# รายงานการวิจัย

เรื่อง

## โครงการวิจัยพัฒนาประยุกต์ใช้ไมโครโพรเซสเซอร์ในงานด้านหุ่นยนต์

## (Application of Microprocessor in Robotics Research)

โดย

นายสมศักดิ์ เดียวสุรินทร์ *

ภาควิชาฟิสิกส์ คณะวิทยาศาสตร์

มหาวิทยาลัยสงขลานครินทร์ หาดใหญ่

เมื่อได้ให้ฉายเพิ่มพูนความรู้ทางวิชาการเพื่อทำวิจัยในหัวข้อข้างต้น

จาก คณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์

1 มิถุนายน 2535 - 31 พฤษภาคม 2536

# โครงการวิจัยพัฒนาประยุกต์ใช้ไมโครโพรเซสเซอร์ในงานด้านหุ่นยนต์

(Application of Microprocessor in Robotics Research)


## บทคัดย่อ

งานวิจัยนี้ได้ทำการออกแบบสร้างหุ่นยนต์ขนาดเล็ก (น้ำหนักประมาณ 2 กิโลกรัม) โครงสร้างหุ่นยนต์ทำด้วยพลาสติกทั้งหมด หุ่นยนต์ขับเคลื่อนด้วย DC มอเตอร์ขนาด 6 V 4 A สองตัวซึ่งต่อกับล้อสองล้อหุ่นยนต์ถูกควบคุมโดยโปรแกรมระบบซึ่งฝังอยู่ใน EPROM 27256 ในบอร์ดไมโครคอนโทรลเลอร์ 8051 คำสั่งให้หุ่นยนต์ทำงานเขียนได้จากผู้ใช้บน PC คอมพิวเตอร์แล้วโหลดคำสั่งลงใน RAM 6116 ในไมโครคอนโทรลเลอร์

# โครงการวิจัยพัฒนาประยุกต์ใช้ไมโครโพรเซสเซอร์ในงานด้านหุ่นยนต์
# (Application of Microprocessor in Robotics Research)

## คำนำ

อุปกรณ์ชนิดใหม่ที่เกิดขึ้นในสาขาวิชาอิเล็กโทรนิกส์เมื่อประมาณ 15 ปีก่อน ซึ่งมีผลต่อชีวิตประจำวันทุก ๆ คนเช่นเดียวกับการเกิดของทรานซิสเตอร์เมื่อสี่ห้าสิบปีที่แล้ว อุปกรณ์นี้เป็นที่รู้จักกันคือ ไมโครโพรเซสเซอร์ (Microprocessor) ซึ่งประกอบด้วยทรานซิสเตอร์เป็นพัน ๆ ตัวรวมกันเป็นวงจรรวม (Integrated Circuit) บรรจุอยู่ภายในชิพ (chip) อันเดียวซึ่งมีพื้นที่ไม่ถึงหนึ่งตารางนิ้ว

ผลของความก้าวหน้าทางเทคโนโลยีนี้ทำให้เรามีไมโครคอมพิวเตอร์ ซึ่งส่วนประกอบที่สำคัญก็คือไมโครโพรเซสเซอร์ใช้กันอย่างแพร่หลายในปัจจุบัน นอกจากนี้ได้มีการนำไมโครโพรเซสเซอร์ไปประยุกต์ใช้ในเครื่องมือวัดและเครื่องควบคุมอิเล็กทรอนิกส์ชนิดต่าง ๆ อีกมากทั้งทางด้านวิทยาศาสตร์ การแพทย์ และอุตสาหกรรม

จุดประสงค์ของโครงการวิจัยนี้ เพื่อนำไมโครโพรเซสเซอร์มาประยุกต์ใช้ในการควบคุมการทำงานพื้นฐานของหุ่นยนต์บ้าน (Home Robot) ซึ่งผู้วิจัยคาดว่าจะได้รับประโยชน์และประสบการณ์ทั้งในการเขียนโปรแกรมภาษาระดับต่ำ (Assembly language) และการออกแบบวงจรอิเล็กโทรนิกส์ควบคู่กันไปด้วย

## ลักษณะของโครงงาน

โครงงานประกอบด้วยงานสองส่วนใหญ่ ๆ คือ งานทางด้านซอฟท์แวร์ (software) และงานทางด้าน (hardware)

งานทางด้านซอฟท์แวร์เป็นการเขียนโปรแกรมเพื่อควบคุมการทำงานของหุ่นยนต์ โปรแกรมที่เขียนเป็นภาษาแอสแซมบลี้ของไมโครโพรเซสเซอร์ หรือไมโครคอนโทรลเลอร์เบอร์ 8051 ของบริษัทอินเทล (Intel) การเขียนโปรแกรมภาษาแอสแซมบลี้จะใช้เอดิเตอร์ของเวิร์ดโพรเซสเซอร์ใด ๆ ก็ได้ เมื่อเขียนโปรแกรมซึ่งอยู่ในรูป source code แล้ว โปรแกรมจะถูกแปลงเป็น object code โดย Assembler A51 object code นี้จะนำไปทดสอบกับ Simulator ของบริษัท Pseudo Corp. เพื่อตรวจสอบว่าโปรแกรมที่เขียนทำงานถูกต้องตามที่ต้องการหรือไม่ก่อนที่จะนำไปบรรจุลงใน EPROM 27256 ซึ่งเป็นหน่วยความจำถาวรอยู่บนบอร์ดไมโครคอนโทรลเลอร์ 8051

งานทางด้านฮาร์ดแวร์ประกอบด้วย การสร้างตัวหุ่นยนต์ และการออกแบบและทดสอบการทำงานของวงจรควบคุมและขับเคลื่อนมอเตอร์ โครงสร้างตัวหุ่นยนต์เป็นพลาสติกใสทำให้มองเห็นส่วนประกอบในหุ่นยนต์ทั้งหมด ฐานด้านล่างของหุ่นยนต์ยึดติดกับมอเตอร์สองตัวซึ่งใช้เป็นตัวขับเคลื่อนหุ่นยนต์ ฐานด้านบนยึดติดกับบอร์ดไมโครคอนโทรลเลอร์ และบอร์ดวงจรควบคุมและขับเคลื่อนมอเตอร์ ที่ล้อด้านหนึ่งจะมีชุดตรวจจับ (sensor) ซึ่งเป็นไอซีมีตัวรับและตัวส่งแสงอยู่ภายในยึดอยู่ใกล้ ชุดตรวจจับนี้มีไว้เพื่อนับจำนวนรอบที่ล้อหมุนซึ่งจะทำให้รู้ระยะการเคลื่อนที่ของหุ่นยนต์

