

## คู่มือหุ่นยนต์ AP114 PIC ROBOT V2.0

## แนะนำอุปกรณ์ต่างๆ ที่สำคัญ

## 1. ไอซีไมโครคอนโทรลเลอร์

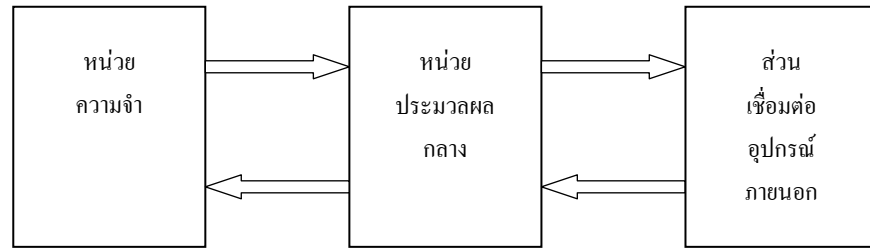
ไอซีไมโครคอนโทรลเลอร์ (Microcontroller) คืออะไร

ไอซีไมโครคอนโทรลเลอร์ คือ อุปกรณ์อิเล็กทรอนิกส์ชนิดหนึ่งซึ่งรวมเอาอุปกรณ์ทางด้านอิเล็กทรอนิกส์ต่างๆ เข้ามาไว้ภายในตัวมัน โดยข้อแตกต่างของไอซีคอนโทรลเลอร์กับไอซีโดยทั่วไปก็คือ ภายในตัวมันสามารถแก้ไขเปลี่ยนแปลงคำสั่งในการทำงานของมันได้ โดยอาศัยโปรแกรมภายในหน่วยความจำ ซึ่งเราสามารถเขียนขึ้นมาได้ด้วยตัวเอง ทำให้เราสามารถนำไอซีไมโครคอนโทรลเลอร์ไปประยุกต์ใช้งานในด้านต่างๆ ได้อย่างมากมาย เช่น เครื่องซักผ้าอัตโนมัติ, ตู้น้ำหยอดเหรียญ, วิทยุ, โทรทัศน์ เป็นต้น ซึ่งอุปกรณ์ต่างๆ เหล่านี้จะมีไอซีไมโครคอนโทรลเลอร์เป็นหัวใจหลักหรือเป็นสมองให้กับเครื่องใช้ไฟฟ้าต่างๆ เหล่านี้ คอยทำการสั่งให้กับอุปกรณ์ที่ต่อร่วมทำงานอย่างถูกต้อง

ความแตกต่างระหว่างไอซีไมโครคอนโทรลเลอร์กับไมโครโปรเซสเซอร์นั้น ก็คือ ไมโครโปรเซสเซอร์ (Microprocessor) จะเป็นอุปกรณ์ที่ทำหน้าที่ในการประมวลผลเท่านั้น ซึ่งตัวมันจะต้องอาศัยอุปกรณ์อื่นๆ เข้ามาช่วยด้วย เช่น หน่วยความจำ (Memory), ส่วนเชื่อมต่ออุปกรณ์ภายนอก (Interface Unit) เป็นต้น จึงจะทำงานได้อย่างสมบูรณ์ ตัวอย่างก็คือ เครื่องคอมพิวเตอร์ที่เราใช้กันอยู่ในปัจจุบันนั่นเอง ส่วนไอซีไมโครคอนโทรลเลอร์นั้นจะมีลักษณะคล้ายกับไมโครโปรเซสเซอร์ก็คือ จะมีส่วนประมวลผลเหมือนกัน แต่จะเพิ่มส่วนของหน่วยความจำและส่วนเชื่อมต่ออุปกรณ์เข้ามาไว้ในตัวมันด้วย ดังนั้นไมโครคอนโทรลเลอร์จึงเหมาะที่จะนำมาใช้ในงานควบคุมที่ไม่ต้องการความซับซ้อนมากนัก

โครงสร้างภายในไอซีไมโครคอนโทรลเลอร์นั้น สามารถแบ่งออกเป็น 3 ส่วนใหญ่ๆ ด้วยกัน ดังนี้

1. หน่วยประมวลผลกลาง (Central Processor Unit : CPU) เป็นส่วนที่เป็นเหมือนกับสมองของตัวไอซีไมโครคอนโทรลเลอร์เลยทีเดียว โดยหน้าที่ของมัน ก็คือ จะทำการประมวลผลข้อมูลต่างๆ ที่เข้ามา แล้วทำการส่งสัญญาณออกไปยังส่วนต่างๆ เพื่อควบคุมการทำงานให้ตรงตามข้อมูลนั้นๆ
2. หน่วยความจำ (Memory) เป็นส่วนที่ใช้ในการเก็บข้อมูลต่างๆ เอาไว้ เพื่อรอการส่งให้กับหน่วยประมวลผลกลางทำการประมวลผลอีกทีหนึ่ง โดยภายในไอซีไมโครคอนโทรลเลอร์นี้ จะมีหน่วยความจำอยู่ 3 แบบ คือ หน่วยความจำโปรแกรม (Program Memory), หน่วยความจำข้อมูลแรม (RAM data Memory) และ หน่วยความจำข้อมูลอีอีพรอม (EEPROM data memory)
  - หน่วยความจำโปรแกรม จะเป็นหน่วยความจำที่ใช้ในการเก็บรักษาคำสั่งควบคุมต่างๆ ที่ผู้พัฒนาโปรแกรมเขียนขึ้น โดยหน่วยประมวลผลกลางจะทำการติดต่อกับส่วนนี้ เพื่อดึงไปประมวลผลและส่งคำสั่งไปควบคุมส่วนอื่นๆ ต่อไป โดยหน่วยความจำโปรแกรมนี้จะคงอยู่ ถึงแม้ว่าจะไม่มีไฟเลี้ยงให้กับตัวไอซีก็ตาม
  - หน่วยความจำข้อมูลแรม จะเป็นหน่วยความจำที่ไอซีไมโครคอนโทรลเลอร์ทุกตัวต้องมีเลขที่เดียว เพราะจะใช้ในการเก็บข้อมูลเกี่ยวกับการประมวลผลทั้งในระหว่างและหลังการประมวลผล แต่ข้อมูลต่างๆ เหล่านี้จะหายไป เมื่อไม่มีไฟเลี้ยงให้กับไอซี
  - หน่วยความจำข้อมูลอีอีพรอม จะเป็นหน่วยความจำพิเศษ ซึ่งไอซีไมโครคอนโทรลเลอร์บางตัวมีและบางตัวไม่มี มีไว้สำหรับเก็บข้อมูลที่ต้องการเก็บรักษาเป็นพิเศษ เมื่อไม่มีไฟเลี้ยงตัวไอซีข้อมูลเหล่านี้ก็จะยังคงอยู่ จนกว่าจะมีการเขียนทับลงไปใหม่
3. ส่วนเชื่อมต่ออุปกรณ์ภายนอก (Interface Unit) จะเป็นส่วนที่ทำหน้าที่ในการติดต่อกับอุปกรณ์ภายนอก โดยการสั่งงานมาจากหน่วยประมวลผลกลางอีกทีหนึ่ง ซึ่งในส่วนนี้เราสามารถที่จะกำหนดให้เป็นแบบอินพุต (รับข้อมูล) หรือแบบเอาต์พุต (ส่งข้อมูล) ก็ได้ ตามการเขียนโปรแกรมของผู้พัฒนาโปรแกรม



รูปแสดงลักษณะโครงสร้างภายในของไมโครคอนโทรลเลอร์

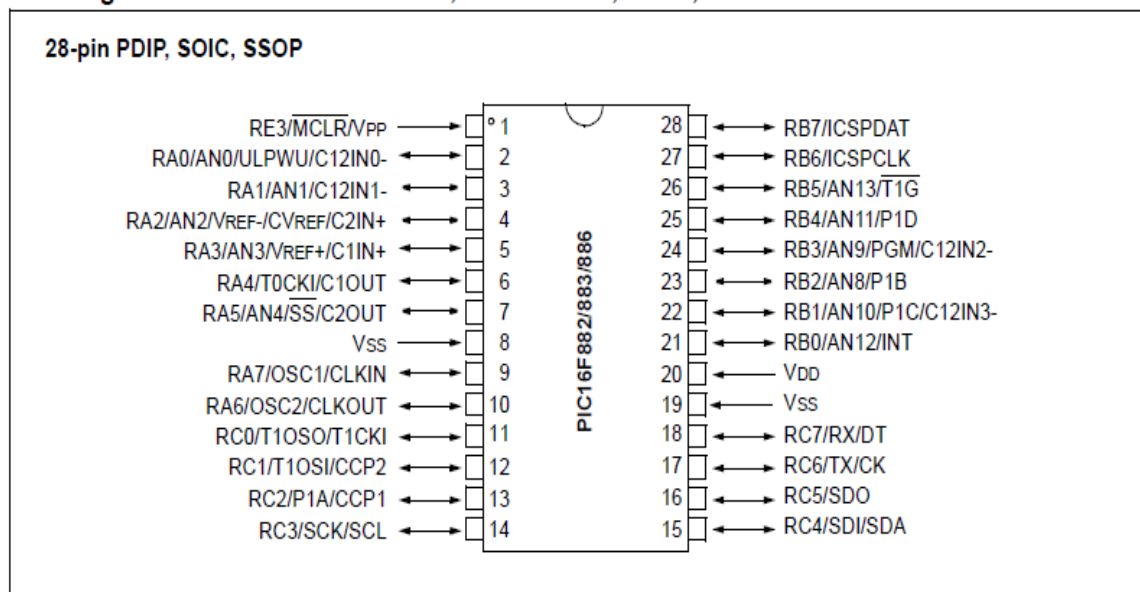
### PIC 16F886 ที่ใช้เป็น CPU ในตัวหุ่นยนต์

ไอซีไมโครคอนโทรลเลอร์ PIC 16F886 ตัวนี้ เป็นไอซีที่ทางบริษัท ไมโครชิพ (Microchip) พัฒนาขึ้นมา เพื่อรองรับการใช้งานที่หลากหลายตามการเขียนโปรแกรมของผู้พัฒนา โดยที่ราคาไม่แพง มีชุดคำสั่งให้ใช้งานมากมาย ทำให้เหมาะแก่การศึกษา

คุณสมบัติของไอซีไมโครคอนโทรลเลอร์ PIC 16F886

- ไอซีใช้ไฟเลี้ยง ตั้งแต่ 2 ถึง 5.5 โวลต์
- ความเร็วในการทำงานสูงสุด 20MHz
- ขนาดของหน่วยความจำโปรแกรม 7 K(words)
- ขนาดของหน่วยความจำข้อมูลแรม 368 bytes
- ขนาดของหน่วยความจำข้อมูลอีพีรอม 256 bytes
- มีจำนวนขาเชื่อมต่อกับอุปกรณ์ภายนอกได้ 24 ช่อง
- สามารถลบและเขียนคำสั่งลงในหน่วยความจำโปรแกรมได้ประมาณ 100,000 ครั้ง
- สามารถลบและเขียนคำสั่งลงในหน่วยความจำอีพีรอมได้ประมาณ 1,000,000 ครั้ง

