง และคำสั่งกำหนดทิศทางของแขนเซอร์โว (servo arms)

รูปแบบ (format) ของคำสั่ง get\ ที่สมบูรณ์มีดังนี้

```
get\p=xxxxx,y\t=z\p=xxxxx,y\....
```

โดยที่

p : รหัสคำสั่งในการควบคุมตำแหน่งมอเตอร์กระแสตรง

xxxxx : ระยะเชิงมุมที่มอเตอร์หมุนไปมีหน่วยเป็นจำนวนพัลซ์ สามารถกำหนดค่าได้ตั้งแต่ 0 ถึง 65535 โดยกำหนดเป็นเลขฐานสิบ 0 - 9

y : ทิศทางการหมุนของมอเตอร์ซึ่งมี 2 ทิศทางได้แก่

"0" เป็นการกำหนดให้มอเตอร์หมุนตามเข็ม (c.w)

"1" เป็นการกำหนดให้มอเตอร์หมุนทวนเข็ม (c.c.w)

t : รหัสคำสั่งกำหนดทิศทางการหมุนของแขนเซอร์โว

z : ทิศทางการหมุนของแขนเซอร์โวมี 2 ทิศทางคือ

"l" เป็นการกำหนดให้แขนเซอร์โวหมุนไปทางซ้าย (turn left)

"r" เป็นการกำหนดให้แขนเซอร์โวหมุนไปทางขวา (turn right)

หมายเหตุ: คำสั่ง p ต้องกำหนดเป็นไว้ลำดับแรกและลำดับสุดท้ายเสมอ

โปรแกรมจะเริ่มด้วยการตรวจสอบรูปแบบคำสั่งย่อยที่ตามหลังคำสั่ง get\ ก่อน ว่าถูกต้องหรือไม่ ไม่ว่าจะเป็นรหัสคำสั่ง, ทิศทางการหมุนมอเตอร์กระแสตรง, ทิศทางการหมุนของแขนเซอร์โว, เครื่องหมายต่างๆ, อักขระของจำนวนพัลซ์ทางตำแหน่งเป็นเลข 0-9 หากตรวจพบว่ามีส่วนใดไม่ตรงกับที่กำหนดไว้ โปรแกรมก็จะส่งข้อความแสดงข้อผิดพลาดนั้นกลับไปยังไมโครคอมพิวเตอร์ทันที

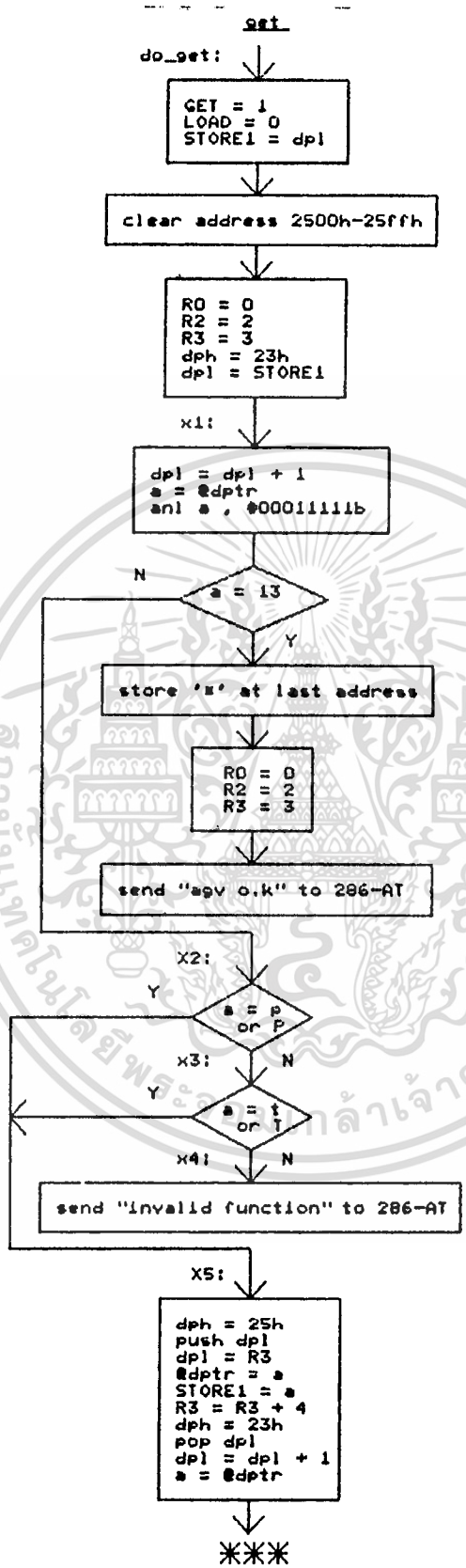
โปรแกรมจะทำการแปลงค่าตั้งทางตำแหน่งซึ่งอยู่ในรูปรหัสแอสกี้ของอักขระ 0 - 9 ไปเป็นเลขฐานสิบหกขนาด 2 ไบท์เพื่อนำไปใช้คำนวณระยะเชิงมุมในขณะที่มอเตอร์หมุน พร้อมกับตรวจสอบว่าค่าที่ส่งมาเกินที่ 65535 หรือไม่ หากเกินก็จะส่งข้อความเตือนไปยังไมโครคอมพิวเตอร์ทันที

หลังจากตรวจสอบคำสั่งย่อยทุกคำสั่งพร้อมกับแปลงอักขระของตำแหน่งเป็นเลขฐานสิบหกเรียบร้อยแล้ว โปรแกรมจะนำข้อมูลการเคลื่อนที่ที่ตรวจสอบและแปลงแล้วไปเก็บยังแอดเดรส 2500h-25xxh

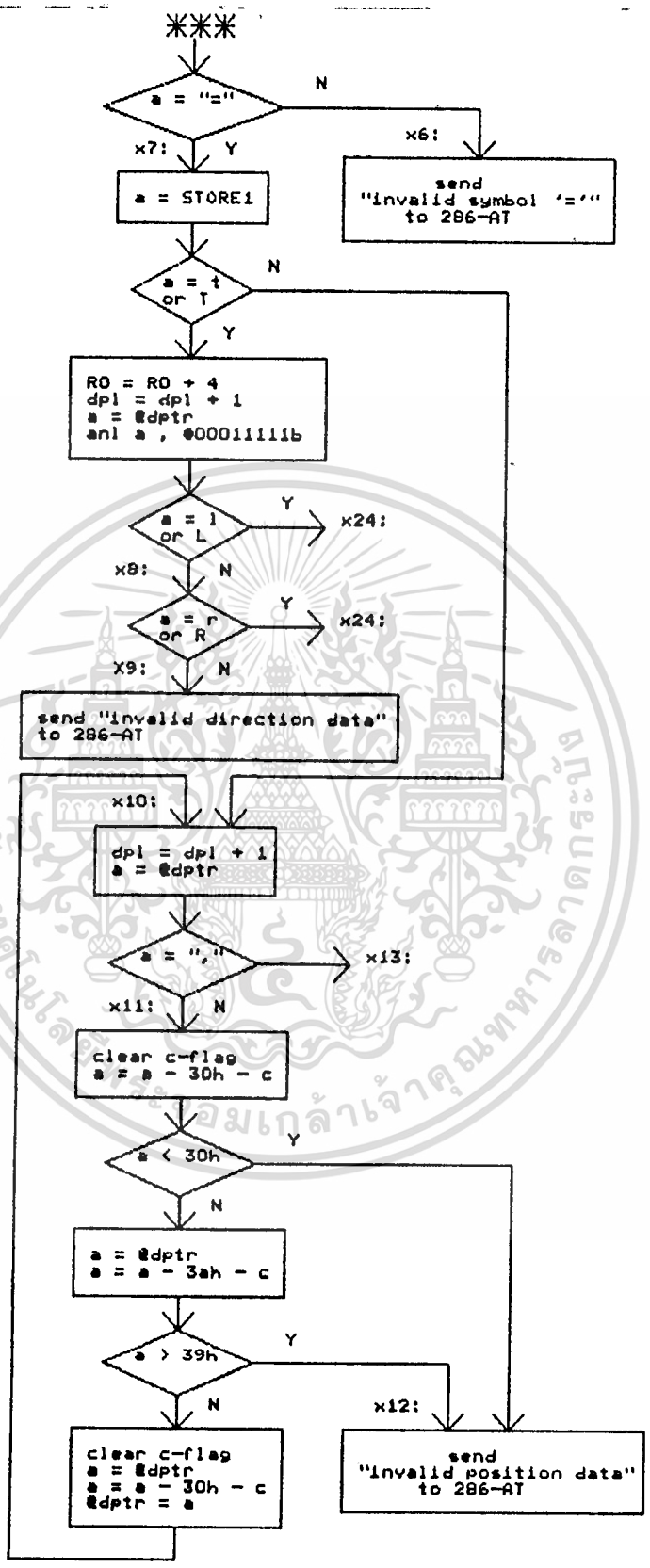
สำหรับลักษณะการเก็บข้อมูลที่แอดเดรส 2500h - 25xxh นั้นแสดงดังรูปที่ 4.10 ส่วนแผนผังการทำงานของโปรแกรมปฏิบัติงานตามคำสั่ง get\ แสดงในรูปที่ 4.11

	address 2503h	address 2502h	address 2501h	address 2500h
address 2503h	position command "P" or "p"	rotate cw = "0" ccw = "1"	position pulse high byte	position pulse low byte
address 2507h	turn command "T" or "t"	direction left = "L" right = "R"	space	space
address 250bh	position command "P" or "p"	rotate cw = "0" ccw = "1"	position pulse high byte	position pulse low byte
address 250fh	turn command "T" or "t"	direction left = "L" right = "R"	space	space
address 2513h	position command "P" or "p"	rotate cw = "0" ccw = "1"	position pulse high byte	position pulse low byte
address 2017h	last "g"	space	space	space

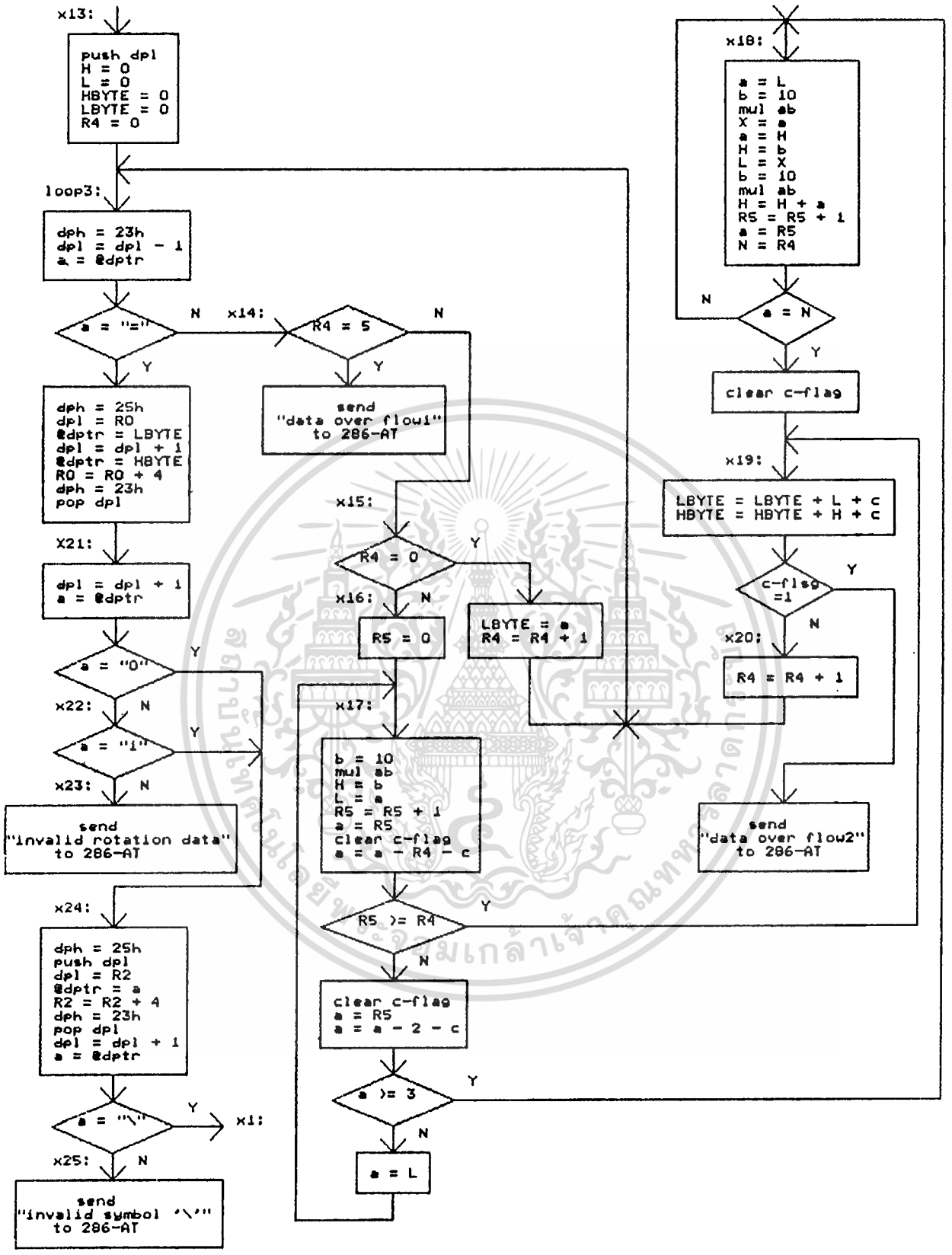
รูปที่ 4.10 ลักษณะข้อมูลการเคลื่อนที่ที่เก็บในแอดเดรส 2500h - 25xxh



รูปที่ 4.11 แผนผังโปรแกรมปฏิบัติตามคำสั่ง get\



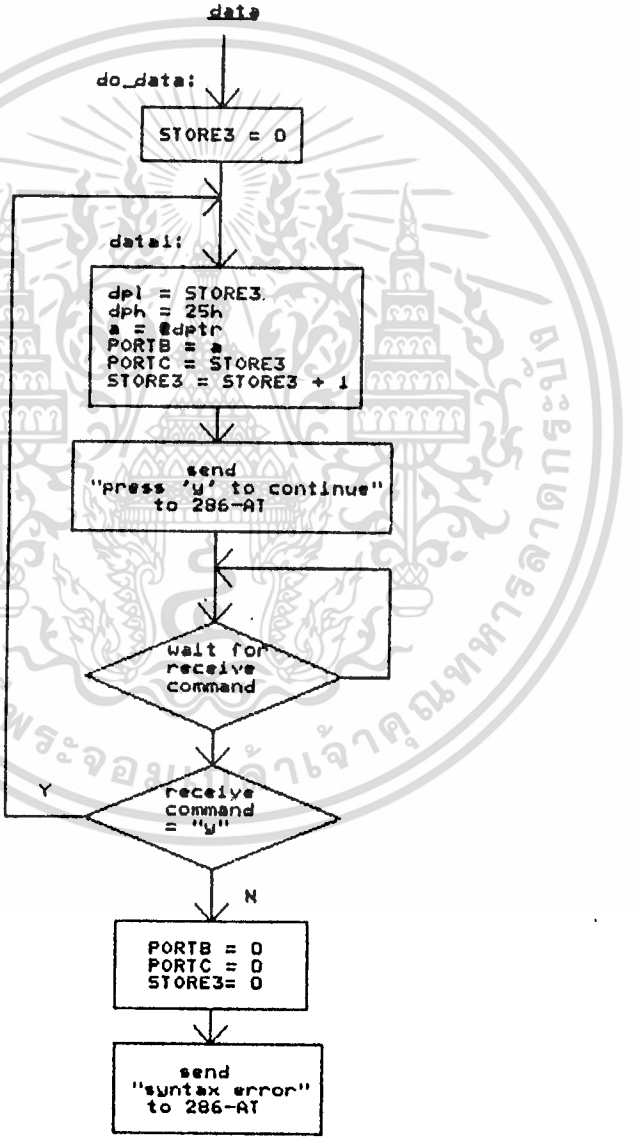
รูปที่ 4.11 แผนผังโปรแกรมปฏิบัติตามคำสั่ง get\ (ต่อ)



รูปที่ 4.11 แผนผังโปรแกรมปฏิบัติตามคำสั่ง get\ (ต่อ)

4.2.3.2 คำสั่ง data

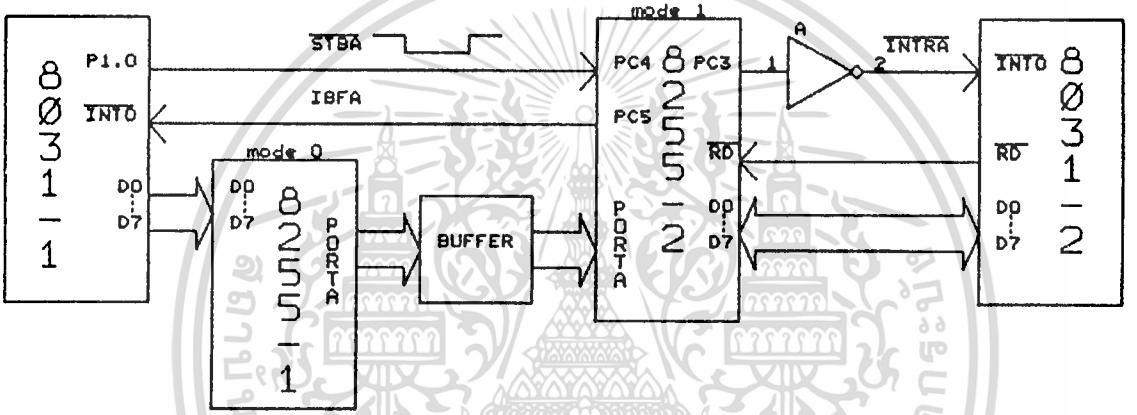
เป็นคำสั่งเรียกดูข้อมูลที่เกี่ยวกับการเคลื่อนที่ซึ่งเก็บในแอดเดรส 2500h-25xxh โดยแสดงออกที่พอร์ท B และพอร์ท C ที่พอร์ท B จะเป็นการแสดงค่าแอดเดรสไบท์ต่ำ (lower byte) ส่วนที่พอร์ท C จะแสดงข้อมูลในแอดเดรสนั้น และสำหรับข้อมูลในแอดเดรสถัดไปก็สามารถเรียกดูได้เช่นกัน , รายละเอียดของโปรแกรมในส่วนคำสั่งนี้แสดงไว้ในรูปที่ 4.12



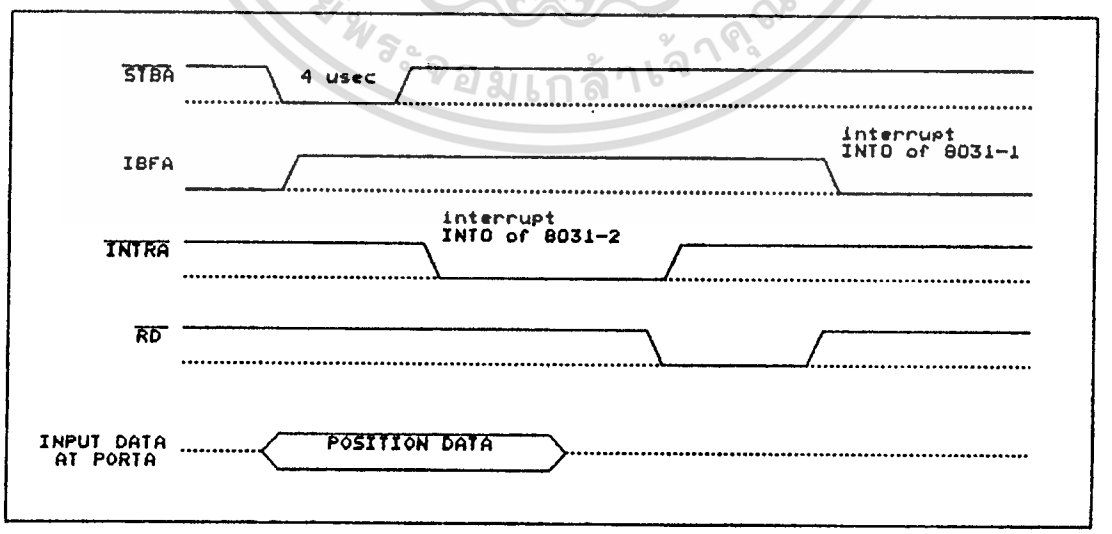
รูปที่ 4.12 แผนผังโปรแกรมปฏิบัติตามคำสั่ง data

4.2.3.3 คำสั่ง load

เป็นคำสั่งที่เกี่ยวข้องกับซีพียูทั้งสองตัวคือ 8031-1 กับ 8031-2 กล่าวคือเป็นคำสั่งให้ 8031-1 ทำการย้ายข้อมูลการเคลื่อนที่ซึ่งเก็บในแอดเดรส 2500h-25xxh ไปเก็บยังแอดเดรส 2000h-20xxh ของ 8031-2 การส่งข้อมูลไปยัง 8031-2 นั้นจะใช้วิธี handshaking ดังบล็อคไดอะแกรมในรูปที่ 4.13 ซึ่งจะเห็นว่าได้ทำการตั้งโหมดการทำงานของ 8255-2 ไว้ที่โหมด 1 แบบสโตรบอินพุททั้งนี้เพื่อให้ 8255-2 ทำหน้าที่ handshaking ส่วนรูปที่ 4.14 นั้นเป็นแผนผังเวลาการทำงานของ 8255-2 ในโหมด 1 แบบสโตรบอินพุท



รูปที่ 4.13 บล็อคไดอะแกรมแสดงวิธี handshaking ในการส่งข้อมูลจาก 8031-1 ไป 8031-2



รูปที่ 4.14 แผนผังเวลาการทำงานของ 8255-2 ในโหมด 1 สโตรบอินพุท

สำหรับรายละเอียดของโปรแกรมในส่วนคำสั่ง load มีดังนี้

ก) โปรแกรมคำสั่ง load ในส่วนไมโครคอนโทรลเลอร์ 8031-1

หลังจากที่ 8031-1 รับทราบคำสั่ง load แล้ว โปรแกรมจะตรวจสอบลอจิกของขา STBA และขา IBF ว่าอยู่ในสภาวะที่ถูกต้องพร้อมที่จะเริ่มทำ handshhaking หรือไม่ นั่นคือจากรูปที่ 68 ลอจิกที่ถูกต้องก่อนจะทำ handshaking คือ STBA=1 และ IBF=0 หากไม่เป็นตามนี้ โปรแกรมจะส่งข้อความแสดงสภาวะของขา STBA และขา IBF กลับไปยังไมโครคอมพิวเตอร์ แต่หากเป็นไปตามเงื่อนไข โปรแกรมจะทำการอีนาเบิ้ลอินเตอรัพท์ INTO พร้อมกับส่งสัญญาณสโตรบ STBA ออกไปที่ขา P1.0 อันเป็นการเริ่มต้นการส่งข้อมูล 8255-2 จะส่งสัญญาณ IBF=1 กลับมาพร้อมกับส่งสัญญาณ INTRA ไปเข้าขาอินเตอรัพท์ INTO ของ 8031-2 โปรแกรมอินเตอรัพท์ INTO ของ 8031-2 จะทำการอ่านข้อมูลจากพอร์ท A ของ 8255-2 เข้าไปเก็บยังแอดเดรส 2000h - 20xxh ขอบขาขึ้นของสัญญาณ RD จะทำให้สัญญาณ IBF เปลี่ยนจาก 1 เป็น 0 กลายเป็นสัญญาณไปเข้าขาอินเตอรัพท์ INTO ของ 8031-1

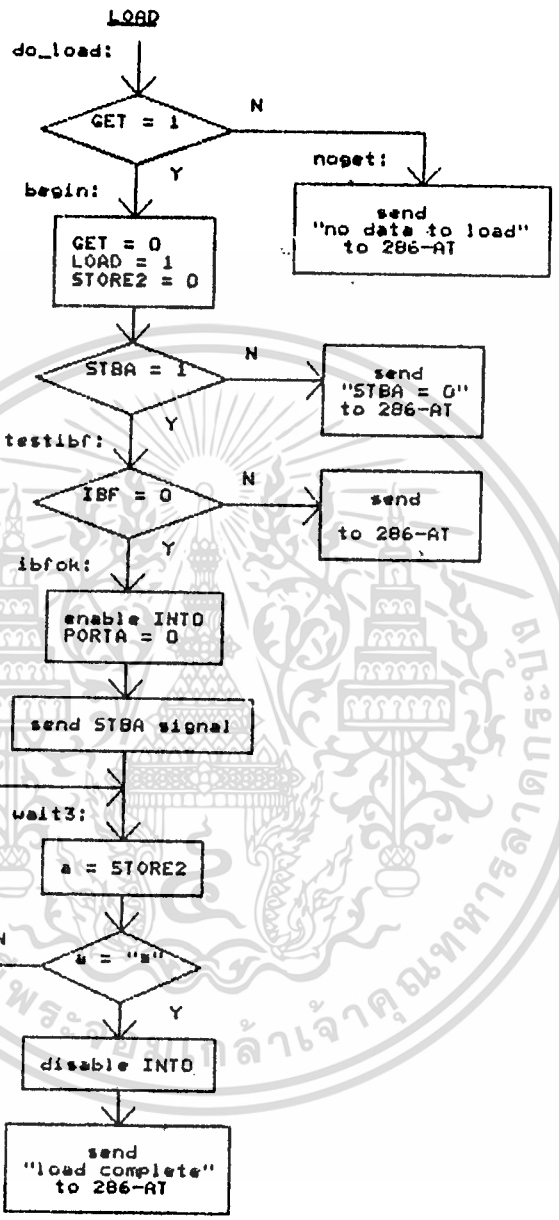
โปรแกรมอินเตอรัพท์ INTO ของ 8031-1 จะทำการดึงข้อมูล 1 ไบท์ในแอดเดรส 2500h ส่งออกพอร์ท A ของ 8255-1 แล้วทำการเลื่อนข้อมูลในไบท์สูงกว่าลงมาแทนที่ ดังนั้นทุกครั้งมีการเรียกใช้อินเตอรัพท์นี้ ข้อมูลในแอดเดรส 2500h - 25xxh จะลดลงเรื่อยๆ เมื่อส่งข้อมูลออกพอร์ท A เรียบร้อยแล้วโปรแกรมจะสร้างสัญญาณสโตรบ STBA ออกที่ขา P1.0 เพื่อเริ่มต้นการ handshaking ครั้งต่อไป จากนั้นจึงจะออกสู่โปรแกรมหลัก

การส่งข้อมูลไปยัง 8031-2 จะสิ้นสุดเมื่ออักขระ "*" ถูกส่งออกไป หลังจากนั้นโปรแกรมหลักจะทำการดิสเอเบิ้ลอินเตอรัพท์ INTO พร้อมกับส่งข้อความกลับไปยังเครื่องคอมพิวเตอร์เพื่อแสดงว่าการ load สิ้นสุดลง สำหรับแผนผังโปรแกรมในส่วนของคำสั่ง load ที่อยู่บนซีพียู 8031-1 นี้แสดงไว้ในรูปที่ 4.15

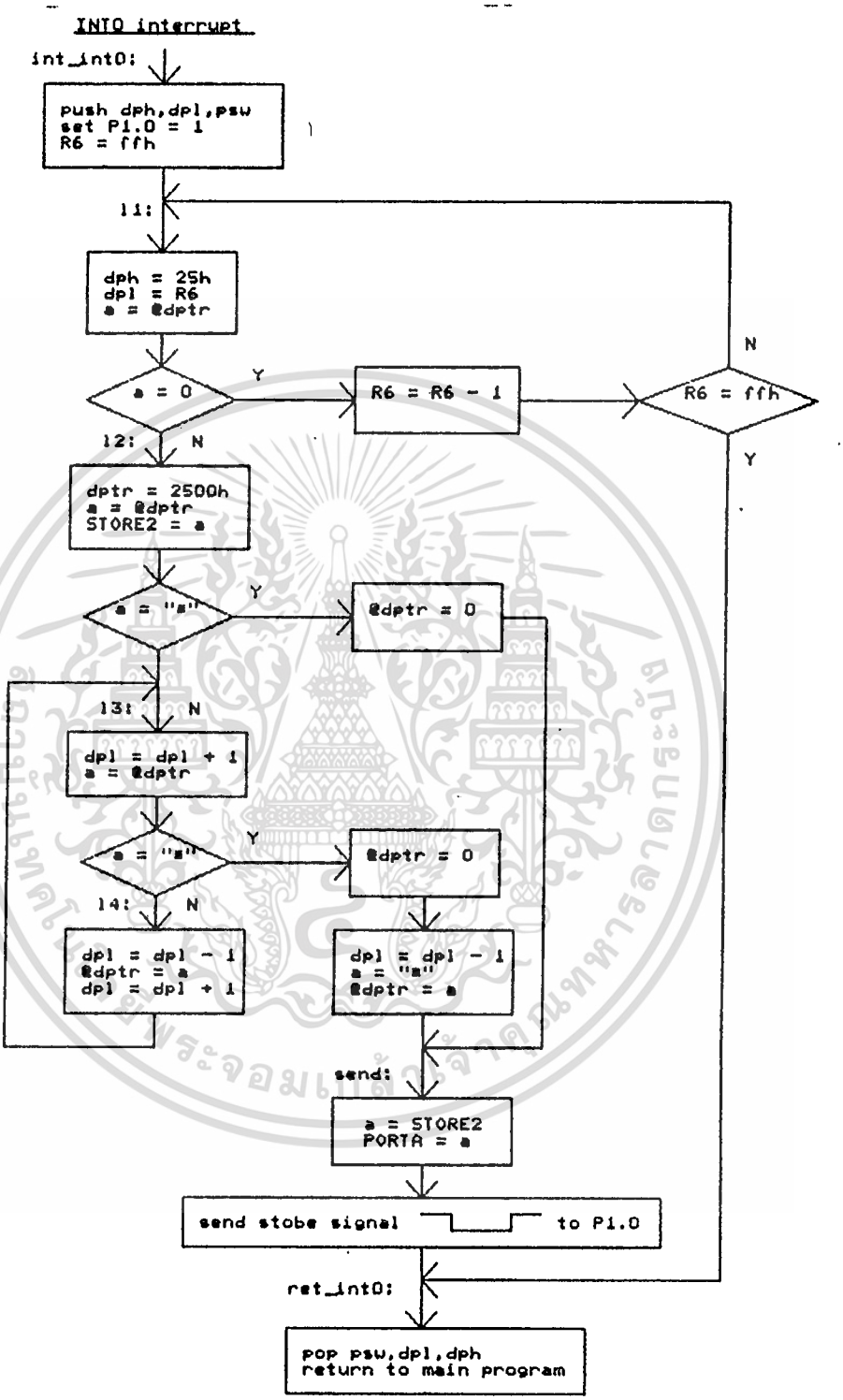
ข) โปรแกรมคำสั่ง load ในส่วนไมโครคอนโทรลเลอร์ 8031-2

โปรแกรมบนซีพียู 8031-2 ที่เกี่ยวข้องกับคำสั่ง load เป็นโปรแกรมอินเตอรัพท์ INTO ซึ่งจะทำงานเมื่อขา INTO ได้รับสัญญาณขอบขาลง INTRA ซึ่งส่งมาจากขา PC3 ของ 8255-2 โปรแกรมอินเตอรัพท์จะทำการอ่านข้อมูลจากพอร์ท A ของ 8255-2 เข้าไปเก็บยังแอดเดรส 2000h - 20xxh

ลักษณะการเก็บข้อมูลนั้นก็ได้แสดงไว้แล้วในรูปที่ 4.10 ส่วนแผนผังการทำงานของโปรแกรมอินเตอรัพท์นี้ก็แสดงไว้แล้วในรูปที่ 4.11 เช่นกัน



รูปที่ 4.15 แผนผังโปรแกรมปฏิบัติตามคำสั่ง load

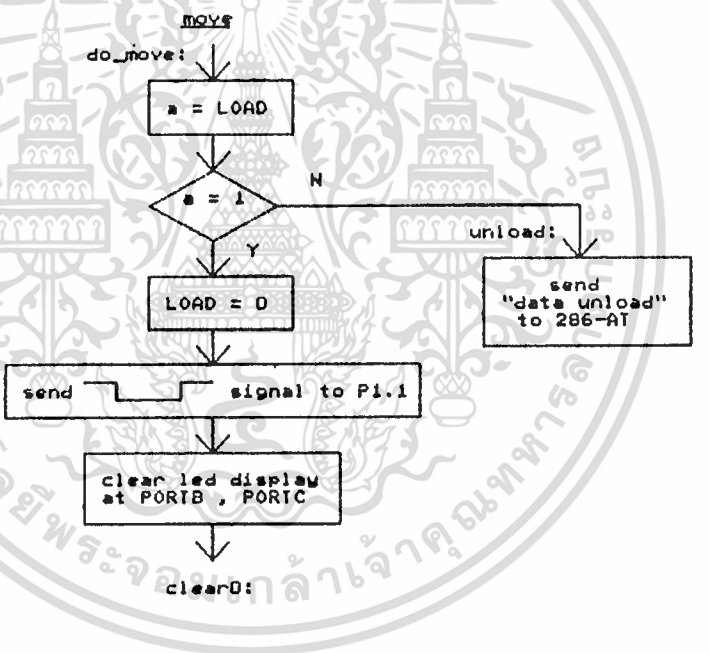


รูปที่ 4.15 แผนผังโปรแกรมปฏิบัติตามคำสั่ง load (ต่อ)

4.2.3.4) คำสั่ง move

เป็นคำสั่งให้มอเตอร์เริ่มหมุนตามข้อมูลที่เก็บในแอดเดรส 2000h -20xxh แต่ซีพียู 8031-1 จะทำคำสั่งนี้ก็ต่อเมื่อได้มีการใช้คำสั่ง load เรียบร้อยแล้วเท่านั้น

เมื่อมีการส่งคำสั่ง move จากไมโครคอมพิวเตอร์ 286-AT มายังซีพียู 8031-1 ซีพียู 8031-1 จะส่งสัญญาณลอจิกศูนย์ออกไปยังขา P1.1 ขณะเดียวกันนั้นซีพียู 8031-2 จะรอรับสัญญาณนี้ที่ขา PC0 ของ 8255-2 เมื่อ 8031-2 ได้รับสัญญาณแล้วก็จะปฏิบัติตามข้อมูลที่กำหนดไว้โดยส่งค่าคำสั่งของความเร็ว (speed command) ให้มอเตอร์เริ่มทำการหมุน สำหรับแผนผังโปรแกรมปฏิบัติตามคำสั่ง move แสดงไว้ในรูปที่ 4.16



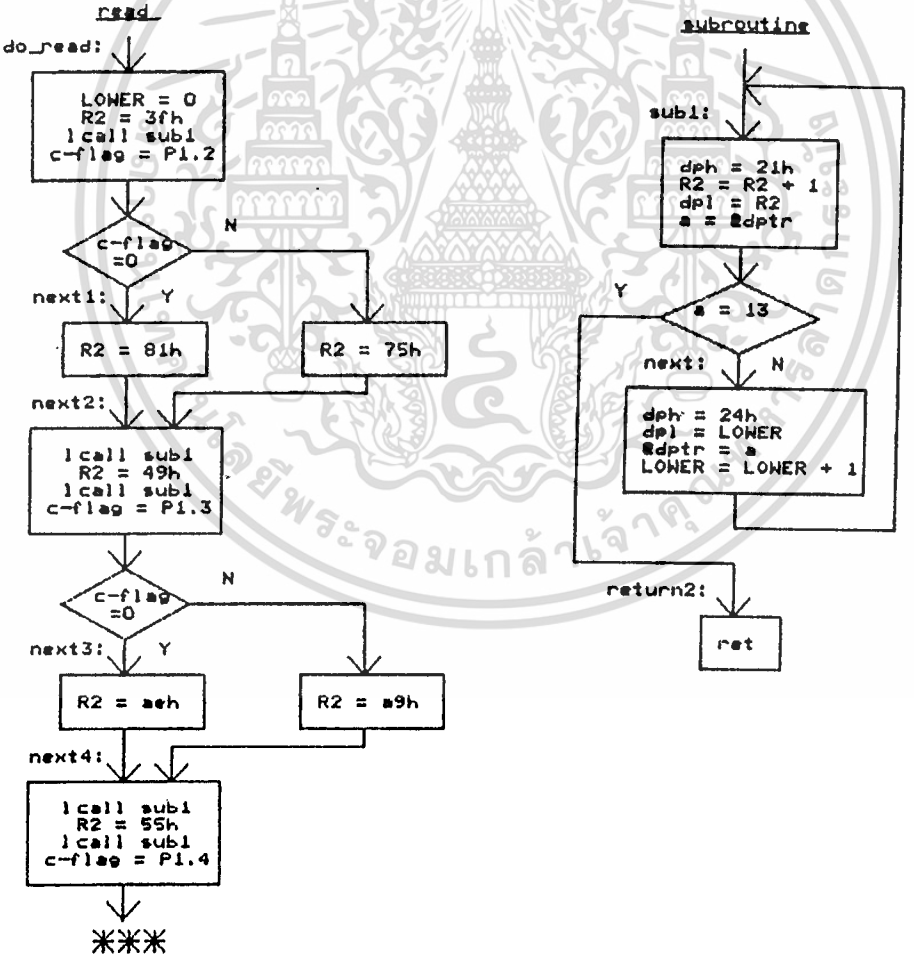
รูปที่ 4.16 แผนผังโปรแกรมปฏิบัติตามคำสั่ง move