## ลักษณะเฉพาะของหุ่นยนต์

1. หุ่นยนต์นี้สามารถควบคุมได้ด้วยโปรแกรมคำสั่ง โปรแกรมคำสั่งเขียนจากผู้ใช้บน PC คอมพิวเตอร์แล้วโหลดจาก PC คอมพิวเตอร์ไปยังหน่วยความจำชั่วคราวในหุ่นยนต์ซึ่งมีไมโครโพรเซสเซอร์ที่เรียกว่าไมโครคอนโทรลเลอร์เบอร์ 8051 ประกอบอยู่ในบอร์ด เมื่อโหลดโปรแกรมเรียบร้อยแล้วหุ่นยนต์ก็จะทำงานภายใต้การควบคุมของโปรแกรมคำสั่ง

2. ขนาดของหุ่นยนต์สูง 35 ซม. เส้นผ่าศูนย์กลางของฐาน 28 ซม. และน้ำหนักทั้งหมดประมาณ 2 กิโลกรัม (รูปที่ 1)

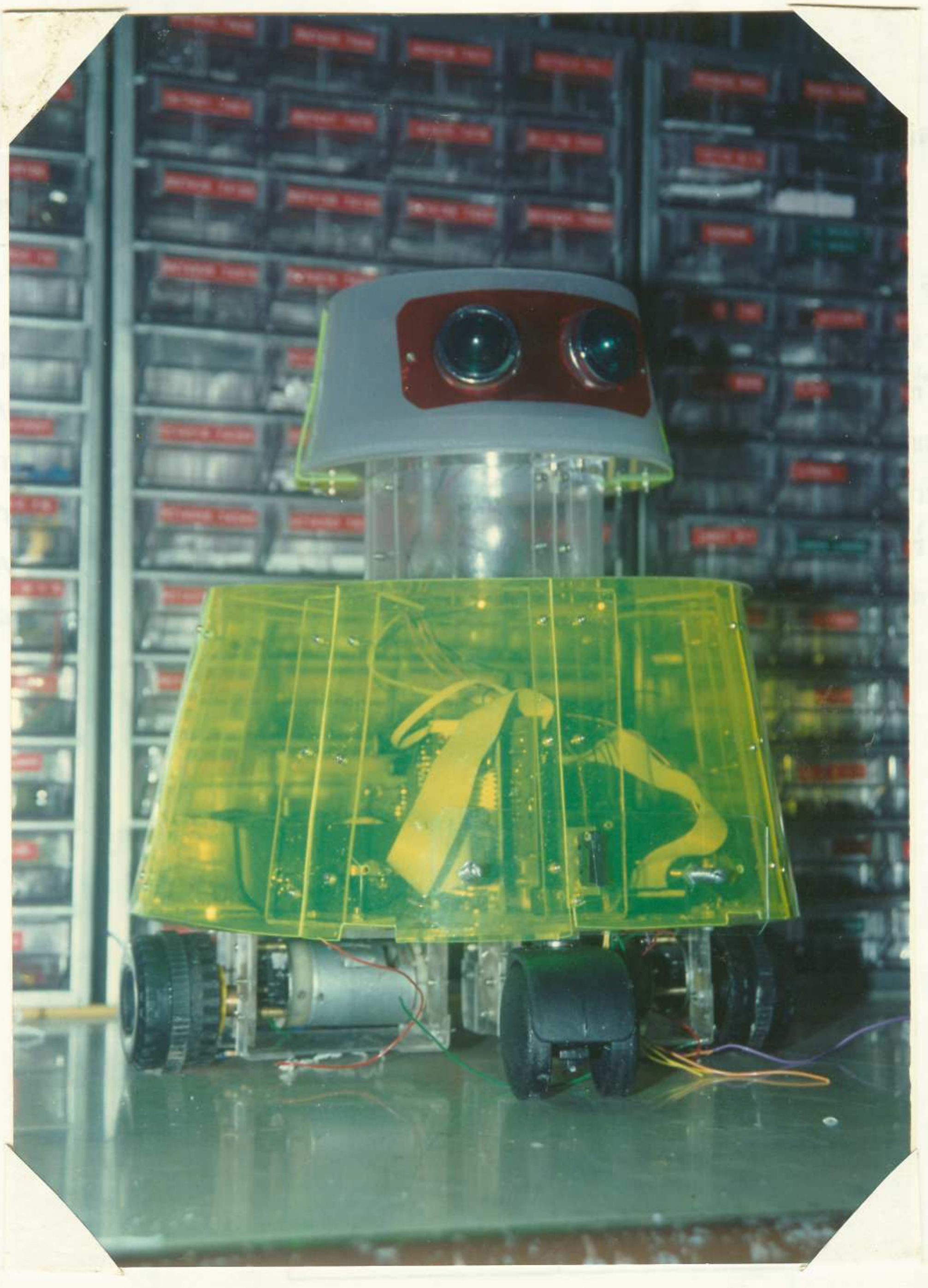
3. หุ่นยนต์ขับเคลื่อนด้วย DC มอเตอร์สำหรับล้อสองล้อ มอเตอร์แต่ละตัวประกอบด้วยเฟืองทดหนึ่งชุดต่อกับล้อ ชุดของมอเตอร์นี้ประกอบอยู่ใต้ฐานพลาสติก

4. สมองของหุ่นยนต์ประกอบด้วยไมโครคอนโทรลเลอร์ 8051 EPROM 27256 และ RAM 6116 ซึ่งทั้งหมดประกอบอยู่บนบอร์ดสำเร็จของบริษัท ETT

5. วงจรควบคุมและขับเคลื่อนมอเตอร์ ซึ่งมีหน้าที่ควบคุมทิศทางการหมุนของมอเตอร์ และยังมีหน้าที่เป็นตัวเชื่อม (interface) ระหว่างบอร์ดไมโครคอนโทรลเลอร์ และอุปกรณ์ภายนอก เช่น ตัวตรวจจับ (sensor) และสวิทช์ต่าง ๆ

6. แหล่งจ่ายไฟตรง 5 V สำหรับบอร์ดไมโครคอนโทรลเลอร์ และแหล่งจ่ายไฟตรง 6 V สำหรับมอเตอร์ทั้งสอง