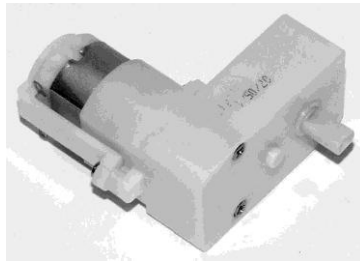
### Pin Diagrams – PIC16F882/883/886, 28-Pin PDIP, SOIC, SSOP



รูปแสดงตำแหน่งและหน้าที่แต่ละขาของไอซี PIC 16F886

## 2. มอเตอร์เกียร์ (Gear Motor)

มอเตอร์เกียร์ถือว่าเป็นชุดส่งกำลังที่สำคัญอีกส่วนหนึ่งของหุ่นยนต์เลยทีเดียว โดยมอเตอร์เกียร์จะประกอบไปด้วยมอเตอร์และชุดเฟือง สาเหตุที่จำเป็นต้องมีมอเตอร์เกียร์ก็เนื่องมาจากมอเตอร์ที่มีจำนวนรอบการหมุนที่สูงเกินไป ทำให้ไม่เหมาะสมกับการนำมาใช้งานโดยตรง ดังนั้นจึงมีความจำเป็นที่จะต้องมีชุดเฟืองเกียร์ เพื่อลดจำนวนรอบในการหมุนลง และนอกจากจะลดจำนวนรอบในการหมุนแล้วยังเป็นตัวช่วยในการทำให้เกิดแรงบิดขึ้น ทำให้หุ่นยนต์สามารถขับเคลื่อนตัวไปได้ ซึ่งในการบอกคุณสมบัติของมอเตอร์เกียร์นั้นจะมีการบอกอัตราทดของเฟืองเกียร์ด้วย เช่น 1:120 นั้นหมายความว่า เมื่อมอเตอร์หมุนไป 120 รอบ ตัวเฟืองเกียร์อันสุดท้าย จะหมุนเพียง 1 รอบ เป็นต้น และเป็นอัตราทดที่เราใช้ในหุ่นของตัวนี้ ถ้าต้องการให้เร็ว หรือ ช้า กว่านี้ก็สามารถเปลี่ยนได้

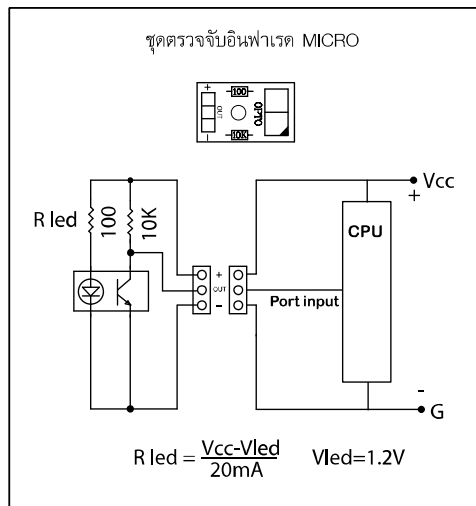


รูปลักษณะของมอเตอร์เกียร์ขนาดเล็ก แบบตัว L

## ตัวหุ่นยนต์ประกอบด้วยอะไรบ้าง

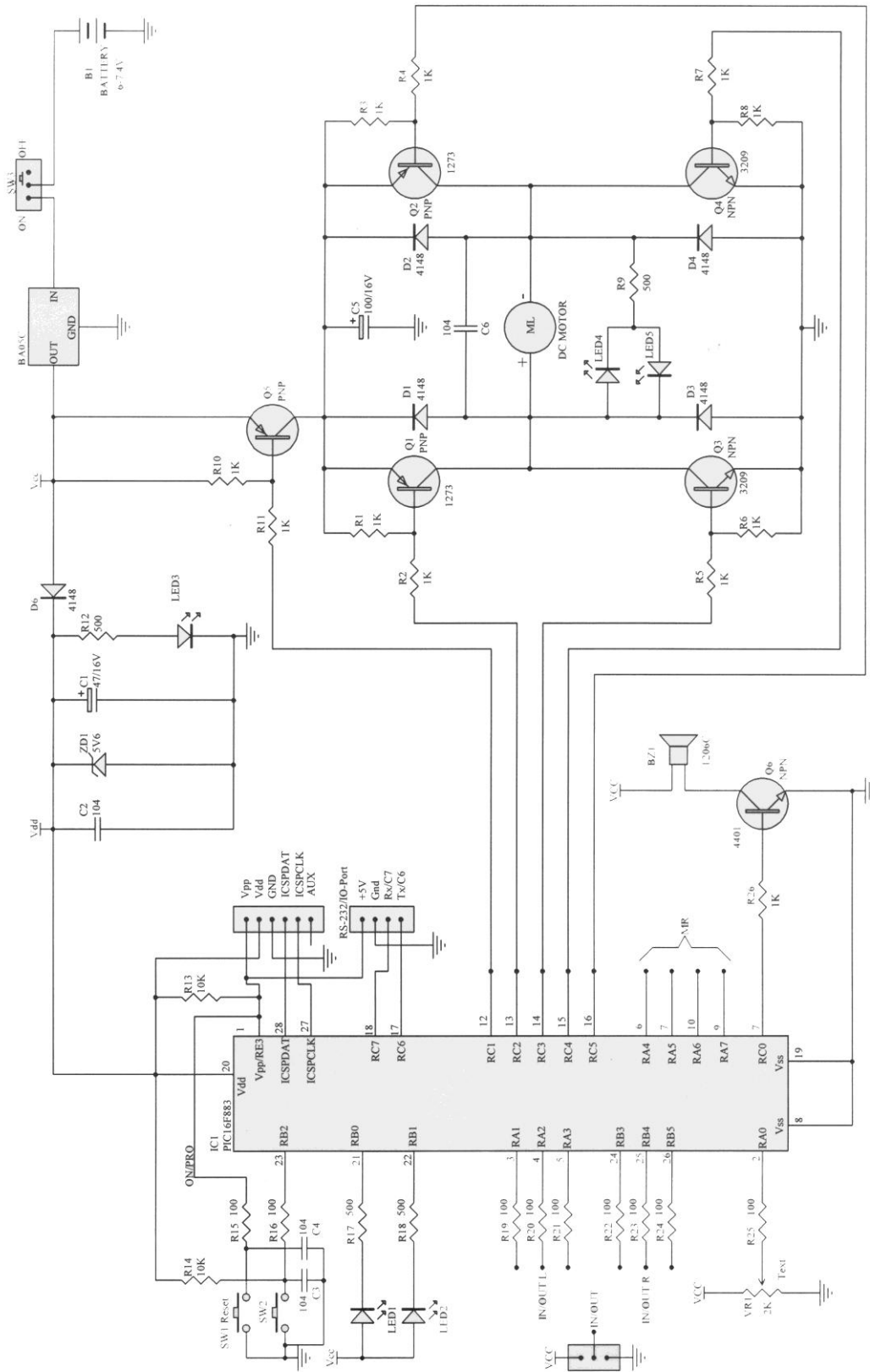
### 1. วงจรอิเล็กทรอนิกส์ (Electronic Circuit)

วงจรอิเล็กทรอนิกส์ที่ใช้ในตัวหุ่นยนต์รุ่นนี้จะอยู่ด้วยกัน 2 บอร์ดใหญ่ๆ คือ บอร์ดเซ็นเซอร์ (Sensor Board), บอร์ดควบคุมและไดร์มอเตอร์ (Control Board and Motor Driver)



วงจรบอร์ดเซ็นเซอร์ (Sensor Board)

บอร์ดเซ็นเซอร์ที่ใช้กับบอร์ดควบคุมชุดนี้เป็นบอร์ดออปโตทรานซิสเตอร์ หลักการทำงานของบอร์ดออปโตทรานซิสเตอร์ คือ ตัว LED INF จะทำการส่งแสงอินฟราเรดตลอดเวลา สำหรับตัวโฟโต้ทรานซิสเตอร์นั้น เมื่อไม่ได้รับแสงอินฟราเรด ทรานซิสเตอร์จะไม่ทำงาน ทำให้แรงดันที่ไหลผ่าน ตัวต้านทานมาที่ขา C มีค่าสูงเท่ากับค่าแรงดัน +5Vcc ทำให้ที่ตำแหน่ง "OUT" จะมีไฟประมาณ +5Vcc โวลต์ แต่เมื่อไรก็ตามที่ตัวโฟโต้ทรานซิสเตอร์ได้รับแสงอินฟราเรด ทรานซิสเตอร์จะทำงาน ที่ตำแหน่ง "OUT" จะมีค่าแรงดันต่ำลง เนื่องมาจากไฟ 5 โวลต์ ที่ไหลผ่านตัวต้านทานจะถูกต่อให้ไหลลงกราวด์ทันที



รูปแสดงวงจรบอร์ดควบคุม (Control Board) และ ไดรฟ์ มอเตอร์

ลักษณะการทำงานของบอร์ดควบคุมนี้จะขึ้นอยู่กับ IC1 16F886 ซึ่งถือว่าเป็นหัวใจของวงจรเลยก็ว่าได้ ซึ่ง IC1 นี้เป็นไอซีไมโครคอนโทรลเลอร์ที่ใช้ในการเก็บชุดคำสั่งต่างๆ ที่เราเขียนขึ้นมา โดยอาศัยบอร์ดเซ็นเซอร์เพื่อเป็นตัวรับรู้ว่ามีภาระตุ่นให้ทำงาน จากนั้นก็จะไปสั่งให้ชุดวงจรไดร์ฟมอเตอร์ส่งไฟไปให้กับมอเตอร์เพื่อทำการหมุนต่อไป สำหรับ IC2 เมื่อได้รับไฟจากแบตเตอรี่ประมาณ 6 โวลต์ ตัว IC2 จะทำการลดไฟที่เหลือเพียง 5 โวลต์ เพื่อให้เหมาะสมกับการจ่ายไฟให้กับ IC1

#### ชุดวงจรไดร์ฟมอเตอร์ (Motor Driver Board)

จะมีวงจรที่เหมือนกันอยู่ 2 ชุด ดังนั้นจะขออธิบายเพียงชุดเดียว เมื่อ IC1 ที่ขา 12 จะทำหน้าที่เป็นตัวจ่ายไฟให้กับชุดวงจรไดร์ฟมอเตอร์ทำงานร่วมกับ TR Q5 และที่ขานี้ยังกำหนดให้ทำงานแบบ PWM ได้อีกด้วย ต่อมาจะเป็นชุดการทำงานของมอเตอร์จะใช้ขา 13-16 ในการควบคุมมอเตอร์ด้านซ้าย ส่วนขา 6-7, 9-10 จะควบคุมด้านขวา คำสั่งการทำงานดูตารางด้านล่าง