4.2.3.5) คำสั่ง read

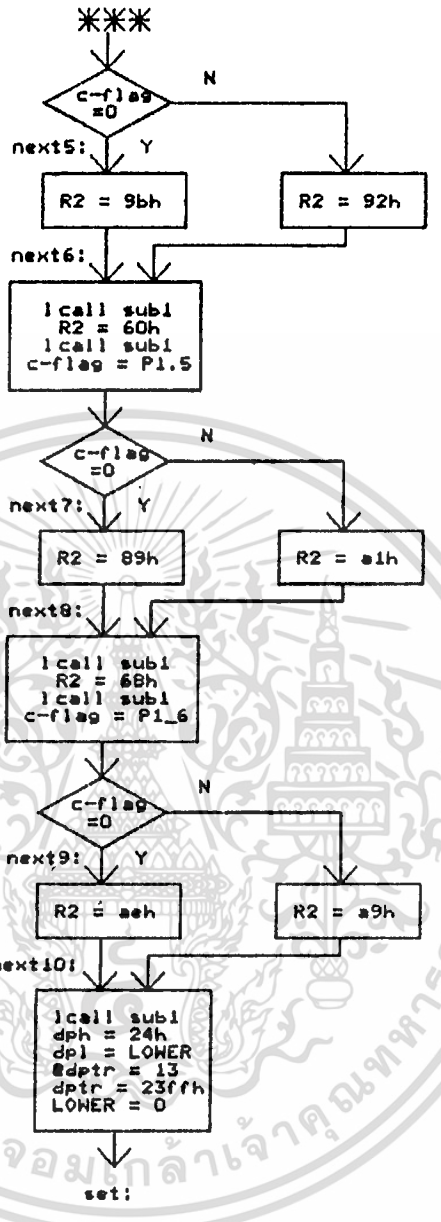
เป็นคำสั่งให้ 8031-1 ทำการอ่านสถานภาพ (status) ที่สำคัญของระบบ จากพอร์ท B แล้วส่งข้อมูกลับไปยังไมโครคอมพิวเตอร์ สำหรับตัวอย่างสถานภาพของระบบที่น่าสนใจ อาทิเช่น

1) สถานีเป้าหมาย (target station) ตรวจสอบว่า AGV หยุดอยู่ที่สถานีเป้าหมายหรือไม่
2) วัตถุกีดขวาง (obstacle) ตรวจสอบว่า AGV มีวัตถุกีดขวางการเคลื่อนที่อยู่หรือไม่
3) แบตเตอรี่ (battery) ตรวจสอบแรงดันแบตเตอรี่ว่ายังอยู่ในระดับปกติหรือไม่
4) การเคลื่อนที่ (moving) ตรวจสอบว่า AGV กำลังเคลื่อนที่อยู่หรือไม่
5) ทางแยก (cross-way) ตรวจสอบว่า AGV หยุดอยู่ที่ทางแยกหรือไม่

แผนผังโปรแกรมในส่วนคำสั่ง read แสดงไว้ในรูปที่ 4.17



รูปที่ 4.17 แผนผังโปรแกรมปฏิบัติตามคำสั่ง read



รูปที่ 4.17 แผนผังโปรแกรมปฏิบัติตามคำสั่ง read (ต่อ)

4.2.3.6 คำสั่ง test

เป็นคำสั่งทดสอบการสื่อสารข้อมูลระหว่างไมโครคอมพิวเตอร์ 286-AT กับซีพียู 8031-1 โดยการนำข้อมูลที่รับมาจากไมโครคอมพิวเตอร์ ส่งกลับคืนไปยังไมโครคอมพิวเตอร์

บทที่ 5

ผลการทดสอบการทำงานของระบบต่างๆ

5.1 ผลการทดสอบระบบสื่อสารข้อมูลระหว่างไมโครคอมพิวเตอร์กับไมโครคอนโทรลเลอร์

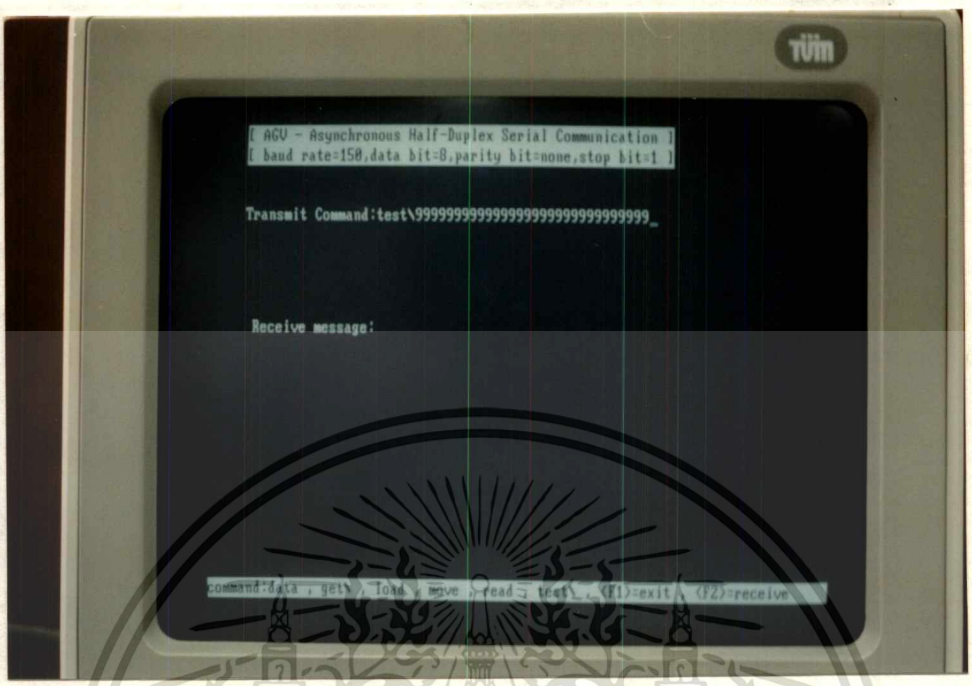
ทำการทดสอบระบบด้วยการป้อนคำสั่งบนไมโครคอมพิวเตอร์ 286-AT ดังรูปที่ 5.1 ตามคำสั่งนี้เป็นการส่งข้อมูลรหัสแอสกี้ของ 9 จำนวน 30 ตัวไปยัง 8031-1 จากนั้น 8031-1 จะส่งข้อมูลเดียวกันกลับมา รหัสแอสกี้ของ 9 คือ 39h หรือ 00111001b เมื่อส่งเป็นข้อมูลอนุกรมขนาด 10 บิทออกไปจะมีลักษณะดังรูปที่ 5.2 ตามรูปสัญญาณช่องบนเป็นการวัดที่พอร์ท RS-232-C ส่วนสัญญาณช่องล่างเป็นการวัดก่อนเข้าวงจรแปลง

ในรูปที่ 5.3 แสดงการแปลงข้อมูลอนุกรมเป็นขบวนพัลซ์ที่พร้อมจะส่งต่อไปยังภาคมอดดูเลท โดยสัญญาณช่องบนเป็นข้อมูลอนุกรมก่อนเข้าวงจรแปลง ส่วนสัญญาณช่องล่างเป็นขบวนพัลซ์ที่ได้หลังผ่านวงจรแปลง ส่วนรูปที่ 5.4 แสดงรายละเอียดที่ชัดเจนยิ่งขึ้นของสัญญาณทั้งสอง

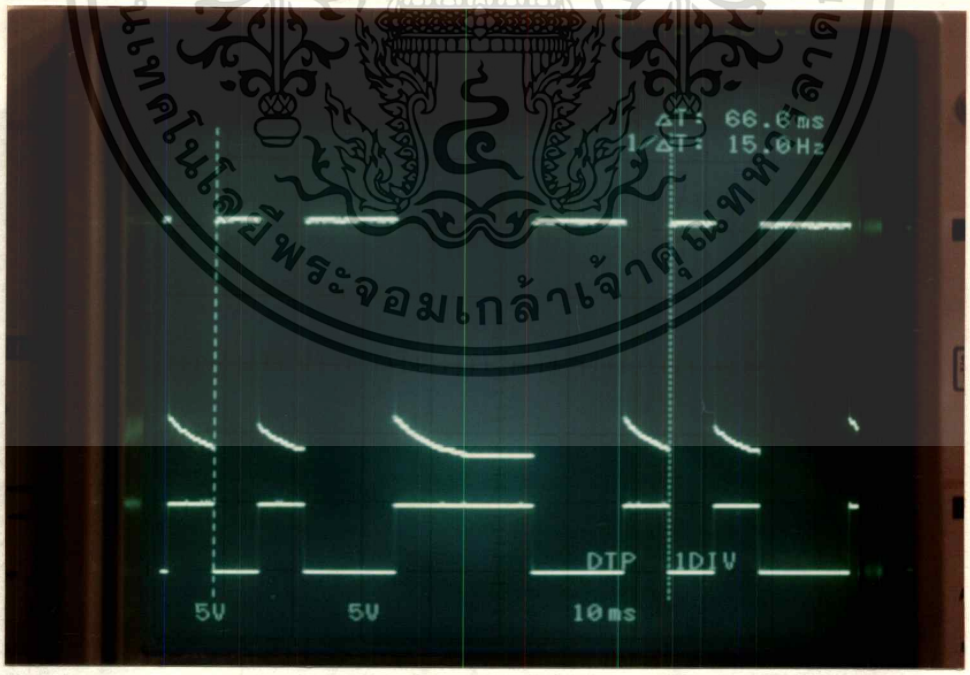
หลังจากนั้นขบวนพัลซ์จะถูกส่งต่อไปเข้าภาคมอดดูเลทได้เป็นสัญญาณพร้อมส่งออกสายอากาศดังรูปที่ 5.5 และรูปที่ 5.6 แสดงรายละเอียดที่ชัดเจนยิ่งขึ้นของสัญญาณเดียวกัน สัญญาณคลื่นวิทยุนี้จะถูกส่งไปเข้ายังภาครับ

ในรูปที่ 5.7 สัญญาณช่องบนเป็นสัญญาณที่รับได้ที่ภาครับและผ่านการแยกสัญญาณเรียบร้อยแล้วโดยสัญญาณนี้จะถูกส่งต่อไปเข้าวงจรแปลงได้ออกมาเป็นสัญญาณช่องล่าง ซึ่งเป็นข้อมูลอนุกรมรหัสแอสกี้ของเลข 9 ที่ถูกแปลงกลับคืนมา และในรูปที่ 5.8 แสดงรายละเอียดที่ชัดเจนยิ่งขึ้นของสัญญาณทั้งสอง

ต่อจากนั้นทำการทดสอบการสื่อสารข้อมูลด้วยข้อความอื่นบ้าง ผลการทดสอบดังตัวอย่างในรูปที่ 5.9 และ 5.10



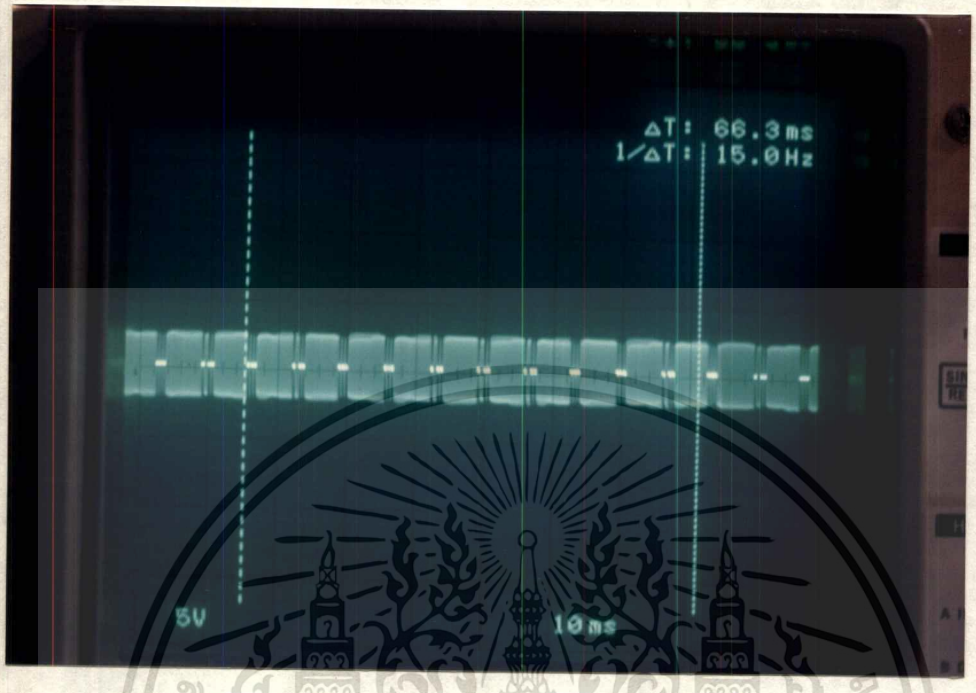
รูปที่ 5.1 การป้อนคำสั่งบนไมโครคอมพิวเตอร์ 286-AT เพื่อทดสอบการสื่อสารข้อมูล



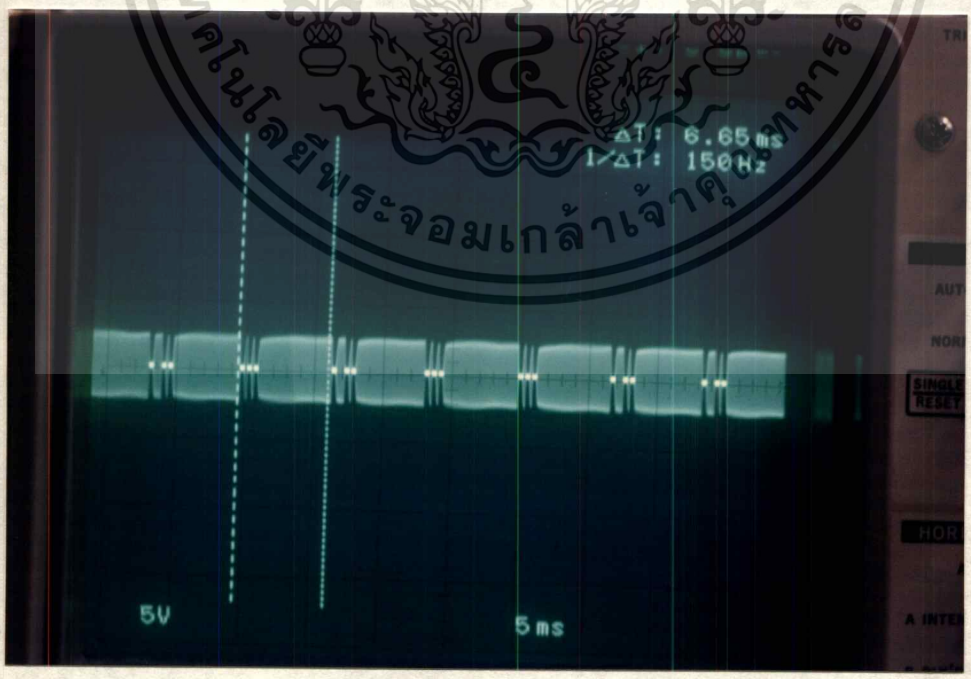
รูปที่ 5.2 ข้อมูลอนุกรมรหัสแอสกี้ของเลข 9

สัญญาณช่องบน : วัดที่พอร์ท RS-232C

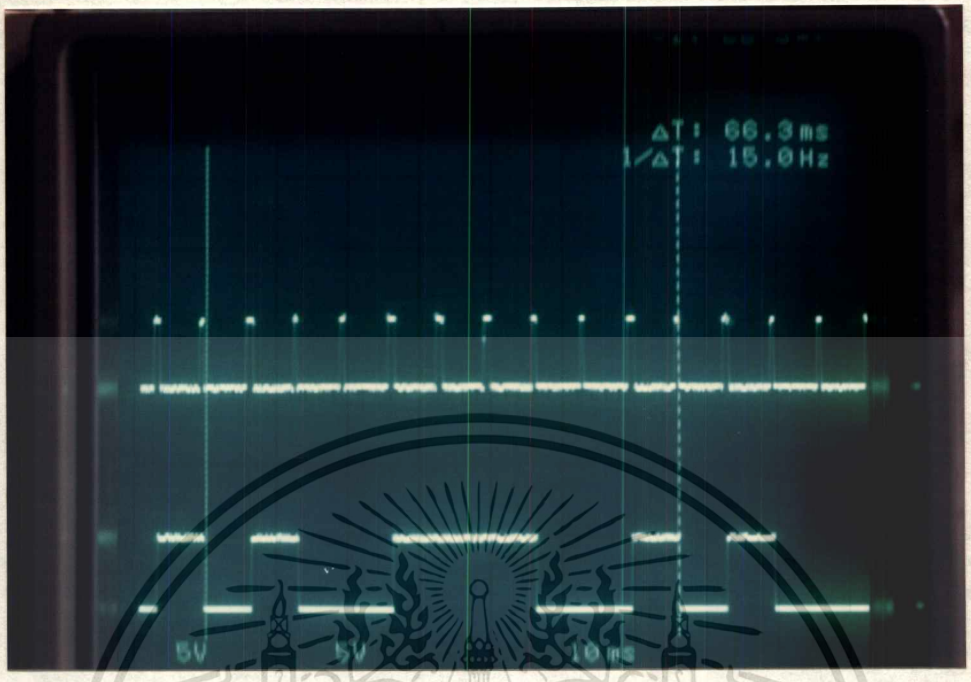
สัญญาณช่องล่าง : วัดก่อนเข้าวงจรแปลง



รูปที่ 5.5 สัญญาณพร้อมส่งออกสายอากาศ



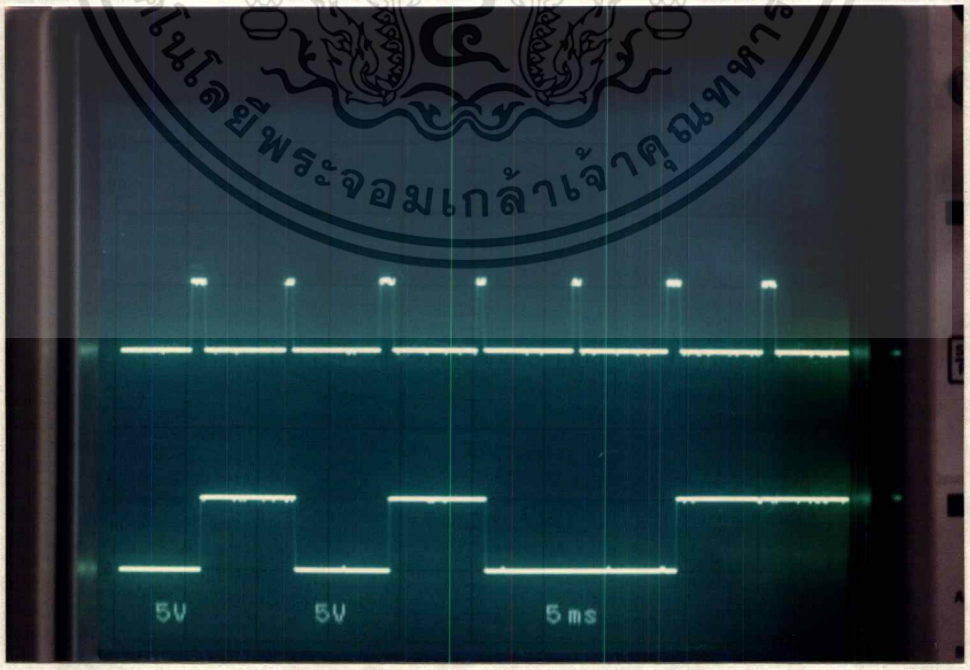
รูปที่ 5.6 รูปขยายของสัญญาณพร้อมส่งออกสายอากาศ



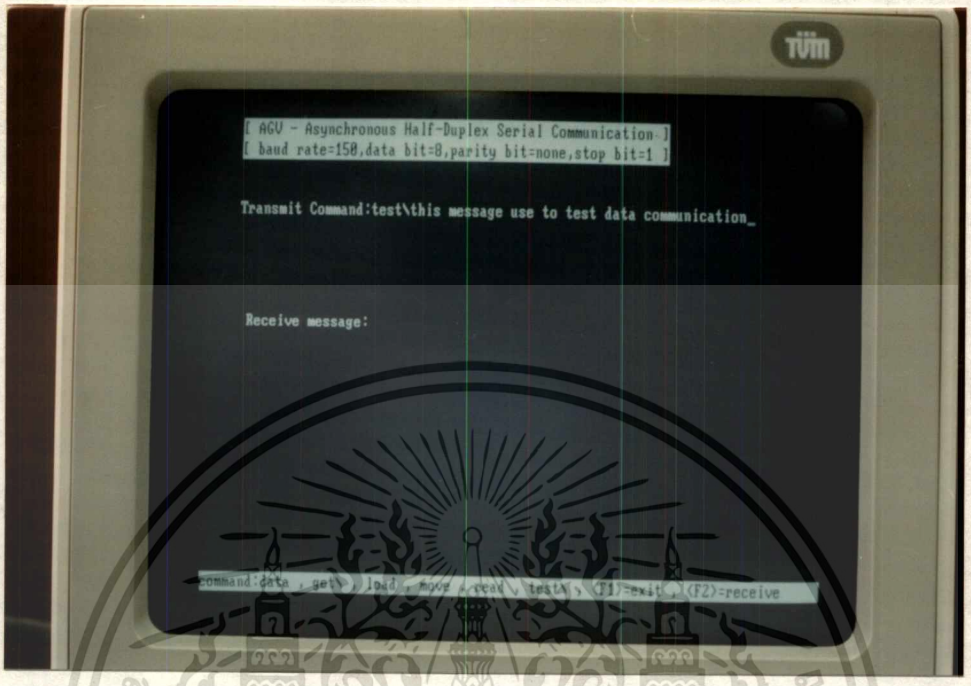
รูปที่ 5.7 สัญญาณที่เครื่องรับหลังผ่านการแยกสัญญาณและข้อมูลอนุกรมที่ถูกแปลงกลับคืนมา

สัญญาณช่องบน: หลังผ่านการแยกสัญญาณ

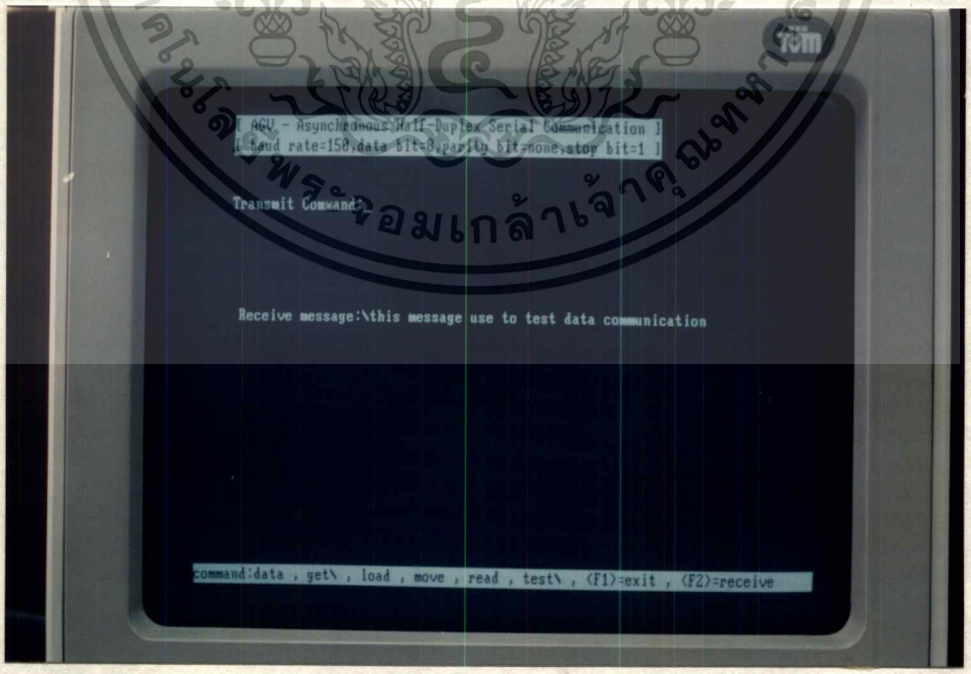
สัญญาณช่องล่าง: ข้อมูลอนุกรมที่แปลงกลับคืนมา



รูปที่ 5.8 รูปขยายสัญญาณที่เครื่องรับหลังผ่านการแยกสัญญาณและข้อมูลอนุกรมที่ถูกแปลงกลับคืนมา



รูปที่ 5.9 การทดสอบการสื่อสารข้อมูลด้วยข้อความอื่น



รูปที่ 5.10 ผลการทดสอบการสื่อสารข้อมูลด้วยข้อความอื่น

5.2 ผลการทดสอบระบบเรียกใช้งานจากสถานีปฏิบัติงาน

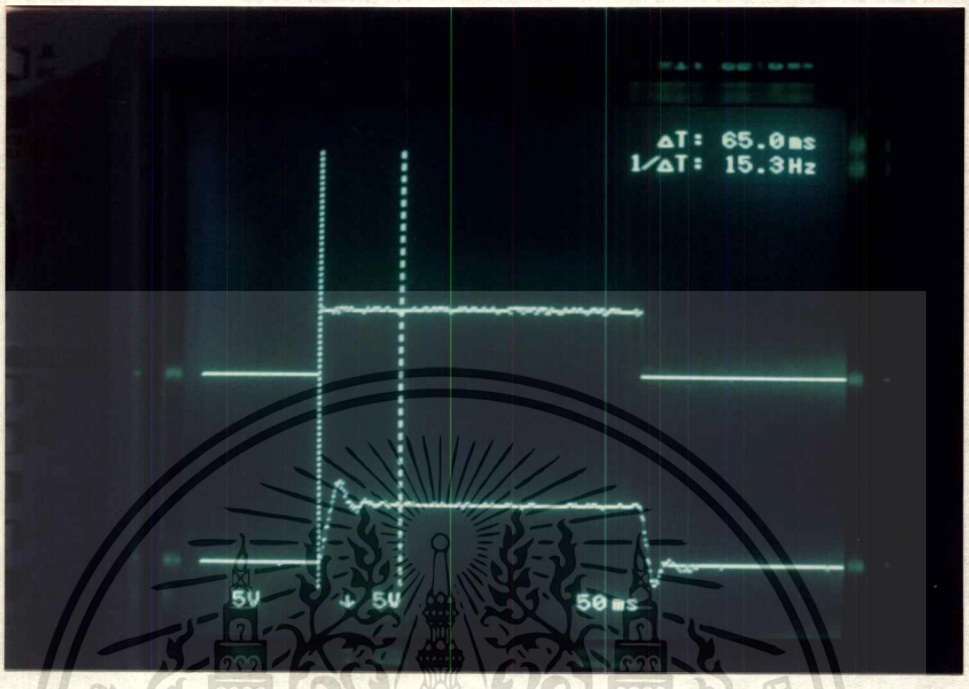
ทำการทดสอบระบบ ด้วยการกดคีย์เรียกจากสถานีปฏิบัติงาน รหัสข้อมูลสถานีนั้นจะถูกส่งเข้ามาเก็บยังไมโครคอมพิวเตอร์ และแสดงผลทางจอภาพดังรูปที่ 5.11 ตามรูปจะเห็นว่ามีการกดคีย์เรียกใช้งานในขณะที่ไมโครคอมพิวเตอร์กำลังปฏิบัติคำสั่ง directory

รูปที่ 5.11 การแสดงผลบนจอภาพเมื่อมีการกดคีย์เรียกใช้งานจากสถานี

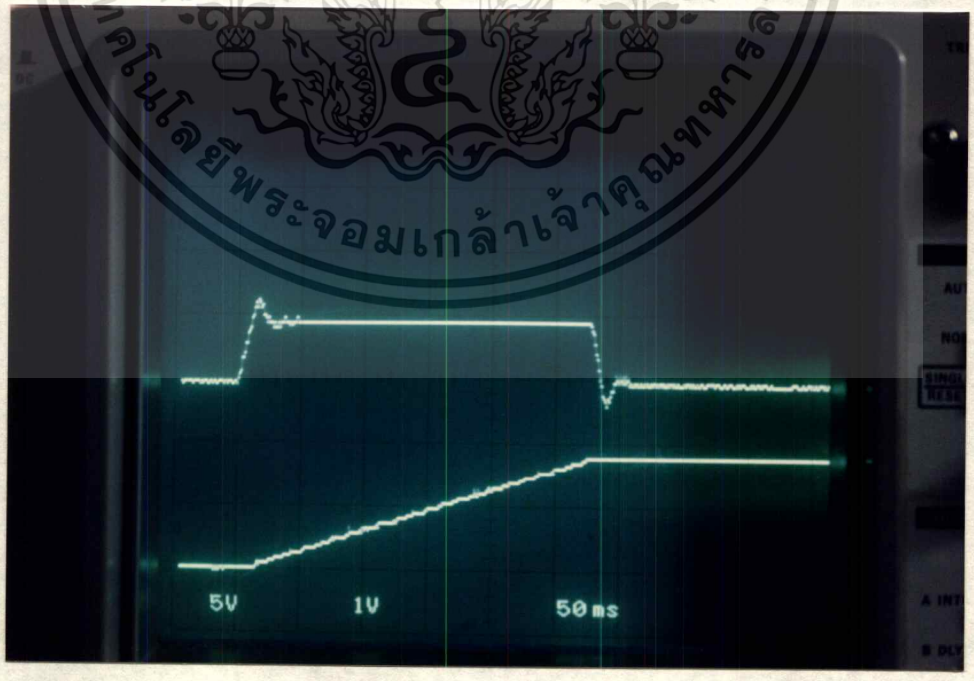
5.3 ผลการทดสอบระบบควบคุมการเคลื่อนที่

ทำการทดสอบระบบควบคุมความเร็วแบบ PI คอนโทรลเลอร์ ด้วยการป้อนสัญญาณสเตปอินพุทให้วงจรโดยตรงแล้ววัดผลตอบสนองทางความเร็วของมอเตอร์จากทาโคมิเตอร์ ผลการทดสอบแสดงดังรูปที่ 5.12 จากนั้นทำการวัดผลตอบสนองทางตำแหน่งของมอเตอร์จากสัญญาณสเตปอินพุทแบบเดียวกันได้ดังรูปที่ 5.13 การวัดตำแหน่งสามารถทำได้โดยการนำค่าเอาท์พุทพัลซ์จากโรตารี่ เอนโคดเตอร์ป้อนเข้าวงจรวัดผลตอบสนองทางตำแหน่งซึ่งมีรายละเอียดดังรูปที่ 5.14

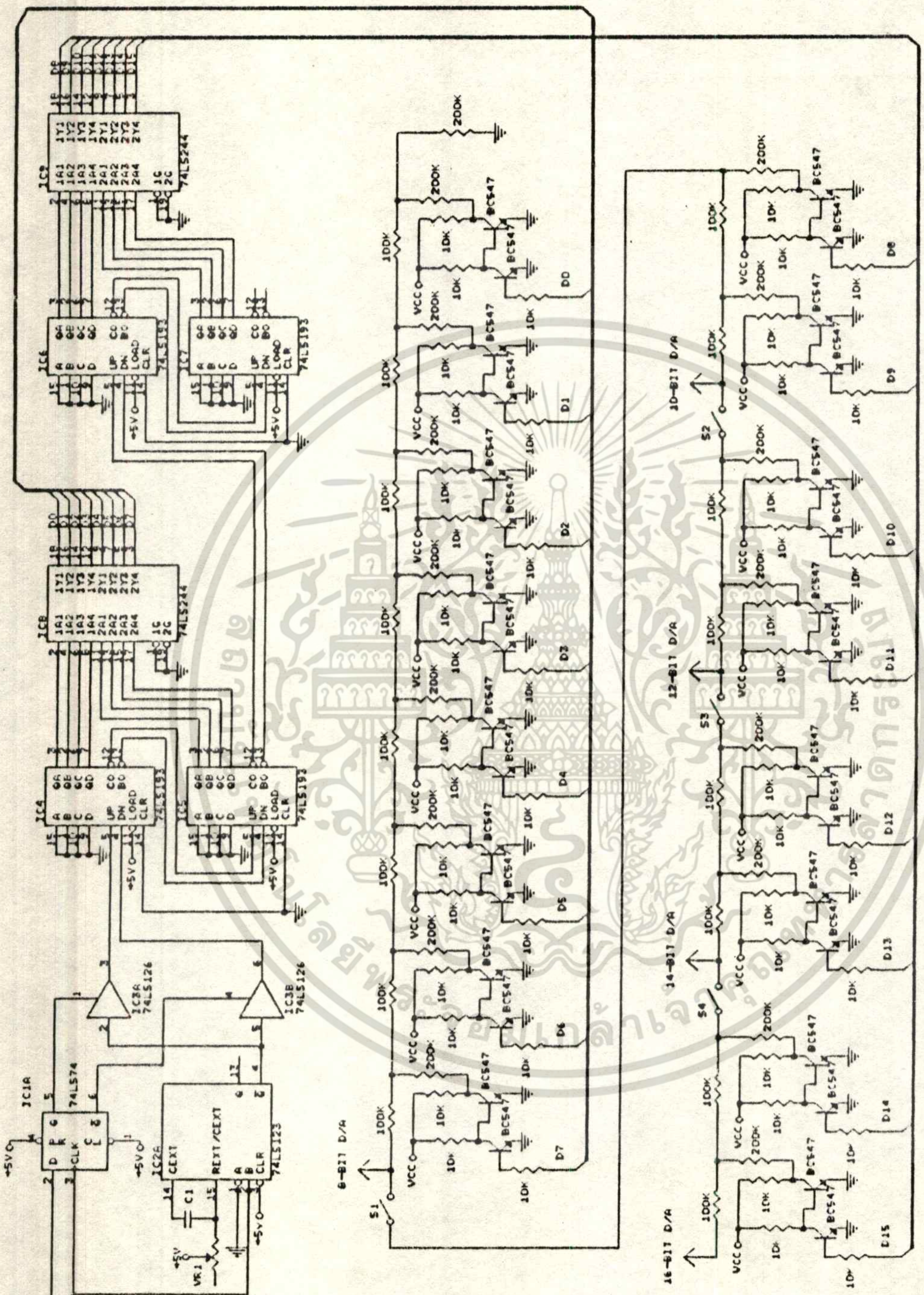
ขั้นต่อไป,ทดสอบระบบควบคุมความเร็วด้วยการเปลี่ยนค่าโหลดโดยการนำมอเตอร์อีกตัวต่อคัปปลิ้งกับเพลามอเตอร์ (shaft) ตัวเดิม ขณะที่มอเตอร์หมุนด้วยความเร็วคงที่ให้ทำการลัดวงจรที่ขั้วอาร์เมเจอร์ของมอเตอร์ตัวที่มาต่อคัปปลิ้ง จากนั้นวัดผลตอบสนองทางความเร็วได้ดังรูปที่ 5.15



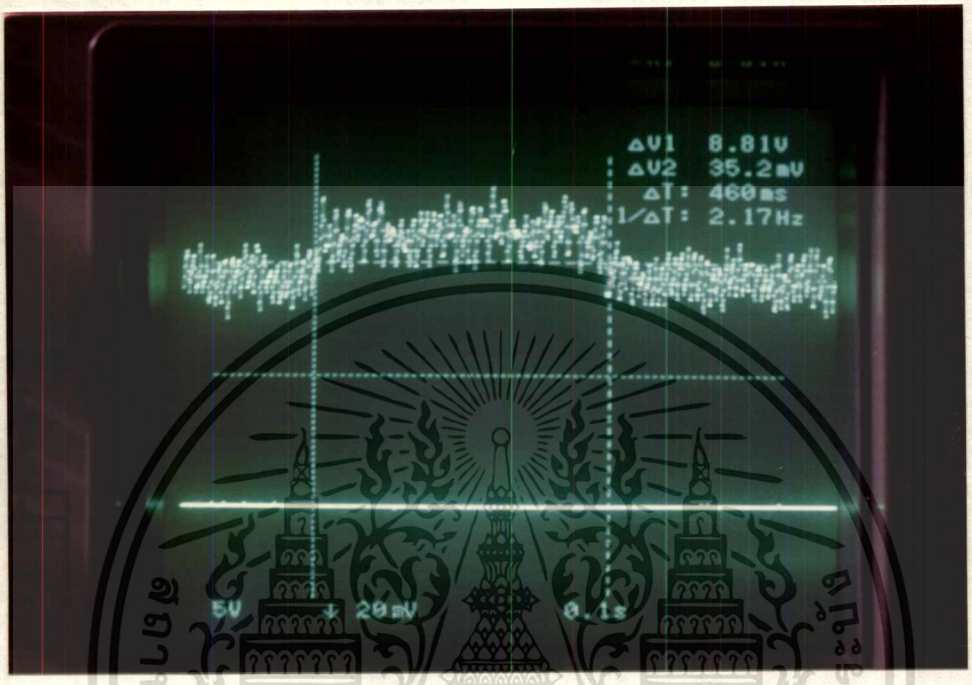
รูปที่ 5.12 ผลตอบสนองทางความเร็วของมอเตอร์กระแสตรงต่อสัญญาณสเตปอินพุท
สัญญาณช่องบน : สเตปอินพุท , สัญญาณช่องล่าง : ความเร็วมอเตอร์