7. โหลดโปรแกรมคำสั่งผ่าน RS-232C ซึ่งเป็นพอร์ตอนุกรมของ PC คอมพิวเตอร์ ด้วยความเร็ว 4800 bps โปรแกรมที่ใช้ในการโหลดโปรแกรมคำสั่งคือ PC plus ของ Procomm —



รูปที่ 1 หุ่นยนต์ต้นแบบ

## วงจรควบคุม

หุ่นยนต์ประกอบด้วยวงจรควบคุมการทำงานสองบอร์ดคือ บอร์ดไมโครคอนโทรลเลอร์ และบอร์ดควบคุมและขับเคลื่อนมอเตอร์

บอร์ดไมโครคอนโทรลเลอร์เป็นบอร์ดคอมพิวเตอร์เอนกประสงค์ของบริษัท ETT ซึ่งในบอร์ดประกอบด้วยไมโครคอนโทรลเลอร์ 8051 หน่วยความจำถาวร EPROM 27256 ขนาด 32K หน่วยความจำชั่วคราว RAM ขนาด 8K และ Programable Phcripheral Interface 8255 โปรแกรมที่ใช้ในการรับคำสั่งจาก PC คอมพิวเตอร์บรรจุอยู่ใน EPROM ตำแหน่ง 0000H-1FFFH ส่วนโปรแกรมสำหรับควบคุมการทำงานบรรจุอยู่ใน EPROM ตำแหน่ง 4000H ขึ้นไป โปรแกรมคำสั่งจะเก็บไว้ในหน่วยความจำชั่วคราว RAM 6116 ตำแหน่ง 2200H ขึ้นไป พอร์ทที่ใช้เชื่อมต่อระหว่างไมโครคอนโทรลเลอร์ และวงจรควบคุมและขับเคลื่อนมอเตอร์คือพอร์ทหนึ่ง (P1) หน้าที่ของขาแต่ละขาของพอร์ทหนึ่งแสดงในตารางที่ 1

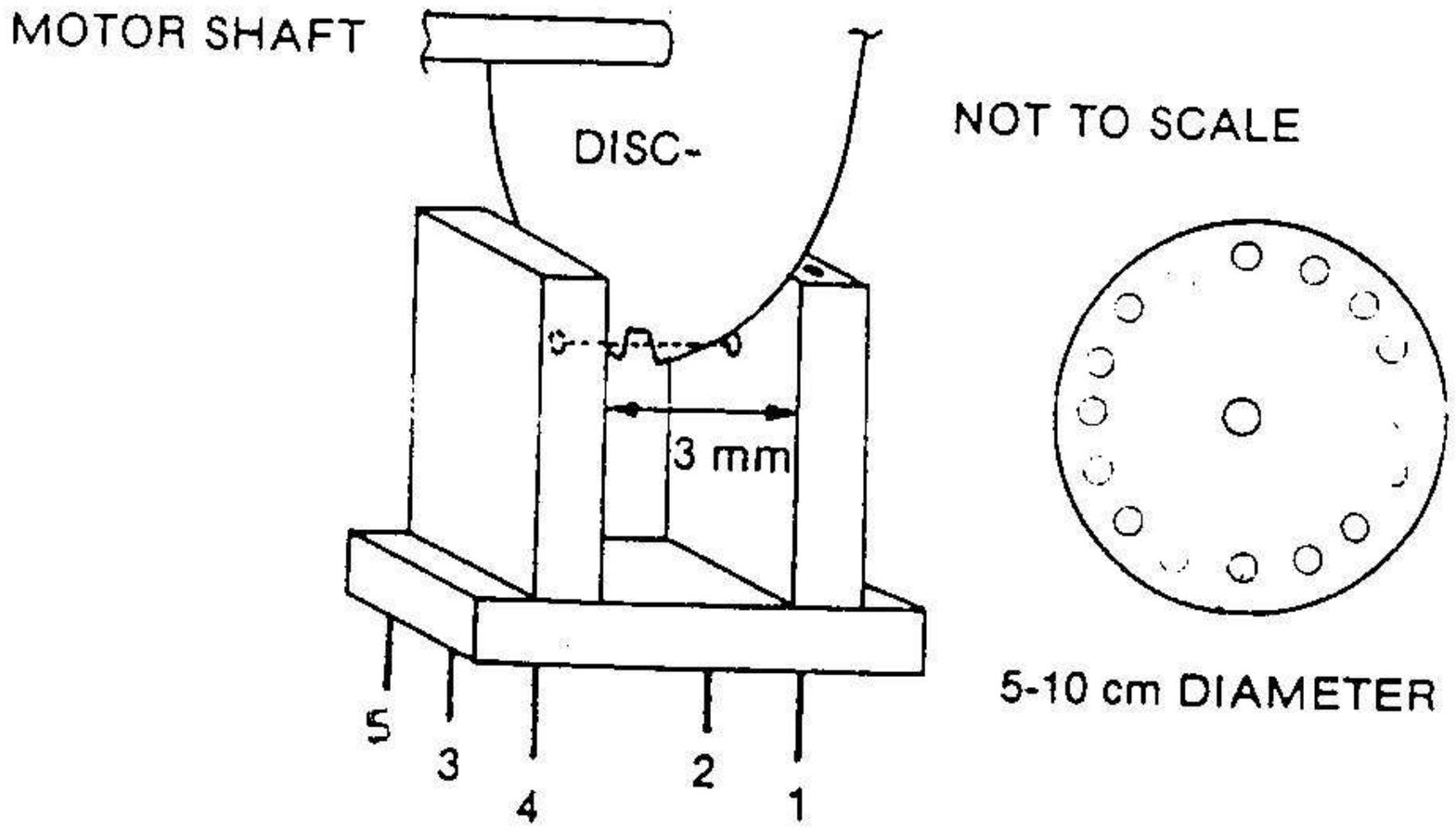
| เลขที่ขา (pin no) | หน้าที่ (Function) |
|---|---|
| P0 | Lear Bumper |
| P1 | Front Bumper |
| P2 | Nose |
| P3 | Beep |
| P4 | Eye |
| P5 | Right Motor |
| P6 | Left Motor |
| P7 | Enable Motors |

ตารางที่ 1 เลขที่ขาและหน้าที่ของพอร์ท P1

บอร์ดควบคุมและขับเคลื่อนมอเตอร์ประกอบด้วยวงจรขับเคลื่อนมอเตอร์ที่มีลักษณะเหมือนกันสองชุด ชุดหนึ่งสำหรับมอเตอร์หนึ่งตัว มอเตอร์แต่ละตัวจะถูกควบคุมโดยวงจรซึ่งมีทรานซิสเตอร์ควบคุมการ