ตารางควบคุมการทำงานของมอเตอร์

|            | Control<br>PWM | มอเตอร์ ขวา |    |    |    | มอเตอร์ ซ้าย |    |    |    |
|------------|----------------|-------------|----|----|----|--------------|----|----|----|
|            |                | 13          | 14 | 15 | 16 | 6            | 7  | 10 | 9  |
| ขา IC      | 12             | 13          | 14 | 15 | 16 | 6            | 7  | 10 | 9  |
| PORT       | C1             | C2          | C3 | C4 | C5 | A4           | A5 | A6 | A7 |
| ไม่ทำงาน   | H              | X           | X  | X  | X  | X            | X  | X  | X  |
| เดินหน้า   | L              | H           | H  | L  | L  | L            | L  | H  | H  |
| ถอยหลัง    | L              | L           | L  | H  | H  | H            | H  | L  | L  |
| เลี้ยวซ้าย | L              | H           | H  | L  | L  | H            | H  | H  | H  |
| เลี้ยวขวา  | L              | H           | H  | H  | H  | L            | L  | H  | H  |
| BREAK      | L              | H           | H  | H  | H  | H            | H  | H  | H  |

-บนบอร์ดมี PORT ให้ใช้งานดังนี้

1. INPUT หรือ OUTPUT 6 PORT คือขา A1-A3 และ B3-B5
2. ขา PORT ควบคุม BUZZER ขา C0
3. ขา PORT ควบคุม LED ขา B0 และ B1
4. ขา PORT ADC มีเก็อกมาเป็นตัวปรับค่า คือ A0
5. ขา PORT RS232 ขา C6 และ C7 (Rx , Tx) หรือจะกำหนดให้เป็น INPUT หรือ OUTPUT ก็ได้
6. มีสวิตช์ RESET เมื่อกดจะหยุดการทำงานของ CPU และเมื่อปล่อยก็จะกลับมาเริ่มต้นทำงานใหม่
7. มีสวิตช์ OK สำหรับไว้เขียนคำสั่งให้รับค่าการทำงานตามการต้องการ เช่น กดเพื่อให้หุ่นทำงาน หรือ กดเพื่อให้หยุดชั่วขณะ
8. ขา PORT โปรแกรม ICSP
  - Vpp เป็นขาไฟที่มาจากเครื่อง โปรแกรม
  - Vdd เป็นขาไฟบวกของวงจรหุ่นยนต์
  - GND เป็นขาไฟลบของวงจรหุ่นยนต์

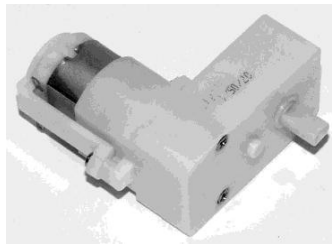
- ICSPDAT / PORT RB7 เป็นขาโปรแกรมข้อมูล หรือกำหนดให้เป็นขา I/O ก็ได้
- ICSPDAT / PORT RB6 เป็นขาโปรแกรมสัญญาณนาฬิกา หรือกำหนดให้เป็นขา I/O ก็ได้
- AUX ไม่ได้ใช้งาน

## 2. ส่วนแมคคานิกส์ (Mechanic Part)

ในส่วนของแมคคานิกส์นั้นจะประกอบไปด้วยหลายส่วนด้วยกัน ได้แก่ ตัวบอดี้หุ่นยนต์, มอเตอร์เกียร์, ล้อใหญ่และล้อหลัง



รูปบอดี้ของหุ่นยนต์



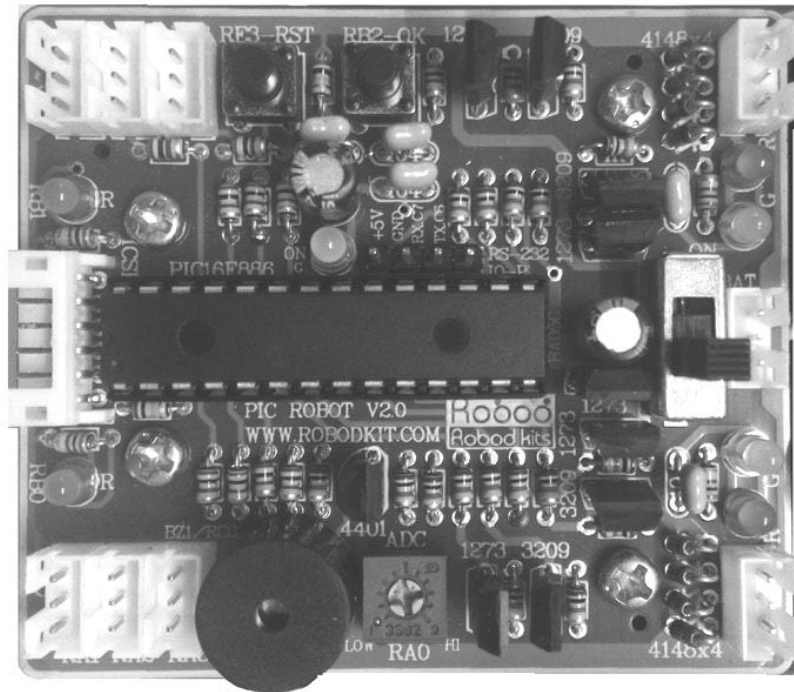
รูปมอเตอร์เกียร์ แบบตัว L



รูปล้อใหญ่

รูปล้อหลัง





รูปแสดงบอร์ดวงจรหุ่นยนต์ PIC ROBOT V2.0

ในชุดประกอบไปด้วย

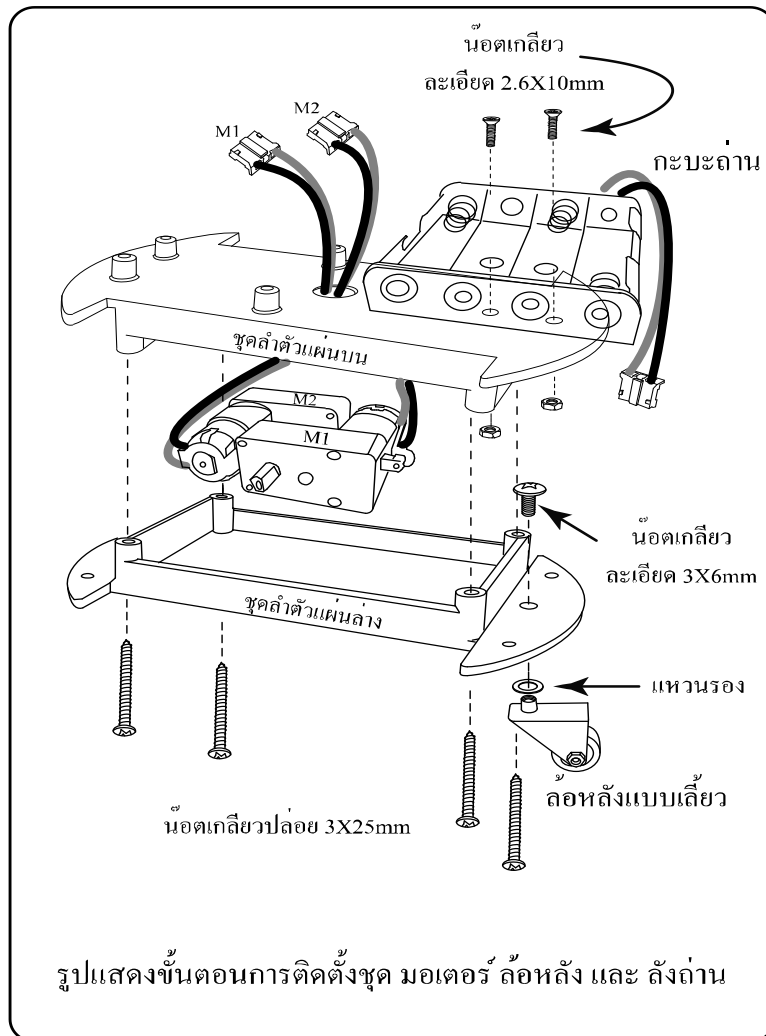
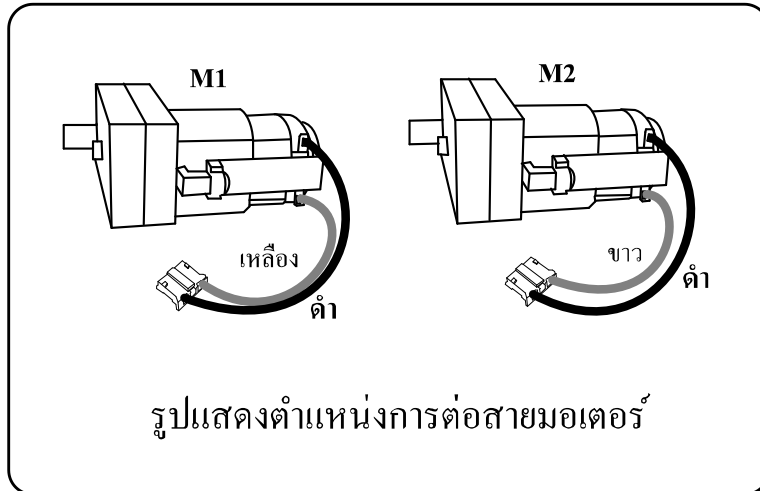
- 1.บอร์ดวงจรอิเล็กทรอนิกส์ 1 ชุด
- 2.บอร์ด เซ็นเซอร์ ออปโตทรานซิสเตอร์ 2 ชุด
- 3.บอร์ดหุ่นยนต์ 1 ชุด
- 4.ล้อข้าง 2 ล้อ
- 5.ชุดล้อหลัง 1 ชุด
- 6.มอเตอร์เกียร์ 1:120 2 ตัว
- 7.ลังถ่าน 4 ก้อน ขนาด AA1 อัน