รูปที่ 5.13 ผลตอบสนองทางตำแหน่งของมอเตอร์กระแสตรงต่อสัญญาณสเตปอินพุท
สัญญาณช่องบน : ความเร็วมอเตอร์ , สัญญาณช่องล่าง : ตำแหน่งมอเตอร์



รูปที่ 5.14 วงจรวัดผลตอบสนองทางตำแหน่ง



รูปที่ 5.15 ผลตอบสนองทางความเร็วมอเตอร์กระแสตรงต่อการเปลี่ยนแปลงโหลด
สัญญาณช่องบน: กระแสอาร์เมเจอร์ , สัญญาณช่องล่าง: ความเร็วมอเตอร์

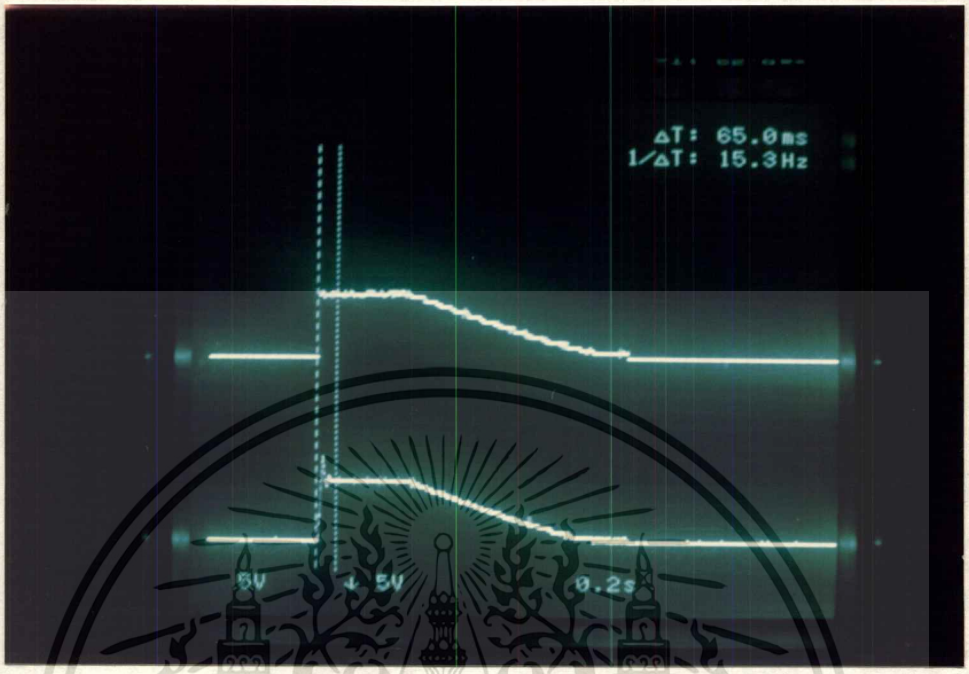
ขั้นตอนต่อ, ทำการทดสอบระบบด้วยการป้อนคำสั่งบนไมโครคอมพิวเตอร์ 286-AT ตามลำดับดังนี้

1) get\p=260,0\

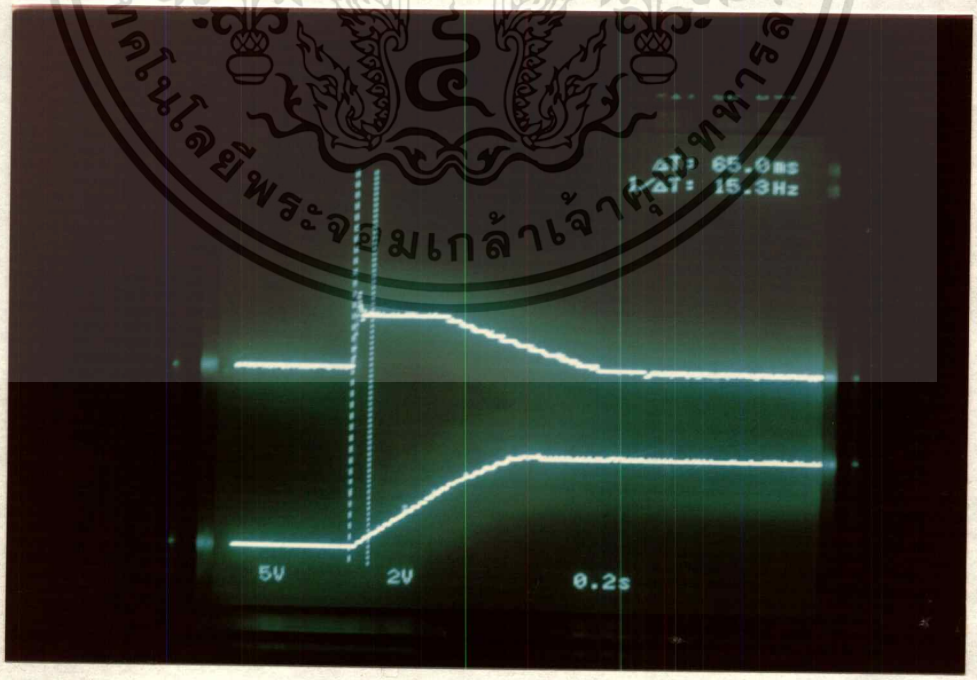
2) load

3) move

ชุดคำสั่งนี้เป็นการสั่งให้มอเตอร์หมุนในทิศตามเข็มนาฬิกา (clockwise) เป็นระยะเชิงมุมเท่าจำนวนพัลซ์ 260 ด้วยความเร็วคงที่แล้วหยุด ทำการวัดผลตอบสนองทางความเร็วของมอเตอร์ได้ดังรูปที่ 5.16 จากนั้นป้อนชุดคำสั่งเดียวกันอีกครั้งแล้วทำการวัดผลตอบสนองทางตำแหน่งได้ดังรูปที่ 5.17



รูปที่ 5.16 ผลตอบสนองทางความเร็วของมอเตอร์กระแสตรงเมื่อป้อนคำสั่งจาก 286-AT
สัญญาณช่องบน: ค่าคำสั่งทางความเร็ว , สัญญาณช่องล่าง: ความเร็วมอเตอร์



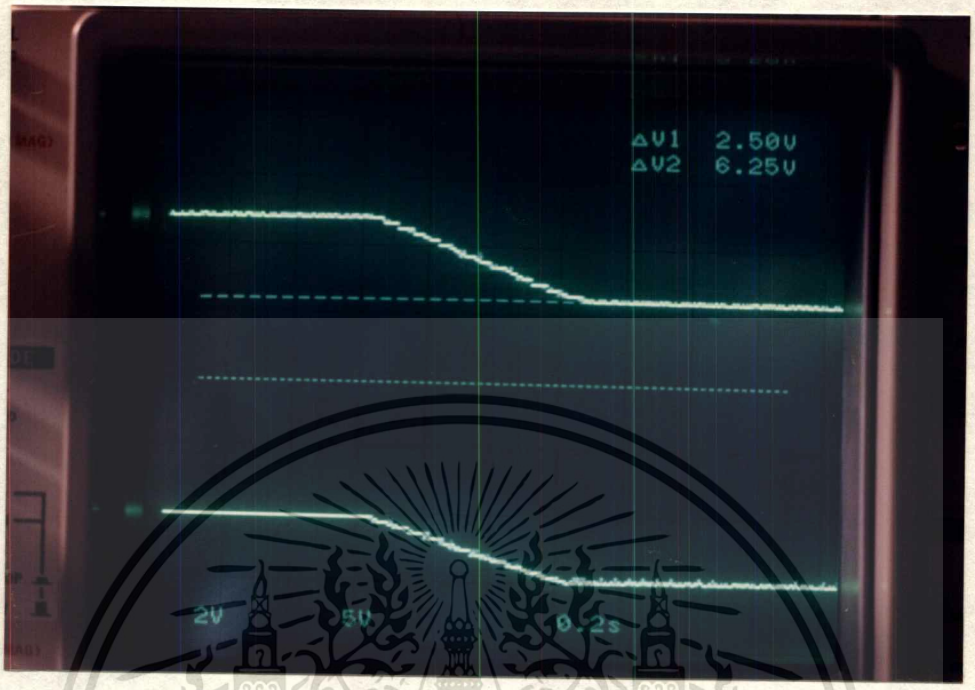
รูปที่ 5.17 ผลตอบสนองทางตำแหน่งของมอเตอร์กระแสตรงเมื่อป้อนคำสั่งจาก 286-AT
สัญญาณช่องบน: ความเร็วมอเตอร์ , สัญญาณช่องล่าง: ตำแหน่งมอเตอร์

5.4 ผลการทดสอบระบบเซนเซอร์

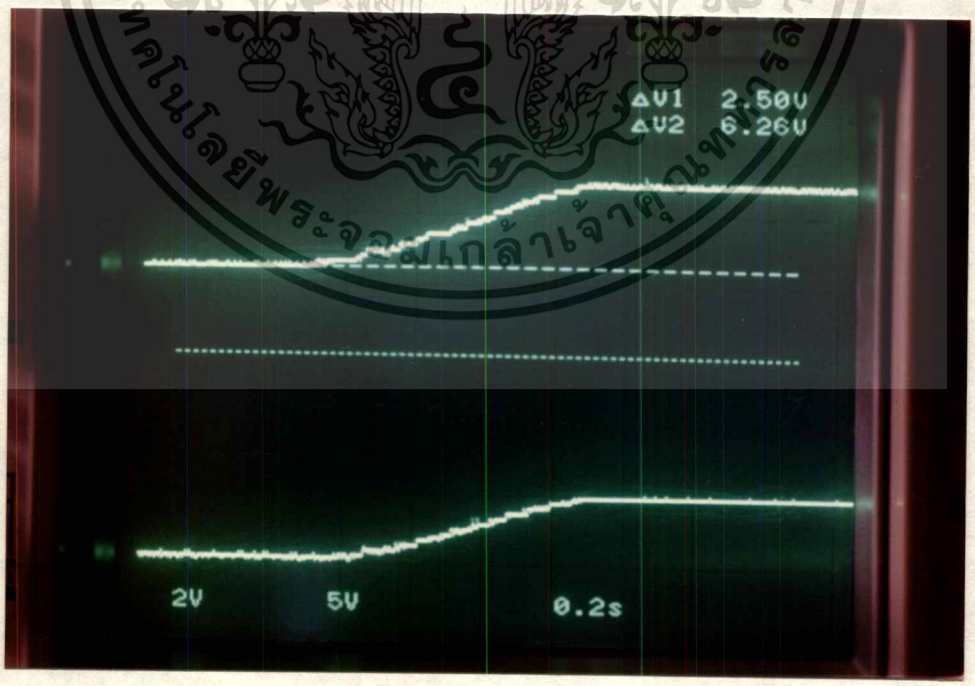
ในการทดสอบระบบตรวจจับวัตถุกีดขวางจะพิจารณาถึงความสัมพันธ์ระหว่างระยะวัตถุกับความเร็วของมอเตอร์ การทดสอบเริ่มด้วยการป้อนสัญญาณอินพุทให้มอเตอร์หมุนด้วยความเร็วคงที่ จากนั้นทำการทดสอบระบบตรวจจับวัตถุกีดขวางด้วยการวางวัตถุกีดขวางในระยะห่างจากตัวอุตร้าโซนิคไม่ต่ำกว่า 1.2 เมตร จากนั้นทำการเลื่อนวัตถุเข้าไปหาตัวอุตร้าโซนิคด้วยความเร็วคงที่ วงจรตรวจจับวัตถุกีดขวางจะให้สัญญาณเอาท์พุทดังรูปที่ 5.18 ตามรูปสัญญาณช่องบนเป็นสัญญาณเอาท์พุทที่ได้จากวงจรตรวจจับ ส่วนสัญญาณช่องล่างเป็นค่าคำสั่งทางความเร็ว (speed command) สำหรับมอเตอร์ที่ได้จากการนำสัญญาณเอาท์พุทจากวงจรตรวจจับส่งต่อไปเข้าชุดวงจรเปรียบเทียบและอนาล็อคสวิทช์ดังได้กล่าวไปแล้วในหัวข้อ 3.3 หน้า 60-61

สำหรับผลตอบสนองทางความเร็วของมอเตอร์กระแสตรงที่มีต่อระยะวัตถุกีดขวางเมื่อเลื่อนวัตถุนั้นเข้าไปใกล้ตัวอุตร้าโซนิคแสดงไว้ในรูปที่ 5.19 ตามรูปสัญญาณช่องบนเป็นค่าคำสั่งทางความเร็วที่แปรผันโดยตรงกับระยะห่างระหว่างวัตถุกีดขวางกับตัวอุตร้าโซนิค ส่วนสัญญาณช่องล่างเป็นผลตอบสนองทางความเร็วของมอเตอร์กระแสตรงที่มีต่อระยะวัตถุนั้น จากนั้นทำการเลื่อนวัตถุกีดขวางห่างออกไปจากตัวอุตร้าโซนิคแล้ววัดผลตอบสนองทางความเร็วได้ดังรูปที่ 5.20

รูปที่ 5.18 สัญญาณเอาท์พุทจากวงจรตรวจจับวัตถุกีดขวางและสัญญาณคำสั่งทางความเร็ว
สัญญาณช่องบน : เอาท์พุทจากวงจรตรวจจับ , สัญญาณช่องล่าง : ค่าคำสั่งทางความเร็ว



รูปที่ 5.19 ผลตอบสนองทางความเร็วของมอเตอร์กระแสตรงเมื่อเลื่อนวัตถุเข้ามาใกล้
สัญญาณช่องบน: ค่าคำสั่งทางความเร็ว : สัญญาณช่องล่าง: ความเร็วมอเตอร์



รูปที่ 5.20 ผลตอบสนองทางความเร็วของมอเตอร์กระแสตรงเมื่อเลื่อนวัตถุห่างออกไป
สัญญาณช่องบน: ค่าคำสั่งทางความเร็ว : สัญญาณช่องล่าง: ความเร็วมอเตอร์

5.5 <u>สรุปผลการทดสอบ</u>

จากการทดสอบระบบสื่อสารด้วยการทดลองส่งเลข 9 ซึ่งมีรหัสแอสกี้คือ 00111001 จากไมโครคอมพิวเตอร์ 286-AT ไปยังไมโครคอนโทรลเลอร์ ข้อมูลอนุกรม 1 เฟรมจะประกอบด้วยบิทเริ่มต้น , บิทข้อมูลที่ 0 - บิทข้อมูลที่ 7 , บิทสิ้นสุด ซึ่งจะตรงกับรหัส 0100111001 ดังรูปที่ 5.2 ข้อมูลอนุกรมจะถูกแปลงเป็นขบวนพัลซ์แล้วมอดดูเลทกับคลื่นพาห์เพื่อส่งออกสายอากาศ ที่ภาครับจะมีการถอดรหัสข้อมูล ซึ่งในที่สุดแล้วก็จะได้ข้อมูลกลับคืนมาดังแสดงในรูปที่ 5.7 , 5.8 และจากรูปที่ 5.9 , 5.10 จะเห็นได้ว่าระบบสื่อสารข้อมูลนี้ทำงานได้อย่างถูกต้อง

การทดสอบระบบเรียกใช้งานจากสถานีปฏิบัติงาน ด้วยการกดคีย์เรียกจากสถานีปฏิบัติงาน ตามรูปที่ 5.11 เป็นการกดคีย์จากสถานี 19 ในขณะที่ไมโครคอมพิวเตอร์กำลังทำคำสั่ง directory อยู่ จะเห็นได้ว่าโปรแกรมควบคุมสามารถถอดรหัสสถานีได้อย่างถูกต้อง

สำหรับการทดสอบระบบควบคุมการเคลื่อนที่ ตามรูปที่ 5.12 เมื่อป้อนสเตปอินพุท PI คอนโทลเลอร์จะควบคุมความเร็วมอเตอร์ให้เข้าสู่ค่าเป้าหมายได้อย่างถูกต้องแม่นยำ โดยมีเซททริ้งไทม์ (settling time) เท่ากับ 65 msec ลักษณะโอเวอชูท (overshoot) ของความเร็วแสดงให้เห็นว่าระบบมีเสถียรภาพ (stability) ส่วนรูปที่ 5.13 แสดงให้เห็นผลตอบสนองทางความเร็วเทียบกับผลตอบสนองทางตำแหน่งเมื่อป้อนสเตปอินพุทแบบเดียวกัน ผลตอบสนองทางตำแหน่งที่วัดได้ตามรูปนี้ ไม่ได้ป้อนกลับให้ 8031-2 แต่อย่างใด สำหรับรูปที่ 5.15 เมื่อเปลี่ยนแปลงค่าโหลดจะเห็นว่าไม่มีผลต่อค่าความเร็วเลย ความจริงแล้วรูปนี้ยังไม่ค่อยชัดเจนเนื่องจากเงื่อนไขในการทดสอบไม่ได้ขับเคลื่อนมอเตอร์ที่พิกัด และเมื่อทำการทดสอบระบบด้วยการป้อนคำสั่งจากไมโครคอมพิวเตอร์ก็จะได้ผลดังรูปที่ 5.16 , 5.17 ในรูปที่ 5.16 เนื่องจากเรากำหนดค่าตำแหน่งเท่ากับ 260 ซึ่งมากกว่า 127 ดังนั้นเมื่อป้อนคำสั่ง move ค่าคำสั่งทางความเร็วจะเท่ากับ 255 ทำให้เหมือนกับการป้อนสเตปอินพุท ผลตอบสนองทางความเร็วในช่วงออกตัวจึงเหมือนรูปที่ 5.12 , 5.13 แต่เนื่องจากในการทดสอบในขั้นนี้มีการป้อนค่าตำแหน่งกลับไปยัง 8031-2 ด้วย ดังนั้นเมื่อผลต่างทางตำแหน่งมีค่าต่ำกว่า 127 มอเตอร์จะเริ่มชะลอความเร็วตามตารางความเร็วที่ 4.1 ซึ่งจะเห็นได้จากรูปที่ 5.16 , 5.17 ว่าความเร็วมอเตอร์ค่อยๆลดลง วิธีนี้ทำให้มอเตอร์หยุดยังตำแหน่งเป้าหมายโดยมีความนุ่มนวลมากขึ้น

ในการทดสอบระบบตรวจจับวัตถุกีดขวางตามรูปที่ 5.18 - 5.20 จะเห็นได้ว่าระยะห่างระหว่างวัตถุกีดขวางกับตัวอุตร้าโซนิคที่เปลี่ยนแปลงไปจะมีผลโดยตรงต่อค่าความเร็วของมอเตอร์ และจากการทดสอบจะเห็นได้ว่าความเร็วของมอเตอร์จะแปรผันโดยตรงกับระยะวัตถุกีดขวาง

บทที่ 6
บทสรุป

6.1 สรุปผลงานวิจัย

วิทยานิพนธ์นี้เป็นการสร้างระบบควบคุม AGV อันประกอบด้วยส่วนต่างๆได้แก่

ก) ระบบสื่อสารข้อมูลระหว่างคอมพิวเตอร์ส่วนกลางกับ AGV ผ่านคลื่นวิทยุ

ใช้วิธีส่งข้อมูลอนุกรมแบบอะซิงโครนัสอาล์ฟดูเพล็กซ์ด้วยอัตราบอด 150 บิท/วินาที ผ่านคลื่นวิทยุแบบ AM ย่านความถี่ 27 MHz จากการทดลองพบว่าระบบสามารถทำงานได้ การแยกกราวน์ของเครื่องรับและเครื่องส่งทุกชุดด้วยออบโตไอโซเลเตอร์ (opto-isolator) ช่วยให้การรับส่งข้อมูลถูกต้องยิ่งขึ้นเนื่องจากเป็นการป้องกันสัญญาณรบกวนที่มาจากเครื่องคอมพิวเตอร์หรือวงจรที่เกี่ยวข้องไม่ให้เข้าไปกวนระบบเครื่องรับ-เครื่องส่ง อย่างไรก็ตามยังไม่ได้มีการทดสอบสมรรถนะของระบบที่มีต่อสัญญาณรบกวน RFI ในระดับต่างๆ อีกทั้งยังไม่ได้มีการทดสอบระยะทางสูงสุดที่สามารถสื่อสารข้อมูลได้โดยไม่เกิดความผิดพลาดของข้อมูล

ข) ระบบเรียกใช้งาน AGV จากพื้นที่ปฏิบัติงาน

การเรียกใช้งานจะใช้วิธีกดคีย์เรียกโดยตรง เนื่องจากวางระบบคีย์บอร์ดเป็นแบบเมตริกซ์ขนาด 8 x 8 ดังนั้นจึงสามารถมีจำนวนสถานีปฏิบัติงานได้ทั้งสิ้น 64 สถานี สัญญาณการกดคีย์จะถูกส่งตามสายแบบกระแสวนลูป ระบบจะทำการอ่านรหัสของสถานีนั้นแล้วส่งสัญญาณอินเตอรัพท์ไปยังไมโครคอมพิวเตอร์ให้มาอ่านข้อมูลเข้าไปเก็บยังหน่วยความจำเพื่อการประมวลผลต่อไป

ค) ระบบควบคุมการเคลื่อนที่ของ AGV

การขับเคลื่อนจะใช้มอเตอร์กระแสตรงซึ่งมีการควบคุมตำแหน่งโดยใช้ไมโครคอนโทรลเลอร์ 8031 นับค่าผิดพลาดทางตำแหน่งแล้วประมวลผลหาค่าความเร็วที่เหมาะสมออกมา ค่าอินพุทของตำแหน่งเป้าหมายสามารถป้อนได้โดยตรงจากไมโครคอมพิวเตอร์ โดยสามารถกำหนดได้ตั้งแต่ 0 ถึง 65535 พัลซ์ ในขณะที่มอเตอร์ยังหมุนไม่ถึงตำแหน่งเป้าหมาย ความเร็วของมอเตอร์จะคงที่ที่ค่าความเร็วค่าหนึ่ง ซึ่งสามารถปรับค่าความเร็วได้จากวงจรควบคุมความเร็ว ความเร็วมอเตอร์นี้ถูกควบคุมด้วยอนาล็อค PI คอนโทรลเลอร์ เมื่อระยะตำแหน่งเป้าหมายลดลงต่ำกว่า +127 มอเตอร์จะชะลอความเร็วลงโดยเป็นไปตามสมการ (4.1),(4.2) จากการทดสอบผลตอบสนองทางตำแหน่งในบทที่ 5 พบว่าสามารถควบคุมให้มอเตอร์หยุดยังตำแหน่งเป้าหมายได้อย่างถูกต้องแม่นยำ ด้านผลตอบสนองทางความเร็วที่มีต่อสัญญาณอินพุท จากการทดสอบแสดงให้เห็นว่าระบบมีเสถียรภาพ

ง) ระบบเซนเซอร์สำหรับ AGV

ระบบเซนเซอร์ต่างๆที่สร้างขึ้นสามารถนำไปประยุกต์ใช้ใน AGV ที่จะมีการสร้างต่อไป ระ

บบ์เซนเซอร์ต่างๆที่สร้างขึ้นได้แก่ ระบบเซนเซอร์โดยใช้โฟโต้เซนเซอร์ , ระบบตรวจจับวัตถุกีดขวาง , ระบบตรวจจับแรงดันแบตเตอรี่ การทดสอบการตรวจจับวัตถุกีดขวางโดยอุตร้าโซนิคพบว่าสามารถตรวจจับวัตถุได้ในระยะประมาณ 1.2 เมตร เมื่อระบบตรวจจับวัตถุกีดขวางความเร็วของมอเตอร์จะแปรผันโดยตรงกับระยะวัตถุ และมอเตอร์จะหยุดเมื่อวัตถุอยู่ห่างจากอุตร้าโซนิคเซนเซอร์ประมาณ 30 เซ็นติเมตร อย่างไรก็ตามถ้านำระบบนี้ไปใช้กับงานจริง AGV จะต้องหยุดคอยวัตถุและจะเคลื่อนที่ต่อไปได้เมื่อวัตถุกีดขวางนั้นเคลื่อนพ้นออกไปเท่านั้น ปัจจุบันในต่างประเทศจึงได้มีการวิจัยและพัฒนาเพื่อให้ AGV มีความฉลาดพอที่จะหลบหลีกสิ่งกีดขวางนั้นเป็นสิ่งที่จะต้องมีการวิจัยและพัฒนาต่อไป

จ) โปรแกรมควบคุม จะมีอยู่สองส่วนคือ โปรแกรมควบคุมในส่วนของไมโครคอมพิวเตอร์ และโปรแกรมควบคุมในส่วนไมโครคอนโทรลเลอร์ซึ่งจะติดตั้งบนตัว AGV โปรแกรมควบคุมในส่วนไมโครคอมพิวเตอร์ประกอบด้วยโปรแกรมสื่อสารข้อมูล , โปรแกรมรับการเรียกจากสถานีปฏิบัติงาน ส่วนโปรแกรมควบคุมในส่วนไมโครคอนโทรลเลอร์จะประกอบด้วยโปรแกรมสื่อสารข้อมูล, โปรแกรมควบคุมการเคลื่อนที่ และโปรแกรมปฏิบัติตามคำสั่งต่างๆ ที่ส่งมาจากไมโครคอมพิวเตอร์

การใช้ไมโครคอมพิวเตอร์มาควบคุมการทำงานของ AGV นั้น มีข้อดีคือทำให้เราสามารถพัฒนาการควบคุมการทำงาน ตลอดจนดัดแปลงแก้ไขโปรแกรมการเคลื่อนที่ของ AGV ได้โดยง่าย ทำให้ระบบ AGV มีความยืดหยุ่นมากขึ้น การเพิ่มจำนวน AGV ในโรงงานสามารถทำได้อย่างมีประสิทธิภาพ ไม่ก่อให้เกิดการวิ่งที่ซ้ำซ้อนกัน ทำให้ระบบการขนถ่ายวัสดุด้วย AGV มีประสิทธิภาพมากขึ้น

ในเรื่องของเทคนิคที่ใช้ในการนำร่องด้วย การตรวจหาตำแหน่งเทปสีดำบนพื้นด้วยแสงอินฟราเรดนั้นมีข้อดีคือต้นทุนการติดตั้งต่ำกว่าเทคนิคอื่น การเปลี่ยนแปลงแก้ไขแผนผังของเส้นทางกระทำได้โดยง่าย มีความยืดหยุ่นสูง แต่อาจมีข้อเสียอยู่บ้างก็ตรงที่เส้นทางมีโอกาสชำรุดเสียหายได้มาก อย่างไรก็ตาม การบำรุงรักษาก็มิใช่เรื่องที่ต้องลงทุนมากแต่อย่างใด

6.2 ข้อเสนอแนะ

ก) การสื่อสารข้อมูลผ่านคลื่นวิทยุ มีโอกาสถูกรบกวนได้จากสัญญาณ RFI ดังนั้นควรหลีกเลี่ยงบริเวณที่มีการใช้งานคลื่นความถี่วิทยุเดียวกัน หรือบริเวณที่มีตัวกำเนิดสัญญาณ RFI ซึ่งจะทำให้การส่งข้อมูลผิดพลาดไป ในทางปฏิบัติจริงจะต้องมีการพัฒนาระบบอีกมากเช่น ควรเปลี่ยนไปใช้ย่านความถี่ที่สูงขึ้นเพราะจะทำให้การสื่อสารข้อมูล ถูกรบกวนจากสัญญาณ RFI น้อยลง

ข) กำลังของเครื่องส่งจะต้องแรงพอที่จะสามารถควบคุม AGV ได้ครอบคลุมพื้นที่ปฏิบัติงาน

ค) โปรแกรมรับส่งข้อมูลอนุกรมควรเปลี่ยนไปใช้วิธีอินเตอรัพท์แทนวิธีโพลลิ่งเพราะจะทำให้ AGV สามารถติดต่อกับคอมพิวเตอร์ส่วนกลางได้อย่างมีประสิทธิภาพมากขึ้น

ง) ในการพัฒนา AGV ในขั้นต่อไปสามารถดูหลักการได้จากภาคผนวก ข

เอกสารอ้างอิง

[1] T.Takahashi , **"Automated Guided Vehicle Systems"** , Proc of the 5TH Inter Conference , IFS (Conferences) Ltd and authors , october 1987.

[2] R.H. Hollier and L.F. Gelders , **"Automated Guided Vehicle Systems"** , Proc of the 6TH Inter Conference , IFS Ltd and authors , October 1988.

[3] D.J Todd , **"Fundamentals of Robot Technology"** , Kogan page Ltd , 1986.

[4] ชูชัย ธนสารตั้งเจริญ , ทินกร ดุ๊ก , **"การสื่อสารข้อมูล"** , ฟิสิกส์เซ็นเตอร์ การพิมพ์.

[5] ยืน ภู่วรวรรณ ,**"เทคโนโลยี ฮาร์ดแวร์ IBM PC"**, บริษัท ซีเอ็ดยูเคชั่น จำกัด , 2533.

[6] สมชาย สุขสวัสดิ์ และเมธี ฉัตรทอง , **"คู่มือวิทยุคมนาคม"** , โรงพิมพ์ภัทรการพิมพ์ , 2534.

[7] โยชิน เปรมปราณีรัชต์, **"ระบบเซอร์โวและอีเลคทรอนิคส์คอนโทรลมอเตอร์"** ,JICA, 2533.

[8] Takashi Kenjo ,**"Power Electronics for the Microprocessor Age"** ,Oxford University press , 1990.

[9] Electro-craft corporation , **"DC Motors Speed Controls Servo Systems"**, 1972.

[10] Steeds W., **"Mechanics of Road Vehicle"** , ILiffe & Sons Ltd., London, 1960.

[11] Gordon Mccomb , **"The Robot Builder's Bonanza 99 Inexpensive Robotics Projects"**, Gordon Mccomb USA , 1987.

[12] Kenneth J.Ayala , **"The 8051 Microcontroller Architecure programming and Application"** , West Publishing Company , 1991.

[13] Ray Duncan , **"Advanced MS-DOS Programming"** , microsoft press , 1986.

[14] Thomas A.Wadlow,**"Memory Resident Programming on the IBM PC"**, Addison-Wesley Publishing Company Inc , U.S.A , 1987.

[15] Robert Jourdain, **"Programmer's Problem Solver for the IBM PC,XT,AT"**, Brady Newyork , 1986.

[16] Harry H.Poole , **"Fundamentals of Robotica Engineering"** , Van Nostrand Reinhold Newyork , 1989.

[17] Gordon M.Mair ,**"Industrial Robotics"**, Prentice Hall Inter Ltd.,1988.

ภาคผนวก ก

รายละเอียดโปรแกรมควบคุมโดยสมบูรณ์

โปรแกรมสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์

```
;*************************************************************
;*   AGV-ASYNCHRONOUS HALF-DUPLEX COMMUNICATION              *
;*   use bios interrupt routine int14h at com1               *
;*   baud rate=150,data bit=8 ,parity bit=none,stop bit=1    *
;*************************************************************
          .286c
cseg      segment public
          assume  cs:cseg ,ds:cseg
          public  main
          org     0100h                     ;prepare psp area
  main:   jmp     initial
;Variable and Data Declaration
TRANS_MESSAGE     db  100  dup(?)
RECEIVE_MESSAGE   db  100  dup(?)
STORE     db      ?
TXCOUNT   dw      ?
RXCOUNT   dw      ?
MESSAGE1  db '     [ AGV - Asynchronous Half-Duplex Serial Communication ]',10,13
          db '     [ baud rate=150,data bit=8,parity bit=none,stop bit=1 ]',10,10,10,13
          db '     Transmit Command:$'
MESSAGE2  db ' Receive message:$'
MESSAGE3  db 'command:data , get\ , load , move , read , test\ , <F1>=exit , <F2>=receiv
SPACE     db '                                                                $'
ERROR1    db '                    break interrupt!
ERROR2    db '                    framming error!
ERROR3    db '                    over run error!
ERROR4    db '                    unknown error!
initial:  push  cs
          pop   ds                          ;define ds
;change to page 1
          mov   ax , 0501h
          int   10h
;clear screen
          mov   ax , 0600h
          mov   cx , 0
          mov   dx , 184fh
          mov   bh , 07
          int   10h
;set attribute of screen
          mov   ax , 0600h
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์(ต่อ)

```
        mov   cx , 0005h
        mov   dx , 013bh
        mov   bh , 70h
        int   10h
        mov   cx , 1800h
        mov   dx , 184fh
        int   10h
;set cursor position
        mov   ah , 02
        mov   dx , 0
        mov   bh , 01
        int   10h
;display message1 to screen
        mov   ah , 09
        mov   dx , offset MESSAGE1
        int   21h
;set cursor position
        mov   ah , 02
        mov   dx , 0a05h
        mov   bh , 01
        int   10h
;display message2 to screen
        mov   ah , 09
        mov   dx , offset MESSAGE2
        int   21h
;set cursor position
        mov   ah , 02
        mov   dx , 1800h
        mov   bh , 01
        int   10h
;display message3 to screen
        mov   ah , 09
        mov   dx , offset MESSAGE3
        int   21h
receive: mov  di , 0
        mov   al , 0
        mov   RECEIVE_MESSAGE[di] , al
;set cursor position
        mov   ah , 02
        mov   dx , 0a16h
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์(ต่อ)

```
          mov   bh , 01
          int   10h
;-------------------------------------------------------------------------------;
;       port status (ah)                    modem status (al)                   ;
; bit7=1 ;time out error          bit7=1 ;receive line signal detected          ;
; bit6=1 ;shift register empty    bit6=1 ;ring indicator                        ;
; bit5=1 ;holding register empty  bit5=1 ;data set ready                        ;
; bit4=1 ;break detected          bit4=1 ;clear to send                         ;
; bit3=1 ;framing error           bit3=1 ;channge to receive line signal detected ;
; bit2=1 ;parity error            bit2=1 ;trailing edge ring indicator          ;
; bit1=1 ;overrun error           bit1=1 ;change in data set ready status       ;
; bit0=1 ;data ready              bit0=1 ;change in clear to send status        ;
;-------------------------------------------------------------------------------;
;---------------------------------;
;        initialize com1          ;
; return:ah = port status         ;
;        al = modem status        ;
;---------------------------------;
          mov   ah , 0                  ;function 0 = configure comm port
          mov   al , 00100011b          ;150,8,n,1
          mov   dx , 0                  ;use port com1
          int   14h
          mov   di , 0
;----------------------------;
;        read status         ;
; return:ah = port status    ;
;        al = modem status   ;
;----------------------------;
   wait:  mov   ah , 3                  ;function 3 = read status
          mov   dx , 0                  ;use port com1
          int   14h
          and   ah , 00000001b          ;data ready yet?
          jz    wait                    ;no keep checking
;-------------------------------------------------;
;        receive character                        ;
; return:if successful, ah bit 7 = 0 ; al = character ;
;        if failed    , ah bit 7 = 1 ; al = unchanged ;
;                       ah bit 0-6 = port status  ;
;-------------------------------------------------;
          mov   ah , 2                  ;function 2 = receive character
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์(ต่อ)

```
        mov   dx , 0                    ;use port com1
        int   14h
        test  ah , 10000000b            ;check for error
        jnz   error
        cmp   di , 0
        jne   n1
        cmp   al , '\'
        jne   wait
    n1: mov   RECEIVE_MESSAGE[di] , al  ;receive suceed
        cmp   al , 13
        je    transmit
;display receive message to screen
        mov   RXCOUNT , di
        mov   bl , byte ptr ds:RXCOUNT
        add   bl , 22
        mov   ah , 2
        mov   dh , 10
        mov   dl , bl
        mov   bh , 1
        int   10h
        mov   dl , RECEIVE_MESSAGE[di]
        mov   ah , 2
        int   21h
        inc   di
        jmp   wait
 error: mov   STORE , ah
;set cursor position
        mov   ah , 2
        mov   dx , 1800h
        mov   bh , 1
        int   10h
        mov   ah , STORE
;check break interrupt
        test  ah , 00010000b
        jz    check2
        mov   ah , 9
        mov   dx , offset ERROR1
        int   21h
        jmp   transmit
;check framming error
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์(ต่อ)

```
check2: test  ah , 00001000b
        jz    check3
        mov   ah , 9
        mov   dx , offset ERROR2
        int   21h
        jmp   transmit
;check overrun error
check3: test  ah , 00000010b
        jz    check4
        mov   ah , 9
        mov   dx , offset ERROR3
        int   21h
        jmp   transmit
check4: mov   ah , 9
        mov   dx , offset ERROR4
        int   21h
;set cursor position and clear transmit command
transmit: mov   ah , 02
        mov   dx , 0516h
        mov   bh , 01
        int   10h
        mov   ah , 09
        mov   dx , offset SPACE
        int   21h
;clear index
        mov   di , 0
        mov   si , 0
        mov   al , '\'
        mov   TRANS_MESSAGE[si] , al
        mov   si , 1
;set cursor position at transmit command line
        mov   ah , 02
        mov   dx , 0416h
        mov   bh , 01
        int   10h
;wait for transmit message
loop2:  mov   ah , 07                   ;wait a character from keyboard
        int   21h
;store data in memory
        mov   TRANS_MESSAGE[si] , al
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์(ต่อ)