รูปที่ 2 วงจรควบคุม/ขับเคลื่อนมอเตอร์

ปิด-เปิดของรีเลย์ (Relay) ตัววงจรแสดงในรูปที่ 2 บิทควบคุมการหมุนของมอเตอร์ส่งมาจากพอร์ทหนึ่งจากไมโครคอนโทรลเลอร์ นอกจากนี้จะควบคุมการขับเคลื่อนมอเตอร์แล้ว ยังมีหน้าที่เชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับอุปกรณ์ภายนอก เช่น ตา จมูก กันชน และสวิทช์ทางแสง (Optical limit switch) ด้วย ตาทั้งสองข้างจะเป็นหลอดไฟซึ่งมีแหล่งจ่ายไฟเป็นแบตเตอรี่ขนาด 3 V อยู่ในตัวของมันเอง ภายในหลอดไฟประกอบด้วย LED หลายตัวบรรจุอยู่ ส่วนที่เป็นจมูกแทนด้วยสวิทช์กดติดปล่อยดับ (push button switch) ติดอยู่บนหัวของหุ่นยนต์ ส่วนกันชน (Bumper) ทั้งสองข้างเป็นแผ่นพลาสติกติดกับไมโครสวิทช์ เมื่อกันชนชนกับวัตถุไมโครสวิทช์จะปิดและส่งสัญญาณไปยังไมโครคอนโทรลเลอร์ ดังนั้นไมโครคอนโทรลเลอร์จะรับรู้สถานะการชนได้จากไมโครสวิทช์ทั้งสองตัวนี้ สำหรับสวิทช์ทางแสงมีหน้าที่ผลิตสัญญาณรูปสี่เหลี่ยม (square wave) ป้อนให้กับไมโครคอนโทรลเลอร์ สัญญาณนี้จะป้อนเข้าที่ขา INTO ลักษณะของสวิทช์ทางแสง (รูปที่ 3) ประกอบด้วยตัวส่งสัญญาณแสงซึ่งเป็น LED และตัวรับซึ่งเป็นโฟโต้ทรานซิสเตอร์ ทั้งตัวรับและส่งบรรจุเป็นหน่วยเดียวกัน โดยมีช่องว่างประมาณ 3 mm เพื่อให้สามารถเอาแผ่นดิสก์สอดเข้าระหว่างช่องว่างนี้ แผ่นดิสก์มีเส้นผ่าศูนย์กลางประมาณ 4 ซม.




รูปที่ 3 สวิทช์ทางแสง (optical limit switch)

เจาะเป็นรู ๆ ตามแนวเส้นรอบวง 16 รู ระยะห่างระหว่างรูมีค่าเท่ากัน แผ่นดิสก์นี้ติดอยู่กับแกนของมอเตอร์ เมื่อมอเตอร์หมุนแผ่นนี้ก็จะหมุนไปด้วย เมื่อรูอยู่ระหว่างแนวทางเดินของแสงระหว่าง LED และโฟโต้ทรานซิสเตอร์ ทรานซิสเตอร์จะนำกระแส (ON) และให้เอาท์พุทออกมาเป็น LOW เมื่อรูไม่อยู่ในแนวทางเดินของแสง ทรานซิสเตอร์ไม่นำกระแส (OFF) และให้เอาท์พุทออกมาเป็น HIGH ดังนั้นขณะที่มอเตอร์หมุนจะมีสัญญาณ HIGH และ LOW สลับไปมาที่เอาท์พุทของโฟโตทรานซิสเตอร์

เราสามารถวัดระยะทางที่ล้อเคลื่อนที่ไปได้โดยการวัดจำนวนพัลซ์ของสัญญาณนี้ สำหรับหุ่นยนต์นี้เส้นรอบวงของล้อเท่ากับ 19 ซม. เนื่องจากตัวนับ (counter) นับได้สูงสุด 255 ดังนั้นระยะเคลื่อนที่สูงสุดของแต่ละคำสั่งคือ 255 x 19/16 = 302.8 ซม. ระยะเคลื่อนที่น้อยที่สุดคือ 19/16 = 1.19 ซม.

## การเขียนโปรแกรมคำสั่ง

โปรแกรมคำสั่งเขียนบน PC โดยใช้เอดิเตอร์ของ side kick หรือ Q เอดิเตอร์ก็ได้ คำสั่งต้องเขียนอยู่ในรูปของภาษาแอสแซมบลี้ของไมโครคอนโทรลเลอร์ 8051 คำสั่งทุก ๆ คำสั่งยกเว้นคำสั่ง E ประกอบด้วยตัวอักษรแล้วตามด้วยพารามิเตอร์ซึ่งเป็นเลขฐานสิบหกสองตัว ตัวอย่างเช่น Fnn เมื่อ nn คือ เลขฐาน 16 มีค่าระหว่าง 01H-FFH คือคำสั่งให้หุ่นยนต์เคลื่อนที่ไปข้างหน้าเป็นระยะทางเท่าที่กำหนดโดยพารามิเตอร์ nn คำสั่งทั้งหมดมี 8 คำสั่ง ดังตารางที่ 2

| คำสั่ง (command) | หน้าที่ (Function) |
|---|---|
| F (Forward) | เคลื่อนที่ไปข้างหน้า |
| B (Backward) | เคลื่อนที่ไปข้างหลัง |
| R (Rotate Right) | หมุนขวา |
| L (Rotate Left) | หมุนซ้าย |
| H (Honk) | ทำเสียงปี๊บ ๆ |
| I (Eye) | เปิดตา |
| P (Pause) | หยุด |
| E (End program) | จบโปรแกรม |