#### เครื่องมือที่จำเป็นในการประกอบหุ่นยนต์ (ไม่มีในชุด)

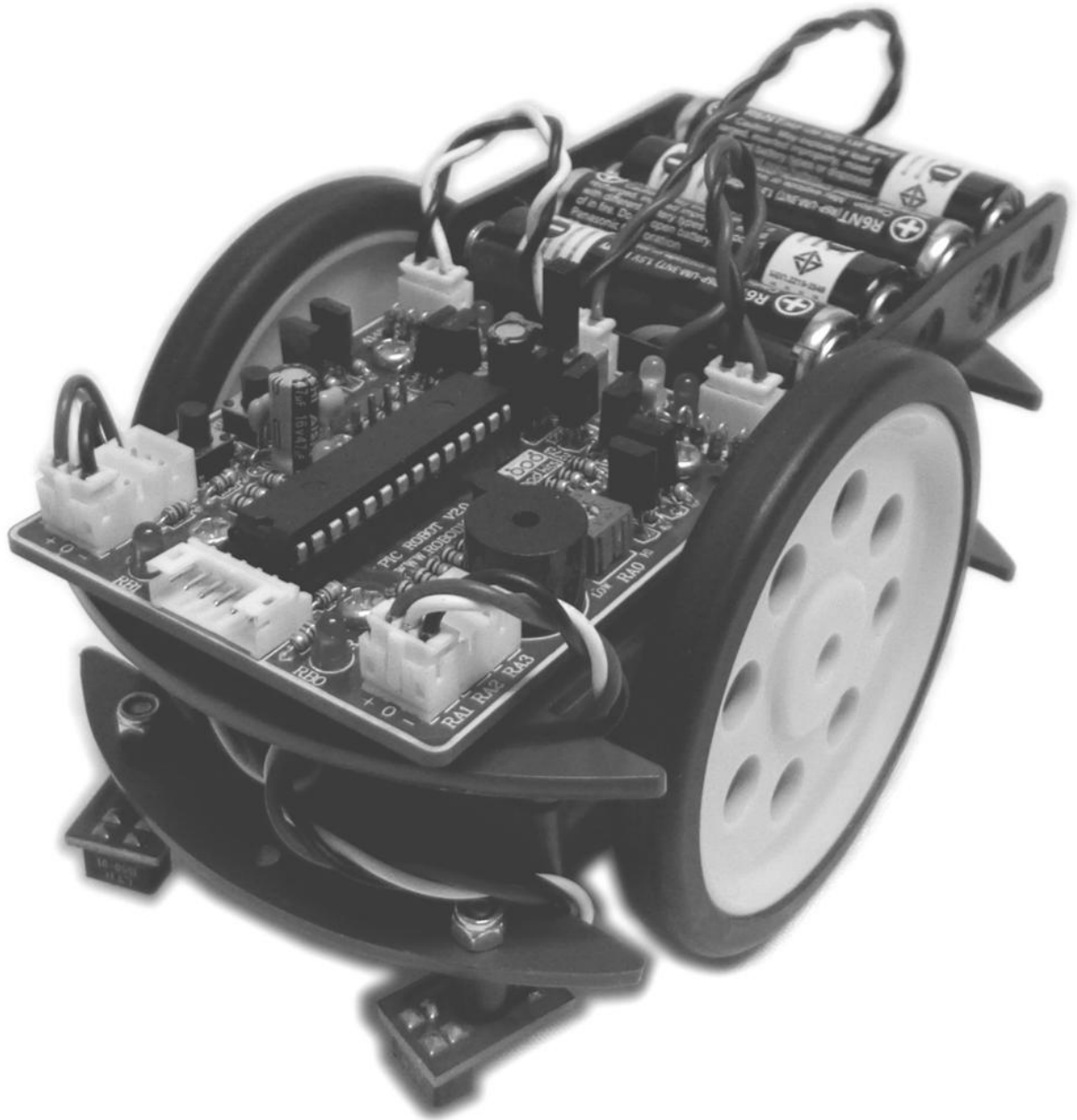
- ไขควงแฉกขนาดเล็ก และใหญ่ อย่างละ 1 ค้าม
- คัตเตอร์ 1 ค้าม
- คีมตัด 1 อัน
- คีมจับ 1 อัน

#### ขั้นตอนการประกอบ

ในการประกอบหุ่นยนต์นี้จะแบ่งออกเป็น 2 ส่วนด้วยกัน คือ ส่วนวงจรอิเล็กทรอนิกส์และส่วนแมคคานิกส์







หุ่นยนต์ AP114 PIC ROBOT V2.0

### Software ที่ใช้กับตัวหุ่นยนต์

ในส่วนของ Software จะมีอยู่ด้วยกัน 2 โปรแกรม คือ โปรแกรมกำหนดการทำงานของหุ่นยนต์ และ โปรแกรมบันทึกข้อมูลลงในไมโครคอนโทรลเลอร์

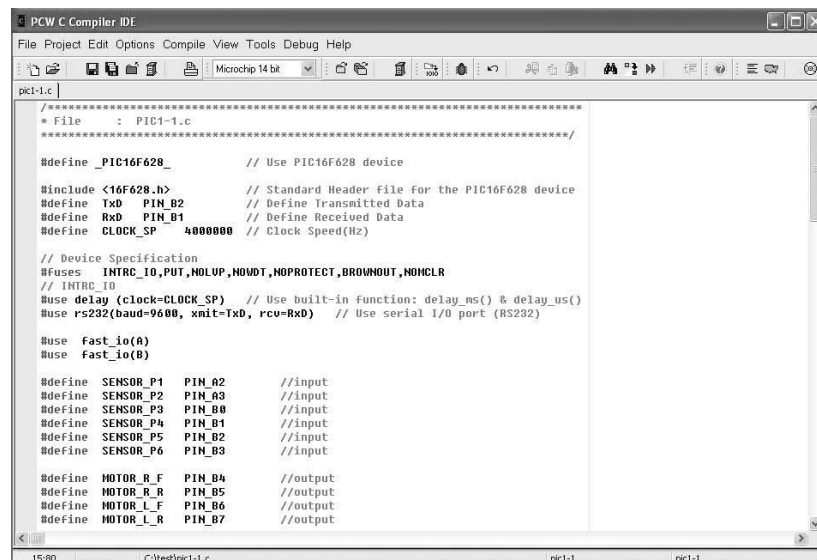
#### 1. โปรแกรมกำหนดการทำงานของหุ่นยนต์

โปรแกรมที่ใช้ในการเขียน เพื่อกำหนดการทำงานของหุ่นยนต์นั้น เราจะใช้โปรแกรมที่เขียนด้วยภาษาซี เนื่องจากในการเขียนโปรแกรมนั้นสามารถเข้าใจได้ง่ายและไม่มีความยุ่งยากสำหรับผู้เรียน ในกรณีที่ผู้เขียนจะใช้ภาษา แอสเซมบลี Assembly ในการเขียนก็สามารถทำได้ตามความถนัดของผู้ใช้งาน ในที่นี้เราจะใช้โปรแกรมที่มีชื่อว่า CCS C compiler ของบริษัท Custom Computer Services ประเทศสหรัฐอเมริกา ซึ่งทางบริษัทได้เปิดให้ทำการดาวน์โหลดฟรี

ใช้ได้ฟรีที่เว็บไซต์ [www.ccsinfo.com](http://www.ccsinfo.com) สำหรับโปรแกรมเวอร์ชันทดลองใช้นี้สามารถใช้งานได้ 30 วัน และสามารถเขียนโปรแกรมได้สูงสุด 2 กิโลไบต์

สำหรับคุณสมบัติขั้นต่ำของเครื่องคอมพิวเตอร์ที่แนะนำให้ใช้กับโปรแกรม CCS C compiler มีดังนี้

- ซีพียู ขนาด 200 MHz ขึ้นไป Intel Pentium หรือ AMD K-6
- หน่วยความจำแรม 64MB
- เนื้อที่ว่างบนฮาร์ดดิสก์ อย่างน้อย 50MB
- CD-ROM
- พอร์ตอนุกรม (Com Port) จำนวน 1 พอร์ต



```

PCW C Compiler IDE
File Project Edit Options Compile View Tools Debug Help
Microchip 14 bit
pic1-1.c
/*****
* File : PIC1-1.c
*****/
#define PIC16F628_ // Use PIC16F628 device
#include <16F628.h> // Standard Header file for the PIC16F628 device
#define TxD PIN_B2 // Define Transmitted Data
#define RxD PIN_B1 // Define Received Data
#define CLOCK_SP 4000000 // Clock Speed(Hz)

// Device Specification
#fuses INTRC_IO,PUT,HOLDUP,NOVDDT,NOVPROTECT,BROWNOUT,NOVCLR
// INTRC_IO
#use delay (clock=CLOCK_SP) // Use built-in function: delay_ms() & delay_us()
#use rs232(baud=9600, xmit=TxD, rcv=RxD) // Use serial I/O port (RS232)

#use Fast_io(A)
#use Fast_io(B)

#define SENSOR_P1 PIN_A2 //input
#define SENSOR_P2 PIN_A3 //input
#define SENSOR_P3 PIN_B0 //input
#define SENSOR_P4 PIN_B1 //input
#define SENSOR_P5 PIN_B2 //input
#define SENSOR_P6 PIN_B3 //input

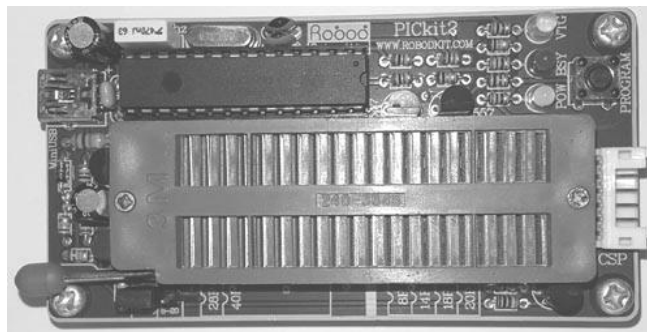
#define MOTOR_R_F PIN_B4 //output
#define MOTOR_R_R PIN_B5 //output
#define MOTOR_L_F PIN_B6 //output
#define MOTOR_L_R PIN_B7 //output

```

รูปหน้าตาของโปรแกรม CCS C compiler

## 2. เครื่องโปรแกรมบันทึกข้อมูลลงในไมโครคอนโทรลเลอร์

เครื่องโปรแกรม PIC นี้ไม่ได้กำหนดให้ใช้ตัวไหนเอาตัวที่ผู้เขียนมีอยู่แล้วก็ได้เพียงแค่มียี่ห้อต่อสายโปรแกรมแบบ ICSP ก็สามารถใช้งานได้ครับ หรือถ้ายังไม่มี ขอแนะนำเครื่องโปรแกรม PIC KIT2V2 ของทาง ROBODKIT ราคา 990.-



รูปแสดงเครื่องโปรแกรม PIC KIT2V2

### ขั้นตอนการติดตั้งโปรแกรม CCS C compiler

ในการลงโปรแกรม CCS C compiler นั้น จะเหมือนกับการลงโปรแกรมอื่นๆ ทั่วไป ซึ่งขั้นตอนที่ง่ายมาก ดังนี้