```
            mov   al , TRANS_MESSAGE[si]
            cmp   al , 13                   ;return?
            je    send
            cmp   al , 0                    ;extended code?
            je    extcode
  ;display character in dl to screen
            mov   dl , al
            mov   ah , 02
            int   21h
  ;increase index si,di
            inc   si
  ;check first character
            cmp   si , 2
            jne   loop2
  ;set cursor position and clear receive message
            mov   ah , 02
            mov   dx , 0a16h
            mov   bh , 01
            int   10h
            mov   ah , 09
            mov   dx , offset SPACE
            int   21h
            mov   ah , 02
            mov   dx , 0b00h
            int   10h
            mov   ah , 09
            mov   dx , offset SPACE
            int   21h
 ;set cursor position
            mov   ah , 02
            mov   dx , 0417h
            mov   bh , 01
            int   10h
            jmp   loop2
extcode:    int   21h
            cmp   al , 59               ;<F1>?
            jne   next
            jmp    quit
  next:     cmp   al , 60               ;<F2> ?
            jne   loop2
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์(ต่อ)

```
;set cursor position and clear receive message
          mov   ah , 02
          mov   dx , 0a16h
          mov   bh , 01
          int   10h
          mov   ah , 09
          mov   dx , offset SPACE
          int   21h
          mov   ah , 02
          mov   dx , 0b00h
          int   10h
          mov   ah , 09
          mov   dx , offset SPACE
          int   21h
          jmp   receive
;set cursor position at transmit command line
   send:  mov   ah , 02
          mov   dx , 0416h
          mov   bh , 01
          int   10h
;clear current line
          mov   ah , 09
          mov   dx , offset SPACE
          int   21h
;-----------------------------------------------;
;                   read status                 ;
; return:ah = port status  , al = modem status  ;
;-----------------------------------------------;
          mov   si , 0                      ;index=0
  wait1:  mov   ah , 3                      ;function 3 = read status
          mov   dx , 0                      ;use port com1
          int   14h
          test  ah , 00100000b              ;register holding empty?
          jz    wait1                       ;no keep checking
;-------------------------------------------------------;
;                   send character                      ;
; return:if successful , ah bit 7 = 0 ; al = unchanged  ;
;        if failed     , ah bit 7 = 1 ; al = unchanged  ;
;                      , ah bit 0-6 = port status       ;
;-------------------------------------------------------;
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์(ต่อ)

```
          mov   ah , 1                    ;function 1 = send character
          mov   dx , 0                    ;use port com1
  loop3:  mov   al , TRANS_MESSAGE[si]    ;register al = character
          int   14h
          test  ah , 10000000b
          jz    next1
          jmp   error
  next1:  cmp   al, 13                    ;end code = return ?
          jne   next2
          jmp   receive
;display character to screen
  next2:  mov   TXCOUNT , si
          mov   bl , byte ptr ds:TXCOUNT
          add   bl , 22
          mov   ah , 2
          mov   dh , 5
          mov   dl , bl
          mov   bh , 1
          int   10h
          mov   dl , TRANS_MESSAGE[si]
          mov   ah , 2
          int   21h
          inc   si
          jmp   wait1
   quit:  mov   ax , 0600h
          mov   cx , 0
          mov   dx , 184fh
          mov   bh , 07
          int   10h
;set cursor position
          mov   ah , 02
          mov   dx , 0
          mov   bh , 0
          int   10h
;return to page 0
          mov   ax , 0500h
          int   10h
          int   20h
cseg      ends
          end   main
```

โปรแกรมรับการกดคีย์จากสถานีปฏิบัติงาน

```
;*******************************************;
; program receive keyword from workstation  ;
;*******************************************;
                cpu "8051.tbl"
                hof "INT8"
A:      equ     0e0h    ;Accumulator
B:      equ     0f0h    ;B Register
PSW:    equ     0d0h    ;Program Status Word
SP:     equ      81h    ;Stack Pointer
DPL:    equ      82h    ;Data Pointer (low byte)
DPH:    equ      83h    ;Data Pointer (high byte)
P0:     equ      80h    ;Port 0
P1:     equ      90h    ;Port 1
P2:     equ     0a0h    ;Port 2
P3:     equ     0b0h    ;Port 3
IP:     equ     0b8h    ;Interrupt Priority Control
IE:     equ     0a8h    ;Interrupt Enable Control
TMOD:   equ      89h    ;Timer/Counter Mode Control
TCON:   equ      88h    ;Timer/Counter Control
TH0:    equ      8ch    ;Timer/Counter 0 (high byte)
TL0:    equ      8ah    ;Timer/Counter 0 (low byte)
TH1:    equ      8dh    ;Timer/Counter 1 (high byte)
TL1:    equ      8bh    ;Timer/Counter 1 (low byte)
SCON:   equ      98h    ;Serial Control
SBUF:   equ      99h    ;Serial Data Buffer
PCON:   equ      87h    ;Power Control
        ;Accumulator
A_0:    equ     0e0h
A_1:    equ     0e1h
A_2:    equ     0e2h
A_3:    equ     0e3h
A_4:    equ     0e4h
A_5:    equ     0e5h
A_6:    equ     0e6h
A_7:    equ     0e7h
        ;B Register
B_0:    equ     0f0h
B_1:    equ     0f1h
B_2:    equ     0f2h
B_3:    equ     0f3h
```

โปรแกรมรับการกดคีย์จากสถานีปฏิบัติงาน(ต่อ)

```
B_4:        equ     0f4h
B_5:        equ     0f5h
B_6:        equ     0f6h
B_7:        equ     0f7h
        ;Program Status Word
P:          equ     0d0h        ;Parity flag
OV:         equ     0d2h        ;Over flow flag
RS0:        equ     0d3h        ;Register bank selector bit 0
RS1:        equ     0d4h        ;Register bank selector bit 1
F0:         equ     0d5h        ;flag 0 ( general purpose )
CY:         equ     0d6h        ;Carry flag
        ;Port 0
P0_0:       equ      80h
P0_1:       equ      81h
P0_2:       equ      82h
P0_3:       equ      83h
P0_4:       equ      84h
P0_5:       equ      85h
P0_6:       equ      86h
P0_7:       equ      87h
        ;Port 1
P1_0:       equ      90h
P1_1:       equ      91h
P1_2:       equ      92h
P1_3:       equ      93h
P1_4:       equ      94h
P1_5:       equ      95h
P1_6:       equ      96h
P1_7:       equ      97h
        ;Port 2
P2_0:       equ     0a0h
P2_1:       equ     0a1h
P2_2:       equ     0a2h
P2_3:       equ     0a3h
P2_4:       equ     0a4h
P2_5:       equ     0a5h
P2_6:       equ     0a6h
P2_7:       equ     0a7h
        ;Port 3
P3_0:       equ     0b0h
```

โปรแกรมรับการกดคีย์จากสถานีปฏิบัติงาน(ต่อ)

```
P3_1:       equ     0b1h
P3_2:       equ     0b2h
P3_3:       equ     0b3h
P3_4:       equ     0b4h
P3_5:       equ     0b5h
P3_6:       equ     0b6h
P3_7:       equ     0b7h
        ;Interrupt Priority Control
PX0:        equ     0b8h        ;External Interrupt 0 priority level
PT0:        equ     0b9h        ;Timer 0 Interrupt priority level
PX1:        equ     0bah        ;External Interrupt 1 priority level
PT1:        equ     0bbh        ;Timer 1 Interrupt priority level
PS:         equ     0bch        ;Serial Port Interrupt priority level
        ;Interrupt Enable Control
EX0:        equ     0a8h        ;External Interrupt 0
ET0:        equ     0a9h        ;Timer 0 overflow interrupt
EX1:        equ     0aah        ;External Interrupt 1
ET1:        equ     0abh        ;Timer 1 overflow interrupt
ES:         equ     0ach        ;Serial Port interrupt
EA:         equ     0afh        ;Enable All Interrupts
        ;Timer/Counter Control
IT0:        equ      88h        ;Interrupt 0 type control bit
IE0:        equ      89h        ;External Interrupt 0 edge flag
IT1:        equ      8ah        ;Interrupt 1 type control bit
IE1:        equ      8bh        ;External Interrupt 1 edge flag
TR0:        equ      8ch        ;Timer 0 run control bit
TF0:        equ      8dh        ;Timer 0 overfow flag
TR1:        equ      8eh        ;Timer 1 run control bit
TF1:        equ      8fh        ;Timer 1 overfow flag
        ;Serial Control
RI:         equ      98h        ;Receive Interrupt flag
TI:         equ      99h        ;Transmit interrupt flag
RB8:        equ      9ah        ;
TB8:        equ      9bh        ;
REN:        equ      9ch        ;Enable Reception
SM2:        equ      9dh        ;Enables multiprocessor communication
SM1:        equ      9eh        ;Serial port mode specifier
SM0:        equ      9fh        ;Serial port mode specifier
        ;additional declaration
PORTA:      equ     0e0e0h
```

โปรแกรมรับการกดคีย์จากสถานีปฏิบัติงาน(ต่อ)

```
PORTB:    equ     0e0e1h
PORTC:    equ     0e0e2h
CONTRL:   equ     0e0e3h
MROW:     equ     070h
MCOLUMN:  equ     071h
PATTERN:  equ     072h
KEYWORD:  equ     073h
BUFFER:   equ     074h
STORE1:   equ     075h
STORE2:   equ     076h
;-----------------------------------------;
; data buffer = internal ram address 50h-6fh ;
;-----------------------------------------;
          org   0000h
          ljmp  power_up
;-----------------------------------------;
; INT0 routine , falling edge of IBF signal  ;
;-----------------------------------------;
          org   0003h
          ljmp  int_int0
;-------------------------;
;      main program       ;
;-------------------------;
          org   0200h
;delay loop for power-up
power_up: mov   r3 , #0ffh
   loop0: mov   r4 , #0ffh
          djnz  r4 , $
          djnz  r3 , loop0
;set STB A signal ( INT1 ) = 1
          setb  P3_3
;reset p3_2 = 0 ( INT0 ) , receive IBF signal
          clr   P3_2
;clean data buffer ( add 50h-6fh ) in internal ram
          mov   r1  , #6fh
   clean: mov   @r1 , #0
          dec   r1
          cjne  r1 , #4fh , clean
;set 8255 to mode 0
;port a = input ; receive data from column of matrix
```

โปรแกรมรับการกดคีย์จากสถานีปฏิบัติงาน(ต่อ)

```
;port b = output: send data to row of matrix
;port c = output: send keyword to port a of another 8255
          mov   a , #10010000b
          mov   DPTR , #CONTRL
          movx  @DPTR , a
;clear output port c
          mov   a , #0
          mov   DPTR , #PORTC
          movx  @DPTR , a
;set INT0 for falling edge interrupt ( receive IBF signal )
          mov   TCON , #000000001b
;enable interrupt ( enable INT0 only )
          mov   IE , #10000001b
          mov   P1 , #10001000b       ;check status
;write zero to port b
   loop1: mov   a , #0
          mov   DPTR , #PORTB
          movx  @DPTR , a
;read data from port a and wait until some keys has been pressed
          mov   DPTR , #PORTA
   loop2: movx  a , @DPTR
          cjne  a , #0ffh , delay1
          sjmp  loop2
;delay for debounce ( change from 1 to 0 )
;time interval = 2 + (2 + 2r4 + 2)r3 = 1.03 ms
  delay1: mov   r3 , #2               ;2 microsec
   loop3: mov   r4 , #0ffh            ;2 microsec
          djnz  r4 , $                ;2 microsec
          djnz  r3 , loop3            ;2 microsec
;initial condition
          mov   r2 , #0
          mov   MROW , #0
          mov   MCOLUMN , #0
          mov   KEYWORD , #1
          mov   PATTERN , #11111110b
;output pattern to port b ( scan each row of matrix )
   loop4: mov   DPTR , #PORTB
          mov   a , PATTERN
          movx  @DPTR , a
;read pattern from port a
```

โปรแกรมรับการกดคีย์จากสถานีปฏิบัติงาน(ต่อ)

```
          mov   DPTR , #PORTA
          movx  a , @DPTR
          setb  c
;rotate acc to the right 1 bit
 bitrot:  rrc   a
;if carry = 1,scan next column, otherwise go to carry0
          jnc   carry0
nextcol:  inc   r2
          cjne  r2 , #08h , bitrot
          mov   r2 , #0
;rotate pattern the left 1 bit
          push  a
          mov   a , PATTERN
          rl    a
          mov   PATTERN , a
;keyword = keyword + 8
          mov   a , KEYWORD
          add   a , #8
          mov   KEYWORD , a
          pop   a
          ljmp  loop4
 carry0:  cjne  a , #11111111b , loop5
;key is correct  so keyword = keyword + mcolumn
          mov   MCOLUMN , r2
          mov   a , KEYWORD
          add   a , MCOLUMN
          mov   KEYWORD ,a
;check data buffer full?
          mov   r0 , #6fh
          cjne  @r0 , #0 , full
          mov   P1 , #0
;IBF signal=1?
          jnb   P3_2 , send
;store keyword in data buffer
          mov   r0 , #6fh
  loop7:  cjne  @r0 , #0 , put
          dec   r0
          cjne  r0 , #4fh , loop7
    put:  inc   r0
          mov   @r0 , KEYWORD
```

โปรแกรมรับการกดคีย์จากสถานีปฏิบัติงาน(ต่อ)

```
          sjmp  loop5
;write keyword to port c
    send: mov   DPTR , #PORTC
          mov   a , KEYWORD
          movx  @DPTR , a
;send strobe signal ( STB A ) to tell 8028G;time=2 + 2(1)= 4 microsec
          mov   P1 , #01010101b
          clr   P3_3
          mov   r3 , #1               ;2 microsec
          djnz  r3 , $                ;2 microsec
          setb  P3_3
          sjmp  loop5
;led display at P1_7 of port 1 to indicate that data buffer full
    full: mov   P1 , #11110000b
;write zero to port b
   loop5: mov   a , #0
          mov   DPTR , #PORTB
          movx  @DPTR , a
;read data from port a
          mov   DPTR , #PORTA
          movx  a , @DPTR
;wait until key up
          cjne  a , #11111111b , loop5
;delay for debounce ( key up )(change from 0 to 1 ):time=2 + (2 + 2r4 + 2)r3 = 1.03 ms
  delay2: mov   r3 , #2               ;2 microsec
   loop6: mov   r4 , #0ffh            ;2 microsec
          djnz  r4 , $                ;2 microsec
          djnz  r3 , loop6            ;2 microsec
          ljmp  loop1
;-------------------------------------------;
; INT0 routine , falling edge of IBF signal ;
;-------------------------------------------;
          org   0900h
int_int0: push  DPH
          push  DPL
          push  PSW
          push  a
          mov   P1 , #00000111b       ;check status
;clear port c
          mov   DPTR , #PORTC
```

โปรแกรมรับการกดคีย์จากสถานีปฏิบัติงาน(ต่อ)

```
          mov   a  , #0
          movx  @DPTR , a
;be sure that STB A = 1
          setb  P3_3
;data buffer empty?
          mov   r1 , #6fh
     11:  cjne  @r1 , #0 , 12
          dec   r1
          cjne  r1 , #4fh , 11
          ljmp  return
     12:  cjne  r1 , #50h , 13
          mov   BUFFER , @r1
          mov   @r1 ,#0
          ljmp  send1
     13:  mov   STORE1 , @r1
          mov   @r1 , #0
     14:  dec   r1
          mov   STORE2 , @r1
          mov   @r1 , STORE1
          mov   STORE1 , STORE2
          cjne  r1 , #50h , 14
          mov   BUFFER , STORE1
;write to port c
   send1: mov   DPTR , #PORTC
          mov   a , BUFFER
          movx  @DPTR , a
;send strobe signal ( STB A ) (p3_3) to 286-AT ; time = 2 + 2(1) = 4  microsec
          clr   P3_3
          mov   r1 , #1               ;2 microsec
          djnz  r1 , $                ;2 microsec
  return: pop   a
          pop   PSW
          pop   DPL
          pop   DPH
          setb  P3_3
          reti
```

โปรแกรมนำรหัสสถานีจาก 8031-A เก็บยังหน่วยความจำและย้ายลงสแต็ค

```
;***********************************************;
; Resident program get station code from 8031-A ;
;       store in memory and move to stack       ;
;***********************************************;
                .286c
cseg            segment byte public
                assume  cs:cseg ,ds:cseg ,es:cseg , ss:cseg
                public  start
                public  new_int0fh
                public  int6fh
                public  data_area
                org     0100h
start:          jmp     initialize
porta           equ     0300h
portb           equ     0301h
portc           equ     0302h
contr           equ     0303h
ocw1_8259       equ     21h
ocw2_8259       equ     20h
parameter       equ     word ptr [bp+8]
old_int0fh_off  dw      ?
old_int0fh_seg  dw      ?
first           db      '--------------------------',10,13
                db      '  AGV CONTROLLER  V.2.0   ',10,13
                db      '  Control Dept , KMITL    ',10,13
                db      '    1992 , A.Nathalang    ',10,13
                db      '--------------------------$',10,13
second          db      'resident file already exist!$',10,13
third           db      'WORKSTATION CALL = NO.$'
forth           db      ,13,'$'
fifth           db      10,13,'DATA AREA IS FULL!$'
sixth           db      'stack[bp+8]=$'
seventh         db      10,13,'$'
data_area       db      100 dup(0)
new_int0fh  proc  far
            cli
            push  bp
            mov   bp , sp
            sub   sp , 20
            push  ds
```

โปรแกรมนำรหัสสเกนิจาก 8031-A เก็บยังหน่วยความจำและย้ายลงสแต็ค(ต่อ)

```
            push  es
            push  ss
            pushf
            pusha
;disable irq7
            mov   al , 11111111b
            out   ocw1_8259 , al
            assume cs:cseg,ds:cseg
            push  cs
            pop   ds
;read 8 bits data from port a
            mov   dx , porta
            in    al , dx
            cmp   al , 0h
            jne   check
            jmp   _out
    check:  mov   di , 0
       n1:  mov   ch , cs:byte ptr data_area[di]
            cmp   ch , 0
            je    n2
            inc   di
            cmp   di , 31
            ja    disp
            jmp   n1
;-----------------;
;  create window  ;
;-----------------;
       n2:  mov   ah , 7
            mov   al , 0
            mov   bh , 70h              ;reverse
            mov   ch , 8
            mov   cl , 18
            mov   dh , 16
            mov   dl , 53
            int   10h
            mov   bh , 07h              ;normal
            mov   ch , 9
            mov   cl , 20
            mov   dh , 15
            mov   dl , 51
```

โปรแกรมนำรหัสสถานีจาก 8031-A เก็บยังหน่วยความจำและย้ายลงสแต็ค(ต่อ)

```
            int   10h
            mov   bh , 70h              ;reverse
            mov   ch , 10
            mov   cl , 22
            mov   dh , 14
            mov   dl , 49
            int   10h
;set cursor position
            mov   ah , 02
            mov   dh , 12
            mov   dl , 24
            mov   bh , 0
            int   10h
;display message third message to screen
            mov   si , 0
            mov   ah , 14               ;write TTY function
            mov   bh , 0                ;define page number
    loop1:  mov   al , cs:byte ptr third[si]
            int   10h
            cmp   al , '.'
            je    next
            inc   si
            jmp   loop1
;read 8 bits data from port a
     next:  mov   dx , porta
            in    al , dx
;store binary data in data_area
            mov   cs:byte ptr data_area[di] , al
            jmp   _next
;display that data_area is full!
     disp:  mov   si , 0
            mov   ah , 14               ;write TTY function
            mov   bh , 0                ;define page number
   _loop1:  mov   al , cs:byte ptr fifth[si]
            int   10h
            cmp   al , '!'
            je    _out
            inc   si
            jmp   _loop1
```

โปรแกรมนำรหัสสถานีจาก 8031-A เก็บยังหน่วยความจำและย้ายลงสแต็ค(ต่อ)

```
;-------------------------------------------------------;
;convert 8 bit binary in al to BCD (number must not > 99)  ;
;-------------------------------------------------------;
    _next:  cbw                         ;extend al (8 bits) to ah:al (16 bits)
            mov   cl , 10
;divide ah:al by cl , unsigned :al = quotient , ah = remainder
            div   cl
            mov   cl , 4                ;no.of step to shift left = 4
            shl   al , cl               ;shift al left  4 bits
            or    al , ah               ;al = al + ah     ( al = 2 digits BCD )
;convert upper nibble to ASCII
            mov   dh , al               ;save al to dh
            shr   al , cl               ;shift al right 4 bits
            add   al , "0"              ;al = al + "0"
;display upper nibble to screen
            mov   ah , 14               ;write TTY function
            mov   bh , 0                ;define page number
            int   10h
;convert lower nible to ASCII
            mov   al , dh
            and   al , 0fh              ;remove upper nibble
            add   al , "0"
;display lower nible to screen
            int   10h
;set cursor position
            mov   ah , 02
            mov   dh , 17h
            mov   dl , 0
            mov   bh , 0
            int   10h
;line feed and return cursor
            mov   si , 0
            mov   ah , 14               ;write TTY function
            mov   bh , 0                ;define page number
    loop2:  mov   al , cs:byte ptr forth[si]
            int   10h
            cmp   al , 13
            je    _out
            inc   si
            jmp   loop2
```

โปรแกรมถ่ายรหัสสถานีจาก 8031-A เก็บข้างหน่วยความจำและย้ายลงสแต็ค(ต่อ)

```
;enable irq7
     _out:  mov   al , 00111100b
            out   ocw1_8259 , al
;send End-Of-Interrupt code
            mov   al , 00100000b
            out   ocw2_8259 , al
            popa
            popf
            pop   ss
            pop   es
            pop   ds
            mov   sp , bp
            pop   bp
            sti
            int   20h
            iret
new_int0fh  endp
    int0fh  proc  far
            cli
            push  bp
            mov   bp , sp
            push  ds
            assume cs:cseg,ds:cseg
            push  cs
            pop   ds
;display message sixth[si]
            mov   si , 0
            mov   ah , 14                ;write TTY function
            mov   bh , 0                 ;define page number
      _lx:  mov   al , cs:byte ptr sixth[si]
            int   10h
            cmp   al , '='
            je    x1
            inc   si
            jmp   _lx
;load pascal parameter from stack to dl
       x1:  mov   dx , parameter         ;load parameter from stack
            cmp   dl , 0
            jne   x3
            mov   al , 0
```

โปรแกรมนำรหัสสถานีจาก 8031-A เก็บยังหน่วยความจำและย้ายลงสแต็ค(ต่อ)

```
            jmp   14
;display pascal parameter to screen
        x3: mov   al , dh
            int   10h
            mov   al , dl              ;dl store parameter from pascal
            int   10h
;line feed and return cursor
            mov   si , 0
       lx2: mov   al , cs:byte ptr seventh[si]
            int   10h
            cmp   al , 13
            je    x2
            inc   si
            jmp   lx2
;load value from data_area ( first in/first out )
        x2: mov   si , 0
            mov   ch , cs:byte ptr data_area[si]
            cmp   ch , 0
            je    13
            mov   al , ch              ;al stores station code
        11: mov   ch , cs:byte ptr data_area[si + 1]
            cmp   ch , 0
            je    12
            mov   cs:byte ptr data_area[si] , ch
            inc   si
            jmp   11
        12: mov   cs:byte ptr data_area[si] , 0
            jmp   14
        13: mov   al , 0
            mov   dl , 0
        14: mov   ah , dl              ;return result to pascal by ax
            mov   parameter , ax
            pop   ds
            mov   sp , bp
            pop   bp
            sti
            iret
    int6fh  endp
initialize: push  cs
            pop   ds
```

โปรแกรมถ่ายรหัสสถานีจาก 8031-A เก็บยังหน่วยความจำและย้ายลงสแต็ค(ต่อ)

```
;disable irq7 by set ocw1_8259
            mov   al , 10111100b
            out   ocw1_8259 , al
;read interupt vector es:bx = segment:offset
            mov   ax , 350fh
            int   21h
;check that file already in ram?
            mov   ax , offset new_int0fh
            cmp   bx , ax
            jne   store
;display message that file has already in ram
            mov   ah , 09
            mov   dx , offset second
            int   21h
            jmp   go_out
    store:  mov   word ptr cs:[old_int0fh_off],bx
            mov   word ptr cs:[old_int0fh_seg],es
;display resident message
            mov   ah , 09h
            mov   dx , offset first
            int   21h
;set interrupt handler address at interrupt vector 0fh (irq7)
            mov   dx , offset new_int0fh
            mov   ax , 250fh
            int   21h
;set interrupt handler address at interrupt vector 6fh (reserve)
            mov   dx , offset int6fh
            mov   ax , 256fh
            int   21h
;-----------------------------------------------------;
; set 8255 to mode 1 ( strobe input ) in control word ;
;          port a : input                             ;
;          port b : output ( led display )            ;
;          port c : pc3 = INTRA                       ;
;                   pc4 = STBA                        ;
;                   pc5 = IBFA                        ;
;                                                     ;
;          D7 = 0          ;define set/reset bit      ;
;             = 1          ;define operation*         ;
;       D6,D5 = 0,0        ;mode 0                    ;
```

โปรแกรมนำรหัสสถานีจาก 8031-A เก็บยังหน่วยความจำและย้ายลงสแต็ค(ต่อ)

```
;             = 0,1         ;mode 1*                    ;
;             = 1,x         ;mode 2                     ;
;         D4 = 0            ;port a is output           ;
;             = 1           ;port a is input*           ;
;         D3 = 0            ;pc6,pc7 is output*         ;
;             = 1           ;pc6,pc7 is input           ;
;         D2 = 0            ;mode 0*                    ;
;             = 1           ;mode 1                     ;
;         D1 = 0            ;port b is output*          ;
;             = 1           ;port a is input            ;
;         D0 = X            ;don't care*                ;
;-------------------------------------------------------;
            mov   dx , contr
            mov   al , 10110000b
            out   dx , al
;-------------------------------------------------------;
; set/reset bit of port c in control word               ;
;         D7 = 0            ;define set/reset bit       ;
;    D6,D5,D4 = x,x,x       ;don't care                 ;
;    D3,D2,D1 = 0,0,0       ;select pc0                 ;
;             = 0,0,1       ;select pc1                 ;
;             = 0,1,0       ;select pc2                 ;
;             = 0,1,1       ;select pc3                 ;
;             = 1,0,0       ;select pc4 (INTE A)        ;
;             = 1,0,1       ;select pc5                 ;
;             = 1,1,0       ;select pc6                 ;
;             = 1,1,1       ;select pc7                 ;
;         D0 = 1            ;set to 1                   ;
;             = 0           ;reset to 0                 ;
;-------------------------------------------------------;
;set INTE A = 1 (enable 8255 interrupt )
            mov   al , 00001001b
            out   dx , al
;show normal operation at led display
            mov   al , 00000001b
            mov   dx , portb
            out   dx , al
;enable irq7
            mov   al , 00111100b
            out   ocw1_8259 , al
```

โปรแกรมนำรหัสสถานีจาก 8031-A เก็บยังหน่วยความจำและย้ายลงสแต็ค(ต่อ)