ตารางที่ 2 คำสั่งและหน้าที่

## รายละเอียดของคำสั่ง

- Fnn (Forward) : คำสั่งนี้ทำให้หุ่นยนต์หมุนล้อไปด้านหน้าทั้งสองล้อ หุ่นยนต์จะเคลื่อนที่ไปด้านหน้าจนกระทั่งจำนวนพัลซ์ที่ป้อนจากสวิทช์ทางแสงมีจำนวนเท่ากับพารามิเตอร์ nn ค่า 01H มีค่าประมาณ 1.19 ซม.
- Bnn (Backward) : คำสั่งนี้ทำให้หุ่นยนต์หมุนล้อไปด้านหลังทั้งสองล้อ
- Rnn (Right) : คำสั่งนี้ทำให้หุ่นยนต์หมุนตามเข็มนาฬิกา (มองจากด้านบน) รอบจุดกึ่งกลางระหว่างล้อสองล้อ โดยที่ล้อซ้ายเคลื่อนไปไปด้านหลังและล้อขวาเคลื่อนที่ไปด้านหน้า ล้อจะหมุนไปเรื่อย ๆ จนกระทั่งจำนวนพัลซ์เท่ากับพารามิเตอร์ nn
- Lnn (Left) : คำสั่งนี้คล้ายกับ Rnn แต่หมุนในทิศตรงข้ามกันคือ หุ่นยนต์จะหมุนทวนเข็มนาฬิกา
- Hnn (Honk) : คำสั่งนี้ทำเสียงปี๊บ ๆ nn ครั้ง ขณะทำเสียงหุ่นยนต์จะหยุดการทำงานอื่นทั้งหมด
- Inn (Eye) : คำสั่งนี้ทำให้ ปิด-เปิดตา ถ้าพารามิเตอร์เป็น 00H ตาไม่สว่าง ค่าอื่น ๆ นอกจาก 0HH จะทำให้ตาสว่าง
- Pnn (Pause) : คำสั่งนี้ทำให้หุ่นยนต์หยุดทำงานใด ๆ เป็นเวลาเท่ากับเลขฐานสิบที่สอดคล้องกับเลขฐานสิบหก มีหน่วยเป็นวินาที เช่น P01 ทำให้หุ่นยนต์หยุดเป็นเวลา 65 วินาที

ข้างล่างนี้เป็นตัวอย่างของการเขียนโปรแกรมคำสั่งเพื่อให้หุ่นยนต์ทำงาน

```
;This is an example of command program written by user to
;operate the robot.

        .equ inrtn0,423ah       ;interrupt routine address
        .org 2200h              ;start from 2200h in data memory
        ljmp 4000h              ;go to code memory
        .org 2203h              ;INT0 routine has move to here
        ljmp inrtn0             ;move INT0 to this address
        .org 2300h              ;store command word at 2300h and up
        .db "r"                 ;rotate right 16
        .db 0ffh                ;16 turn of the wheel
        .db "p"                 ;pause for
        .db 01h                 ;65 second
        .db "i"                 ;turn on eye
        .db 024h
        .db "f"                 ;go forward
        .db 01h                 ;22cm
        .db "i"
        .db 00h                 ;turn off eye
        .db "e"                 ;end command
        .end
```

รูปแบบโปรแกรมคำสั่งเป็นภาษาแอสแซมบลี้ของ 8051 ผู้ใช้หุ่นยนต์ต้องพิมพ์ 6 บรรทัดตามตัวอย่างเสมอ บรรทัดแรกคือที่อยู่ของอินเทอรัพท์รูทีน บรรทัดที่สองบอกให้เก็บโปรแกรมนี้ที่แอดเดรส 2200H ใน RAM บรรทัดที่สามคือที่อยู่ของโปรแกรมระบบ บรรทัดที่สี่และห้าบอกให้ไมโครคอนโทรลเลอร์ไปที่แอดเดรส 423aH ซึ่งเป็นแอดเดรสเริ่มต้นของ อินเทอรัพท์รูทีน บรรทัดที่ 6 คือแอดเดรสเริ่มต้นของคำสั่งอยู่ที่ 2300H หลังจากบรรทัดที่ 6 ผู้ใช้ก็เขียนคำสั่งที่มีอยู่ตามด้วยพารามิเตอร์ โปรแกรมคำสั่งนี้บอกให้หุ่นยนต์ทำงานดังนี้

- หมุนขวา 16 รอบ (รอบล้อ)
- หยุดพัก 65 วินาที
- เปิดตา
- เดินไปข้างหน้า 22 ซม.
- ปิดตา
- ส่งเสียงปี้บ ๆ

หุ่นยนต์จะส่งเสียงไปเรื่อย ๆ จนกว่าผู้ใช้จะกดปุ่น Nosc แล้วมันจะทำงานซ้ำแบบเดิม

## การโหลดโปรแกรมคำสั่ง

เมื่อเขียนโปรแกรมคำสั่งตามรูปแบบที่กำหนดแล้ว ขั้นต่อไปต้องแปลงคำสั่งซึ่งอยู่ในรูป source code ให้เป็น object code โดยใช้แอสแซมเบลอร์ของ 8051 ซึ่งมีชื่อว่า A51 หลังจากได้ object code แล้ว การโหลด object code จากคอมพิวเตอร์ไปที่หุ่นยนต์ให้ทำตามขั้นตอนดังนี้

1. เสียบสายเชื่อมต่อระหว่างหุ่นยนต์และพอร์ทอนุกรม RS-232 ของ PC คอมพิวเตอร์
2. เปิดเครื่องหุ่นยนต์แล้วกดปุ่น reset บนหัวของหุ่นยนต์ จะปรากฏข้อความดังนี้บนจอ PC คอมพิวเตอร์

```
ET  DEBUGGER-31
Version 1.0
By ETT Co., Ltd. 1991
```

3. ใส่แผ่นดิสก์ซึ่งมีโปรแกรม PROCOM ลงในดิสก์ไดรฟ์ แล้วเรียกโปรแกรมนี้โดยพิมพ์ PCP
4. ส่งไฟล์จาก PC ไปยัง ET DEBUGGER-31 ทำได้ดังนี้
   - HR < ENTER >
   - กด PgUp บนคีย์บอร์ด
   - เลือกการส่งไฟล์แบบ ASCII (กดเลข 4)
   - พิมพ์ชื่อไฟล์ที่ต้องการส่ง < กด ENTER >
5. เมื่อส่งโปรแกรมเข้าไปในหุ่นยนต์แล้วจะปรากฏข้อมูลของคำสั่งแสดงบนจอของ PC ข้อมูลนี้จะเป็นตัวเลขซึ่งอยู่ในรูปของ object code ดังตัวอย่าง

```
:0322000002400099
:032203000242 3A5A
:0B23000072FF700169246601690065 2E
:00000001FF
```