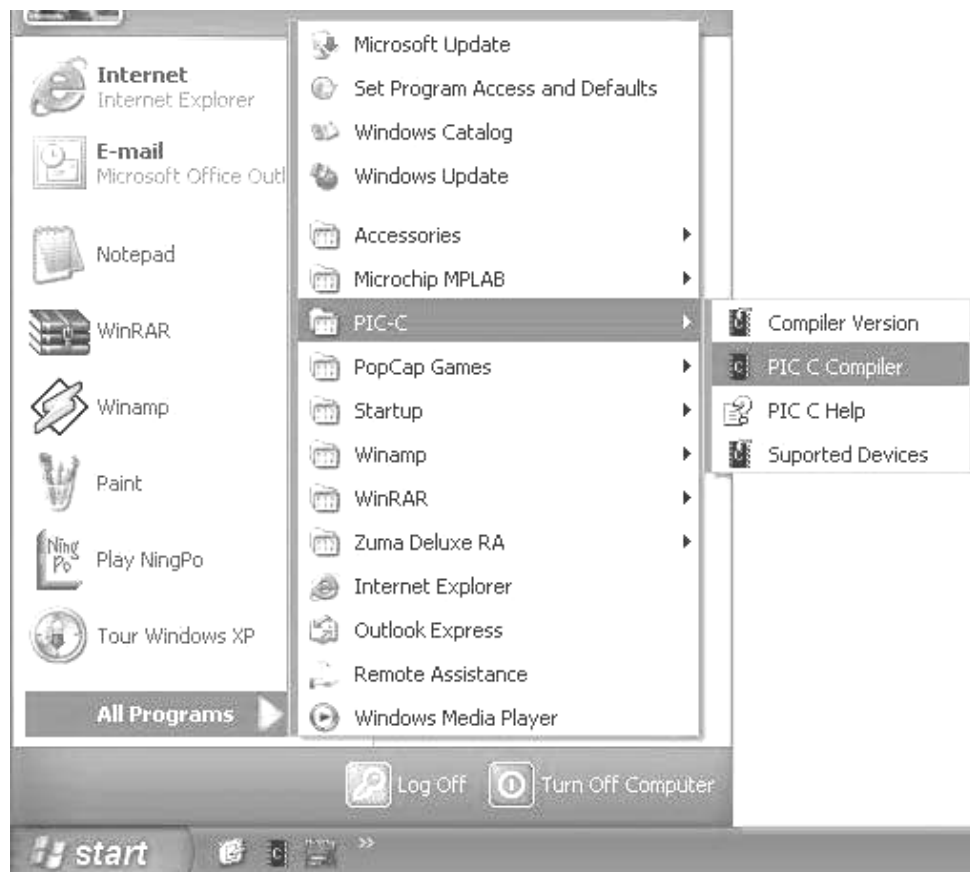
1. ทำการดับเบิลคลิกไฟล์ pcwhupd.exe จะเป็นการเริ่มเข้าสู่ขั้นตอนการติดตั้งโปรแกรม
2. เมื่อเริ่มเข้าสู่กระบวนการติดตั้งโปรแกรม ให้ทำการคลิกปุ่ม Next โปรแกรมจะแสดงรายละเอียดของโปรแกรมขึ้นมา จากนั้นก็ให้ทำการคลิกปุ่ม Next จะแสดงหน้าต่างใหม่ขึ้นมา ซึ่งในหน้าต่างนี้จะถามถึงตำแหน่งที่เราต้องการติดตั้ง โดยปกติตัวติดตั้งจะกำหนดมาให้อยู่แล้ว
3. เมื่อเลือกตำแหน่งที่ต้องการติดตั้งโปรแกรมได้แล้ว ก็ให้ทำการกดปุ่ม Next โปรแกรมก็จะทำการติดตั้งจนกระทั่งเสร็จ ก็ให้ทำการกดปุ่ม Finish ก็เป็นอันเสร็จ

### เริ่มต้นใช้งานโปรแกรม CCS C compiler

#### การเปิดโปรแกรม

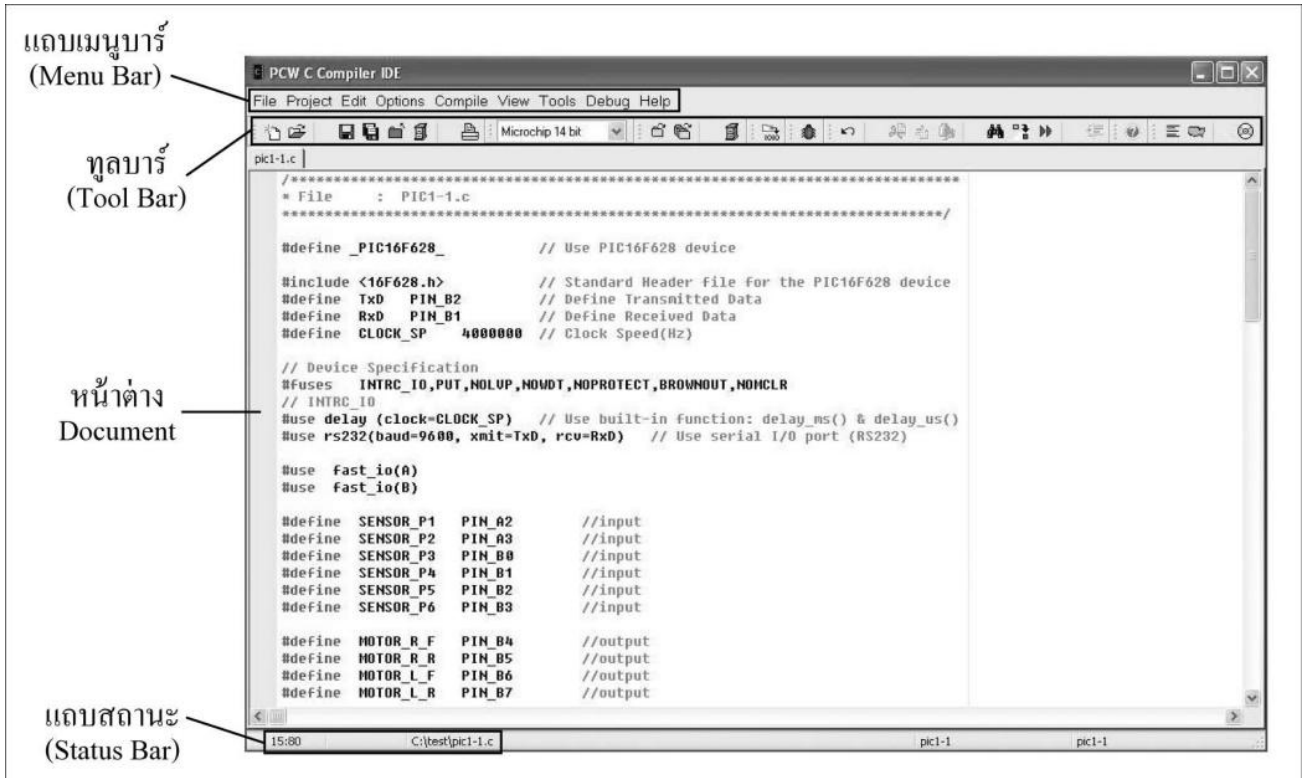
การเปิดโปรแกรม CCS C compiler ทำได้โดย

- 1.คลิกที่ปุ่ม START ของ Windows
- 2.เลื่อนไปที่ All Programs
- 3.เลื่อนไฮไลท์เมาส์ไปที่กลุ่มไอคอน PIC-C
- 4.คลิกที่ไอคอน PIC C Compiler



รูปแสดงลำดับขั้นการเปิดโปรแกรม CCS C compiler

## ส่วนประกอบของหน้าจอโปรแกรม



## 1. หน้าต่าง Document

หน้าต่าง Document คือส่วนที่ใช้สำหรับใส่คำสั่งที่เราต้องการเขียน วิธีใช้งานหน้าต่างนี้จะคล้ายกับการใช้งานโปรแกรมเวิร์ดโปรเซสเซอร์ทั่วไป เช่น การพิมพ์ข้อความ, การคัดลอก, การลบข้อความ เป็นต้น

## 2. แถบเมนูบาร์ (Menu Bar)

คือ ส่วนที่เก็บคำสั่งสำหรับการทำงานต่างๆ เอาไว้ ซึ่งบางคำสั่งสามารถเรียกใช้จากแถบเครื่องมือได้ แต่บางคำสั่งจะมีลักษณะเฉพาะในแถบเมื่อนั้น เราสามารถเปิดเมนูต่างๆ ขึ้นมาใช้งาน โดยการคลิกที่ชื่อเมนูและเลื่อนเมาส์ไปคลิกยังคำสั่งที่ต้องการ หากคำสั่งนั้นมีเมนูย่อย (สังเกตจากลูกศรที่อยู่ด้านขวาของเมื่อนั้น) ให้เลื่อนเมาส์ไปที่คำสั่งนั้น จะปรากฏกรอบเมนูย่อยแสดงขึ้นมา จากนั้นก็คลิกเลือกคำสั่งที่ต้องการ

## 3. แถบเครื่องมือ (Toolbar)

คือแถบเครื่องมือที่เก็บปุ่มคำสั่งต่างๆ ที่จำเป็นต้องใช้งานบ่อยๆ ไว้ เมื่อนำเมาส์ไปชี้ตามปุ่มต่างๆ เหล่านั้น จะมีข้อความขึ้นมาเพื่ออธิบายถึงหน้าที่ของปุ่มต่างเหล่านั้น เช่น ปุ่ม Create New File, ปุ่ม Open File, ปุ่ม Save File เป็นต้น

## 4. แถบสถานะ (Status Bar)

คือแถบแสดงสถานะที่อยู่ด้านล่างของหน้าต่าง Document ซึ่งประกอบด้วย 2 ส่วน คือ ทางด้านซ้ายจะเป็นตัวบอกตำแหน่งของเคอร์เซอร์ว่าอยู่บรรทัดและตัวอักษรที่เท่าไร เช่น 32:5 หมายความว่า ตอนนี้เคอร์เซอร์อยู่บรรทัดที่ 32 ตัวอักษรที่ 5 เป็นต้น และทางด้านขวา จะบอกสถานะที่ในการเก็บไฟล์ที่เรากำลังเขียนอยู่ว่าเก็บอยู่ที่ใด เช่น c:\project\test.c เป็นต้น

ในกรณีที่ต้องการศึกษาเกี่ยวกับ CCS เรามีหนังสือที่จะใช้ในการศึกษา คือ All about CCS C คอมไพเลอร์ และ PIC Works Examples C Source Code ของ APPSOFTTECH จำหน่ายครับ

## พื้นฐานการเขียนโปรแกรมภาษาซีและ CCS C Compiler

### 1. โครงสร้างในการเขียนภาษาซี

โครงสร้างของภาษาซีนั้น จะประกอบด้วยการทำงานอยู่ 2 ส่วน คือ ส่วนของ โปรแกรมหลัก (Main Code Programming) และ ส่วนของ โปรแกรมย่อยหรือฟังก์ชัน (Function Code Programming)

การทำงานของโปรแกรมโดยหลักๆ แล้วจะอยู่ในส่วนของโปรแกรมหลัก ซึ่งอาจจะมีการเรียกใช้งานในส่วนของโปรแกรมย่อยบ้าง โดยโปรแกรมย่อยนี้จะเป็นโปรแกรมที่ถูกเรียกใช้งานบ่อยๆ เช่น โปรแกรมเกี่ยวกับเวลา, เงื่อนไขที่กระทำเหมือนกัน เป็นต้น ถ้าเราไม่ทำการแยกโปรแกรมต่างๆ เหล่านี้ออกจากโปรแกรมหลักแล้ว อาจจะทำให้โปรแกรมรวมทั้งหมดมีขนาดใหญ่เกินกว่าที่จะสามารถนำไปใช้งานได้ ดังนั้นโปรแกรมย่อยจึงมีประโยชน์อย่างมากในการลดขนาดของโปรแกรมรวมทั้งหมด ให้ลดลงนั่นเอง

ลักษณะในการเขียนโปรแกรมหลักนี้ เราจะเริ่มต้นด้วย main ตามด้วยปีกกาใหญ่ { } ส่วนโปรแกรมที่เขียนจะเขียนอยู่ภายในปีกกาใหญ่นั้นจะเป็นตัวกำหนดการทำงานของโปรแกรม เช่น

ตัวอย่างโปรแกรม

```
#include <stdio.h> // Preprocessor directives (header file)
void main(void)
{
    printf ( "\nHello World\n" ); //แสดงผลลัพธ์ด้วยฟังก์ชันมาตรฐาน printf
}
```

คำอธิบายโปรแกรม

โดยโปรแกรมที่ยกตัวอย่างนี้ เมื่อทำการรันโปรแกรม ตัวโปรแกรมจะทำการแสดงข้อความ "Hello World" บนหน้าจอคอมพิวเตอร์