```
;send EOI command
            mov   al , 00100000b
            out   ocw2_8259 , al
            sti
;make it a resident file
            assume cs:cseg,ds:cseg
            push  cs
            pop   ds
            mov   dx , offset initialize
            int   27h
;enable irq7
   go_out:  mov   al , 00111100b
            out   ocw1_8259 , al
;send EOI command
            mov   al , 00100000b
            out   ocw2_8259 , al
            sti
            int   20h
cseg        ends
            end   start
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์,แปลรหัสและปฏิบัติตามคำสั่ง

```
;*************************************************************************;
; 8031-1 Serial Communication, Code translation and Command executing ;
;*************************************************************************;
          cpu "8051.TBL"
          hof "INT8"
A:        equ     0e0h          ;Accumulator
B:        equ     0f0h          ;B Register
PSW:      equ     0d0h          ;Program Status Word
SP:       equ      81h          ;Stack Pointer
DPL:      equ      82h          ;Data Pointer (low byte)
DPH:      equ      83h          ;Data Pointer (high byte)
P0:       equ      80h          ;Port 0
P1:       equ      90h          ;Port 1
P2:       equ     0a0h          ;Port 2
P3:       equ     0b0h          ;Port 3
IP:       equ     0b8h          ;Interrupt Priority Control
IE:       equ     0a8h          ;Interrupt Enable Control
TMOD:     equ      89h          ;Timer/Counter Mode Control
TCON:     equ      88h          ;Timer/Counter Control
TH0:      equ      8ch          ;Timer/Counter 0 (high byte)
TL0:      equ      8ah          ;Timer/Counter 0 (low byte)
TH1:      equ      8dh          ;Timer/Counter 1 (high byte)
TL1:      equ      8bh          ;Timer/Counter 1 (low byte)
SCON:     equ      98h          ;Serial Control
SBUF:     equ      99h          ;Serial Data Buffer
PCON:     equ      87h          ;Power Control
;Accumulator
A_7:      equ     0e7h
A_6:      equ     0e6h
A_5:      equ     0e5h
A_4:      equ     0e4h
A_3:      equ     0e3h
A_2:      equ     0e2h
A_1:      equ     0e1h
A_0:      equ     0e0h
;B Register
B_7:      equ     0f7h
B_6:      equ     0f6h
B_5:      equ     0f5h
B_4:      equ     0f4h
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
B_3:        equ     0f3h
B_2:        equ     0f2h
B_1:        equ     0f1h
B_0:        equ     0f0h
;Program Status Word ( PSW , BIT7 - BIT0 )
CY:         equ     0d6h        ;Carry flag
F0:         equ     0d5h        ;flag 0 ( general purpose )
RS1:        equ     0d4h        ;Register bank selector bit1
RS0:        equ     0d3h        ;Register bank selector bit0
OV:         equ     0d2h        ;Over flow flag
P:          equ     0d0h        ;Parity flag
;Port 0 ( BIT7 - BIT0 )
P0_7:       equ      87h
P0_6:       equ      86h
P0_5:       equ      85h
P0_4:       equ      84h
P0_3:       equ      83h
P0_2:       equ      82h
P0_1:       equ      81h
P0_0:       equ      80h
;Port 1 ( BIT7 - BIT0 )
P1_7:       equ      97h
P1_6:       equ      96h
P1_5:       equ      95h
P1_4:       equ      94h
P1_3:       equ      93h
P1_2:       equ      92h
P1_1:       equ      91h
P1_0:       equ      90h
;Port 2 ( BIT7 - BIT0 )
P2_7:       equ     0a7h
P2_6:       equ     0a6h
P2_5:       equ     0a5h
P2_4:       equ     0a4h
P2_3:       equ     0a3h
P2_2:       equ     0a2h
P2_1:       equ     0a1h
P2_0:       equ     0a0h
;Port 3
P3_7:       equ     0b7h        ;read
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
P3_6:     equ     0b6h        ;write
P3_5:     equ     0b5h        ;TI interrupt
P3_4:     equ     0b4h        ;T0 interrupt
P3_3:     equ     0b3h        ;INT1 interrupt
P3_2:     equ     0b2h        ;INT0 interrupt
P3_1:     equ     0b1h        ;TXD interrupt
P3_0:     equ     0b0h        ;RXD interrupt
;Interrupt Priority Control ( IP , bit7 - bit0 )
PS:       equ     0bch        ;Serial Port Interrupt priority level
PT1:      equ     0bbh        ;Timer 1 Interrupt priority level
PX1:      equ     0bah        ;External Interrupt 1 priority level
PT0:      equ     0b9h        ;Timer 0 Interrupt priority level
PX0:      equ     0b8h        ;External Interrupt 0 priority level
;Interrupt Enable Control ( IE , bit7 - bit0 )
EA:       equ     0afh        ;Enable All Intterrupts
ES:       equ     0ach        ;Serial Port interrupt
ET1:      equ     0abh        ;Timer 1 overflow interrupt
EX1:      equ     0aah        ;External Interrupt 1
ET0:      equ     0a9h        ;Timer 0 overflow interrupt
EX0:      equ     0a8h        ;External Interrupt 0
;Timer/Counter Control ( TCON , bit7 - bit0 )
TF1:      equ     8fh         ;Timer 1 overfow flag
TR1:      equ     8eh         ;Timer 1 run control bit
TF0:      equ     8dh         ;Timer 0 overfow flag
TR0:      equ     8ch         ;Timer 0 run control bit
IE1:      equ     8bh         ;External Interrupt 1 edge flag
IT1:      equ     8ah         ;Interrupt 1 type control bit
IE0:      equ     89h         ;External Interrupt 0 edge flag
IT0:      equ     88h         ;Interrupt 0 type control bit
;Serial Control ( SCON , bit7-bit0 )
SM0:      equ     9fh         ;Serial port mode specifier
SM1:      equ     9eh         ;Serial port mode specifier
SM2:      equ     9dh         ;Enables multiprocessor communication
REN:      equ     9ch         ;Enable Reception
TB8:      equ     9bh         ;
RB8:      equ     9ah         ;
TI:       equ     99h         ;Transmit interrupt flag
RI:       equ     98h         ;Receive interrupt flag
;Additional Declaration
PORTA:    equ     4000h
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
PORTB:    equ     4001h
PORTC:    equ     4002h
CONTRL:   equ     4003h
LOWER:    equ     7fh
STORE1:   equ     7eh
STORE2:   equ     7dh
STORE3:   equ     7ch
HBYTE:    equ     7bh
LBYTE:    equ     7ah
H:        equ     79h
L:        equ     78h
N:        equ     77h
X:        equ     76h
INT1:     equ     75h
LOAD:     equ     74h
GET:      equ     73h
;-------------------------------------------------;
; external program memory     address 0000h - 1fffh ;
; external data memory        address 2000h - 3fffh ;
;  - transmit message area    address 2100h - 22d0h ;
;  - receive message area     address 2300h - 23ffh ;
;  - data area for "read"     address 2400h - 24ffh ;
;  - data area for "data\"    address 2500h - 25ffh ;
;-------------------------------------------------;
          org   0000h
          ljmp  power_up
;------------------------;
; INT0 interrupt handler ;
;------------------------;
          org   0003h
          ljmp  int_int0
;------------------------;
; INT1 interrupt handler ;
;------------------------;
          org   0013h
          ljmp  int_int1
;--------------------------;
; RXD,TXD interrupt handler ;
;--------------------------;
          org   0023h
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
          ljmp  serial
;-------------------------------------------------;
; define communication message   address 0100h - 02d0h ;
;-------------------------------------------------;
          org   0100h
    0100  dfb   "\agv ready to receive your message!",13,10
    0125  dfb   "\syntax Error!",13,10
    0135  dfb   "\agv o.k!",13,10
    0140  dfb   "\Target=",13,10
    014a  dfb   "\obstacle=",13,10
    0156  dfb   "\battery=",13,10
    0161  dfb   "\move=",13,10
    0169  dfb   "\cross-way=",13,10
    0176  dfb   "not reach ",13,10
    0182  dfb   "reach ",13,10
    018a  dfb   "finish ",13,10
    0193  dfb   "normal ",13,10
    019c  dfb   "low ",13,10
    01a2  dfb   "doing ",13,10
    01aa  dfb   "no ",13,10
    01af  dfb   "discover ",13,10
    01ba  dfb   "\invalid function",13,10
    01cd  dfb   "\invalid position data",13,10
    01e5  dfb   "\invalid rotation data",13,10
    01fd  dfb   "\invalid direction data",13,10
    0216  dfb   "\invalid symbol '='",13,10
    022b  dfb   "\invalid symbol '\'",13,10
    0240  dfb   "\data overflow 1",13,10
    0252  dfb   "\data overflow 2",13,10
    0264  dfb   "\STBA=0",13,10
    026d  dfb   "\IBF=1",13,10
    0275  dfb   "\press 'y' to continue!",13,10
    028e  dfb   "\load complete",13,10
    029e  dfb   "\moving complete",13,10
    02b0h dfb   "\data unload",13,10
    02be  dfb   "\no data to load",13,"*",10
          org   0300h
;-------------------------;
;delay time for power-up  ;
;-------------------------;
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์,แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
power_up: mov   r1 , #0
   loop:  mov   r0 , #0ffh
          djnz  r0 , $
          inc   r1
          cjne  r1 , #0fh , loop      ;delay time for power-up
          mov   IE , #00000000b
;set 'move' signal to 1
          setb  P1_1
;set STBA signal to 1
          setb  P1_0
;set stack area address 08h - 7fh
          mov   SP , #07h
;-----------------------------------------------;
; set serial communication , baudrate = 150 bps ;
;-----------------------------------------------;
; set serial for high level priority by let PS in IP = 1
          mov   IP , #00010000b
;-----------------------------------------------;
; select timer1,set to mode 2                   ;
; ( 8 bit auto reload with TH1 ) and let GATE = 0 ;
; TMOD:GATE,C/T,M1,M0,GATE,C/T,M1,M0            ;
;      ----timer1---- ----timer0----            ;
;-----------------------------------------------;
          mov   TMOD , #00100000b
;-------------------------------------------------------;
; set TR1=1 and set INT0,INT1 for detect falling edge signal ;
; TCON:TF1,TR1,TF0,TR0,IE1,IT1,IE0,IT0                  ;
;-------------------------------------------------------;
          mov   TCON , #01000101b
;define value for timer1 ( define baud rate = 150 bps )
          mov   TH1  , #040h
;----------------------;
; set 8255 mode 0      ;
;   port a = output    ;
;   port b = output    ;
;   port c = output    ;
;----------------------;
          mov   DPTR , #CONTRL
          mov   a , #10000000b
          movx  @DPTR , a
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
;set initial value to 8255
          mov   a , #0
          mov   DPTR , #PORTA
          movx  @DPTR , a
          mov   a , #88h
          mov   DPTR , #PORTB
          movx  @DPTR , a
          mov   DPTR , #PORTC
          movx  @DPTR , a
;wait agv_mpu2 start_up
  wlagv2: mov   c , P3_3
          jc    wlagv2
  w2agv2: mov   c , P3_3                ;P3_3=0
          jnc   w2agv2
          mov   a , #0                  ;P3_3=1
          mov   DPTR , #PORTB
          movx  @DPTR , a
          mov   DPTR , #PORTC
          movx  @DPTR , a
;-----------------------------------------------;
; transfer data from program-memory to data memory ;
; address 0100h - 02d0h to address 2100h - 22d0h   ;
;-----------------------------------------------;
          mov   r2 , #01h
          mov   r3 , #21h
          mov   DPL , #0
 loop1:   mov   DPH , r2
          mov   a , #0
          movc  a , @a + DPTR           ;read message from program memory
          mov   DPH , r3
          movx  @DPTR , a               ;store message to data memory
          mov   STORE1 , a
          inc   DPL
          mov   a , DPL
          cjne  a , #0h , nn1
          mov   r2 , #02h
          mov   r3 , #22h
   nn1:   mov   a , STORE1
          cjne  a , #2ah , loop1        ;compare to '*'
          mov   DPTR , #201ffh          ;point to first message
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
          mov   INT1 , #0
          mov   LOAD , #0
          mov   GET , #0
;-----------------------------------------;
;   prepare value for transmit serial data   ;
;-----------------------------------------;
    set:  mov   TCON , #01000101b
          mov   IE , #10010000b        ;enable TXD
;-------------------------------------------------------------;
; set SCON and start to transmit serial data                  ;
; bit 7,6 = 0,1    set serial mode 1 (8 bits,no parity,1 stop) ;
; bit 5   = 0      SM2 = 0                                    ;
; bit 4   = 0      REN = 0 , disable reception                ;
; bit 3,2 = 0,0                                               ;
; bit 1   = 1      TI = 1 , start to transmit                 ;
; bit 0   = 0      RI = 0                                     ;
;-------------------------------------------------------------;
          mov   SCON , #01000010b
   wait:  movx  a , @DPTR              ;wait for the end of transmit
          cjne  a , #13 , wait
          mov   IE , #00000000b        ;disable TXD
          jnb   TI , $                 ;wait until TI = 1
          clr   TI                     ;stop transmit message
;-------------------------------------------------;
; clear receive serial data ( adress 2300h - 23ffh ) ;
;-------------------------------------------------;
 clear0:  mov   a , #0
          mov   DPTR , #22ffh
          movx  @DPTR , a
          mov   DPTR , #2300h
 clear1:  movx  @DPTR , a
          inc   DPL
          cjne  a , DPL , clear1
          mov   DPTR , #22ffh
;-------------------------------------------------;
; set SCON for receive serial data                ;
; bit 7,6 = 0,1      serial mode 1                ;
; bit 5   = 1        SM2=1,RI will set if         ;
;                      a) RI=0                    ;
;                      b) receive stop bit = 1    ;
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
; bit 4   = 1        REN = 1 , enable reception   ;
; bit 3,2 = 0,0                                   ;
; bit 1   = 0        TI = 0                       ;
; bit 0   = 0        RI = 0                       ;
;-------------------------------------------------;
         mov   SCON , #01110000b
         mov   TCON , #01000101b
         mov   IE , #10010100b        ;enable RXD,INT1
  wait1: mov   a , INT1
         cjne  a , #1 , contin
         mov   INT1 , #0
         mov   DPTR , #229dh          ;send "moving complete" to 286-AT
         ljmp  set
 contin: movx  a , @DPTR              ;wait for the end of receive message
         cjne  a , #13 , wait1
         mov   IE , #00000000b        ;disable RXD,INT1
;-----------------------------------------------------------------------;
; change ASCII code of big character and small character to the same value ;
; -big-  -ASCII code-    : -small-  -ASCII code-       same value      ;
;  DATA  44h,41h,54h,41h :  data    64h,61h,74h,61h    04h,01h,14h,01h  ;
;  GET\  47h,45h,54h,5ch :  get\    67h,65h,74h,5ch    07h,05h,14h,1ch  ;
;  LOAD  4ch,4fh,41h,44h :  load    6ch,6fh,61h,64h    0ch,0fh,01h,04h  ;
;  MOVE  4dh,4fh,56h,45h :  move    6dh,6fh,76h,65h    0dh,0fh,16h,05h  ;
;  READ  52h,45h,41h,44h :  read    72h,65h,61h,64h    12h,05h,01h,04h  ;
;  TEST  54h,45h,53h,54h :  test    74h,65h,73h,74h    14h,05h,13h,14h  ;
;-----------------------------------------------------------------------;
         mov   DPTR , #2301h
  loop2: movx  a , @DPTR              ;change first 4 characters to same code
         anl   a , #00011111b
         movx  @DPTR , a
         inc   DPL
         mov   a , #05
         cjne  a , DPL , loop2
;what is the command!
;check 'data'?
         mov   DPTR , #2301h
         movx  a , @DPTR
         cjne  a , #04h , l_get       ;a="d" or "D" ?
         inc   DPL
         movx  a , @DPTR
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
          cjne  a , #01h , rest         ;a="a" or "A" ?
          inc   DPL
          movx  a , @DPTR
          cjne  a , #14h , rest         ;a="t" or "T" ?
          inc   DPL
          movx  a , @DPTR
          cjne  a , #01h , rest         ;a="a" or "A" ?
          ljmp  do_data
;check 'get\'?
  l_get:  mov   DPTR , #2301h
          movx  a , @DPTR
          cjne  a , #07h , l_load       ;a="g" or "G" ?
          inc   DPL
          movx  a , @DPTR
          cjne  a , #05h , rest         ;a="e" or "E" ?
          inc   DPL
          movx  a , @DPTR
          cjne  a , #14h , rest         ;a="t" or "T" ?
          inc   DPL
          movx  a , @DPTR
          cjne  a , #1ch , rest         ;a="\" ?
          ljmp  do_get
;check 'load'?
 l_load:  mov   DPTR , #2301h
          movx  a , @DPTR
          cjne  a , #0ch , l_move       ;a="l" or "L" ?
          inc   DPL
          movx  a , @DPTR
          cjne  a , #0fh , rest         ;a="o" or "O" ?
          inc   DPL
          movx  a , @DPTR
          cjne  a , #01h , rest         ;a="a" or "A" ?
          inc   DPL
          movx  a , @DPTR
          cjne  a , #04h , rest         ;a="d" or "D" ?
          ljmp  do_load
    rest: ljmp  syntax
;check 'move'?
 l_move:  mov   DPTR , #2301h
          movx  a , @DPTR
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
        cjne  a , #0dh , l_read      ;a="m" or "M" ?
        inc   DPL
        movx  a , @DPTR
        cjne  a , #0fh , syntax      ;a="o" or "O" ?
        inc   DPL
        movx  a , @DPTR
        cjne  a , #16h , syntax      ;a="v" or "V" ?
        inc   DPL
        movx  a , @DPTR
        cjne  a , #05h , syntax      ;a="e" or "E" ?
        ljmp  do_move
;check 'read'?
 l_read: mov   DPTR , #2301h
        movx  a , @DPTR
        cjne  a , #12h , l_test      ;a="r" or "R" ?
        inc   DPL
        movx  a , @DPTR
        cjne  a , #05h , syntax      ;a="e" or "E" ?
        inc   DPL
        movx  a , @DPTR
        cjne  a , #01h , syntax      ;a="a" or "A" ?
        inc   DPL
        movx  a , @DPTR
        cjne  a , #04h , syntax      ;a="d" or "D" ?
        ljmp  do_read
;check 'test'?
 l_test: mov   DPTR , #2301h
        movx  a , @DPTR
        cjne  a , #14h , l_yes       ;a="t" or "T" ?
        inc   DPL
        movx  a , @DPTR
        cjne  a , #05h , syntax      ;a="e" or "E" ?
        inc   DPL
        movx  a , @DPTR
        cjne  a , #13h , syntax      ;a="s" or "S" ?
        inc   DPL
        movx  a , @DPTR
        cjne  a , #14h , syntax      ;a="t" or "T" ?
        ljmp  do_test
 l_yes:  mov  DPTR , #2301h
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
          movx  a , @DPTR
          cjne  a , #19h , syntax
          mov   DPL , STORE3
          ljmp  data1
;bad command!
 syntax:  mov   a , #0
          mov   DPTR , #PORTB
          movx  @DPTR , a
          mov   DPTR , #PORTC
          movx  @DPTR , a
          mov   STORE3 , #0
          mov   DPTR , #2124h          ;send "syntax error" to 286-AT
          ljmp  set
;--------------------------------;
;    read and display data       ;
;  from address 2500h - 25xxh    ;
;  port C = DPL , port B = data  ;
;--------------------------------;
do_data:  mov   STORE3 , #0
  data1:  mov   DPL , STORE3
          mov   DPH , #25h
          movx  a , @DPTR
          mov   DPTR , #PORTB
          movx  @DPTR , a
          mov   DPTR , #PORTC
          mov   a , STORE3
          movx  @DPTR , a
          inc   STORE3
          mov   DPTR , #2274h          ;send "press 'y' to continue" to 286-AT
          ljmp  set
;--------------------------------;
; receive data and classify     ;
; into adress 2500h - 25ffh     ;
;--------------------------------;
 do_get:  mov   GET , #1
          mov   LOAD , #0
          mov   STORE1 , DPL
;clear adress 2500h - 25ffh
          mov   a , #0
          mov   DPTR , #2500h
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
 clear2:  movx  @DPTR , a
          inc   DPL
          cjne  a , DPL , clear2
;define value
          mov   r0 , #0
          mov   r2 , #2
          mov   r3 , #3
          mov   DPH , #23h
          mov   DPL , STORE1
     x1:  inc   DPL
          movx  a , @DPTR
          anl   a , #00011111b
          cjne  a , #13 , x2            ;a=<return>?
          mov   DPH , #25h
          mov   DPL , r3
          mov   a , #2ah
          movx  @DPTR , a               ;store '*' at last address
          mov   r0 , #0
          mov   r2 , #2
          mov   r3 , #3
          mov   DPTR , #2134h           ;send "agv o.k" to 286-AT
          ljmp  set
     x2:  cjne  a , #10h , x3           ;a=p? or a=P?
          ljmp  x5
     x3:  cjne  a , #14h , x4           ;a=t or a=T?
          sjmp  x5
     x4:  mov   DPTR , #2169h           ;send "invalid function" to 286-AT
          ljmp  set
     x5:  mov   DPH , #25h
          push  DPL                     ;push DPL
          mov   DPL , r3
          movx  @DPTR , a
          mov   STORE1 , a              ;save p or P or t or T
          mov   a , r3                  ;r3 = r3 + 4
          add   a , #4
          mov   r3 , a
          mov   DPH , #23h
          pop   DPL                     ;restore DPL
          inc   DPL
          movx  a , @DPTR
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
        cjne  a , #3dh , x6          ;a="="?
        sjmp  x7
    x6: mov   DPTR , #2215h          ;send "invalid symbol '='" to 286-AT
        ljmp  set
    x7: mov   a , STORE1
        cjne  a , #14h , x10         ;if a < > t or T go to xx2
        mov   a , r0
        add   a , #4
        mov   r0 , a                 ;r0 = r0 + 4
        inc   DPL
        movx  a , @DPTR
        anl   a , #00011111b
        cjne  a , #0ch , x8          ;a="l" or "L"?
        ljmp  x24
    x8: cjne  a , #12h , x9          ;a="r" or "R"?
        ljmp  x24
    x9: mov   DPTR , #21fch          ;send "invalid direction data" to 286-AT
        ljmp  set
   x10: inc   DPL
        movx  a , @DPTR
        cjne  a , #2ch , x11         ;a=","?
        ljmp  x13
;-----------------------------------------;
; convert ASCII decimal string ( 0 - 9) to BCD ;
;-----------------------------------------;
   x11: clr   c                      ;clear carry flag before subtraction
        subb  a , #30h
        jc    x12                    ;if a < 30h ( c = 1 ) then go to x12
        movx  a , @DPTR              ;a >= 30h
        subb  a , #3Ah
        jnc   x12                    ;if a > 39h ( c = 0 ) then go to x12
        clr   c
        movx  a , @DPTR
        subb  a , #30h
        movx  @DPTR , a
        ljmp  x10
   x12: mov   DPTR , #21cch          ;send "invalid position data" to 286-AT
        ljmp  set
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
;-----------------------------------------------------;
; convert BCD decimal string to 2 bytes binary number ;
;-----------------------------------------------------;
   x13:  push  DPL
         mov   H , #0
         mov   L , #0
         mov   HBYTE , #0
         mov   LBYTE , #0
         mov   r4 , #0
  loop3: mov   DPH , #23h
         dec   DPL
         movx  a , @DPTR
         cjne  a , #3dh , x14          ;a="="?
         mov   DPH , #25h
         mov   DPL , r0
         mov   a , LBYTE
         movx  @DPTR , a               ;save LBYTE
         inc   DPL
         mov   a , HBYTE
         movx  @DPTR , a               ;save HBYTE
         mov   a , r0
         add   a , #4
         mov   r0 , a                  ;r0 = r0 + 4
         mov   DPH , #23h
         pop   DPL
         ljmp  x21
   x14:  cjne  r4 , #5 , x15
         mov   DPTR , #223fh           ;send "data over flow !" to 286-AT
         ljmp  set
   x15:  cjne  r4 , #0 , x16
         mov   LBYTE , a
         inc   r4
         ljmp  loop3
   x16:  mov   r5 , #0
   x17:  mov   b , #10
         mul   ab
         mov   H , b
         mov   L , a
         inc   r5
         mov   a , r5
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
        clr   c
        subb  a , r4                ;a = a - r4 - c
        jnc   x19                   ;if r5 >= r4 go to x19
        clr   c                     ;r5 < r4
        mov   a , r5
        subb  a , #2                ;a = a - 2 - c
        jnc   x18                   ;if a >= 3 go to x18
        mov   a , L
        ljmp  x17
x18:    mov   a , L
        mov   b , #10
        mul   ab
        mov   X , a
        mov   a , H
        mov   H , B
        mov   L , X
        mov   b , #10
        mul   ab
        mov   X , a
        mov   a , H
        add   a , X                 ;H = H + a
        mov   H , a
        inc   r5
        mov   a , r5
        mov   H , r4
        cjne  a , H , x18
        clr   c
x19:    mov   a , LBYTE
        addc  a , L
        mov   LBYTE , a
        mov   a , HBYTE
        addc  a , H
        mov   HBYTE , a
        jnc   x20                   ;if c = 1 , data overflow
        mov   DPTR , #2251h         ;send "data overflow 2" to 286-AT
        ljmp  set
x20:    inc   r4
        ljmp  loop3
x21:    inc   DPL
        movx  a , @DPTR
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
            cjne  a , #30h , x22          ;a="0"?
            sjmp  x24
      x22:  cjne  a , #31h , x23          ;a="1"?
            sjmp  x24
      x23:  mov   DPTR , #21e4h           ;send "invalid rotation data" to 286-AT
            ljmp  set
      x24:  mov   DPH , #25h
            push  DPL                     ;save DPL
            mov   DPL , r2
            movx  @DPTR , a
            mov   a , r2                  ;r2 = r2 + 4
            add   a , #4
            mov   r2 , a
            mov   DPH , #23h
            pop   DPL                     ;pop DPL
            inc   DPL
            movx  a , @DPTR
            cjne  a , #5ch , x25          ;a="\"?
            ljmp  x1
      x25:  mov   DPTR , #222ah           ;send "invalid symbol '\'" to 286-AT
            ljmp  set
;-------------------------------------------------;
; load data to from address 2500 - 25xxh to agv_mpu2 ;
; 1) INTEA=1 ( bit in 8255 of mpu2 )              ;
; 2) check STBA=1 ?,IBFA=0 ? ,and                 ;
; 3) enable INT0                                  ;
; 4) send data to port a                          ;
; 5) create STBA signal ( 4 usec )                ;
; ------------------------------------------------;
do_load:  mov   a , GET
          cjne  a , #1 , noget
          sjmp  begin                 ;get=1 , data is ready to load
  noget:  mov   DPTR , #225dh         ;send "no data to load" to 286-AT
          ljmp  set
  begin:  mov   GET , #0
          mov   LOAD , #1
          mov   STORE2 , #0
;check STBA=1 ? (P1_0)
          jb    P1_0 , testibf        ;if STBA=0 , send 'STBA=0' to 286-AT
          mov   DPTR , #2260h
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
          ljmp  set
;check IBFA=0 ? (P3_2)
testibf:  jnb   P3_2 , ibfok          ;if IBF=1 , send 'IBF=1' to 286-AT
          mov   DPTR , #2266h
          ljmp  set.
;enable INTO
  ibfok:  mov   IE , #10000001b
;clear port a
          mov   a , #0
          mov   DPTR , #PORTA
          movx  @DPTR , a
;create STBA signal ( 4 usec )
          clr   P1_0
          mov   r7 , #1               ;2 usec
          djnz  r7 , $                ;2 usec
          setb  P1_0                  ;INTO of agv_mpu2 is interrupted
 wait3:   mov   a , STORE2
          cjne  a , #2ah , wait3      ;a="*"?
;disable INTO
          mov   IE , #10000000b
          mov   DPTR , #228dh         ;send "load complete" to 286-AT
          ljmp  set
;-------------------------------;
; send 'move' signal to agv_mpu2 ;
;-------------------------------;
do_move:  mov   a , LOAD
          cjne  a , #1 , unload
          mov   LOAD , #0
          clr   P1_1                  ;load=1,data has been load to agv_mpu2
          mov   r1 , #0
 loop4:   mov   r0 , #0ffh
          djnz  r0 , $
          inc   r1
          cjne  r1 , #0ffh , loop4    ;delay time for 'move' signal
          setb  P1_1
;clear led display
          mov   a , #0
          mov   DPTR , #PORTB
          movx  @DPTR , a
          mov   DPTR , #PORTC
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
          movx  @DPTR , a
          ljmp  clear0
   unload: mov   DPTR , #22afh          ;send "data unload" to 286-AT
          ljmp  set
 ;-------------------------------------------------------;
 ; read status of agv from port1 and send data to 286-AT ;
 ;-------------------------------------------------------;
 do_read:
 ;-------------------------------------------------------------------;
 ; P1.6     ;   P1.5   ;  P1.4    ;   P1.3   ;   P1.2   ;   logic   ;
 ;cross-way ;   move   ;  battery ; obstacle ;  target  ;           ;
 ;----------;----------;----------;----------;----------;-----------;
 ;   no     ;   doing  ;  normal  ; no       ; not reach;     1     ;
 ; discover ;   finish ;  low     ; discover ; reach    ;     0     ;
 ;-------------------------------------------------------------------;
 ;check target status(P1_2)
          mov   LOWER , #0
          mov   r2 , #3fh
          lcall sub1                    ;read "\Target=" and store to 24xxh
          mov   c , P1_2
          jnc   next1
          mov   r2 , #75h               ;a=1:"not reach"
          sjmp  next2
  next1:  mov   r2 , #81h               ;a=0:"reach"
  next2:  lcall sub1                    ;read status and store to 24xxh
 ;check obstacle status(P1_3)
          mov   r2 , #49h
          lcall sub1                    ;read "\obstacle=" and store to 24xxh
          mov   c , P1_3
          jnc   next3
          mov   r2 , #0a9h              ;a=1:"no"
          sjmp  next4
  next3:  mov   r2 , #0aeh              ;a=0:"discover"
  next4:  lcall sub1                    ;read status and store to 24xxh
 ;check battery status(P1_4)
          mov   r2 , #55h
          lcall sub1                    ;read "\battery=" and store to 24xxh
          mov   c , P1_4
          jnc   next5
          mov   r2 , #92h               ;a=1:"normal"
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
          sjmp  next6
  next5:  mov   r2 , #9bh              ;a=0:"low"
  next6:  lcall subl                   ;read status and store to 24xxh
;check move status(P1_5)
          mov   r2 , #60h
          lcall subl                   ;read "\move=" and store to 24xxh
          mov   c , P1_5
          jnc   next7
          mov   r2 , #0a1h             ;a=1:"doing"
          sjmp  next8
  next7:  mov   r2 , #89h              ;a=0:"finish"
  next8:  lcall subl                   ;read status and store to 24xxh
;check cross-way status(P1_6)
          mov   r2 , #68h
          lcall subl                   ;read "\cross-way=" and store to 24xxh
          mov   c , P1_6
          jnc   next9
          mov   r2 , #0a9h             ;a=1:"no"
          sjmp  next10
  next9:  mov   r2 , #0aeh             ;a=0:"discover"
 next10:  lcall subl                   ;read status and store to 24xxh
          mov   DPH , #24h
          mov   DPL , LOWER            ;point to last character
          mov   a , #13
          movx  @DPTR , a              ;store "return" code to last character
          mov   DPTR , #231ffh
          mov   LOWER , #0
          ljmp  set
;-----------------------------------------------;
; Subroutine for read status from address 21xxh ;
; and store to address 24xxh                    ;
; together with accumulate number in LOWER      ;
;-----------------------------------------------;
   subl:  mov   DPH , #21h
          inc   r2
          mov   DPL , r2
          movx  a , @DPTR
          cjne  a , #13 , next
          ljmp  return2
  next:   mov   DPH , #24h
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง (ต่อ)

```
          mov   DPL , LOWER
          movx  @DPTR , a
          inc   LOWER
          ljmp  sub1
return2:  ret
;-------------------------------------;
;     test serial communication       ;
; send transmit message back to 286-AT ;
;-- ----- ------------------------------;
do_test:  mov   DPTR , #2304h
          ljmp  set
;-------------------------------------------------------;
;                INT0 interrupt handler                 ;
; receive falling edge of IRF A signal and transfer data ;
; from data stack adress 2500h - 251fh ( 32 byte )       ;
; 4 byte/1 operation : so we can define upto 8 operations ;
; to agv_mpu2 through port a by first-in first out method ;
;--- --- ------------------ ------------------------------;
          org   0a00h
int_int0: push  DPH
          push  DPL
          push  PSW
;set STBA = 1
          setb  P1_0
;check data stack empty?
          mov   r6 , #0ffh
     11:  mov   DPH , #25h
          mov   DPL , r6
          movx  a , @DPTR
          cjne  a , #0, 12              ;if data is not empty , go to 12
          dec   r6
          cjne  r6 , #0ffh , 11
          ljmp  ret_int0                ;data is empty
     12:  mov   DPTR , #2500h
          movx  a ,  @DPTR
          mov   STORE2 , a
          cjne  a , #2ah , 13           ;a = '*' ?
          mov   a , #0
          movx  @DPTR , a               ;clear contents in that address
          ljmp  send
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
     13:  inc   DPL
          movx  a , @DPTR
          cjne  a , #2ah , 14          ;a = '*' ?
          mov   a , #0
          movx  @DPTR , a              ;clear contents in that address
          dec   DPL
          mov   a , #2ah
          movx  @DPTR , a
          sjmp  send
     14:  dec   DPL
          movx  @DPTR , a
          inc   DPL
          ljmp  13
    send: mov   a , STORE2
          mov   DPTR , #PORTA
          movx  @DPTR , a
;send strobe signal signal ( 4 microsec )
          clr   P1_0
          mov   r6 , #1                ;2 microsec
          djnz  r6 , $                 ;2 microsec
          setb  P1_0
ret_int0: pop   PSW
          pop   DPL
          pop   DPH
          reti
;---------------------------------------------;
;            INT1 interrupt handler           ;
; receive signal moving complete from agv_mpu2 ;
;---------------------------------------------;
          org   0e00h
int_int1: mov   INT1 , #1
          reti
;---------------------------------------------;
;           RXD,TXD interrupt handler         ;
;           communication with 286-AT         ;
;---------------------------------------------;
          org   1000h
 serial:  jb    RI , receive          ;jump if RI = 1
          jb    TI , transmit         ;jump if TI = 1
          reti
```

โปรแกรมสื่อสารข้อมูลกับไมโครคอมพิวเตอร์, แปลรหัสและปฏิบัติตามคำสั่ง(ต่อ)

```
transmit: clr   TI                      ;clear TI in SCON
          inc   DPTR
          movx  a , @DPTR
          mov   SBUF , a
          reti
receive:  clr   RI                      ;clear RI in SCON
          inc   DPTR
          mov   a , DPL
          cjne  a , #0 , next11
          mov   a , SBUF
          cjne  a , #5ch , error
          sjmp  next12
 error:   dec   DPL
          dec   DPH
          reti
next11:   mov   a , SBUF
next12:   movx  @DPTR , a
          reti
```

โปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรงและควบคุมการหมุนแขนเซอร์โว

```
;*************************************************************;
; Program control position of DC motor and rotate Servo arms ;
;*************************************************************;
          cpu "8051.TBL"
          hof "INT8"
A:        equ     0e0h          ;Accumulator
B:        equ     0f0h          ;B Register
PSW:      equ     0d0h          ;Program Status Word
SP:       equ      81h          ;Stack Pointer
DPL:      equ      82h          ;Data Pointer (low byte)
DPH:      equ      83h          ;Data Pointer (high byte)
P0:       equ      80h          ;Port 0
P1:       equ      90h          ;Port 1
P2:       equ     0a0h          ;Port 2
P3:       equ     0b0h          ;Port 3
IP:       equ     0b8h          ;Interrupt Priority Control
IE:       equ     0a8h          ;Interrupt Enable Control
TMOD:     equ      89h          ;Timer/Counter Mode Control
TCON:     equ      88h          ;Timer/Counter Control
TH0:      equ      8ch          ;Timer/Counter 0 (high byte)
TL0:      equ      8ah          ;Timer/Counter 0 (low byte)
TH1:      equ      8dh          ;Timer/Counter 1 (high byte)
TL1:      equ      8bh          ;Timer/Counter 1 (low byte)
SCON:     equ      98h          ;Serial Control
SBUF:     equ      99h          ;Serial Data Buffer
PCON:     equ      87h          ;Power Control
;Accumulator
A_7:      equ     0e7h
A_6:      equ     0e6h
A_5:      equ     0e5h
A_4:      equ     0e4h
A_3:      equ     0e3h
A_2:      equ     0e2h
A_1:      equ     0e1h
A_0:      equ     0e0h
;B Register
B_7:      equ     0f7h
B_6:      equ     0f6h
B_5:      equ     0f5h
B_4:      equ     0f4h
```

โปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรงและควบคุมการหมุนแขนเซอร์โว(ต่อ)

```
B_3:        equ     0f3h
B_2:        equ     0f2h
B_1:        equ     0f1h
B_0:        equ     0f0h
;Program Status Word ( PSW , BIT7 - BIT0 )
CY:         equ     0d7h        ;Carry flag
F0:         equ     0d5h        ;flag 0 ( general purpose )
RS1:        equ     0d4h        ;Register bank selector bit1
RS0:        equ     0d3h        ;Register bank selector bit0
OV:         equ     0d2h        ;Over flow flag
P:          equ     0d0h        ;Parity flag
;Port 0 ( BIT7 - BIT0 )
P0_7:       equ      87h
P0_6:       equ      86h
P0_5:       equ      85h
P0_4:       equ      84h
P0_3:       equ      83h
P0_2:       equ      82h
P0_1:       equ      81h
P0_0:       equ      80h
;Port 1 ( BIT7 - BIT0 )
P1_7:       equ      97h
P1_6:       equ      96h
P1_5:       equ      95h
P1_4:       equ      94h
P1_3:       equ      93h
P1_2:       equ      92h
P1_1:       equ      91h
P1_0:       equ      90h
;Port 2 ( BIT7 - BIT0 )
P2_7:       equ      0a7h
P2_6:       equ      0a6h
P2_5:       equ      0a5h
P2_4:       equ      0a4h
P2_3:       equ      0a3h
P2_2:       equ      0a2h
P2_1:       equ      0a1h
P2_0:       equ      0a0h
;Port 3
P3_7:       equ      0b7h
```

โปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรงและควบคุมการหมุนแบบเซอร์โว (ต่อ)

```
P3_6:     equ     0b6h
P3_5:     equ     0b5h
P3_4:     equ     0b4h
P3_3:     equ     0b3h
P3_2:     equ     0b2h
P3_1:     equ     0b1h
P3_0:     equ     0b0h
;Interrupt Priority Control ( IP , bit7 - bit0 )
PS:       equ     0bch          ;Serial Port Interrupt priority level
PT1:      equ     0bbh          ;Timer 1 Interrupt priority level
PX1:      equ     0bah          ;External Interrupt 1 priority level
PT0:      equ     0b9h          ;Timer 0 Interrupt priority level
PX0:      equ     0b8h          ;External Interrupt 0 priority level
;Interrupt Enable Control ( IE ,bit7 - bit0 )
EA:       equ     0afh          ;Enable All Interrupts
ES:       equ     0ach          ;Serial Port interrupt
ET1:      equ     0abh          ;Timer 1 overflow interrupt
EX1:      equ     0aah          ;External Interrupt 1
ET0:      equ     0a9h          ;Timer 0 overflow interrupt
EX0:      equ     0a8h          ;External Interrupt 0
;Timer/Counter Control ( TCON , bit7 - bit0 )
TF1:      equ     8fh           ;Timer 1 overfow flag
TR1:      equ     8eh           ;Timer 1 run control bit
TF0:      equ     8dh           ;Timer 0 overfow flag
TR0:      equ     8ch           ;Timer 0 run control bit
IE1:      equ     8bh           ;External Interrupt 1 edge flag
IT1:      equ     8ah           ;Interrupt 1 type control bit
IE0:      equ     89h           ;External Interrupt 0 edge flag
IT0:      equ     88h           ;Interrupt 0 type control bit
;Serial Control ( SCON , bit7-bit0 )
SM0:      equ     9fh           ;Serial port mode specifier
SM1:      equ     9eh           ;Serial port mode specifier
SM2:      equ     9dh           ;Enables multiprocessor communication
REN:      equ     9ch           ;Enable Reception
TB8:      equ     9bh           ;
RB8:      equ     9ah           ;
TI:       equ     99h           ;Transmit interrupt flag
RI:       equ     98h           ;Receive Interrupt flag
;Additional declaration
PORTA:    equ     4000h
```

โปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรงและควบคุมการหมุนแขนเซอร์โว(ต่อ)

```
PORTB:    equ       4001h
PORTC:    equ       4002h
CONTRL:   equ       4003h
LOBYTE:   equ       70h
HIBYTE:   equ       71h
MROTATE:  equ       72h
PHASE_B:  equ       73h
LOCOUNT:  equ       74h
HICOUNT:  equ       75h
PBSTATUS: equ       76h
LOWER:    equ       77h
STEERING: equ       78h
CHECK:    equ       79h
          org   0000h
          ljmp  power_up
;-------------------------;
; int0 interrupt routine ;
;-------------------------;
          org   0003h
          ljmp  int_int0
;-----------------------;
; T0 interrupt routine ;
;-----------------------;
          org   000bh
          ljmp  int_t0
;---------------------------------;
; accerelate/decerelate data table ;
;---------------------------------;
          org   0100h
          dfb   080h,08bh,08fh,093h,096h,099h,09bh,09dh
          dfb   09fh,0a1h,0a3h,0a5h,0a7h,0a8h,0aah,0abh
          dfb   0adh,0aeh,0afh,0b1h,0b2h,0b3h,0b4h,0b6h
          dfb   0b7h,0b8h,0b9h,0bah,0bbh,0bch,0bdh,0beh
          dfb   0bfh,0c0h,0c1h,0c2h,0c3h,0c4h,0c5h,0c6h
          dfb   0c7h,0c8h,0c9h,0c9h,0cah,0cbh,0cch,0cdh
          dfb   0ceh,0ceh,0cfh,0d0h,0d1h,0d2h,0d2h,0d3h
          dfb   0d4h,0d5h,0d5h,0d6h,0d7h,0d8h,0d8h,0d9h
          dfb   0dah,0dah,0dbh,0dch,0dch,0ddh,0deh,0deh
          dfb   0dfh,0e0h,0e0h,0e1h,0e2h,0e2h,0e3h,0e4h
          dfb   0e4h,0e5h,0e6h,0e6h,0e7h,0e7h,0e8h,0e9h
```

โปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรงและควบคุมการหมุนแขนเซอร์โว(ต่อ)

```
        dfb   0e9h,0eah,0eah,0ebh,0ech,0ech,0edh,0edh
        dfb   0eeh,0eeh,0efh,0f0h,0f0h,0f1h,0f1h,0f2h
        dfb   0f2h,0f3h,0f4h,0f4h,0f5h,0f5h,0f6h,0f6h
        dfb   0f7h,0f7h,0f8h,0f8h,0f9h,0f9h,0fah,0fah
        dfb   0fbh,0fbh,0fch,0fch,0fdh,0fdh,0feh,0ffh
        dfb   000h,001h,001h,002h,002h,003h,003h,004h
        dfb   004h,005h,005h,006h,006h,007h,007h,008h
        dfb   008h,009h,009h,00ah,00ah,00bh,00bh,00ch
        dfb   00dh,00dh,00eh,00eh,00fh,00fh,010h,011h
        dfb   011h,012h,012h,013h,013h,014h,015h,015h
        dfb   016h,016h,017h,018h,018h,019h,019h,01ah
        dfb   01bh,01bh,01ch,01dh,01dh,01eh,01fh,01fh
        dfb   020h,021h,021h,022h,023h,023h,024h,025h
        dfb   025h,026h,027h,027h,028h,029h,02ah,02ah
        dfb   02bh,02ch,02dh,02dh,02eh,02fh,030h,031h
        dfb   031h,032h,033h,034h,035h,036h,036h,037h
        dfb   038h,039h,03ah,03bh,03ch,03dh,03eh,03fh
        dfb   040h,041h,042h,043h,044h,045h,046h,047h
        dfb   048h,049h,04bh,04ch,04dh,04eh,050h,051h
        dfb   052h,054h,055h,057h,058h,05ah,05ch,05eh
        dfb   060h,062h,064h,066h,069h,06ch,070h,074h
;---------------;
;  main program ;
;---------------;
        org   0210h
power_up: mov   r1 , #0
  delay0: mov   r0 , #0ffh
        djnz  r0 , $
        inc   r1
        cjne  r1 , #0fh , delay0      ;delay time for power-up
        mov   P1 , #80h               ;initialize D/A at 0 volt
;-----------------------------------------------------------;
;  set 8255 to mode 1                                       ;
;  group A mode 1 ( strobe input ) , group B mode 0         ;
;  port a : input                                           ;
;  port b : output                                          ;
;  port c : pc0-pc2 = input                                 ;
;               pc3 = INTRA                                 ;
;               pc4 = STBA                                  ;
;               pc5 = IBFA                                  ;
```

โปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรงและควบคุมการหมุนแบบเซอร์โว(ต่อ)