object code ของโปรแกรมคำสั่ง

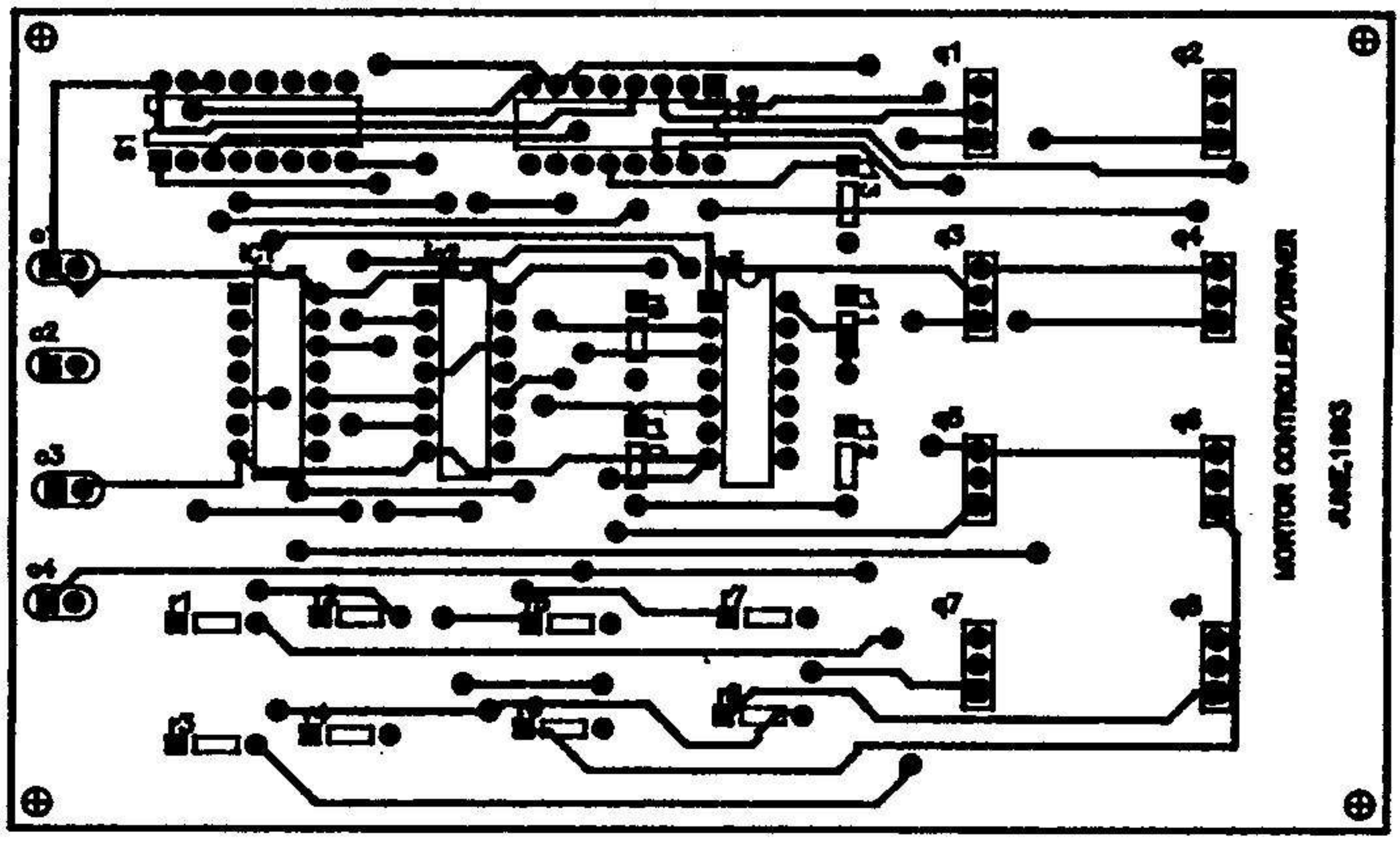
6. กด G < ENTER >
7. ถอดสายเชื่อมต่อระหว่าง PC กับหุ่นยนต์
8. กด Nosc

# สรุป

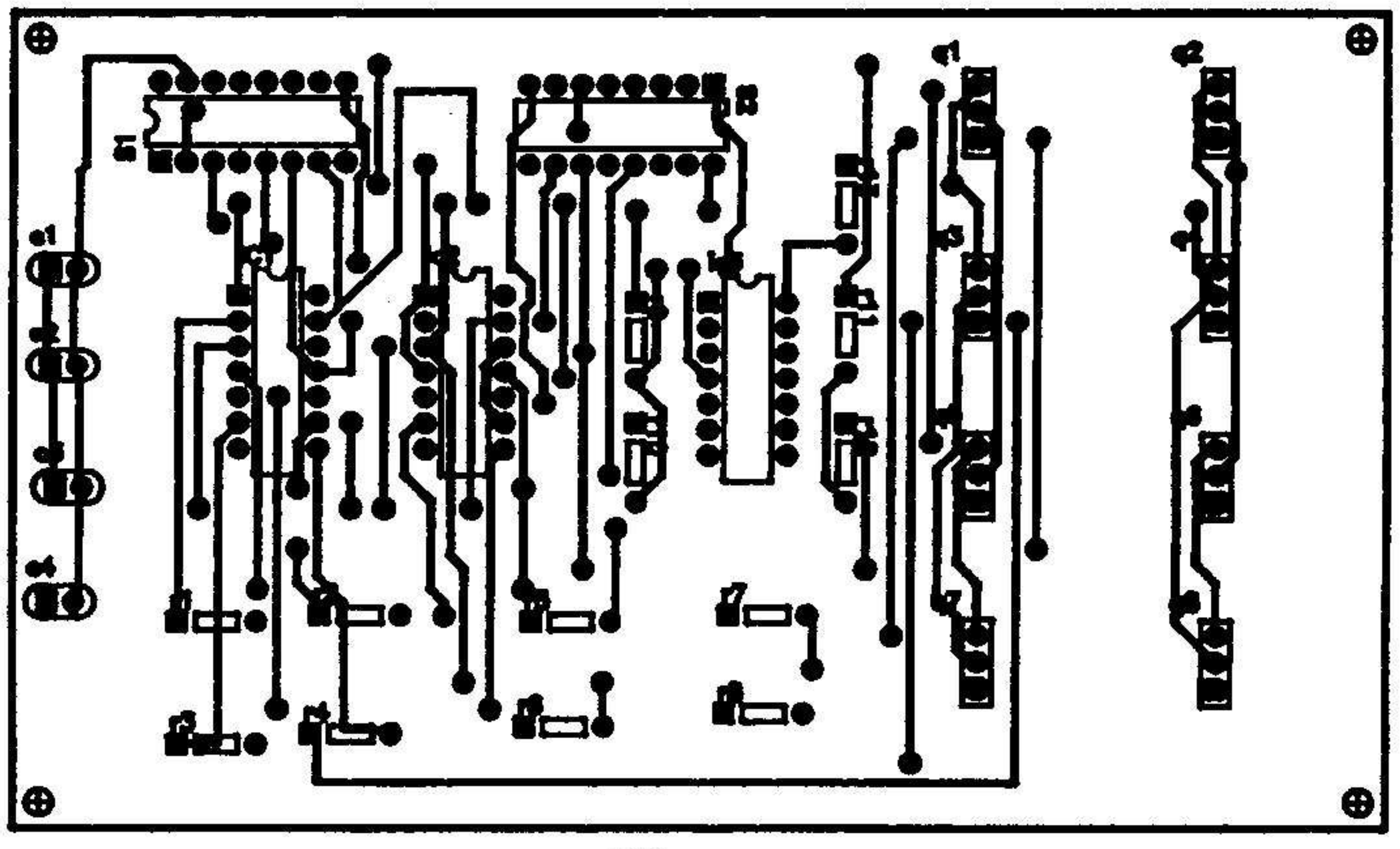
จากการทดสอบการทำงานของหุ่นยนต์นี้พบว่า ได้ผลดีพอสมควร จากเป้าหมายให้หุ่นยนต์ทำหน้าที่ได้เจ็ดอย่าง หุ่นยนต์ทำงานได้สมบูรณ์แบบห้าหน้าที่คือ หมุนซ้าย หมุนขวา ทำเสียงปี๊บ ๆ ทำตากระพริบ และหยุด ส่วนสองหน้าที่ที่ทำงานได้ไม่สมบูรณ์ คือ การเคลื่อนที่ไปข้างหน้าและหลัง การเคลื่อนที่ข้างหน้าและหลังหุ่นยนต์เคลื่อนที่ได้ไม่เป็นเส้นตรง สาเหตุที่เคลื่อนที่เป็นเส้นตรงไม่ได้ก็เพราะมอเตอร์ทั้งสองที่ใช้ในการขับเคลื่อนล้อหมุนด้วยความเร็วไม่เท่ากัน ดังนั้นถ้าจะพัฒนาหุ่นยนต์ต้นแบบนี้ให้สมบูรณ์มากขึ้น ผู้ออกแบบจะต้องดัดแปลงแก้ไขในส่วนของวงจรควบคุมและขับเคลื่อนมอเตอร์ให้สามารถควบคุมอัตราเร็วของการหมุนของมอเตอร์ด้วย