ส่วนโปรแกรมย่อยนั้นจะมีลักษณะการเขียนที่เหมือนกับโปรแกรมหลัก แต่ชื่อของโปรแกรมเราสามารถเปลี่ยนได้ตามความต้องการ เพื่อสะดวกในการใช้งาน เช่น Delay\_time, Go\_left เป็นต้น ตัวอย่างการเขียนจะเป็นดังนี้

ตัวอย่างโปรแกรม

```
#include <stdio.h> // Preprocessor directives (header file)
// ฟังก์ชัน โปรแกรมหน่วงเวลา
Void time_delay(void)
{
    Delay_ms(1000); //หน่วงเวลาเป็นเวลา 1 วินาที
}

// Main โปรแกรม
void main(void)
{
    while(TRUE)
    {
        printf ( "\nHello \n" ); //แสดงผลลัพธ์ด้วยฟังก์ชันมาตรฐาน printf
        time_delay(); //เรียกใช้งาน โปรแกรมย่อย time_delay
        printf ("\nWorld\n" ); //แสดงผลลัพธ์ด้วยฟังก์ชันมาตรฐาน printf
        time_delay(); //เรียกใช้งาน โปรแกรมย่อย time_delay
    }
}
```

ตัวอย่างโปรแกรม

จากโปรแกรมดังกล่าว เมื่อทำการรันโปรแกรม ที่หน้าจอคอมพิวเตอร์จะทำการแสดงข้อความ “Hello” ก่อน แล้วทิ้งช่วงประมาณ 1 วินาที แล้วจึงทำการแสดงข้อความ “World” แล้วทิ้งช่วงประมาณ 1 วินาที แล้วก็กลับไปแสดงข้อความ “Hello” ใหม่ จะเป็นอย่างไรไปเรื่อย เนื่องมาจากคำสั่ง while (TRUE)

จะสังเกตเห็นว่า ในการเขียนโปรแกรมนั้นจะเขียนโปรแกรมหลักไว้ด้านล่างและเขียนโปรแกรมย่อยไว้ข้างบน สาเหตุที่เป็นอย่างนี้ เนื่องมาจากภายในโปรแกรมจะทำการไล่ลำดับการทำงานจากบนลงล่าง แต่เมื่อมีการเรียกใช้งานโปรแกรมย่อยจะเรียกจากข้างบนนั่นเอง

## 2.การกำหนดค่าของตัวแปรต่างๆ (Declaration) ในโปรแกรมภาษาซีที่ใช้กับ CCS C Compiler

### - ชนิดของข้อมูล(data type)

| ชนิดข้อมูล ภาษา C<br>สำหรับ CCS C | ขนาด (Byte)     | ไม่คิดเครื่องหมาย (unsigned)                  | คิดเครื่องหมาย (signed)          |
|-----------------------------------|-----------------|---|----------------------------------|
| int1                              | 1 บิต (ตัวเลข)  | 0 or 1  | -                                |
| int8                              | 8 บิต (ตัวเลข)  | 0 to 255                                      | -128 to 127                      |
| int16                             | 16 บิต (ตัวเลข) | 0 to 65535                                    | -32768 to +32767                 |
| int32                             | 32 บิต (ตัวเลข) | 0 to 4,294,967,295                            | -2,147,483,648 to +2,147,483,648 |
| float32                           | 32 บิต (ทศนิยม) | $1.5 \times 10^{-45}$ to $3.4 \times 10^{38}$ |                                  |

### - การคำนวณทางคณิตศาสตร์

สัญลักษณ์ในการคำนวณทางคณิตศาสตร์จะเหมือนกับที่เราเคยเรียนกันมา จะต่างเพียงบางตัวเท่านั้น เช่น ตัวหาร ในภาษาซีจะใช้เป็นเครื่องหมาย / ในการคำนวณแต่ละครั้งเราจะต้องกำหนดค่าของตัวแปรแต่ละตัวก่อน แล้วจึงนำไปคำนวณตัวอย่างเช่น

```
int x, y, z; //กำหนดให้ x, y และ z เป็นตัวแปรขนาด 8 บิต
x = 10; y = 5; //กำหนดให้ x มีค่าเท่ากับ 10 และ y มีค่าเท่ากับ 5
z = x + y //ค่าของ z มีค่าเท่ากับ x + y นั่นคือ 15
```

### 3.การทำงานแบบมีเงื่อนไข

การทำงานในลักษณะนี้จะเป็นการตรวจสอบเงื่อนไขตามที่เราดังเอาไว้ ถ้าเป็นไปตามเงื่อนไขที่เราดังเอาไว้ ก็จะไปทำงานในเงื่อนไขนั้น ถ้าไม่ใช่ก็จะกระโดดข้ามไป คำสั่งในลักษณะนี้มีอยู่ 2 คำสั่ง คือ

#### - คำสั่ง if ..... else .....

เป็นคำสั่งที่ทำการตรวจสอบเงื่อนไขที่เรากำหนด ถ้าเป็นจริงก็จะทำงานในปีกกาของ if แต่ถ้าไม่ใช่ก็จะทำงานในปีกกาของ else เช่น

#### ตัวอย่างโปรแกรมที่ 1

```
if (a==1) //ถ้า a มีค่าเท่ากับ 1 จะทำให้ b มีค่าเท่า 2 ถ้าไม่ใช่ก็จะกระโดดข้ามไป
{
    b = 2;
}
```

#### ตัวอย่างโปรแกรมที่ 2

```
if (a==1) //ถ้า a มีค่าเท่ากับ 1 จะทำให้ b มีค่าเท่า 2
{
    b = 2;
}
```

```
else //ถ้า a ไม่เท่ากับ 1 จะทำให้ b มีค่าเท่ากับ 0
{
    b = 0;
}
```

ตัวอย่างโปรแกรมที่ 3

```
if (a==1) //ถ้า a มีค่าเท่ากับ 1 จะทำให้ b มีค่าเท่า 2
{
    b = 2;
}
else if (a==2) //ถ้า a มีค่าเท่ากับ 2 จะทำให้ b มีค่าเท่ากับ 0
{
    b = 0;
}
```

#### - คำสั่ง switch

เป็นคำสั่งที่ใช้ตรวจสอบหลายๆ เงื่อนไขในครั้งเดียว ซึ่งถ้าเราใช้คำสั่ง if ... else ... อาจจะทำให้คำสั่งนั้นยาวเกินความจำเป็น ตัวอย่างเช่น

```
switch (a) //ตรวจสอบ a
{
    case 0 : b = 1; //ถ้า a มีค่าเท่ากับ 0 จะทำให้ b มีค่าเท่ากับ 1 แล้วจึงหยุดการทำงานด้วย break
    break;
    case 1 : b = 2; //ถ้า a มีค่าเท่ากับ 1 จะทำให้ b มีค่าเท่ากับ 2 แล้วจึงหยุดการทำงานด้วย break
    break;
    case 2 : b = 3; //ถ้า a มีค่าเท่ากับ 2 จะทำให้ b มีค่าเท่ากับ 3 แล้วจึงหยุดการทำงานด้วย break
    break;
}
```

#### 4.การทำงานแบบวนลูป

เป็นคำสั่งที่สั่งให้โปรแกรมทำงานอยู่ภายในลูป ซึ่งจะทำงานไปเรื่อยๆ จนกระทั่งเงื่อนไขจะเป็นเท็จ ก็จะหลุดออกจากลูปไป

##### - คำสั่ง while()

เป็นคำสั่งที่จะทำการตรวจสอบเงื่อนไขว่าเป็นจริงหรือไม่ ถ้าเป็นจริงก็จะทำงานในคำสั่ง while ไปเรื่อยๆ จนกว่าจะเป็นเท็จจึงจะหลุดออกจากลูป ตัวอย่างโปรแกรม

```
while (a<10) //ถ้า a มีค่าน้อยกว่า 10 ก็ให้ทำการบวกค่าไปที่ละ 1 จนมีค่าเท่ากับ 10 จึงหลุดไป
{
    a++; //บวก a ที่ละหนึ่งเรื่อยๆ และเก็บค่าที่บวกไว้ที่ a
}
```

##### - คำสั่ง do ..... while()

เป็นคำสั่งที่แตกต่างจากคำสั่ง while ตรงที่จะกระทำคำสั่งภายในลูปก่อน 1 ครั้ง แล้วจึงจะตรวจสอบเงื่อนไข ถ้าเป็นจริงก็จะทำงานในลูป แต่ถ้าเป็นเท็จจึงจะหลุดออกจากลูป ตัวอย่างโปรแกรม

```
do
{
    a++; //บวก a ที่ละหนึ่งเรื่อยๆ
}
while (a<10); //ถ้า a มีค่าน้อยกว่า 10 ก็ให้ทำการบวกค่าไปที่ละ 1 จนมีค่าเท่ากับ 10 จึงหลุดไป
```

##### - คำสั่ง for

เป็นคำสั่งที่จะทำงานตามจำนวนที่กำหนดเอาไว้ เมื่อครบแล้วก็จะหลุดออกจากลูปไป ตัวอย่างโปรแกรม



```

for (i=0; i<10; i++)      //กำหนดให้ i เท่ากับศูนย์ แล้วบวกไปเรื่อยๆ จนเท่ากับ 10 จึงหยุดไป
{
    a++;      //บวก a ทีละหนึ่งเรื่อยๆ
    a = a+1;  //นำค่า a บวกหนึ่ง
}

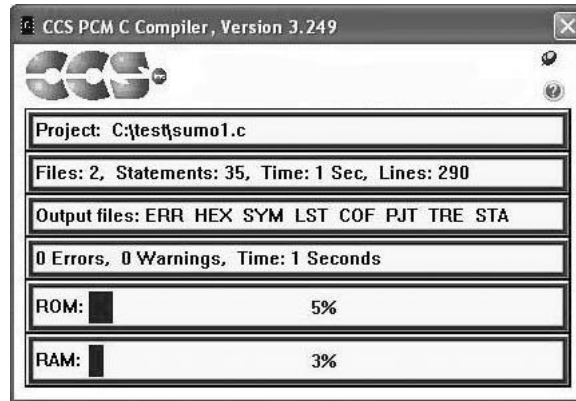
```

จากที่กล่าวมาข้างต้นนั้นจะเป็นคำสั่งที่ใช้งานเป็นหลักอยู่แต่ยังมีคำสั่งอื่นๆอีกตรงนี้คงต้องไปหาอ่านในหนังสือ ภาษา C เอาละกันครับ ไม่เช่นนั้นเราคงยังไม่ได้เขียนโปรแกรมให้กับหุ่นของเราเป็นแน่

### คอมไพล์จากภาษาซีไปสู่ HEX FILE

หลังจากที่เราเขียนโปรแกรมเสร็จเรียบร้อยแล้ว เราจะต้องทำการเปลี่ยนจากภาษาซีให้เป็น HEX FILE ก่อน เพื่อให้เราจะได้โหลดโปรแกรมลงบนตัวไอซีไมโครคอนโทรลเลอร์ได้ โดยขั้นตอนมีดังนี้