```
;           pc6-pc7 = input                                    ;
;                D7 = 0   ;define set/reset bit                ;
;                   = 1   ;define operation *                 ;
;   group A:  D6,D5 = 0,0 ;mode 0                              ;
;                   = 0,1 ;mode 1 *                           ;
;                   = 1,x ;mode 2                             ;
;                D4 = 0   ;port a is output                   ;
;                   = 1   ;port a is input *                  ;
;                D3 = 0   ;pc6,pc7 is output                  ;
;                   = 1   ;pc6,pc7 is input *                 ;
;   group B:     D2 = 0   ;mode 0 *                            ;
;                   = 1   ;mode 1                             ;
;                D1 = 0   ;port b is output *                 ;
;                   = 1   ;port b is input                    ;
;                D0 = 0   ;pc0,pc1,pc2 is output              ;
;                   = 1   ;pc0,pc1,pc2 is input *             ;
;-------------------------------------------------------------;
         mov   dptr , #CONTRL
         mov   a , #10111001b
         movx  @dptr , a
;-------------------------------------------------------------------------;
; for 8255 mode1 , we must be set INTE A = 1 ( enable 8255 interrupt )    ;
; CONTRL:   D7        D6 D5 D4     D3 D2 D1          D0                   ;
;        set/reset   don't care    select bit     logic to be set        ;
;          0         0  0  0       (bit pc4)          1                  ;
;-------------------------------------------------------------------------;
         mov   a , #00001001b
         movx  @DPTR , a
;-----------------------------------------;
; set interrupt enable                    ;
; IE:EA,-,ET2,ES,ET1,EX1,ET0,EX0          ;
; description IE                          ;
;   EA  = disable(0)/enable(1)  all interrupt ;
;   ET2 = 8052 only                       ;
;   ES  = disable(0)/enable(1) TXD,RXD    ;
;   ET1 = disable(0)/enable(1) T1         ;
;   EX1 = disable(0)/enable(1) INT1       ;
;   ET0 = disable(0)/enable(1) T0         ;
;   EX0 = disable(0)/enable(1) INT0       ;
;-----------------------------------------;
```

โปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรงและควบคุมการหมุนแบบเซอร์โว(ต่อ)

```
;disable all interrupt
          mov   IE , #00000000b
;--------------------------------;
; set interrupt priority        ;
; high level      lowlevel      ;
;     T0          INT0      high;
;                 INT1          ;
;                 T1            ;
;                 RXD,TXD   low ;
;--------------------------------;
          mov   IP , #00000010b
;-------------------------------------------------------------------------;
; prepare for T0 interrupt,T1 interrupt and some condition for INT0,INT1  ;
; - TMOD: GATE,C/T,M1,M0,GATE,C/T,M1,M0                                   ;
;         ----timer1---- ----timer0----                                   ;
;   description TMOD ( x = 0 or 1 )                                       ;
;   GATE: 0    not allow INTx to control Tx pin                           ;
;         1    allow INTx to control Tx pin                               ;
;    C/T: 0    select  timer x                                            ;
;         1    select  counter x                                          ;
;  M1,M0: 1,0  8-bit auto-reload THx holds a value which is to be reload  ;
;              into TLx each time it overflow                             ;
; - TCON: TF1,TR1,TF0,TR0,IE1,IT1,IE0,IT0                                 ;
;   description TCON  (set=1,clear=0)                                     ;
;     TF1: T1 overflow flag set/clear by hardware                         ;
;    *TR1: T1 control bit   set/clear by software                         ;
;     TF0: T0 overflow flag set/clear by hardware                         ;
;    *TR0: T0 control bit   set/clear by software                         ;
;     IE1: INT1 edge flag   set when falling edge signal is received      ;
;    *IT1: INT1 signal type control bit,  1=falling edge , 0=low level    ;
;     IE0: INT0 edge flag   set when high to low edge signal is received  ;
;    *IT0: INT0 signal type control bit,  1=falling edge , 0=low level    ;
;-------------------------------------------------------------------------;
;select counter 0 , counter 1 , set to mode 2 and let GATE = 0
          mov   TMOD , #01100110b
;set TR0,TR1 = 1 and set INT1,INT0 for detect falling edge signal
          mov   TCON , #01010101b
;define value for counter 0
          mov   TL0 , #0ffh             ;define value in TL0
          mov   TH0 , #0ffh             ;define auto-reload value of mode 2
```

โปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรงและควบคุมการหมุนแขนเซอร์โว(ต่อ)

```
;define value for counter 1
          mov   TL1 , #0ffh             ;define value in TL1
          mov   TH1 , #0ffh             ;define auto-reload value pf mode 2
;create zero signal to agv_mpu1
          mov   PBSTATUS , #00010101b
          mov   a , PBSTATUS
          mov   DPTR , #PORTB
          movx  @DPTR , a
          mov   r1 , #0
pb5delay: mov   r0 , #0ffh
          djnz  r0 , $
          inc   r1
          cjne  r1 , #1fh , pb5delay
;-------------------------------------------------------------------------;
;    PB7    ;   PB6     ;    PB5  ;  PB4  ;  PB3 ;  PB2  ; PB1  ;   PB0    ;
;operation  ; error     ;send to  ;detect ; turn ; normal; turn ; detect    ;
;display    ; display   ;INT1 mpu1;sensor1; right;forward; left ;guide way ;
;-----------;-----------;---------;-------;------;-------;------;----------;
;normal "0" ;normal "0" ; falling ;no "0" ;no "0"; no "0";no "0"; no  "0"  ;
; busy  "1" ; occur "1" ; edge    ;yes "1";yes"1"; yes"1";yes"1"; yes "1"  ;
;-------------------------------------------------------------------------;
  start:  mov   PBSTATUS , #00110101b
          mov   P1 , #80h               ;initialize D/A at 0 volt
          mov   a , PBSTATUS
          mov   DPTR , #PORTB
          movx  @DPTR , a               ;define initial status at port b
          mov   LOWER , #0ffh
          mov   CHECK , #0
;-------------------------------------------------;
; enable INT0 interrupt ( wait for INTRA signal ) ;
;-------------------------------------------------;
          mov   TCON , #01010101b
          mov   IE , #10000001b
  wait1:  mov   a , CHECK
          cjne  a , #2ah , wait1
;-------------------------;
; disable INT0 interrupt ;
;-------------------------;
          mov   IE , #00000000b
          orl   PBSTATUS , #10000000b  ;display agv_mpu2 is busy at PB7
```

โปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรงและควบคุมการหมุนแบบเซอร์โว(ต่อ)

```
          mov   DPTR , #PORTB
          mov   a , PBSTATUS
          movx  @DPTR , a
;wait for 'move' signal at PC0
  wait2:  mov   DPTR , #PORTC
          movx  a , @dptr
          anl   a , #00000001b          ;and a with #00000001b
          jnz   wait2                   ;if signal =1 ,wait2
;transfer data from adress 2000-20xxh to 2500-25xxh
          mov   DPL , #0
     n1:  mov   DPH , #20h
          movx  a , @DPTR
          mov   DPH , #25h
          movx  @DPTR , a
          cjne  a , #2ah , n2
          sjmp  n3
     n2:  inc   DPL
          ljmp  n1
;load data from data storage address 2500h - 25xxh
     n3:  mov   r3 , #3
;begin to run
     n4:  mov   DPH , #25h
          mov   DPL , r3
          movx  a , @DPTR
          cjne  a , #10h , n5           ;if a <> "p or P" go to n5
          sjmp  posit
     n5:  cjne  a , #14h , n6           ;if a <> "t or T" go to n6
          ljmp  turn
     n6:  cjne  a , #2ah , n7           ;if a <> "*" go to n7
;a="*" , clear contents in this adress ( 25xxh )
          mov   a , #0
          movx  @DPTR , a
;wait until move signal at PC0 = 1
  wait3:  mov   dptr , #PORTC
          movx  a , @dptr
          anl   a , #00000001b          ;and a with #00000001b
          jz    wait3                   ;if signal =0 ,wait3
          mov   r2 , #05h
  delay1: mov   r1 , #0ffh
  delay2: mov   r0 , #0ffh
```

โปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรงและควบคุมการหมุนแขนเซอร์โว(ต่อ)

```
        djnz  r0 , $
        dec   r1
        cjne  r1 , #0 , delay2
        dec   r2
        cjne  r2 , #0 , delay1
;send interrupt signal to INT1 of agv_mpu1 by create falling edge signal
        orl   PBSTATUS , #00100000b
        mov   a , PBSTATUS
        mov   DPTR , #PORTB
        movx  @DPTR , a              ;make sure that PB5 = 1
        anl   a , #11011111b         ;make falling edge signal at PB5
        mov   PBSTATUS , a
        movx  @DPTR , a              ;send interrupt signal
        mov   r1 , #0
delay3: mov   r0 , #0ffh
        djnz  r0 , $
        inc   r1
        cjne  r1 , #0fh , delay3     ;time delay
        orl   PBSTATUS , #00100000b
        mov   a , PBSTATUS
        mov   DPTR , #PORTB
        movx  @DPTR , a              ;set PB5 to 1
        ljmp  start
;display invalid function by set PB6
   n7:  orl   PBSTATUS , #01000000b
        mov   a , PBSTATUS
        mov   DPTR , #PORTB
        movx  @DPTR , a
        sjmp  $                      ;system halt
;------------------------------;
; do function = "p or P" ;
;-------- - -- ---------;
posit:  mov   a , #0                 ;clear contents in that address
        movx  @DPTR , a
;read the rotation of motor and store to MROTATE
        dec   DPL
        movx  a , @DPTR
        cjne  a , #30h , defrot
        mov   MROTATE , #0
        sjmp  label0
```

โปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรงและควบคุมการหมุนแขนเซอร์โว(ต่อ)

```
 defrot: mov   MROTATE , #1
 label0: mov   a , #0                  ;clear contents in that address
         movx  @DPTR , a
;read upper byte position command and store to HIBYTE
         dec   DPL
         movx  a , @DPTR
         mov   HIBYTE , a
         mov   a , #0                  ;clear contents in that address
         movx  @DPTR , a
;read lower byte position command and store to LOBYTE
         dec   DPL
         movx  a , @DPTR
         mov   LOBYTE , a
         mov   a , #0                  ;clear contents in that address
         movx  @DPTR , a
;define initial condition
         mov   PHASE_B , #0
         mov   LOCOUNT , #0
         mov   HICOUNT , #0
;bigin to run
         mov   a , HIBYTE
         jnz   label1                  ;jump to label1 if HIBYTE <> #0
         mov   a , LOBYTE
         jz    label5                  ;jump to label5 if LOBYTE = #0
         clr   c
         subb  a , #07fh               ;a = a - c - #data
                                       ;if a < #data , borrow(carry) flag is set
         jc    label3                  ;jump to label3 if LOBYTE < #7fh
 label1: mov   a , MROTATE
         jz    label2                  ;jump to label2 if MROTATE = 0
         mov   DPL , #80h
         ljmp  label6
 label2: mov   DPL , #7fh
         ljmp  label6
 label3: mov   a , MROTATE
         jz    label4                  ;jump to label4 if MROTATE = 0
         mov   a , LOBYTE
         cpl   a                       ;complement a
         mov   DPL , a
         ljmp  label6
```

โปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรงและควบคุมการหมุนแบบเซอร์โว(ต่อ)

```
 label4:  mov   DPL , LOBYTE
          ljmp  label6
 label5:  mov   DPL , #0
;-------------------------;
; enable T0,INT0,INT1,T1 ;
;-------------------------;
 label6:  mov   TCON , #01010101b
          mov   IE , #10001111b
;send speed command from table to D/A
          mov   DPH , #01h
          mov   a , #0
          movc  a , @a + DPTR
          mov   P1 , a
;wait until position command is zero
  wait4:  mov   a , HIBYTE
          cjne  a , #0 , wait4
  wait5:  mov   a , LOBYTE
          cjne  a , #0 , wait5
;-------------------------;
; disable all interrupt ;
;-------------------------;
          mov   IE , #00000000b
;send speed command from table to D/A
          mov   DPTR , #0100h
          mov   a , #0
          movc  a , @a + DPTR
          mov   P1 , a
;delay time
 label7:  mov   r2 , #0fh
 delay4:  mov   r1 , #0fffh
 delay5:  mov   r0 , #0ffh
          djnz  r0 , $
          djnz  r1 , delay5
          djnz  r2 , delay4
;increate pointer to data storage
          clr   c
          mov   a , #4
          addc  a , r3
          mov   r3 , a
;go to receive next command
```

โปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรงและควบคุมการหมุนแขนเซอร์โว(ต่อ)

```
        ljmp  n4
; ------ ---------------;
; do Function "L or T" ;
; ---------------------;
   turn: mov   a , #0
         movx  @DPTR , a            ;clear contents in that address
         dec   DPL
         movx  a , @DPTR
         cjne  a , #0ch , t1        ;if a <> "l or L" go to t1
;turn left
         mov   a , #0
         movx  @DPTR , a            ;clear contents in that address
         mov   a , PBSTATUS
         anl   a , #11110000b       ;inhibit guide way,turn right and normal
         orl   a , #00000010b       ;turn left
         mov   PBSTATUS , a
         mov   DPTR , #PORTB
         movx  @DPTR , a            ;output signal to servo system
         ljmp  label7
    t1:  cjne  a , #12h , t2        ;if a <> "r or R" go to t2
;turn right
         mov   a , #0
         movx  @DPTR , a            ;clear contents in that address
         mov   a , PBSTATUS
         anl   a , #11110000b       ;inhibit guide way,turn left and normal
         orl   a , #00001000b       ;turn right
         mov   PBSTATUS , a
         mov   DPTR , #PORTB
         movx  @DPTR , a            ;output signal to servo system
         ljmp  label7
;display invalid steering by set PB6 ,clear PB7
    t2:  mov   a , PBSTATUS
         anl   a , #01111111b
         orl   a , #01000000b
         mov   PBSTATUS , a
         mov   DPTR , #PORTB
         movx  @DPTR , a
         sjmp  $                    ;system halt
```

โปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรงและควบคุมการหมุนแขนเซอร์โว(ต่อ)

```
;-------------------------------------------------------------;
;       int0 interrupt routine ( receive INTR A signal )      ;
;  read data from port a and store to data area add 2000-20ffh  ;
;  ---  ---------------------------------------------------- ---;
          org   0900h
int_int0: push  DPH
          push  DPL
          push  PSW
;read data from agv_mpu1 , reset INTRA,IBFA automatically
          mov   a , LOWER
          cjne  a , #0ffh , 11
          mov   DPTR , #PORTA
          movx  a , @DPTR               ;only want to create read signal
          inc   LOWER
          ljmp  ret_int0
     11:  mov   DPTR , #PORTA
          movx  a , @DPTR
          mov   DPH , #20h
          mov   DPL , LOWER
          movx  @DPTR , a               ;store data to adress 20xxh
          mov   CHECK , a
     12:  inc   LOWER
          mov   a , LOWER
          cjne  a , #0ffh , ret_int0    ;if address = 20ffh "data is full"
          orl   PBSTATUS , #01000000b   ;display error to led at PB6
          mov   DPTR , #PORTB
          mov   a , PBSTATUS
          movx  @DPTR , a
ret_int0: pop   PSW
          pop   DPL
          pop   DPH
          reti
;-------------------------------------;
;          T0 interrupt routine       ;
; ( receive signal from rotary encoder) ;
; ------------------------------------;
          org   0c00h
 int_t0:  push  PSW
          push  DPH
          push  DPL
```

โปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรงและควบคุมการหมุนแขนเซอร์โว(ต่อ)

```
          push  a
;ckeck current direction at PC7
          mov   dptr , #PORTC
          movx  a , @dptr
          anl   a , #10000000b
          jz    lab1                ;jump to lab1 if pc7 = 0
          mov   PHASE_B , #1        ;curent rotation is c.w
          sjmp  next1
   lab1:  mov   PHASE_B , #0        ;curent rotation is c.c.w
  next1:  mov   a , MROTATE         ;check the rotation command
          jz    lab2                ;jump to lab2 if MROTATE = 0 (c.w)
          mov   a , PHASE_B
          jnz   next2               ;jump to next2 if PHASE_B = 1 (c.c.w)
          sjmp  lab3                ;jump to lab3 because it get wrong rotation
   lab2:  mov   a , PHASE_B
          jz    next2               ;jump to next2 if PHASE_B = 0 (c.w)
;increate position command 1 pulse
   lab3:  clr   c
          mov   a , LOBYTE
          addc  a , #1h             ;a = a + c + #1
          mov   LOBYTE , a
          mov   a , HIBYTE
          addc  a , #0h             ;a = a + c + #0
          mov   HIBYTE , a
          clr   c
;decreate position counter 1 pulse
          dec   LOCOUNT
          mov   a , LOCOUNT
          cjne  a , #0ffh , ret_t0
          mov   a , HICOUNT
          cjne  a , #0h , x
          inc   LOCOUNT
          ljmp  ret_t0
       x: dec   HICOUNT
          ljmp  ret_t0
  next2:  mov   a , HIBYTE
          jnz   lab9                ;jump to lab9 if HIBYTE <> #0
          mov   a , LOBYTE
          jz    lab10               ;jump to lab10 if LOBYTE = #0
          clr   c
```

โปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรงและควบคุมการหมุนแขนเซอร์โว(ต่อ)

```
        subb   a , #07fh              ;a = a - c - #data
                                      ;if a < #data , borrow(carry) flag is set
        jc     lab7                   ;jump to lab7 if LOBYTE < #7fh
lab4:   dec    LOBYTE
lab5:   mov    a , MROTATE
        jz     lab6                   ;jump to lab6 if MROTATE = 0
        mov    DPL , #80h
        ljmp   lab11
lab6:   mov    DPL , #7fh
        ljmp   lab11
lab7:   dec    LOBYTE
        mov    a , MROTATE
        jz     lab8                   ;jump to lab8 if MROTATE = 0
        mov    a , LOBYTE
        cpl    a                      ;complement a
        mov    DPL , a
        ljmp   lab11
lab8:   mov    DPL , LOBYTE
        ljmp   lab11
lab9:   mov    a , LOBYTE
        jnz    lab4                   ;jump to lab4 if LOBYTE <> #0
        dec    HIBYTE                 ;borrow high byte position command
        mov    LOBYTE , #0ffh         ;to low byte position command
        ljmp   lab5
lab10:  mov    DPL , #0
        ljmp   lab12
lab11:  clr    c
        mov    a , LOCOUNT
        addc   a , #1
        mov    LOCOUNT , a
        mov    a , HICOUNT
        addc   a , #0
        mov    HICOUNT , a
;send speed command from table to D/A
lab12:  mov    DPH , #01h
        mov    a , #0
        movc   a , @a + dptr
        mov    P1 , a
ret_10: pop    a
        pop    DPL
```

โปรแกรมควบคุมตำแหน่งมอเตอร์กระแสตรงและควบคุมการหมุนแขนเซอร์โว (ต่อ)