โอกาสที่จะนำหุ่นยนต์นี้ไปประยุกต์ใช้งานด้านต่าง ๆ อาจเป็นไปได้ถ้าสามารถเพิ่มหน้าที่การทำงานเพิ่มขึ้นให้แก่หุ่นยนต์นี้ เช่น เพิ่มหน้าที่ในการพูดตามเสียงที่เราบันทึกไว้ในหน่วยความจำ หรือเพิ่มเติมวงจรเพื่อบังคับให้หุ่นยนต์เคลื่อนที่ไปตามทางซึ่งมีแถบสีเป็นตัวนำทาง โดยเพิ่มตัวตรวจจับสี (color sensor) เข้ากับระบบที่มีอยู่ เป็นต้น

# ภาคผนวก




1)



2)

ลายพิมพ์วงจรควบคุม/ขับเคลื่อนมอเตอร์

1) ด้านบน

2) ด้านล่าง

# อุปกรณ์สำหรับแผงวงจรควบคุม/ขับเคลื่อนมอเตอร์

| | |
|---|---|
| $R_1$-$R_{10}$ | 220 phm 1/4 W |
| $R_9$-$R_{13}$ | 10 K 1/4 W |
| IC1 | 74LS04 |
| IC2 | 74LS08 |
| IC3 | 74LS14 |
| C1-C4 | 0.1 uF |
| Q1-Q4 | transistor TIP 32 |
| S1-S2 | 16-pin DIP socket |
| L1-L4 | Relay 6 V |