เมื่อทำการเขียนโปรแกรมจนเสร็จเรียบร้อยแล้ว ให้เข้าไปที่เมนู Compile จากนั้นเลือก Compile (หรือจะกดปุ่ม F9 ก็ได้) โปรแกรมจะแสดงหน้าต่างแสดงรายละเอียดเกี่ยวกับการคอมไพล์ทั้งหมดขึ้นมา โดยไฟล์ที่คอมไพล์ไม่ผ่านหรือมีข้อผิดพลาดใด โปรแกรมจะแสดงกรอบขึ้นมาแจ้งถึงข้อผิดพลาดนั้น แต่ถ้าคอมไพล์ผ่าน โปรแกรมจะสร้างไฟล์อื่นๆ ขึ้นมา โดยหนึ่งในนั้นก็คือ ไฟล์ HEX ที่เราจะนำไปโหลดลงตัวไอซีไมโครคอนโทรลเลอร์นั่นเอง



รูปแสดงผลการคอมไพล์ไฟล์ที่คอมไพล์ผ่าน

มาถึงขั้นตอนที่น่าจะพอเข้าใจโปรแกรม CCS C compiler บ้างแล้วนะครับ การใช้งาน และการเขียนโปรแกรมอย่างละเอียดขอแนะนำหนังสือของ APPSOFTTECH หาซื้อได้ที่นี้เลยครับ [www.robodkit.com](http://www.robodkit.com)

ตัวอย่างโปรแกรมที่ 1 กับการสั่งงานให้ LED ทำงาน

#### EX\_1LED

```

1.  /*****
2.  *Work File : EX_1LED
3.  *Target   : AP114 PIC ROBOD V2
4.  *Compiler : CCS C Compiler
5.  *Copyright : Advance Project Group Co.,ltd
6.  *Device   : PIC16F886
7.  *****/
8.  #include <16F886.h>      // Standard Header file for the PIC16F886 device
9.  #define CLOCK_SP 400000 // Clock Speed(Hz)
10. //Device Specification
11. #fuses INTRC_IO , MCLR

```



```

12. #use delay (clock=CLOCK_SP) // Use built-in function: delay_ms() & delay_us()
13. #define Led1 PIN_B0 // Port LED1
14. #define Led2 PIN_B1 // Port LED2
15. void main(void)
16. {
17. // Set port input = 1 , output = 0,
18.     set_tris_b(0b00111100); // Set RB0-RB1,RB6-RB7 = output, RB2-RB5 = input
19.     while (TRUE)
20.     {
21.         output_low(LED1);
22.         output_low(LED2);
23.         delay_ms(500);
24.         output_high(LED1);
25.         output_high(LED2);
26.         delay_ms(500);
27.     }
28. }

```

### อธิบายการทำงานของโปรแกรม

- บรรทัดที่ 1-7 เป็นการอธิบายเพื่อทำความเข้าใจโค้ดโปรแกรมที่ได้เขียนไว้ ในส่วนนี้จะไม่มีหรือไม่มีก็ได้เพราะเวลาคอมไพล์ มันจะเข้าไป ในตำแหน่งนี้จะใช้เครื่องหมายคอมเม้นแบบหลายบรรทัดเราจะใช้แบ็กสแลชแล้วตามด้วยเครื่องหมายดอกจัน แล้วใส่คำอธิบายที่บรรทัดก็ได้เมื่อจบก็ใส่ดอกจันแล้วตามด้วยแบ็กสแลช เช่น /\* คำอธิบาย \*/
- บรรทัดที่ 8 เป็นการเรียกใช้เฮดเดอร์ไฟล์ ในที่นี้เราใช้ IC เบอร์ 16F886 ในการทำงาน จึงใช้เฮดเดอร์ไฟล์ เบอร์ 16F886.h โดยที่ .h หมายถึง เฮดเดอร์ไฟล์ (Header file) เพื่อใช้ในการประมวลผลเบื้องต้น
- บรรทัดที่ 9 เป็นการกำหนดความถี่ในการใช้งานในที่นี้กำหนดไว้ที่ 4Mhz หรือ 4000000Hz
- บรรทัดที่ 10 เป็นการอธิบายแบบบรรทัดเดียวเราจะใช้แบ็กสแลช สองอันติดกันแล้วตามด้วยคำอธิบายให้อยู่ในบรรทัดเดียวเท่านั้น เช่น //คำอธิบาย
- บรรทัดที่ 11 เป็นการกำหนดคุณสมบัติต่างๆของ CPU เช่น
  - ออสซิลเลเตอร์ที่ใช้ทำงาน (LP,XT,HS,RCIO,EC,INTRC) ในที่นี้ใช้ INTRC\_IO คือใช้ความถี่ภายในและให้ขา OSC เป็น IO
  - การเปิดใช้ซารีเซท (MCLR)
- บรรทัดที่ 12 เรียกใช้ฟังก์ชัน หน่วงเวลา
- บรรทัดที่ 13-14 กำหนดชื่อเรียก Led1 และ Led2 ที่ขา B0 และ B1
- บรรทัดที่ 15 เป็นชื่อฟังก์ชันหลักของภาษา C ในการเริ่มต้น โค้ดโปรแกรมจะต้องมีฟังก์ชันที่ชื่อว่า main() เสมอ
- บรรทัดที่ 18 เป็นการกำหนดทิศทางของ PORT CPU จะให้เป็น INPUT or OUTPUT เช่น เมื่อต้องการให้เป็น INPUT ให้กำหนดคิบนั้นเป็น 1 และถ้าเป็น OUTPUT ให้กำหนดคิบนั้นเป็น 0
- บรรทัดที่ 19 คำสั่ง while(TRUE) เป็นคำสั่งให้ทำงานแบบวนลูปไม่รู้จบ

- บรรทัดที่ 21-26 เรียกใช้ฟังก์ชัน `output_low()` กำหนดให้สถานะขาพอร์ต RB0 และ RB1 เป็นลอจิก “0” และ `output_high()` กำหนดให้สถานะขาพอร์ต RB0 และ RB1 เป็นลอจิก “1” และหน่วงเวลาด้วยฟังก์ชัน `delay_ms()` ทุกๆ 0.5 วินาที เพื่อให้เห็นการติดดับสลับกันของ LED

### ใบงานทดลองการทำงานของโปรแกรม

- ทดลองเพิ่มหรือลดค่าในในการหน่วงเวลา `delay_ms(.....)` แล้วทำการ **Compiler** และ โหลดลงตัวหุ่นสังเกตการณ์ทำงาน
- ทดลองใส่เครื่องหมาย // หน้า `output_low(LED1);` และ `output_high(LED1);` แล้วทำการ **Compiler** และ โหลดลงตัวหุ่น สังเกตการณ์ทำงาน
- ทดลองเขียนโปรแกรมให้ LED1 และ LED2 ติดดับ สลับกัน

ตัวอย่างโปรแกรมที่ 2 กับการสั่งงานให้ Buzzer ทำงาน

#### EX\_2Buzzer

```

1.  /*****
2.  *Work File : EX_2Buzzer
3.  *Target   : AP114 PIC ROBOD V2
4.  *Compiler : CCS C Compiler
5.  *Copyright : Advance Project Group Co.,Ltd
6.  *Device   : PIC16F883
7.  *****/
8.  #include <16F886.h>           // Standard Header file for the PIC16F883 device
9.  #define CLOCK_SP 4000000 // Clock Speed(Hz)
10. // Device Specification
11. #fuses INTRC_IO,MCLR
12. #use delay (clock=CLOCK_SP) // Use built-in function: delay_ms() & delay_us()
13. #define Buzzer PIN_C0        // Port Buzzer
14. void main(void)
15. {
16. // Set port input = 1 , output = 0,
17.     set_tris_c(0b11111110); // Set RC0 = output , C1-C7 = input
18.     while (TRUE)
19.     {
20.         output_high(Buzzer);
21.         delay_ms(500);
22.         output_low(Buzzer);
23.         delay_ms(500);
24.     }
25. }
```

### อธิบายการทำงานของโปรแกรม

- บรรทัดที่ 13 กำหนดชื่อเรียก Buzzer ที่ขา C0
- บรรทัดที่ 17 ให้พอร์ต C0 เป็น OUTPUT

- บรรทัดที่ 20-23 เรียกใช้ฟังก์ชัน output\_high() กำหนดให้สถานะขาพอร์ต RC0 เป็นลอจิก “1” Buzzer ทำงาน และ output\_low() กำหนดให้สถานะขาพอร์ต RC0 เป็นลอจิก “0” Buzzer หยุดทำงาน และ หน่วงเวลาด้วย ฟังก์ชัน delay\_ms() ทุกๆ 0.5 วินาที เพื่อให้ Buzzer มีเสียง สลับ กับหยุด

ตัวอย่างโปรแกรมที่ 3 กับการรับคำสั่งการกดสวิทช์

EX\_3Switch

```

1.  /******
2.  *Work File : EX_3Switch
3.  *Target   : AP114 PIC ROBOD V2
4.  *Compiler : CCS C Compiler
5.  *Copyright : Advance Project Group Co.,ltd
6.  *Device   : PIC16F883
7.  *****/
8.  #include <16F886.h>           // Standard Header file for the PIC16F883 device
9.  #define CLOCK_SP 4000000 // Clock Speed(Hz)
10. // Device Specification
11. #fuses INTRC_IO,MCLR
12. #use delay (clock=CLOCK_SP) // Use built-in function: delay_ms() & delay_us()
13. #define Switch_1 PIN_B2 // Port Switch input
14. #define Buzzer PIN_C0 // Port Buzzer output
15. void main(void)
16. {
17. // Set port input = 1 , output = 0,
18.     set_tris_b(0b11111111); // Set RB0-RB7 = input
19.     set_tris_c(0b11111110); // Set RC0 = output , C1-C7 = input
20.     while (TRUE)
21.     {
22.         if(!input(Switch_1))
23.         {
24.             delay_ms(20);
25.             if(!input(Switch_1)){output_high(Buzzer);}
26.         }
27.         else {output_low(Buzzer);}
28.     }
29. }
```

อธิบายการทำงานของโปรแกรม

- บรรทัดที่ 13 , 14 เป็นการกำหนดขาสวิทช์ที่ RB2 และ Buzzer ที่ RC0
- บรรทัดที่ 18 , 19 ให้พอร์ต C0 เป็น OUTPUT พอร์ต B2 เป็น INPUT
- บรรทัดที่ 22 ใช้คำสั่ง if() ในการตรวจจับการกดสวิทช์ด้วยฟังก์ชัน input() ปกติสวิทช์จะมีลอจิกเป็น “1” เมื่อกดสวิทช์ จะได้ลอจิกเป็น “0” เครื่องหมาย “!” ที่อยู่หน้าฟังก์ชัน input() นั้นมีไว้ตรวจสอบว่าค่าที่รับได้เป็น “0” หรือไม่