```
pop   DPH
pop   PSW
reti
```

ภาคผนวก ข

แนวทางพัฒนา AGV ในขั้นต่อไป

ภาคผนวก ช
แนวทางพัฒนา AGV ในขั้นต่อไป

โครงการสร้าง AGV ที่เกี่ยวข้องกับวิทยานิพนธ์

จากวิทยานิพนธ์ที่ได้ทำการศึกษาระบบควบคุมในส่วนต่างๆของ AGV จึงมีความเป็นไปได้ที่จะนำมาสร้าง AGV จริง ลักษณะของ AGV ตามโครงการจะสร้างขึ้นมาเพื่อใช้สำหรับทำงานภายในโรงงาน การขับเคลื่อนจะใช้มอเตอร์กระแสตรงซึ่งควบคุมตำแหน่งด้วยไมโครคอนโทรลเลอร์และควบคุมความเร็วด้วยอนาล็อค PI คอนโทรลเลอร์ ลักษณะโครงสร้างของ AGV ประกอบด้วยล้อ 4 ล้อ ขับเคลื่อน (drive) ด้วยล้อหลัง สำหรับการเลี้ยว (steering) ใช้ล้อหน้าซึ่งควบคุมการเลี้ยวด้วยชุดเซอร์โวซึ่งได้รับสัญญาณควบคุมจากไมโครคอนโทรลเลอร์เช่นกัน AGV จะถูกควบคุมให้เคลื่อนที่ไปตามเส้นทางนำร่อง (guide path) ที่กำหนดไว้ สำหรับเทคนิคการนำร่องจะใช้โฟโต้เซนเซอร์ตรวจหาแนวแถบสีเข้มซึ่งทาเป็นแนวอยู่ตามพื้นอาคาร และยังใช้เทคนิคเดียวกันตรวจหาตำแหน่งทางแยกและตำแหน่งสถานีเป้าหมายอีกด้วย

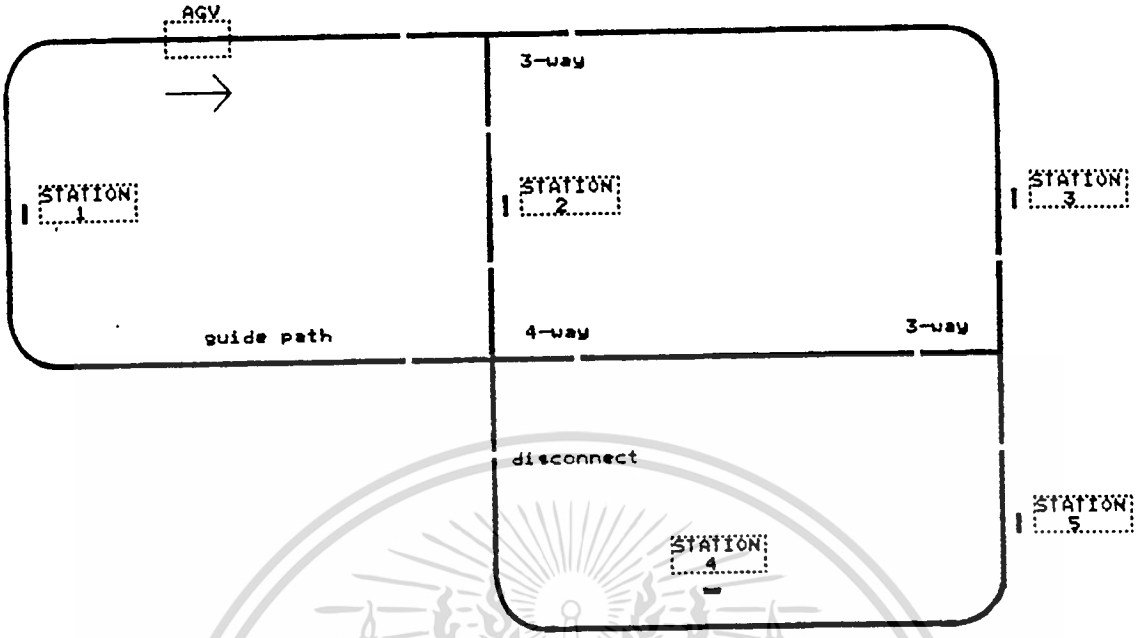
AGV สามารถติดต่อสื่อสารกับคอมพิวเตอร์ส่วนกลาง (central computer) ได้ การสื่อสารจะใช้วิธีส่งข้อมูลอนุกรมแบบอะซิงโครนัส อาร์ฟดูเพล็กซ์ ผ่านคลื่นวิทยุ AM ย่านความถี่ 27 MHz

การทำงานของ AGV จะเริ่มจากผู้ใช้ (user) ซึ่งอยู่ที่สถานีปฏิบัติงาน (work station) ทำการกดคีย์เรียก AGV สัญญาณกดคีย์จะถูกส่งตามสาย (wire) ไปเข้าคอมพิวเตอร์ส่วนกลาง

เมื่อคอมพิวเตอร์รับทราบแล้วก็จะทำการคำนวณหาวิถีการเคลื่อนที่ต่อไป จากนั้นจะส่งคำสั่งพร้อมข้อมูลการเคลื่อนที่ อาทิเช่น ระยะทาง , ทิศทางการเลี้ยว เป็นต้น ไปยัง AGV ลักษณะของคำสั่งและข้อมูลการเคลื่อนที่จะอยู่ในรูปข้อมูลอนุกรมรหัสแอสกี้ (ASCII code) จากนั้นคอมพิวเตอร์จะสั่งให้ AGV เริ่มทำงาน AGV จะเคลื่อนที่ไปตามเส้นทางที่กำหนดไว้ ดังตัวอย่างรูปที่ ข.1 ด้วยความเร็วคงที่และไปหยุดยังสถานีเป้าหมาย ในการควบคุมตำแหน่งใช้ไมโครคอนโทรลเลอร์ทำหน้าที่นับพัลซ์ของระยะทางที่ได้จากโรตารี่เอนโคดเดอร์ (rotary encoder) แล้วทำการประมวลผลเพื่อหาค่าความเร็วที่เหมาะสม (speed command) เพื่อส่งต่อไปให้ตัวควบคุม PI ต่อไป

ในระหว่างที่ AGV เคลื่อนที่จะมีการตรวจหาตำแหน่งสถานีเป้าหมายตลอดจนตรวจหาตำแหน่งของทางแยกที่ต้องผ่าน กรณีถ้ามีวัตถุกีดขวางบนเส้นทาง AGV จะหยุดรอจนกว่าจะตรวจพบว่าวัตถุนั้นเคลื่อนย้ายออกไปแล้วจึงจะเคลื่อนที่ต่อไปได้ เมื่อ AGV เคลื่อนที่ถึงตำแหน่งสถานีเป้าหมายแล้ว AGV จะส่งข้อความตอบกลับไปยังคอมพิวเตอร์ส่วนกลาง เพื่อแสดงการว่าการทำงานเสร็จสิ้นลง นอกจากนี้สภาวะ (status) ต่างๆของ AGV สามารถเรียกดูได้จากคอมพิวเตอร์ได้โดยตรง

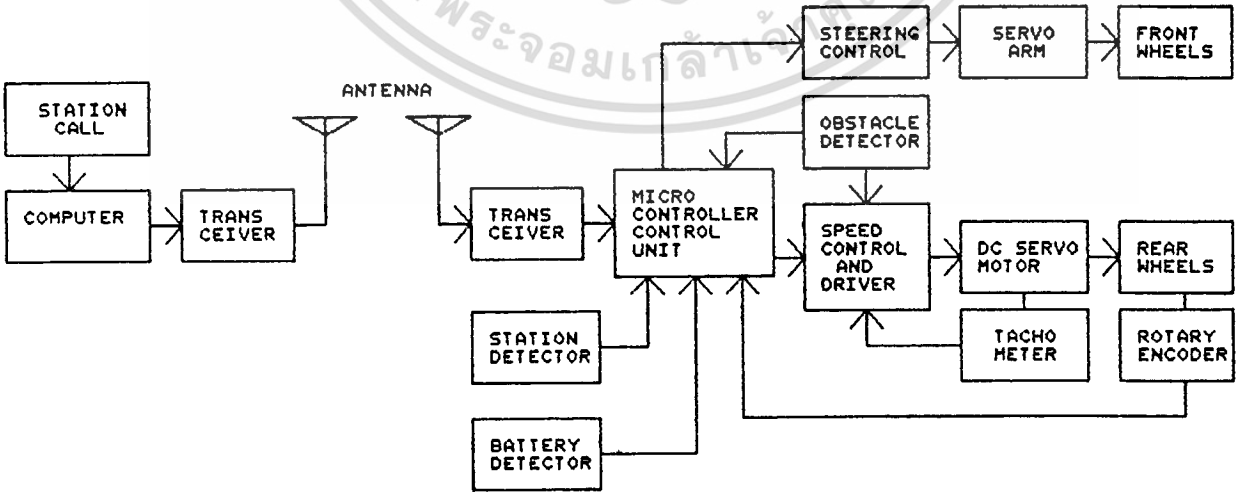
กรณีที่มีการเรียกใช้ AGV จากหลายสถานี AGV จะเคลื่อนที่ไปหาทีละสถานี โดยเรียงลำดับการ



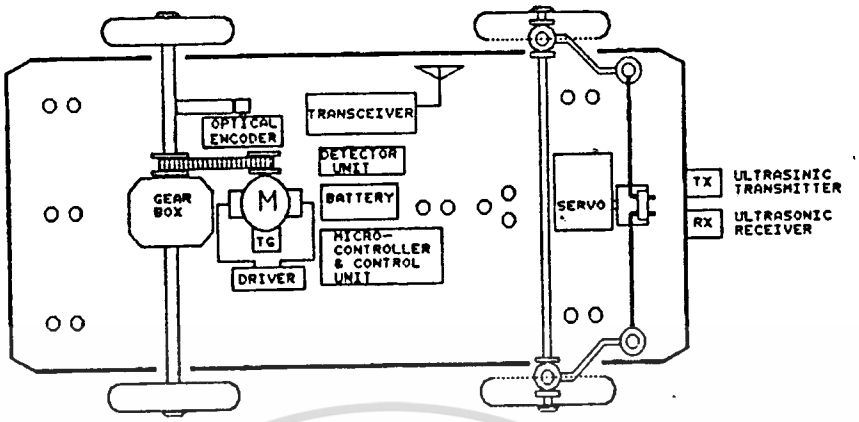
รูปที่ ข.1 ตัวอย่างทางเดินนำร่อง,ทางแยกและตำแหน่งสถานีปฏิบัติงาน

เรียกก่อนหลัง นอกจากนี้ AGV ยังมีระบบตรวจวัดแรงดันของแบตเตอรี่อีกด้วย

ในรูปที่ ข.2 แสดงบล็อคไดอะแกรมของ AGV ทั้งระบบ ส่วนในรูปที่ ข.3 แสดงรูปลักษณะของ AGV ตามโครงการ



รูปที่ ข.2 บล็อคไดอะแกรมแสดงระบบ AGV

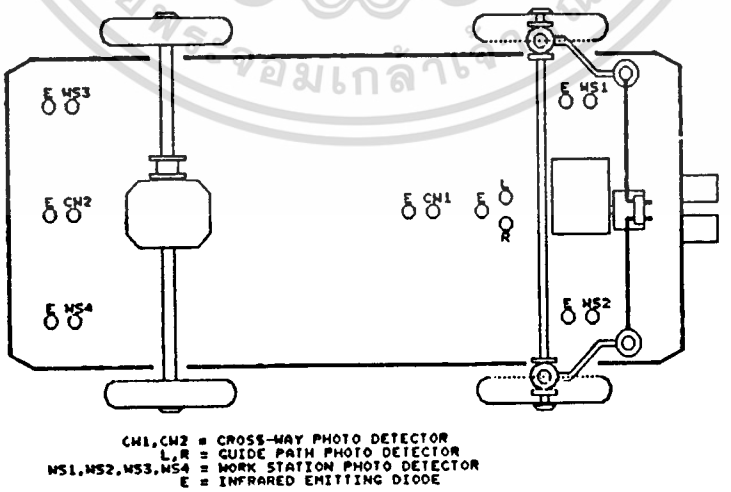


รูปที่ ข.3 แสดงรูปลักษณะของ AGV ตามโครงการ

ตำแหน่งโฟโต้เซนเซอร์บน AGV

ในรูปที่ ข.4 แสดงตำแหน่งโฟโต้เซนเซอร์ที่ติดตั้งบน AGV โฟโต้เซนเซอร์แต่ละคู่มีวงจรตรวจจับดังรูปที่ 3.23 ซึ่งได้อธิบายไปแล้ว แต่มีหน้าที่ในการทำงานต่างกันกล่าวคือ

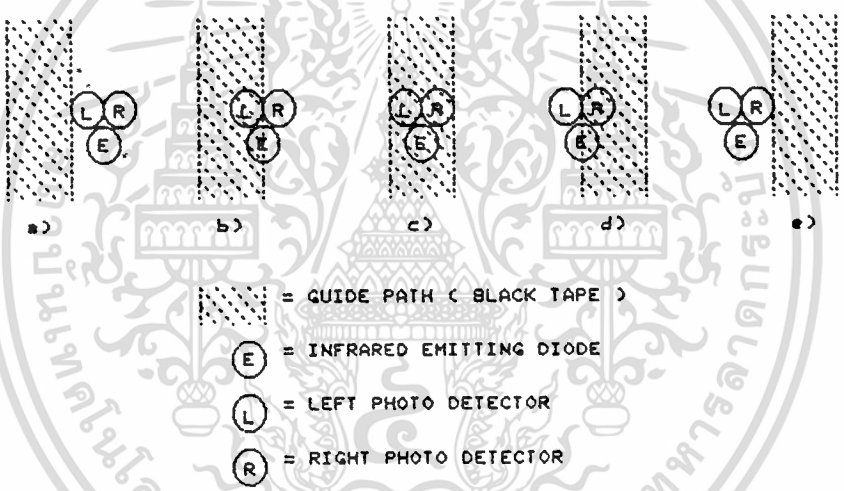
- โฟโต้ดีเทคเตอร์ L , R ทำหน้าที่ตรวจหาตำแหน่งเส้นทางนำร่อง
- โฟโต้ดีเทคเตอร์ CW1 , CW2 ทำหน้าที่ตรวจหาตำแหน่งทางแยก
- โฟโต้ดีเทคเตอร์ WS1 , WS2 , WS3 และ WS4 ทำหน้าที่ตรวจหาตำแหน่งสถานีปฏิบัติงาน



รูปที่ ข.4 ตำแหน่งโฟโตเซนเซอร์ที่ติดตั้งบน AGV

วิธีการนำร่อง

โฟโต้ดีเทคเตอร์ L , R ทำหน้าที่ตรวจหาตำแหน่งเส้นทาง (แนวเทปสีดำ) แล้วส่งค่าแรงดันเอาท์พุทไปยังวงจรควบคุมการเลี้ยว จากรูปที่ ข.5a) เป็นตำแหน่งขณะที่ AGV เคลื่อนที่กินทางขวามากเกินไป กรณีนี้แรงดันเอาท์พุทที่ได้จากโฟโต้ดีเทคเตอร์ L และ R จะเป็น "0" ทั้งคู่ ถัดมาคือรูปที่ ข.5b) กรณีนี้แรงดันเอาท์พุทที่ได้จากโฟโต้ดีเทคเตอร์ L จะเป็น "1" ส่วนโฟโต้ดีเทคเตอร์ R จะเป็น "0" ถัดมาคือรูปที่ ข.5c) แรงดันเอาท์พุทที่ได้จากโฟโต้ดีเทคเตอร์ L และ R จะเป็น "1" ทั้งคู่ ถัดมาคือรูปที่ ข.7d) แสดงถึงกรณีที่ตำแหน่ง AGV เริ่มกินทางซ้ายมาก แรงดันเอาท์พุทที่ได้จากโฟโต้ดีเทคเตอร์ L และ R จะเป็น "0" และ "1" ตามลำดับ สำหรับรูปสุดท้ายคือ ข.5e) แสดงตำแหน่งขณะ AGV กินทางซ้ายมากเกินไป กรณีนี้แรงดันเอาท์พุทที่ได้จากโฟโต้ดีเทคเตอร์ L และ R จะเป็น "0" ทั้งคู่

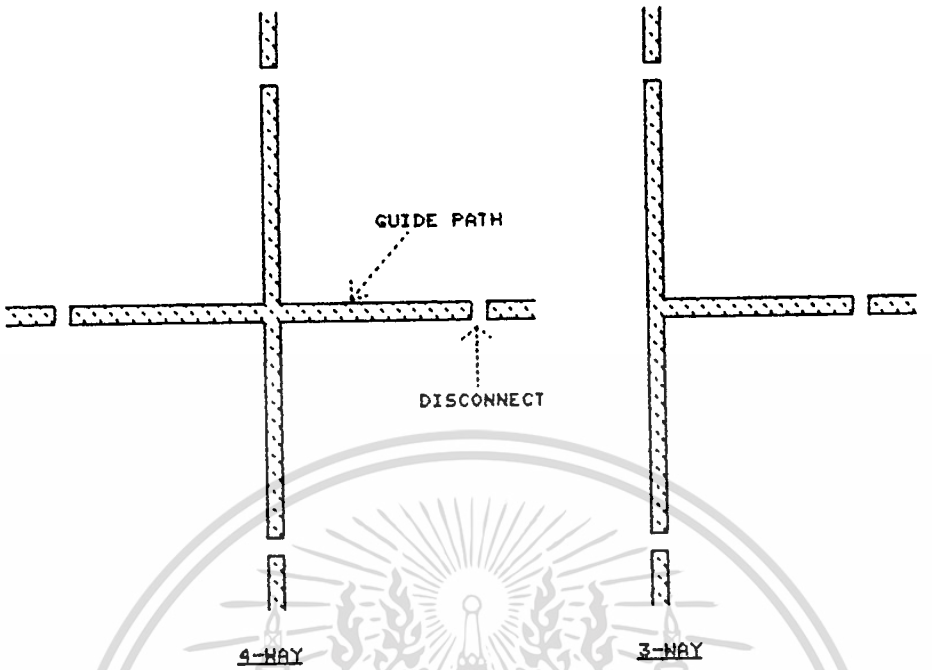


รูปที่ ข.5 แสดงตำแหน่งต่างๆของโฟโต้เซนเซอร์ L,R ในการตรวจหาเทปสี

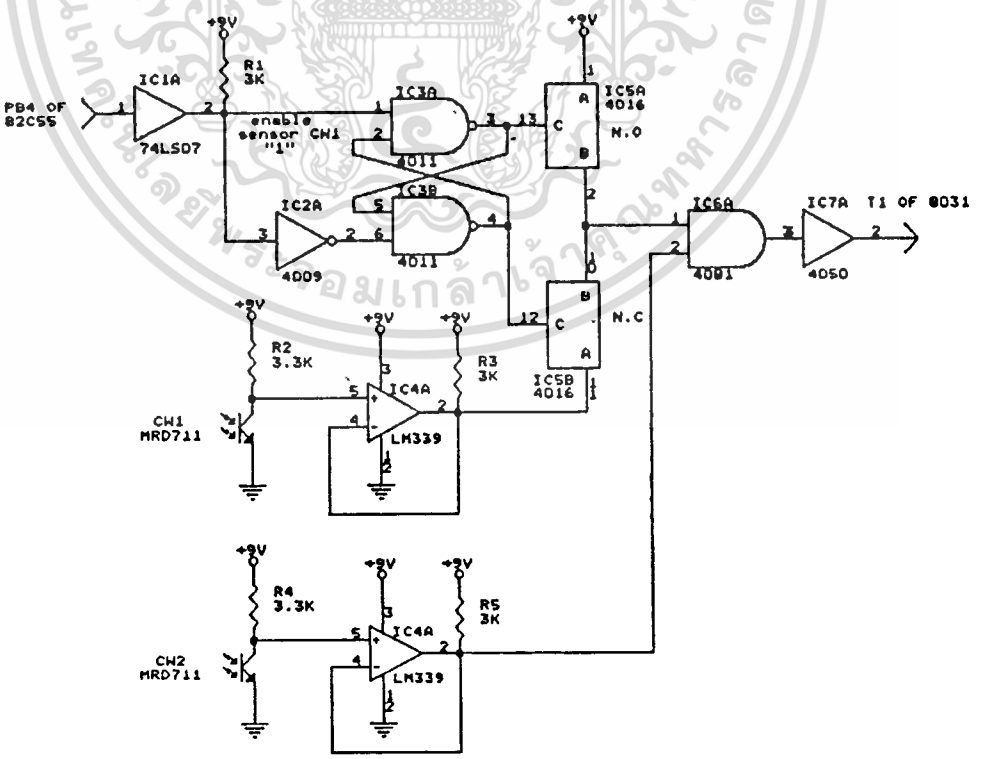
วิธีการเลี้ยว

โครงสร้างล้อของ AGV ในงานวิจัยนี้เป็นระบบ 4 ล้อ เลี้ยวโดยใช้ 2 ล้อหน้ามีเซอร์โวมอเตอร์ควบคุมมุมในการเลี้ยว ส่วน 2 ล้อหลังเป็นล้อขับให้ AGV เคลื่อนที่ไปได้ การเลี้ยวจะเกิดขึ้นเมื่อ AGV เคลื่อนไปถึงทางแยก (cross-way) ลักษณะของทางแยกต่างๆ เป็นดังรูปที่ ข.6

จากรูปจะเห็นว่าที่ตำแหน่งใกล้ทางแยกจะมีการเว้นว่างแนวเส้นเทปสีดำไว้ จุดประสงค์ก็เพื่อให้โฟโต้เซนเซอร์ CW1 , CW2 สามารถตรวจจับได้ ทำให้ AGV รู้ว่าขณะนี้เคลื่อนที่มาถึงทางแยกแล้ว สำหรับวงจรตรวจจับทางแยก แสดงดังรูปที่ ข.7 ส่วนตำแหน่งต่างๆของ AGV ขณะเลี้ยวแสดงในรูปที่ ข.8



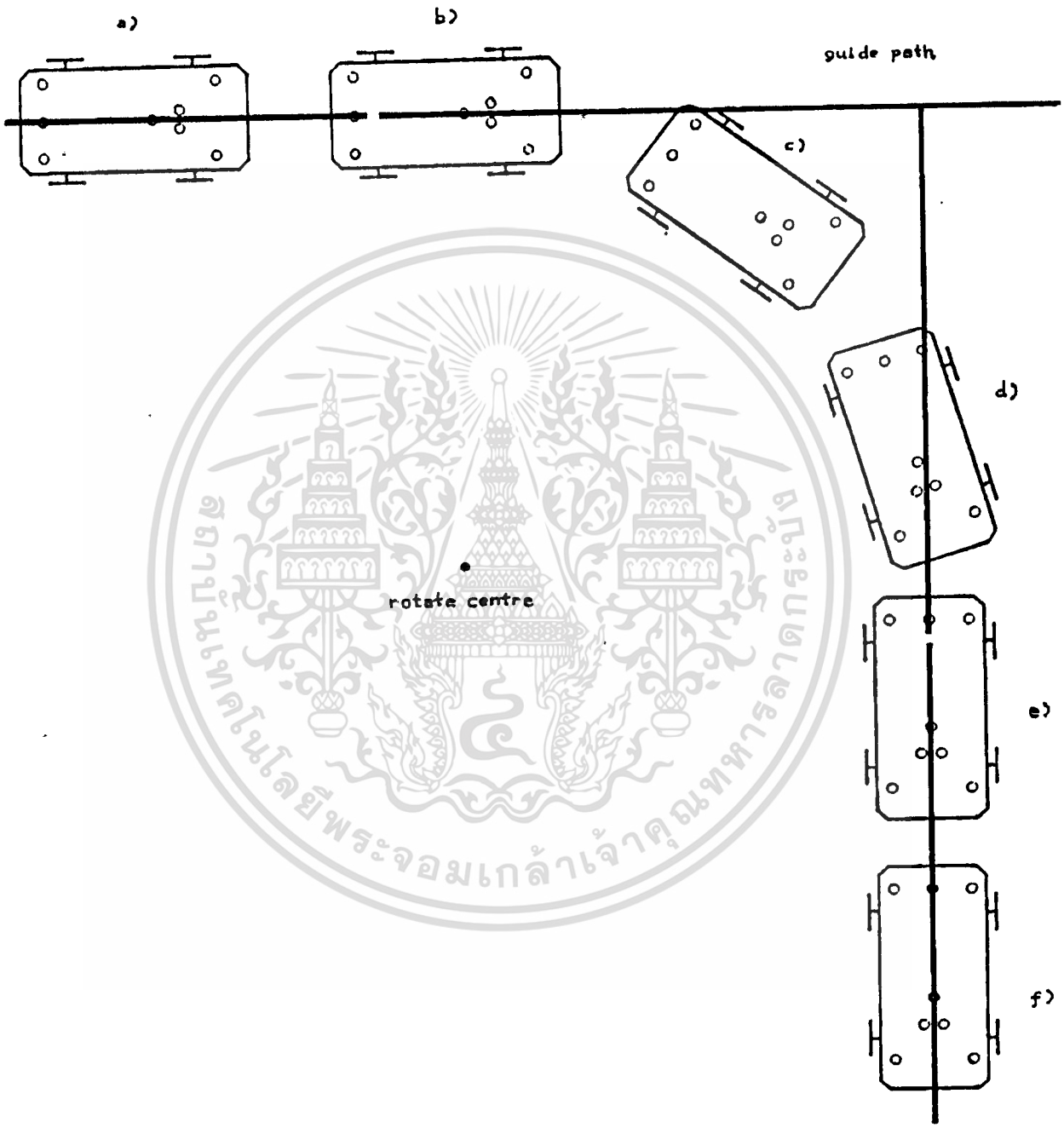
รูปที่ ข.6 ลักษณะของทางแยก



รูปที่ ข.7 วงจรตรวจจับทางแยก

รูปที่ ข.8 เป็นการแสดงตำแหน่งต่างๆของ AGV ขณะเลี้ยว เมื่อ AGV เคลื่อนที่มาใกล้ทางแยก ดังรูปที่ ข.8a) โฟโต้เซนเซอร์ CW1 จะเป็นตัวแรกที่ตรวจจับตำแหน่งช่องว่างได้ โดยแรงดันเอาท์พุทที่ได้จาก CW1 จะเปลี่ยนจาก "1" เป็น "0" เป็นสัญญาณขอบขาลงไปเข้าขาอินเตอรัพท์ T1 ของซีพียู 8031-2

โปรแกรมอินเตอรัพท์ T1 จะลบค่าตั้งทางตำแหน่ง (position setpoint) ปัจจุบัน ที่เหลืออยู่ออกหมดพร้อมกับลดความเร็วในการเคลื่อนที่ของ AGV ลง จากนั้นโปรแกรมจะรอจนกว่าโฟโต้เซนเซอร์ CW2 ส่งสัญญาณขอบขาลงเข้ามายังขา T1 อีกครั้งดังรูปที่ ข.8b) จึงจะกลับไปสู่โปรแกรมหลัก สัญญาณจาก CW2 เป็นการบอกให้ AGV เริ่มต้นเลี้ยวโดย AGV จะเริ่มเลี้ยวไปตามทิศทางที่โปรแกรมไว้โดยจะวิ่งพ้นออกจากแนวเทปสีดำ สำหรับรูปที่ ข.8c) นั้นเป็น AGV ขณะเลี้ยวและอยู่นอกแนวเส้นเทปสีดำซึ่งขณะนี้โฟโต้เซนเซอร์ L,R จะตรวจจับไม่พบเส้นเทป การเลี้ยวจะสิ้นสุดลงทันทีที่โฟโต้เซนเซอร์ L,R ตรวจจับขอบแนวเส้นเทปได้ (รูปที่ ข.8d)) จากนั้น AGV ก็จะถูกควบคุมโดยโฟโต้เซนเซอร์ L,R ให้วิ่งไปตามแนวเส้นเทปเหมือนเดิม การวัดตำแหน่งในระยะทางถัดไปจะเริ่มใหม่ทันทีที่ CW2 ส่งขอบขาลงให้ขา T1 อีกครั้ง (รูปที่ ข.8e)) โดยโปรแกรมจะทำการโหลดค่าตั้งทางตำแหน่ง (position setpoint) ถัดไปจากหน่วยความจำและสั่งให้ AGVเคลื่อนที่ต่อไป (รูปที่ ข.8f))



รูปที่ ข.8 แสดงตำแหน่งต่างๆของ AGV ขณะทำการเลี้ยว

วงจรควบคุมการนำร่องและการเลี้ยว

วงจรควบคุมการนำร่องและการเลี้ยวแสดงได้ดังรูปที่ ข.9 จากรูปแรงดันเอาท์พุทที่ได้จากโฟโต้ดีเทคเตอร์ L,R คือ VR และ VL จะแยกไปใช้ 2 ทาง ทางหนึ่งไปเข้าตัวเปรียบเทียบ (IC2), และเกท (IC4) และ ขา PC6 ของไอซี 8255 ตามลำดับ จุดประสงค์ก็เพื่อบอกให้ AGV รู้ว่าขณะนี้ AGV เคลื่อนที่อยู่ตรงแนวเส้นเทปสีหรือไม่ กรณีที่ AGV อยู่บนเส้นทาง PC6 จะมีลอจิกเป็น "1" ส่วนกรณีที่ AGV อยู่นอกเส้นทาง PC6 จะมีลอจิกเป็น "0"

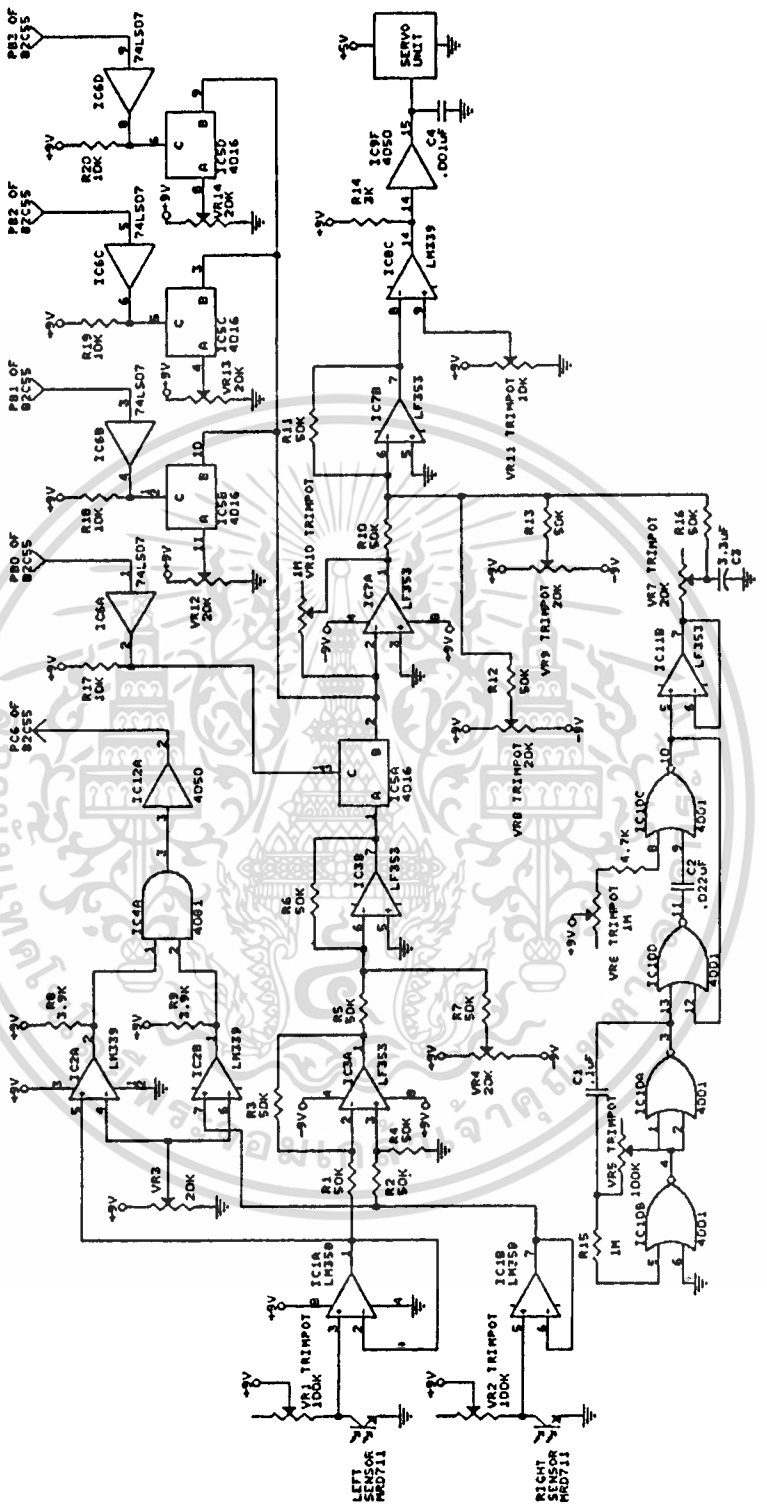
ขณะเดียวกัน VR และ VL จะถูกส่งไปเข้าวงจรขยายสัญญาณแตกต่าง (difference amp) IC3A ได้เอาท์พุทเท่ากับ (VR - VL) และจะถูกส่งต่อไปเข้าวงจรขยายสัญญาณรวม (summing amp) โดยรวมกับระดับแรงดันคงที่ค่าหนึ่ง จุดประสงค์เพื่อยกระดับแรงดันของสัญญาณ (VR - VL)

เอาท์พุทจากวงจรขยายสัญญาณรวมจะถูกส่งต่อไปเข้าวงจรพัลซ์วิทมอดดูเลชั่น เพื่อรวมกับสัญญาณรูปสามเหลี่ยมความถี่ประมาณ 100 Hz ได้เป็นสัญญาณพัลซ์สี่เหลี่ยมความถี่ประมาณ 100 Hz ที่สามารถปรับความกว้างของพัลซ์ (pulse width) ได้จากการเปลี่ยนแปลง VR , VL สัญญาณพัลซ์ที่ได้นี้จะส่งไปเข้าวงจรควบคุมการหมุนของเซอร์โวมอเตอร์ ทำให้ AGV มีการปรับทิศทางของตัวเอง

กรณีที่ AGV อยู่บนแนวเทปสีพอดี VR,VL จะเท่ากันคือ +9 โวลท์ ลอจิกที่ขา PC6 จะเท่ากับ "1" พัลซ์ที่สร้างขึ้นจะไปควบคุมเซอร์โวให้ปรับตัวมาอยู่ในตำแหน่งสมดุลย์ ทำให้ AGV ยังคงเคลื่อนที่ในตรงต่อไป

กรณีที่ AGV เบนไปทางขวามากเกินไป VR จะเท่ากับ 0 โวลท์ ส่วน VL จะเท่ากับ +9 โวลท์ ลอจิกที่ขา PC6 จะเท่ากับ "0" เป็นการบอก AGV ว่าขณะนี้ออกนอกเส้นทางแล้ว พัลซ์ที่สร้างขึ้นจะไปควบคุมเซอร์โวให้หมุนด้วยมุมที่ทำให้ AGV ปรับตัวกลับมาอยู่ในแนวตรงตามเดิม

กรณีที่ AGV เบนไปทางซ้ายมากเกินไป VR จะเท่ากับ +9 โวลท์ ส่วน VL จะเท่ากับ 0 โวลท์ ลอจิกที่ขา PC6 จะเท่ากับ "0" เป็นการบอก AGV ว่าขณะนี้ออกนอกเส้นทางแล้ว พัลซ์ที่สร้างขึ้นจะไปควบคุมเซอร์โวให้หมุนด้วยมุมที่ทำให้ AGV ปรับตัวกลับมาอยู่ในแนวตรงตามเดิม

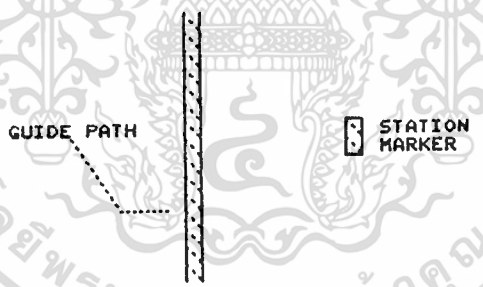


รูปที่ ข.9 วงจรควบคุมการนำร่องและการเลี้ยว

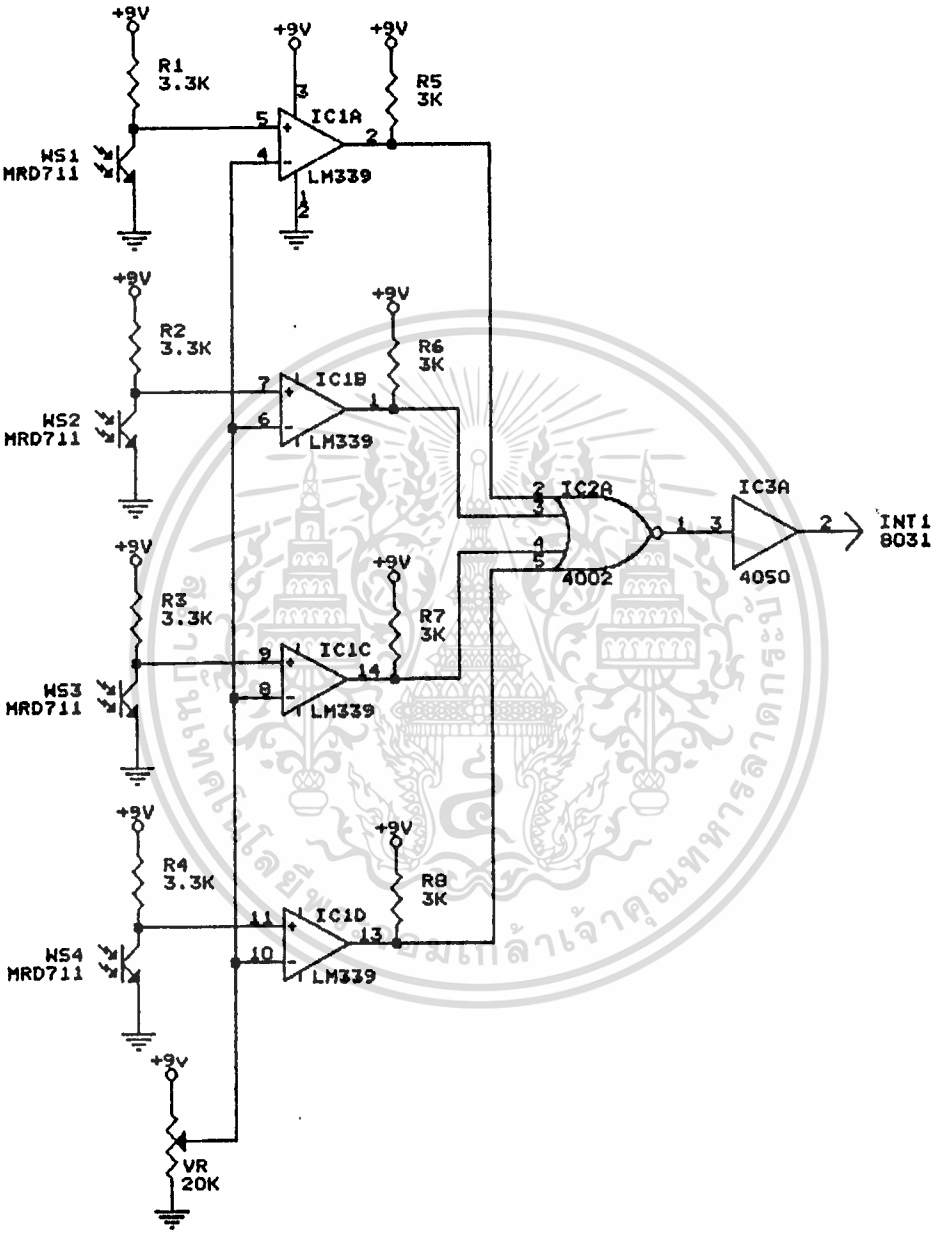
วิธีการหาตำแหน่งสถานีปฏิบัติงาน

AGV สามารถตรวจหาตำแหน่งสถานีได้โดยใช้โฟโต้เซนเซอร์เช่นเดียวกับการนำร่อง โดยจะทำการตรวจหาจุดแสดงตำแหน่ง (marker) ซึ่งในที่นี้ใช้เป็นเทปสีดำเช่นเดียวกับการนำร่อง ลักษณะการติดเทปสีดำเพื่อแสดงตำแหน่งสถานีเป็นดังรูปที่ ข.10 ตำแหน่งการติดเทปสีนั้นจะต้องอยู่ในแนวตรงกับที่โฟโต้เซนเซอร์ WS1,WS3, และ WS2,WS4 ตามรูปที่ ข.4 เคลื่อนที่ผ่านพอดี เมื่อ AGV เคลื่อนที่เข้ามาใกล้ตำแหน่งสถานี ความเร็วของ AGV ข้าลงตามค่าตารางความเร็วที่เก็บไว้ในหน่วยความจำ และเมื่อเคลื่อนมาถึงจุดแสดงตำแหน่งสถานี โฟโต้เซนเซอร์ WS1 หรือ WS2 จะตรวจจับได้และส่งสัญญาณไปเข้าขาอินเตอรัพท์ของ 8031-2 โปรแกรมอินเตอรัพท์จะเคลียร์ค่าเออเรอร์สะสมที่เกิดจากการเคลื่อนที่ทิ้งไปพร้อมกับตั้งค่าระยะที่ถูกต้องเสียใหม่ พร้อมกันนั้น AGV จะลดความเร็วลงมาอยู่ในระดับเหมาะสม ทำให้ AGV สามารถหยุดได้ในทันทีที่ WS3 หรือ WS4 ตรวจจับตำแหน่งสถานีได้

ด้วยวิธีนี้เราสามารถจะหยุด AGV ตรงตำแหน่งสถานีที่ต้องการพอดี สำหรับวงจรตรวจจับตำแหน่งสถานีที่ใช้แสดงดังรูปที่ ข.11



รูปที่ ข.10 เครื่องหมายแสดงตำแหน่งสถานีปฏิบัติ



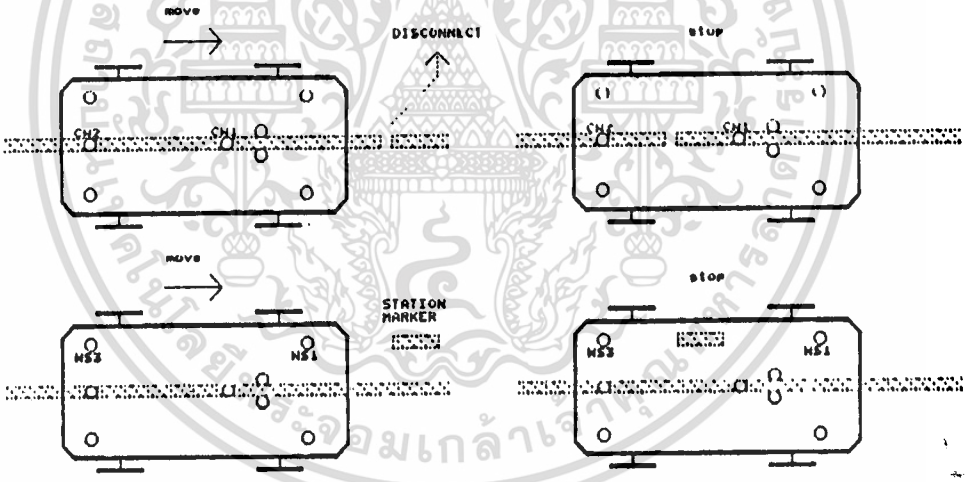
รูปที่ ข.11 วงจรตรวจจับตำแหน่งสถานีปฏิบัติงาน

รายละเอียดเพิ่มเติมในโปรแกรมควบคุมการเคลื่อนที่

1). โปรแกรมหลัก (main program)

กำหนดขั้นตอนการทำงานคือ

ก) หลังทำการรีเซท 8031-2 โปรแกรมจะสั่งให้ AGV เคลื่อนที่ไปหยุดยังทางแยกหรือสถานีใดๆ เองโดยอัตโนมัติ โดยในตอนเริ่มต้น AGV จะปรับแนวการเคลื่อนที่ให้อยู่บนแนวเส้นเทปด้วยโฟโต้เซนเซอร์ L,R จากนั้น AGV จะเคลื่อนที่ไปตามแนวเส้นเทปอย่างช้าๆ โฟโต้เซนเซอร์ CW1 จะคอยตรวจหาตำแหน่งทางแยก ขณะที่โฟโต้เซนเซอร์ WS1,WS2 จะคอยตรวจหาตำแหน่งสถานี หาก AGV พบสิ่งใดก่อนก็ตาม AGV จะหยุดทันที ลักษณะการหยุดเป็นดังรูปที่ ข.12



รูปที่ ข.12 แสดงลักษณะการหยุดของ AGV ที่ทางแยกหรือสถานี หลังจากทำการรีเซท

ข) 8031-2 ส่งสัญญาณไปยังขา P3.3 ของ 8031-1 เพื่อบอกให้ 8031-1 ทราบว่าขณะนี้พร้อมแล้วที่จะรับข้อมูลเกี่ยวกับการเคลื่อนที่จาก 8031-2 ข้อมูลนี้เป็นข้อมูลที่รับมาจากไมโครคอมพิวเตอร์เมื่อใช้คำสั่ง get\ การส่งข้อมูลจาก 8031-1 ไปยัง 8031-2 นั้นจะใช้วิธี handshaking โดย 8031-2 จะรอรับสัญญาณอินเตอรัพท์ INTRA ที่ส่งมายังขา INTO โปรแกรมย่อยอินเตอรัพท์ INTO จะทำการอ่านข้อมูลเก็บไว้ในหน่วยความจำ ดังจะได้กล่าวต่อไป

ค) 8031-2 รอรับสัญญาณ 'move' จาก 8031-1 สัญญาณ 'move' เป็นสัญญาณลอจิกศูนย์ที่เกิดขึ้นหลังจากป้อนคำสั่ง move ที่ไมโครคอมพิวเตอร์ เป็นการสั่งให้ AGV เริ่มเคลื่อนที่ตามข้อมูลได้รับมา แต่ก่อนที่ AGV จะเคลื่อนที่จะต้องมีการคำนวณและตรวจสอบเงื่อนไขบางอย่างก่อน

ง) เมื่อได้รับสัญญาณ 'move' แล้ว โปรแกรมจะตรวจสอบรหัสคำสั่งการเคลื่อนที่ , ทิศทางการหมุน จากนั้นจะโหลดค่าระยะทางซึ่งมีขนาด 16 บิท ไปแยกเก็บในตัวแปร HIBYTE และ LOBYTE โดย HIBYTE นั้นเก็บค่าไบท์บน(upper byte) ส่วน LOBYTE เก็บค่าไบท์ล่าง(lower byte) ระยะทางดังกล่าวนี้อยู่ในรูปของจำนวนพัลซ์ จากนั้นโปรแกรมจะตรวจสอบเงื่อนไขของจำนวนพัลซ์ใน HIBYTE และ LOBYTE เพื่อนำไปเทียบหาความเร็วที่เหมาะสมในตารางเก็บค่าความเร็ว (ตารางที่ 4.1 ในหัวข้อ 4.4.4) ขั้นต่อไปโปรแกรมจะสั่งให้ AGV เริ่มเคลื่อนตัวอย่างช้าๆออกจากทางแยกหรือสถานีที่หยุดรอตอนแรกพร้อมทั้งรอรับสัญญาณลอจิกศูนย์จากโฟโต้เซนเซอร์ตัวใดตัวหนึ่งคือ CW2 หรือ WS3 หรือ WS4

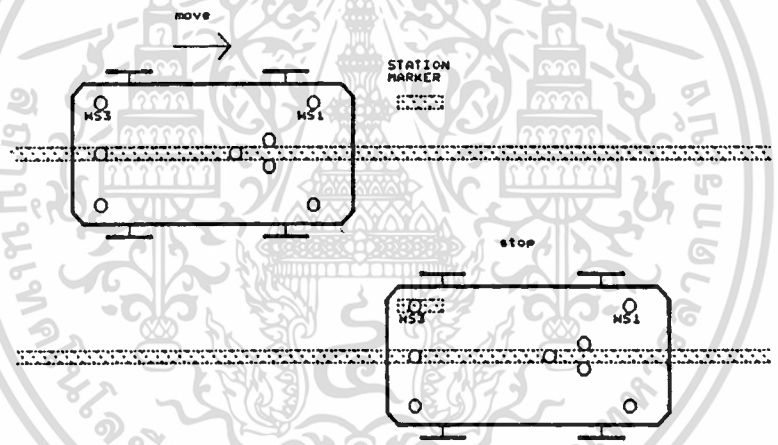
จ) เมื่อได้รับสัญญาณจากโฟโต้เซนเซอร์ดังกล่าวแล้ว โปรแกรมจะอีนาเบิ้ลอินเตอรัพท์ T0 เพื่อเริ่มต้นนับจำนวนพัลซ์ซึ่งจะสัมพันธ์กับระยะทางที่ AGV เคลื่อนที่ไปได้ โปรแกรมจะนับค่าระยะทางไปพร้อมกับลดค่าใน LOBYTE ลงไปเรื่อยๆ เมื่อ LOBYTE มีค่าเป็นศูนย์จะมีการขอยืม (borrow) จาก HIBYTE การนับระยะทางจะทำไปเรื่อยๆ จนกว่าค่าใน HIBYTE และ LOBYTE เหลือศูนย์ทั้งคู่ AGV จึงจะหยุดเคลื่อนที่ จากนั้นโปรแกรมจะกลับไปตรวจสอบรหัสคำสั่งชุดต่อไป

ฉ) กรณีที่ AGV เคลื่อนที่ผ่านทางแยกต่างๆ หรือผ่านสถานีอื่นที่ไม่ใช่สถานีเป้าหมายและยังอยู่ห่างไกลจากสถานีเป้าหมาย สัญญาณจากโฟโต้เซนเซอร์ CW1,CW2,WS1,WS2,WS3 และ WS4 จะไม่มีผลต่อการเคลื่อนที่แต่อย่างใด เนื่องจากโปรแกรมจะทำการดิสเอเบิ้ลอินเตอรัพท์ INT1 และ T1 เอาไว้

ช) กรณีควบคุมให้ AGV ทำการเลี้ยว เมื่อ AGV เคลื่อนที่มาใกล้ทางแยก โปรแกรมจะทำการอีนาเบิ้ลอินเตอรัพท์ T1 เพื่อคอยสัญญาณอินเตอรัพท์จากโฟโต้เซนเซอร์ CW1 ซึ่งตรวจพบตำแหน่งทางแยก ทันทีที่ CW1 พบทางแยก โปรแกรมอินเตอรัพท์ T1 จะเคลียร์พัลซ์ที่เหลือใน HIBYTE และ LOBYTE ให้เป็นศูนย์ พร้อมกับลดความเร็วในการเคลื่อนที่ของ AGV ลง จากนั้นจะรอจนกว่าโฟโต้เซนเซอร์ CW2 ส่งสัญญาณขอบขาลงเข้าขา T1 อีกครั้งจึงจะกลับสู่โปรแกรมหลัก จากนั้นโปรแกรมจะกลับไปตรวจสอบรหัสคำสั่งและทิศทางการเลี้ยวแล้วเริ่มทำการเลี้ยว เมื่อ AGV เคลื่อนพ้นออกจากแนวเส้นเทปสีดำแล้วโฟโต้เซนเซอร์ L,R จะตรวจจับไม่พบเส้นเทป การเลี้ยวจะสิ้นสุดลงทันทีที่โฟโต้เซนเซอร์ L,R ตรวจจับขอบแนวเส้นเทปได้อีกครั้ง จากนั้นโปรแกรมจะกำหนดให้โฟโต้เซนเซอร์ L,R ทำหน้าที่ควบคุม AGV ให้วิ่งไปตามแนวเส้นเทปเหมือนเดิม สำหรับการนับพัลซ์ทางตำแหน่งของระยะทางถัดไปนั้นจะเริ่มอีกครั้งหลังจากที่ CW2 ส่งสัญญาณขอบขาลงให้ขา T1 แล้ว โดยโปรแกรมจะกลับไปตรวจสอบรหัสคำสั่งชุดใหม่ยังหน่วยความจำ สำหรับตำแหน่งต่างๆของ AGV ระหว่างเลี้ยวได้

แสดงไว้แล้วตามรูปที่ ข.8

ช) การควบคุมให้ AGV สามารถหยุดตรงตำแหน่งสถานีที่ต้องการได้อย่างแม่นยำ เมื่อ AGV เคลื่อนที่มาใกล้ถึงสถานีเป้าหมาย โปรแกรมหลักจะทำการอินาเบิ้ลอินเตอรัพท์ INT1 เองโดยอัตโนมัติ สัญญาณอินเตอรัพท์ INT1 มาจากการที่เซนเซอร์ WS1 หรือ WS2 ตรวจจับตำแหน่งสถานีเป้าหมายได้ โปรแกรมอินเตอรัพท์ INT1 นี้จะทำให้ AGV สามารถหยุดที่ตำแหน่งสถานีเป้าหมายได้อย่างแม่นยำโดย โปรแกรมนี้จะกำจัดเออเรอร์สะสมของตำแหน่งที่เกิดขึ้นจากการลื่นไถล (slip) ของล้อจากการเคลื่อนที่เป็นระยะไกลๆ ด้วยการตั้งค่าระยะทางที่เหลือเสียใหม่ สำหรับระยะทางใหม่นั้นก็เป็นระยะห่างระหว่างเซนเซอร์คู่หน้า (WS1,WS2) กับเซนเซอร์คู่หลังนั่นเอง (WS3,WS4) ในรูปที่ ข.13 แสดงการหยุดของ AGV ที่ตำแหน่งสถานีเป้าหมาย