Cable, ribbon 16-pin DIP

Socket, 14-pin DIP (สำหรับ IC1, IC2 และ IC3)

```
;This is the program to control Robot operations. The program
;accept the command word in the assembly format. There are eight
;command words for the operations,each command word has it 's
;own address routine. The origin of program is at 4000h in the
;code memory. The command word program has to be loaded at 2300h
;in data memory.

        .equ comadr,2300h     ;begining location of command address
        .equ savcnt,30h       ;counter address
        .equ comend,65h       ;code of end command word
        .equ ncmds,07h        ;number of command word
        .equ buff,40h         ;buffer for command word
        .equ nobeep,50h       ;buffer for number of beep

        .org 4000h
        mov p1,#7Fh           ;set up port 1
cbeep:  lcall fbeep           ;make sound beep
        clr p1.4              ;turn on eye
        jnb p1.2,start        ;poll nose
        sjmp cbeep
start:  setb p1.4             ;turn off eye
        mov dptr,#comadr
nextc:  movx a,@dptr          ;get command word then compare to
        push dph              ;end command word
        push dpl
        cjne a,#comend,word
        ljmp endcom           ;jump to endcom

word:   mov buff,a            ;save command word at buffer
        mov a,#00h
        mov dptr,#table       ;point to begining of command table
        mov r3,#ncmds         ;store number of commands in r3

comwd:  movc a,@a+dptr
        cjne a,buff,nmatch    ;compare code in buffer to code in
        mov a,#00h            ;the table,if it is the same then
        inc dptr              ;get the address of the code
        movc a,@a+dptr
        mov r1,a
        mov a,#00h
        inc dptr
        movc a,@a+dptr
        mov r2,a
        mov dph,r1
        mov dpl,r2
        mov a,#00h
        jmp @a+dptr

nmatch: inc dptr
        inc dptr              ;it is not first code in the table
        inc dptr              ;it must be checked with the next
        mov a,#00h            ;codes until it is matched
        djnz r3,comwd
        ljmp error            ;error code

        .org 4150h
table:  .db "l"               ;rotate left
        .dw lcom
        .db "b"               ;go backward
        .dw bcom
        .db "r"               ;rotate right
        .dw rcom
        .db "f"               ;go forward
        .dw fcom
```

```
            .db "h"                 ;make sound beep
            .dw hcom
            .db "i"                 ;turn on eye
            .dw icom
            .db "p"                 ;pause
            .dw pcom

lcom:       pop  dpl
            pop  dph
            inc  dptr
            movx a,@dptr            ;get parameter of commmand "l"
            mov savcnt,a            ;save at savcnt
            setb p1.7               ;enable motors
            setb p1.5               ;turn on left motor clockwise
            clr p1.6                ;turn on right motor counter clockwise
            mov ie,#81h             ;enable INT0
            mov tcon,#01h           ;enable trigger for INT0
            mov r4,#00h             ;clear counter
allowl:     mov a,r4                ;get number of count
            cjne a,savcnt,allowl    ;compare counts in counter to savcnt
            clr p1.7                ;turn off motor
            mov ie,#00h             ;disable interrupt
            mov r4,#00h             ;clear  counter
            inc dptr
            ljmp nextc              ;get next command word

bcom:       pop dpl
            pop dph
            inc dptr
            movx a,@dptr            ;get parameter of command "b"
            mov savcnt,a            ;save at savcnt
            setb p1.7               ;enable motors
            clr p1.5                ;turn on left motor counter clockwise
            clr p1.6               ;turn on right motor counter clockwise
            mov ie,#81h             ;enable INT0
            mov tcon,#01h           ;enable edge trigger for INT0
            mov r4,#00h             ;clear counter
allowb:     mov a,r4                ;get number of count
            jnb p1.0,bstop          ;poll back bumper
            cjne a,savcnt,allowb    ;compare counts in counter to savcnt
bstop:      clr p1.7                ;turn off motor
            mov ie,#00h             ;disable interrupt
            mov r4,#00h             ;clear counter
            inc dptr
            ljmp nextc              ;get next command word

rcom:       pop dpl
            pop dph
            inc dptr
            movx a,@dptr            ;get parameter of command "r"
            mov savcnt,a            ;save at savcnt
            setb p1.7               ;enable motors
            clr p1.5               ;turn on left motor counter clockwise
            setb p1.6               ;turn on right motor clockwise
            mov ie,#81h             ;enable INT0
            mov tcon,#01h           ;enable edge trigger for INT0
            mov r4,#00h             ;clear counter
allowr:     mov a,r4                ;get number of count
            cjne a,savcnt,allowr    ;compare counts in counter to savcnt
            clr p1.7                ;turn off motor
            mov ie,#00h             ;disable interrupt
            mov r4,#00h             ;clear counter
            inc dptr
```

```
        ljmp nextc              ;get next command word

fcom:   pop dpl
        pop dph
        inc dptr
        movx a,@dptr            ;get parameter of command "f"
        mov savcnt,a            ;save at savcnt
        setb p1.7               ;enable motors
        setb p1.5               ;turn on left motor clockwise
        setb p1.6               ;turn on right motor clockwise
        mov ie,#81h             ;enable INT0
        mov tcon,#01h           ;enable edge trigger for INT0
        mov r4,#00h             ;clear counter
allowf: mov a,r4                ;get number of count
        jnb p1.1,fstop          ;poll front bumper
        cjne a,savcnt,allowf    ;compare counts in  counter to savcnt
fstop:  clr p1.7                ;turn off motor
        mov ie,#00h             ;disable interrupt
        mov r4,#00h             ;clear counter
        inc dptr
        ljmp nextc              ;get next command word

hcom:   pop dpl
        pop dph
        inc dptr
        movx a,@dptr            ;get parameter of command "h"
        mov nobeep,a            ;save parameter in nobeep buffer
repeat: acall fbeep             ;call "Beep routine"
        djnz nobeep,repeat
        inc dptr
        ljmp nextc              ;get next command word

icom:   pop dpl
        pop dph
        inc dptr
        movx a,@dptr            ;get parameter of command "i"
        cjne a,#00h,eye         ;if a=00 turn off eye
        setb p1.4               ;otherwise turn on eye
eye:    clr p1.4
        inc dptr
        ljmp nextc              ;get next command word

pcom:   pop dpl
        pop dph
        inc dptr
        movx a,@dptr            ;get parameter of command "p"
        mov r6,a                ;uses r5 as a counter
pause:  mov a,#0ffh             ;load parameter 65 sec for timer
        mov b,#0ffh
        acall timer
        djnz r6,pause
        inc dptr
        ljmp nextc              ;get next command word

endcom: lcall cbeep

error:  clr p1.4
        lcall cbeep

;Interrupt/ This is interrupt routine for INT0

inrtn0: inc r4                  ;count the number of pulse from
        reti                    ;the hole signal
```

```
;Timer/ The time delay routine name "Timer" uses timer0 and
      ;register a and b to generat delays from 1 to 65,535d
      ;milliseconds. The calling program loads register a(lsb)
      ;and b(msb) with desired delay in milliseconds. Loading a
      ;delay of 0000h results in an immediate return.

      .equ onemshi,0fah     ;2's coplement of 535h=facbh
      .equ onemslo,0cbh

timer: push tl0             ;save timer0 contens
      push th0
      cjne a,#00h,go        ;test for a=0
      orl a,b               ;a=00, test for b=00
      jz done               ;a will be 00 if a=b=00
      clr a                 ;b is not 00, clear a
go:   anl tcon,#0cfh        ;clear timer0 over flow and run
                            ;flags in TCON
      anl tmod,#0f0h        ;clear T0 part of TMOD, set T0 for
      orl tmod,#01h         ;timer operation, mode 1 (16bit)
onems: mov tl0,#onemslo     ;set T0 to count up from facbh
      mov th0,#onemshi
      orl tcon,#10h         ;start timer0
wait: jbc tf0,dwnab         ;poll T0 overflow flag
      sjmp wait             ;loop until T0 overflows
dwnab: anl tcon,#0efh       ;stop T0
      djnz acc,onems        ;count a down and loop until zero
      cjne a,b,bdown        ;if a=b=00 then done,return
      sjmp done
bdown: dec b                ;decrement b and count again
      sjmp onems
done: pop th0               ;restore T0 contens
      pop tl0
      ret

;Beep routine/This beep routine uses r5 as counter to count
            ;number of sound beep.

fbeep: mov r5,#30h          ;set number of pulse/beep
pbeep: mov a,#01h           ;set a=01 for 1 ms
      setb p1.3             ;high for 1 ms
      acall timer
      mov a,#01h
      clr p1.3              ;low for 1 ms
      acall timer
      djnz r5,pbeep         ;loop until r5=00
      mov r5,#30h
nbeep: mov a,#01h           ;set a=01 for 1 ms
      clr p1.3              ;low for 1 ms
      acall timer
      mov a,#01h
      clr p1.3              ;low for 1ms
      acall timer
      djnz r5,nbeep         ;loop until r5=00
      ret
      .end
```