รูปที่ ข.13 แสดงการหยุดของ AGV ที่ตำแหน่งสถานีเป้าหมาย

ฌ) เมื่อ AGV หยุดตรงตำแหน่งที่ต้องการแล้ว โปรแกรมจะส่งสัญญาณ 'moving complete'ให้ 8031-1 ทราบ เพื่อให้ 8031-1 ส่งข้อความแสดงการทำงานสิ้นสุดไปยังไมโครคอมพิวเตอร์ต่อไป

2) <u>โปรแกรมอินเตอรัพท์เพิ่มเติม</u>

ก) โปรแกรมย่อยอินเตอรัพท์ INT1 จะทำงานเมื่อมีสัญญาณขอบขาลงจากโฟโต้เซนเซอร์ WS1 หรือ WS2 เข้ามาที่ขาอินเตอรัพท์ INT1 อินเตอรัพท์ INT1 นี้จะอินาเบิ้ลเฉพาะเมื่อ AGV เคลื่อนที่ใกล้ถึงเป้าหมายเท่านั้น โปรแกรมนี้จะตั้งค่าระยะทางในตัวแปร HIBYTE และ LOBYTE เสียใหม่

โดยระยะทางใหม่นี้ จะเท่ากับความยาวระหว่างตำแหน่งเซนเซอร์คู่หน้า (WS1,WS2) กับตำแหน่งเซนเซอร์คู่หลัง (WS3,WS4) ผลจากการตั้งค่าระยะทางใหม่นี้จะทำให้ AGV ปรับความเร็วและหยุดได้อย่างสอดคล้องกับระยะทางจริง ทำให้การหยุดมีความแม่นยำมากขึ้น

ข) โปรแกรมย่อยอินเตอรัพท์ T1 จะทำงานเมื่อมีสัญญาณขอบขาลงจากโฟโต้เซนเซอร์ CW1 เข้ามาที่ขาอินเตอรัพท์ T1 อินเตอรัพท์นี้จะอินาเบิ้ลทุกครั้งที่ค่าพัลซ์ทางตำแหน่งในตัวแปร HIBYTE และ LOBYTE ลดลงต่ำกว่าค่า 3fffh ซึ่งเป็นค่าระยะทางที่ตั้งไว้ ทันทีที่เกิดการอินเตอรัพท์โปรแกรมจะทำการเคลียร์ค่าพัลซ์ทางตำแหน่งที่เหลืออยู่ในตัวแปร HIBYTE และ LOBYTE ให้เป็นศูนย์ จากนั้นจะรอสัญญาณขอบขาลงจากเซนเซอร์ CW2 ก่อนจึงจะกลับไปสู่โปรแกรมหลัก

ภาคผนวก ค

บทความที่ได้รับการตีพิมพ์ในระหว่างศึกษาปริญญาโท

มหาวิทยาลัยเชียงใหม่

EECON XIII

คณะวิศวกรรมศาส

การประชุมทางวิชาการวิศวกรรมไฟฟ้า ครั้งที่ 13

The 13th Conference of Electrical Engineering

8-9 พฤศจิกายน 2533

ณ ภาควิชาวิศวกรรมไฟฟ้า

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่

ระบบควบคุมการเคลื่อนที่ในช่วงออกตัวและหยุดภายในเวลาสั้นที่สุดด้วยสัญญาณอินพุทแบบขั้นบันไดต่อเนื่อง

(Start-Stop of Motion Control System With Minimum Time By Using Piecewise Continuous Input)

อนุตรชัย ณถลาง*
โยธิน เปรมปราณีรัชต์**
จงกล งามวิวิทย์***

บทคัดย่อ

ผลงานวิจัยนี้เสนอวิธีควบคุมการเคลื่อนที่ในช่วงออกตัวและหยุดของดีซีมอเตอร์ภายในเวลาสั้นที่สุดด้วยสัญญาณอินพุทแบบขั้นบันไดต่อเนื่องที่มีลักษณะเป็นฟังก์ชั่น 2 สเตป โดยอธิบายถึงการออกแบบขนาดของสเตปแรกและช่วงการหน่วงเวลาของสเตปที่สองอย่างเหมาะสมกับระบบเพื่อให้ได้ผลตอบสนองของการเคลื่อนที่ในช่วงออกตัวและหยุดภายในเวลาสั้นที่สุด โดยปราศจากโอเวอร์ชูท และเข้าสู่ตำแหน่งเป้าหมายได้อย่างถูกต้องเที่ยงตรง นอกจากนี้ยังได้เสนอผลการจำลองระบบด้วยไมโครคอมพิวเตอร์และผลการทดลองผลตอบสนองการเคลื่อนที่ในช่วงออกตัวและหยุดของระบบต่อสัญญาณอินพุทแบบขั้นบันไดต่อเนื่อง เปรียบเทียบต่อสัญญาณสเตปอินพุท

Abstract

This paper presents a start-stop control method of the DC Motor motion control system with minimum time by using a piecewise continuous input , a combination of two steps function. The paper described is how to design the magnitude of the first step and the delay time in the second step of the piecewise continuous input for satisfaction with the characteristic of the motion control system in order to get a minimum time of motion response which has no overshoot in the start-stop period. Then the system can be obtained with the high accuracy of the desired position. Additionally , the microcomputer simulation and experimental results of the motion response of the start-stop period to the piecewise continuous input in comparison to the step input are also presented.

---

* นักศึกษาปริญญาโท ภาควิชาวิศวกรรมระบบควบคุม คณะวิศวกรรมศาสตร์
** รองศาสตราจารย์ ภาควิชาวิศวกรรมระบบควบคุม คณะวิศวกรรมศาสตร์
*** ผู้ช่วยศาสตราจารย์ ภาควิชาวิศวกรรมระบบควบคุม คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าฯลาดกระบัง

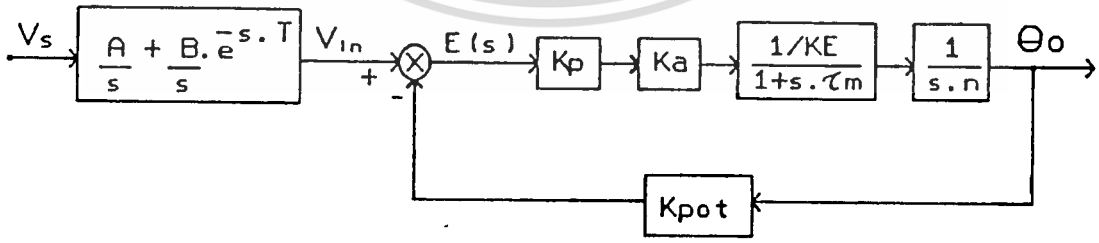
1. บทนำ

การควบคุมการเคลื่อนที่ของโหลดด้วยดีซีมอเตอร์กระทำได้ง่ายและเป็นที่นิยมใช้อย่างกว้างขวางในระบบขับเคลื่อนต่างๆ เช่นระบบควบคุมการเคลื่อนที่ของแขนหุ่นยนต์[1][2], ระบบ CNC และระบบขับเคลื่อนแผ่นจานเก็บข้อมูลของเครื่องคอมพิวเตอร์เป็นต้น ปัญหาหลักของระบบควบคุมการเคลื่อนที่โดยทั่วๆไปได้แก่ ปัญหาการออกตัว(start)และการหยุด(stop)ให้ได้รวดเร็วหรือภายในเวลาน้อยที่สุด[3][4] โดยปราศจากโอเวอร์ชูท เพื่อให้สามารถกำหนดตำแหน่งการหยุดได้ตามค่าเป้าหมายอย่างถูกต้องเที่ยงตรงภายในเวลาสั้นที่สุด ผลงานการวิจัยนี้ได้เสนอวิธีการควบคุมการเคลื่อนที่ในช่วงการออกตัวและหยุดภายในเวลาสั้นที่สุดด้วยการป้อนสัญญาณอินพุทแบบขั้นบันไดต่อเนื่องที่มีลักษณะเป็นฟังก์ชั่น 2 สเตป โดยอธิบายถึงการออกแบบขนาดของสเต็ปแรกและช่วงหน่วงเวลา(delay time)ของสเตปที่สองอย่างเหมาะสมกับระบบ จากนั้นจะใช้สัญญาณขั้นบันไดแบบต่อเนื่องนี้ควบคุมระบบเฉพาะในช่วงออกตัวและหยุดเพื่อให้ดีซีมอเตอร์ออกตัวและหยุดได้อย่างรวดเร็วโดยปราศจากโอเวอร์ชูท และเข้าสู่ตำแหน่งเป้าหมายได้อย่างถูกต้องเที่ยงตรง

จากผลการทดสอบระบบด้วยการจำลองการทำงานของระบบด้วยไมโครคอมพิวเตอร์และจากผลการทดลองแสดงให้เห็นว่า ผลตอบสนองของระบบในช่วงออกตัวและช่วงหยุดที่มีต่อสัญญาณอินพุทแบบขั้นบันไดต่อเนื่องเปรียบเทียบต่อสัญญาณสเตปอินพุทนั้น โอเวอร์ชูทของตำแหน่งลดลงได้ 100 % ซึ่งทำให้เสถียรภาพของระบบดีขึ้นและที่สำคัญก็คือ การเข้าสู่ตำแหน่งในช่วงออกตัวและช่วงหยุดจะใช้เวลาน้อยกว่าเดิมมาก

2. การออกแบบสัญญาณขั้นบันไดต่อเนื่อง

ลักษณะของระบบควบคุมการเคลื่อนที่ดีซีมอเตอร์สำหรับงานวิจัยนี้แสดงไว้ในรูปที่ 1. สัญญาณสเตปอินพุท (Vs) จะถูกป้อนผ่านวงจรสร้างสัญญาณอินพุทแบบขั้นบันไดต่อเนื่อง(Vin) ซึ่งได้ทำการคำนวณขนาดต่างๆของสัญญาณไว้แล้วจากผลการซิมูเลชั่นระบบด้วยไมโครคอมพิวเตอร์



รูปที่ 1.ระบบควบคุมการเคลื่อนที่ดีซีมอเตอร์ที่มีการป้อนสัญญาณอินพุทแบบขั้นบันไดต่อเนื่อง

ระบบมีพารามิเตอร์ดังนี้ Kp : proportional gain

Ka : amplifier gain

KE : voltage constant

τ_m : mechanical time constant

Kpot : potentiometer gain

n : gear ratio

จากรูปที่ 1. สามารถเขียน transfer function ของระบบได้เป็น

$$\frac{\Theta_{o(s)}}{V_{in(s)}} = \frac{\frac{Kp.Ka}{KE.n.\tau_m}}{s^2 + \frac{1}{\tau_m}\cdot s + \frac{Kp.Ka.Kpot}{KE.n.\tau_m}} \quad \ldots\ldots(1)$$

หรือ

$$\frac{\Theta_{o(s)}}{V_{in(s)}} = \frac{\frac{1}{Kpot}\cdot\omega_n^2}{s^2 + 2\cdot\zeta\cdot\omega_n\cdot s + \omega_n^2} \quad \ldots\ldots(2)$$

โดยที่ $\omega_n = \sqrt{\frac{Kp.Ka.Kpot}{KE.n.\tau_m}}$; undamped natural frequency

$\zeta = \frac{1}{2\cdot\omega_n\cdot\tau_m}$; damping ratio

2.1 ผลตอบสนองของตำแหน่งในช่วงออกตัวและช่วงหยุดที่มีต่อสัญญาณสเตปอินพุท

กำหนดให้ระบบมีค่า damping factor (ζ) = 0.123 ส่วนค่าพารามิเตอร์ต่างๆของระบบมีดังนี้

KE = 0.019 โวลท์/เรเดียน/วินาที

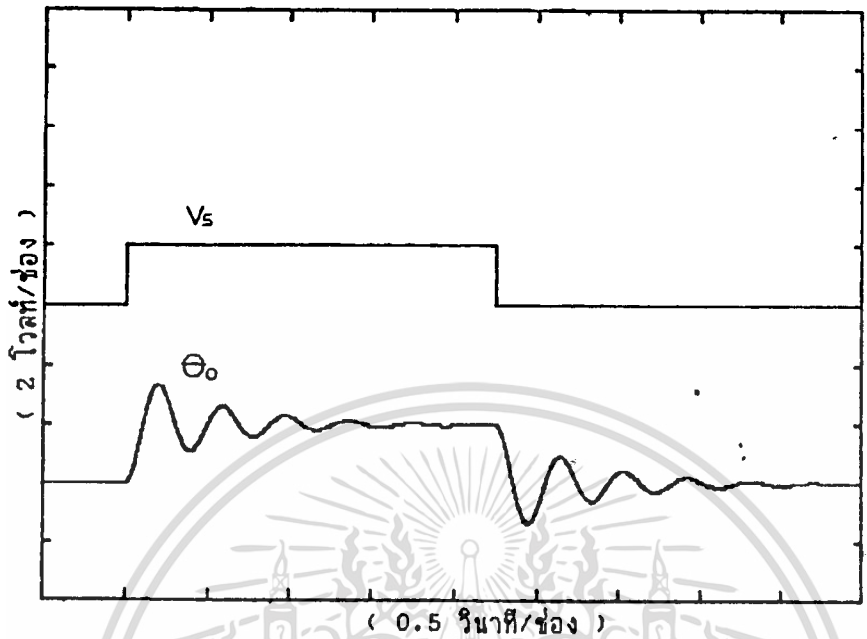
τ_m = 0.25 วินาที

Kpot = 3 โวลท์/เรเดียน

Ka = 3.8

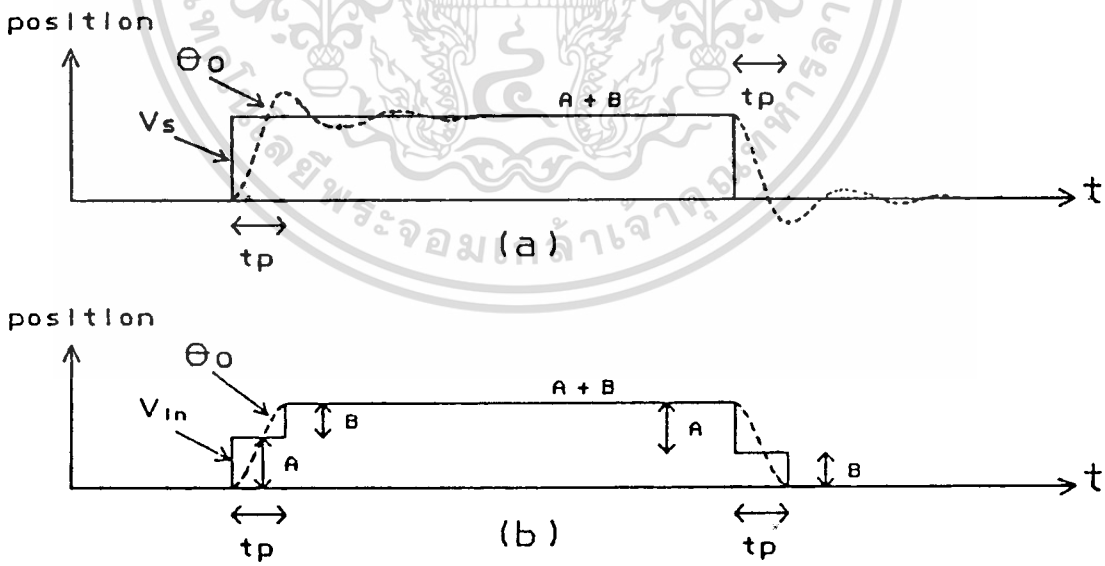
n = 14

จากสมการ(1)และ(2)คำนวณหาค่า proportional gain(Kp)ได้เท่ากับ 1.544 จากนั้นเริ่มทำการชิมูเลชั่นด้วยไมโครคอมพิวเตอร์ กำหนดค่าสัญญาณเซ็ทพอยท์ของตำแหน่งเท่ากับ 2 โวลท์ (หรือ Θ_s มีค่าเท่ากับ 0.666 เรเดียน) ผลของการชิมูเลชั่นแสดงตามรูปที่ 2.



รูปที่ 2. ผลตอบสนองทางตำแหน่งในช่วงออกตัวและหยุดของดิซิมอเตอร์ต่อสัญญาณสเตปอินพุท
โดยที่ t_p = 0.195 วินาที , t_s = 2.04 วินาที และ M_p = 67.68 %

2.2 ผลการตอบสนองของตำแหน่งในช่วงออกตัวและหยุดที่มีต่อสัญญาณอินพุทแบบขั้นบันไดต่อเนื่อง
ลักษณะสัญญาณอินพุทแบบขั้นบันไดต่อเนื่องมีลักษณะเป็นฟังก์ชั่น 2 สเตปดังรูปที่ 3.



รูปที่ 3. ลักษณะของสัญญาณแบบขั้นบันไดต่อเนื่อง

ถ้ากำหนดให้สัญญาณเซ็ทพอยท์ของตำแหน่งที่ต้องการ (Θ_R) มีขนาดเท่ากับ A + B โวลท์ การออกแบบคือการคำนวณหาค่าขนาดของสเตปแรก (A) , ช่วงหน่วงเวลาระหว่างสเตปแรกและสเตปที่สอง(T) และค่าขนาดของสเตปที่สอง (B)

นิจารณารูปที่ 3(a),3(b) และจากสมการ (2) จะได้ผลตอบสนองทางตำแหน่งในช่วงออกตัวที่มีต่อสัญญาณสเตปอินพุทที่เวลาขณะตำแหน่งเกิดโอเวอร์ชูทสูงสุด (peak time) เป็น

$$\Theta_o(tp) = \frac{(A+B)}{Kpot} \cdot \left(1 + e^{-\zeta \cdot \pi / \sqrt{1-\zeta^2}}\right) \qquad \ldots.(3)$$

และผลตอบสนองของตำแหน่งที่ steady state คือ

$$\Theta_o(\infty) = \frac{(A+B)}{Kpot} \qquad \ldots.(4)$$

โดยที่

$$tp = \frac{\pi}{\omega_n \cdot \sqrt{1-\zeta^2}} \quad ; \text{peak time} \qquad \ldots.(5)$$

ในการออกแบบสัญญาณอินพุทแบบขั้นบันไดต่อเนื่อง เราจะกำหนดให้ช่วงหน่วงเวลา (T) เท่ากับ tp และสัดส่วนของสเตป A ต่อ (A + B) เป็น

$$\frac{A}{(A+B)} = \frac{\Theta_o(\infty)}{\Theta_o(tp)}$$

ทำให้ได้ว่า

$$A = \frac{\Theta_o(\infty)}{\Theta_o(tp)} \cdot Kpot \cdot \Theta_o(\infty) \qquad \ldots.(6)$$

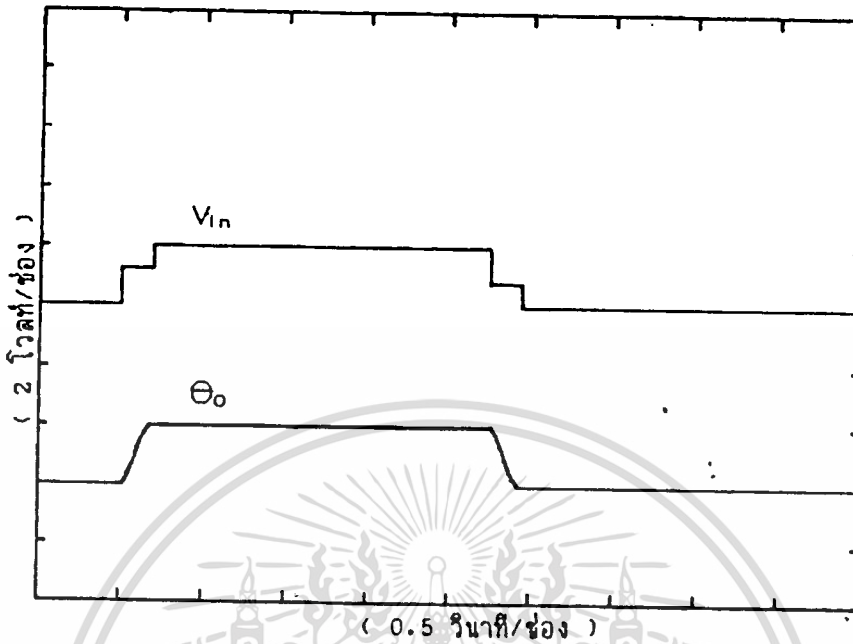
$$B = (A+B) - A \qquad \ldots.(7)$$

ฉะนั้นจากข้อมูลของระบบในหัวข้อ 2.1

ζ = 0.123 , ω_n = 16.27 , Kpot = 3 โวลท์/เรเดียน ,tp = 0.195 วินาที และขนาดสัญญาณอินพุทซึ่งเป็นเซ็ทพอยท์ของตำแหน่ง(A + B) = 2 โวลท์ (Θ_s = 0.666 เรเดียน) ทำการคำนวณหาลักษณะของสัญญาณขั้นบันไดต่อเนื่องจากสมการ (3),(4),(6) และ(7) ได้ดังนี้

1) ขนาดของสเตปแรก (A) = 1.193 โวลท์
2) ช่วงหน่วงเวลา (T) = 0.195 วินาที
3) ขนาดของสเตปที่สอง (B) = 0.807 โวลท์

สำหรับผลการซิมูเลชั่นระบบที่มีการป้อนสัญญาณอินพุทแบบขั้นบันไดต่อเนื่องที่คำนวณได้นี้แสดงไว้ในรูปที่ 4. ซึ่งจากการซิมูเลชั่นจะเห็นได้ว่าเมื่อทำการป้อนสัญญาณอินพุทแบบขั้นบันไดต่อเนื่องอย่างเหมาะสมจะทำให้ได้ผลคือ โอเวอร์ชูทของตำแหน่งเป็นศูนย์และระบบเข้าสู่ค่าคงที่ของตำแหน่งในช่วงออกตัวภายในเวลา 0.195 วินาที ลดลงจากเดิมคือ 2.04 วินาที คิดเป็นเปอร์เซนต์ได้ 90.44 เปอร์เซนต์



รูปที่ 4. ผลตอบสนองของตำแหน่งในช่วงออกตัวและหยุดของดีซีมอเตอร์ที่มีต่อสัญญาณขั้นบันไดต่อเนื่อง โดยที่ tp = 0.195 วินาที, ts = 0.195 วินาที และ Mp = 0 %

3. ความเที่ยงตรงของระบบ

จากรูปที่ 1. และสมการที่ (2) จะหาความสัมพันธ์ระหว่าง E(s) และ Vin(s) ได้ดังนี้

$$\frac{E(s)}{V_{in}(s)} = \frac{s^2 + 2\cdot\zeta\cdot\omega_n\cdot s}{s^2 + 2\cdot\zeta\cdot\omega_n\cdot s + \omega_n^2}$$

และเมื่อป้อนสัญญาณอินพุทแบบขั้นบันไดต่อเนื่อง

$$V_{in}(s) = \frac{A}{s} + \frac{B}{s}\cdot e^{-s\cdot tp}$$

จากทฤษฎี final value, steady state error(e_{ss}) = $\lim_{s\to 0} s.E(s)$

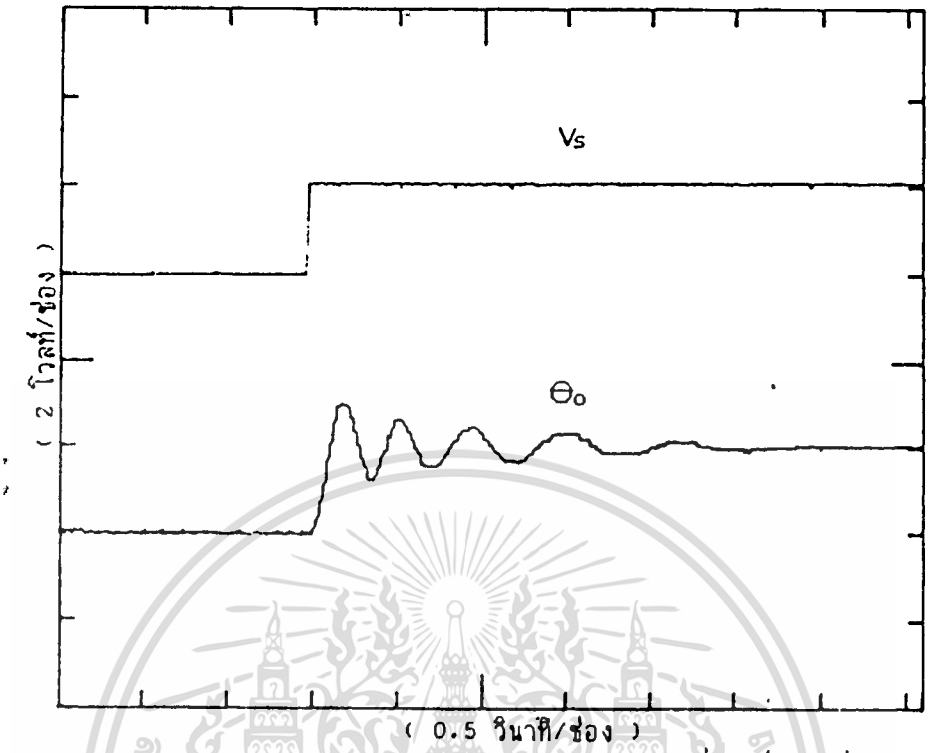
ดังนั้นจะได้ว่า

$$e_{ss} = \lim_{s\to 0} \frac{s\cdot(s^2 + 2\cdot\zeta\cdot\omega_n\cdot s)\cdot\left(\frac{A}{s} + \frac{B}{s}\cdot e^{-s\cdot tp}\right)}{s^2 + 2\cdot\zeta\cdot\omega_n\cdot s + \omega_n^2} = 0$$

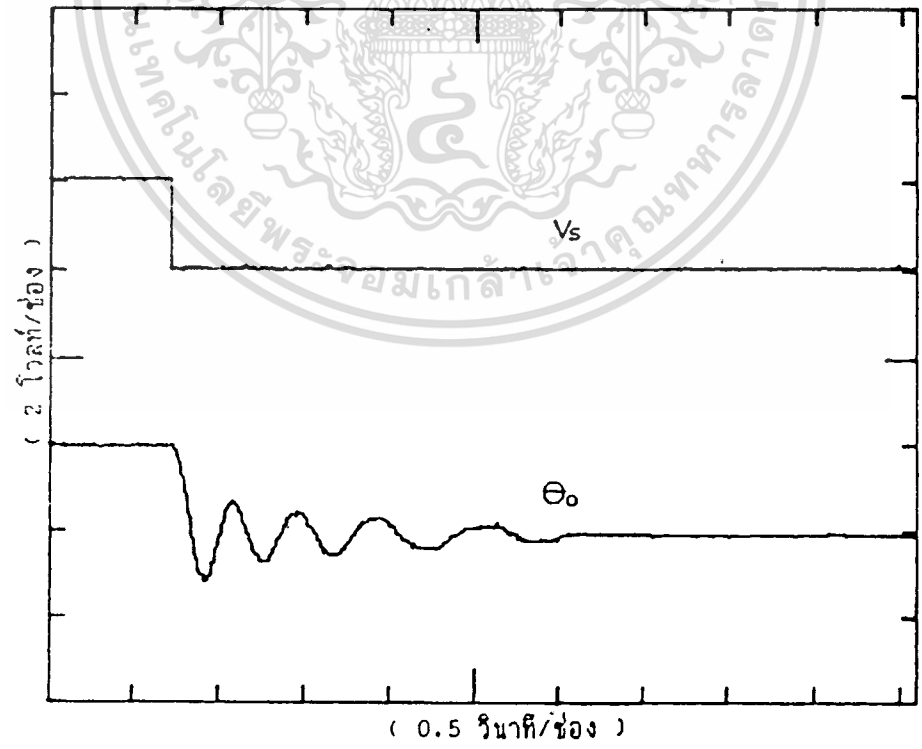
ซึ่งแสดงว่าระบบยังมีความเที่ยงตรงดีแม้จะป้อนสัญญาณอินพุทเป็นแบบขั้นบันไดต่อเนื่อง

4. ผลการทดลอง

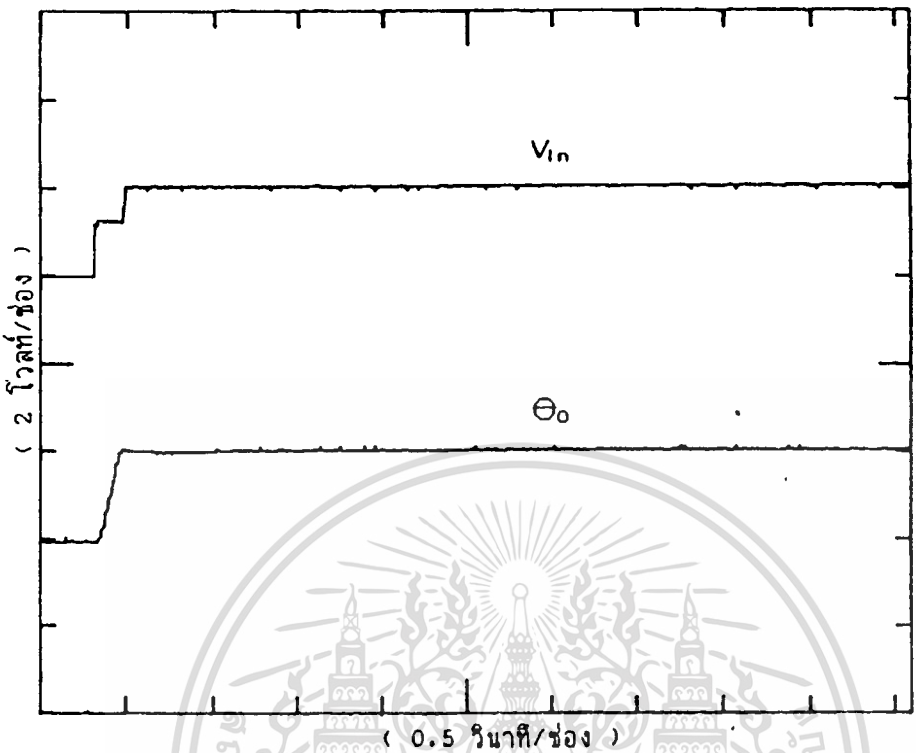
ทำการสร้างระบบขับเคลื่อนดีซีมอเตอร์ตามบล็อกไดอะแกรมรูปที่ 1. แล้วทดลองวัดผลตอบสนองของตำแหน่งที่มีต่อสัญญาณสเตปอินพุทและต่อสัญญาณแบบขั้นบันไดต่อเนื่องซึ่งคำนวณได้จากหัวข้อ 2.2 ทั้งในช่วงออกตัวและหยุดโดยการวัดที่เอาท์พุทของ pot ด้วย storage oscilloscope



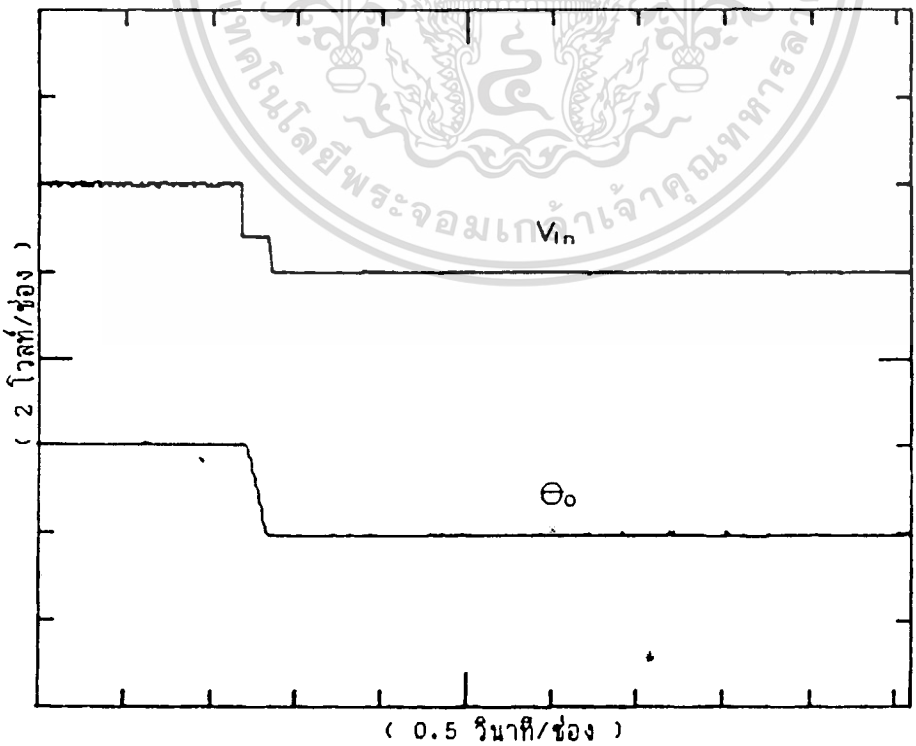
รูปที่ 5. ผลตอบสนองของตำแหน่งในช่วงออกตัวต่อสัญญาณสเตปอินพุทที่ค่าเซ็ทพอยท์ (A + B) เท่ากับ = 2 โวลท์ (Θ_s = 0.666 เรเดียน)



รูปที่ 6. ผลตอบสนองของตำแหน่งในช่วงหยุดต่อสัญญาณสเตปอินพุทที่ Θ_s = -0.666 เรเดียน



รูปที่ 7. ผลตอบสนองของตำแหน่งในช่วงออกตัวต่อสัญญาณขั้นบันไดต่อเนื่องที่ค่าเซ็ทพอยท์(A + B) เท่ากับ 2 โวลท์ หรือ Θ_s = 0.666 เรเดียน



รูปที่ 8. ผลตอบสนองของตำแหน่งในช่วงหยุดต่อสัญญาณขั้นบันไดต่อเนื่องที่ Θ_s = -0.666 เรเดียน

4.6 เปรียบเทียบผลการทดลอง

จากผลการทดลองในรูปที่ 5 ถึง 8 นำมาเปรียบเทียบค่าต่างๆไว้ในตารางที่ 1.นี้

	ช่วงออกตัว(start)		ช่วงหยุด(stop)	
	สเตป (Vs,โวลท์)	ชั้นบันไดต่อเนื่อง (Vin,โวลท์)	สเตป (Vs,โวลท์)	ชั้นบันไดต่อเนื่อง (Vin,โวลท์)
พีคไทม์(tp),วินาที	0.194	0.191	0.195	0.192
เซททริ้งไทม์(ts),วินาที	2.5	0.191	2.42	0.192
โอเวอร์ชูท(Mp),เปอร์เซนต์	60.1	0	62.5	0

ตารางที่ 1. แสดงการเปรียบเทียบผลตอบสนองของตำแหน่งระหว่างสัญญาณสเตปอินพุท และสัญญาณชั้นบันไดต่อเนื่อง

5. สรุป

จากผลการทดลองในรูปที่ 5,6,7, และ 8 สรุปให้เห็นชัดเจนขึ้นในตารางที่ 1. ซึ่งแสดงให้เห็นว่าในช่วงออกตัว สามารถลดโอเวอร์ชูทลงได้ 100 % และลดเวลาในการเข้าสู่ตำแหน่งเป้าหมายได้ 92.36 % ส่วนในช่วงหยุดโอเวอร์ชูทลงได้ 100 % เช่นกันและเวลาในการเข้าสู่ตำแหน่งเป้าหมายลดลงได้ 92.06 %

ผลการทดลองแสดงให้เห็นว่า การออกแบบสัญญาณอินพุทแบบชั้นบันไดต่อเนื่องที่เหมาะสมตามหัวข้อ 2.2 ทำให้สามารถควบคุมตำแหน่งในการออกตัวและหยุดของดีซีมอเตอร์ให้เข้าสู่ค่าเป้าหมายได้ภายในเวลาน้อยที่สุดและไม่มีโอเวอร์ชูท ซึ่งแสดงให้เห็นว่าระบบมีเสถียรภาพที่ดีขึ้น และสามารถกำหนดการเข้าสู่ตำแหน่งเป้าหมายได้อย่างเที่ยงตรง เหมาะสมที่จะนำไปประยุกต์ใช้ในงานควบคุมการเคลื่อนที่ที่มีการเปลี่ยนตำแหน่งไปมาบ่อยๆ และเป็นระบบที่มีค่าโหลดคงที่

เอกสารอ้างอิง

[1]. W.G.Holzbock,"Robotic Technology",Van Nostrand Co.,1986.

[2]. Francois Lhote,etal,"Robot Components and Systems",Kogan page,1983.

[3]. M.A.Berger,"Six Steps to an Optimum Servomotor",Machine Design ,April,pp 248-252,1978.

[4]. B.C.Kuo,"DC Motor and Control Systems",SRL Publishing Co.,1978.

ประวัติผู้เขียน

จบปริญญาตรีวิทยาศาสตร์บัณฑิต สาขาฟิสิกส์ จากมหาวิทยาลัยเชียงใหม่ เมื่อปี 2528 เข้าทำงานบริษัทซีเกทเทคโนโลยี (ประเทศไทย) จำกัด ในตำแหน่งผู้ควบคุมการตรวจสอบคุณภาพ จากนั้นลาออกไปเข้าการปิโตรเลียมแห่งประเทศไทย เมื่อเดือนมีนาคม 2529 ตำแหน่งนักวิทยาศาสตร์ กองวิชาการ โรงแยกก๊าซธรรมชาติ จ.ระยอง แล้วย้ายไปเป็นวิศวกรงานเดินเครื่องประจำกะ C กองเดินเครื่อง โรงแยกก๊าซธรรมชาติ ระหว่างทำงานเคยเดินทางไปอบรมเกี่ยวกับ "Plant Operation" ที่ประเทศแคนาดา" จากนั้นได้ลาออกเมื่อเดือน พฤษภาคม 2531 มาศึกษาต่อปริญญาโทที่ สถาบันเทคโนโลยี พระจอมเกล้าลาดกระบัง คณะวิศวกรรมศาสตร์ ภาควิชาวิศวกรรมระบบควบคุม ระหว่างการศึกษาได้ทำงานเป็นอาจารย์พิเศษสอนวิชา laboratory นักศึกษาชั้นปีที่ 2 และ 3 เป็นเวลา 2 ปีการศึกษา และระหว่างศึกษาต่อได้มีผลงานวิจัยที่ได้รับการตีพิมพ์ดังนี้

1) "การประยุกต์ใช้อินติกรัลคอนโทรลเพื่อควบคุมการเข้าสู่ค่าคงที่ของความเร็วดีซีมอเตอร์ภายในเวลาน้อยที่สุดด้วยสัญญาณอินพุทแบบขั้นบันไดต่อเนื่อง" , วิศวกรรมสาร , เล่มที่ 4 , 2533.

2) "ระบบควบคุมการเคลื่อนที่ในช่วงออกตัวและหยุดภายในเวลาสั้นที่สุดด้วยสัญญาณอินพุทแบบขั้นบันไดต่อเนื่อง" , การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 13 , 2533.