4. บรรทัดที่ 24 เมื่อตรวจสอบการกดสวิตช์เป็น "0" ก็จะเข้าไปทำงานในปีกกา และเรียกใช้ฟังก์ชัน delay\_ms() เพื่อทำการหน่วงเวลาประมาณ 20ms เหตุที่ต้องมีส่วนนี้เพื่อป้องกันการกดสวิตช์ที่ซ้ำกัน
5. บรรทัดที่ 25 เมื่อหน่วงเวลาแล้วก็จะเข้ามาตรวจสอบการกดสวิตช์อีกครั้งถ้ายังเป็น "0" อยู่ ก็จะเรียกฟังก์ชัน output\_high() เพื่อให้ Buzzer ทำงาน
6. บรรทัดที่ 27 เมื่อมีการปล่อยสวิตช์หรือสวิตช์มีสถานะลอจิก "1" ก็จะไปที่คำสั่ง else เพื่อเรียกฟังก์ชัน output\_low() เพื่อให้ Buzzer หยุดทำงาน

ตัวอย่างโปรแกรมที่ 4 กับการทำงานการรับค่า ADC ทำให้ LED ติดดับ สลับกัน

#### EX\_4ADC

```

1.  /******
2.  *Work File : EX_4ADC
3.  *Target   : AP114 PIC ROBOD V2
4.  *Compiler : CCS C Compiler
5.  * Copyright : Advance Project Group Co.,ltd
6.  *Device   : PIC16F886
7.  *****/
8.  #include <16f886.h> // Standard Header file for the PIC16F886 device
9.  #define CLOCK_SP 8000000 // Clock Speed(Hz)
10. #device *=16 ADC=10
11. // Device Specification
12. #fuses INTRC_IO,PUT,NOLVP,NOWDT,NOPROTECT,NOBROWNOUT,MCLR
13. #use delay (clock=CLOCK_SP) // Use built-in function: delay_ms() & delay_us()
14. #define Led1 PIN_B0 // Port LED1
15. #define Led2 PIN_B1 // Port LED2
16. void main(void)
17. {
18. int16 read_adc_ch0;
19. // Set port input = 1 , output = 0,
20. set_tris_a(0b00001111); // Set RA0-RA3 = input,RA4-RA7 = output
21. set_tris_b(0b00111100); // Set RB0-RB1,RB6-RB7 = output, RB2-RB5 = input
22. set_tris_c(0b00000000); // Set RC0-7 = output
23. setup_adc_ports(sAN0); //A0
24. setup_adc(ADC_CLOCK_INTERNAL); //Frc
25. set_adc_channel(0); //AN0 or RA0
26. while (TRUE)
27. {
28. read_adc_ch0 = read_adc();
29. if(read_adc_ch0 > 512)
30. {
31. output_low(LED1);
32. output_high(LED2);
33. delay_ms(500);
34. }
35. else

```

```

36.      {
37.          output_low(LED2);
38.          output_high(LED1);
39.          delay_ms(500);
40.      }
41.  }
42. }

```

### อธิบายการทำงานของโปรแกรม

1. บรรทัดที่ 18 สร้างตัวแปรชื่อ read\_adc\_ch0 เพื่อไว้เก็บค่าที่อ่านได้จาก ADC
2. บรรทัดที่ 20-22 เป็นการกำหนดทิศทางขา INPUT และ OUTPUT
3. บรรทัดที่ 23 เป็นการกำหนดขาพอร์ต อะนาล็อก AN0
4. บรรทัดที่ 24 กำหนดสัญญาณนาฬิกาให้กับโมดูล A/D แบบ Frc คือการใช้สัญญาณนาฬิกาจากภายใน
5. บรรทัดที่ 25 กำหนดช่องสัญญาณที่จะอ่านค่า อะนาล็อก ตัวอย่างกำหนดไว้ที่ “0”
6. บรรทัดที่ 28 เรียกใช้ฟังก์ชัน read\_adc() แล้วนำมาเก็บไว้ที่ตัวแปรชื่อ read\_adc\_ch0 ค่าที่อ่านได้จะอยู่ในช่วง 0-1027
7. บรรทัดที่ 29 ตรวจสอบค่าที่อ่านมาได้มากกว่า 512 หรือไม่
8. บรรทัดที่ 31-33 ทำมากกว่าให้เข้ามาทำในปีกกา if คือ ให้ LED1 ติด LED2 ดับ และ หน่วงเวลา 0.5วินาที
9. บรรทัดที่ 35-40 เมื่อค่าที่อ่านมาไม่มากกว่า 512 จะมาทำงานที่ else คือ ให้ LED1 ดับ LED2 ติด และ หน่วงเวลา 0.5วินาที

ตัวอย่างโปรแกรมที่ 5 กับการควบคุมการทำงานของมอเตอร์

### EX\_5motor

```

1.  /******
2.  *Work File : EX_5motor
3.  *Target   : AP114 PIC ROBOD V2
4.  *Compiler : CCS C Compiler
5.  * Copyright : Advance Project Group Co.,ltd
6.  *Device   : PIC16F886
7.  *****/
8.  #include <16f886.h>           // Standard Header file for the PIC16F886 device
9.  #define CLOCK_SP 8000000     // Clock Speed(Hz)
10. // Device Specification
11. #fuses INTRC_IO,PUT,NOLVP,NOWDT,NOPROTECT,NOBROWNOUT,MCLR
12. #use delay (clock=CLOCK_SP)  // Use built-in function: delay_ms() & delay_us()
13. #define M_on PIN_C1          // Port Control Vcc for Motor
14. void Stop()
15. {
16. // Control motor right
17.     output_high(PIN_C2);
18.     output_low(PIN_C3);
19.     output_low(PIN_C4);
20.     output_high(PIN_C5);

```

```
21. // Control motor left
22.     output_high(PIN_A4);
23.     output_low(PIN_A5);
24.     output_low(PIN_A6);
25.     output_high(PIN_A7);
26. }
27. void M_break()
28. {
29. // Control motor right
30.     output_high(PIN_C2);
31.     output_high(PIN_C3);
32.     output_high(PIN_C4);
33.     output_high(PIN_C5);
34. // Control motor left
35.     output_high(PIN_A4);
36.     output_high(PIN_A5);
37.     output_high(PIN_A6);
38.     output_high(PIN_A7);
39. }
40. void go()
41. {
42. // Control motor right
43.     output_high(PIN_C2);
44.     output_high(PIN_C3);
45.     output_low(PIN_C4);
46.     output_low(PIN_C5);
47. // Control motor left
48.     output_low(PIN_A4);
49.     output_low(PIN_A5);
50.     output_high(PIN_A6);
51.     output_high(PIN_A7);
52. }
53. void back()
54. {
55. // Control motor right
56.     output_low(PIN_C2);
57.     output_low(PIN_C3);
58.     output_high(PIN_C4);
59.     output_high(PIN_C5);
60. // Control motor left
61.     output_high(PIN_A4);
62.     output_high(PIN_A5);
63.     output_low(PIN_A6);
64.     output_low(PIN_A7);
65. }
66. void M_L_go()
```

```

67. {
68.     output_low(PIN_A4);
69.     output_low(PIN_A5);
70.     output_high(PIN_A6);
71.     output_high(PIN_A7);
72. }
73. void M_R_go()
74. {
75.     output_high(PIN_C2);
76.     output_high(PIN_C3);
77.     output_low(PIN_C4);
78.     output_low(PIN_C5);
79. }
80. void main(void)
81. {
82. // Set port input = 1 , output = 0,
83.     set_tris_a(0b00001111); // Set RA0-RA3 = input,RA4-RA7 = output
84. // set_tris_b(0b00111100); // Set RB0-RB1,RB6-RB7 = output, RB2-RB5 = input
85.     set_tris_c(0b00000000); // Set RC0-7 = output
86.     output_low(M_on);
87.     while (TRUE)
88.     {
89.         Stop(); delay_ms(500);
90.         go(); delay_ms(500);
91.         Stop(); delay_ms(500);
92.         back(); delay_ms(500);
93.         M_break(); delay_ms(500);
94.         M_L_go(); delay_ms(500);
95.         M_break(); delay_ms(500);
96.         M_R_go(); delay_ms(500);
97.         M_break(); delay_ms(500);
98.     }
99. }

```

#### อธิบายการทำงานของโปรแกรม

- บรรทัดที่ 13 กำหนดค่า M\_on PIN\_C1 ในตำแหน่งขานี้จะต่ออยู่กับ Q5 เพื่อทำหน้าที่ในการจ่ายไฟให้กับวงจรควบคุมการทำงานของมอเตอร์ ตามตารางข้างต้นที่ได้กล่าวไว้แล้ว
- บรรทัดที่ 14-26 สร้างฟังก์ชัน Stop() เพื่อสั่งให้มอเตอร์หยุดทำงาน
- บรรทัดที่ 27-39 สร้างฟังก์ชัน back() ฟังก์ชันนี้จะคล้ายกับ Stop() แต่มอเตอร์จะหยุดการหมุนเลยและจะไม่มีแรงเฉื่อยเหมือน Stop()
- บรรทัดที่ 40-52 สร้างฟังก์ชัน go() จะสั่งให้มอเตอร์เดินหน้า
- บรรทัดที่ 53-65 สร้างฟังก์ชัน back() จะสั่งให้มอเตอร์ถอยหลัง
- บรรทัดที่ 66-72 สร้างฟังก์ชัน M\_L\_go() จะสั่งให้มอเตอร์ซ้ายเดินหน้า
- บรรทัดที่ 73-79 สร้างฟังก์ชัน M\_R\_go() จะสั่งให้มอเตอร์ขวาเดินหน้า

8. บรรทัดที่ 83และ85 เป็นการกำหนดทิศทางของขา PORT A และ C
9. บรรทัดที่ 86 กำหนดค่าควบคุมมอเตอร์ทำงานจ่ายไฟให้กับวงจรควบคุมมอเตอร์
10. บรรทัดที่ 89-97 เรียกใช้งานฟังก์ชันการทำงานของมอเตอร์ที่ละฟังก์ชันแล้วตามด้วยการหน่วงเวลา

ตัวอย่างโปรแกรมที่ 6 กับการรับสัญญาณ sensorINF

EX\_6 sensorINF

```

1.  /******
2.  *Work File : EX_6sensorINF
3.  *Target   : AP114 PIC ROBOD V2
4.  *Compiler : CCS C Compiler
5.  * Copyright : Advance Project Group Co.,ltd
6.  *Device   : PIC16F886
7.  *****/
8.  #include <16f886.h>           // Standard Header file for the PIC16F886 device
9.  #define CLOCK_SP 8000000     // Clock Speed(Hz)
10. // Device Specification
11. #fuses INTRC_IO,PUT,NOLVP,NOWDT,NOPROTECT,NOBROWNOUT,MCLR
12. #use delay (clock=CLOCK_SP)  // Use built-in function: delay_ms() & delay_us()

13. #define INF_L  PIN_A1  // Port Infrared left
14. #define INF_R  PIN_B5  // Port Infrared right
15. #define Led1   PIN_B0  // Port LED1
16. #define Led2   PIN_B1  // Port LED2
17. void main(void)
18. {
19. // Set port input = 1 , output = 0,
20.     set_tris_a(0b00001111); // Set RA0-RA3 = input,RA4-RA7 = output
21.     set_tris_b(0b00111100); // Set RB0-RB1,RB6-RB7 = output, RB2-RB5 = input
22. // set_tris_c(0b00000000); // Set RC0-7 = output
23.     while (TRUE)
24.     {
25.         if(!input(INF_L)) output_low(LED1);
26.         else output_high(LED1);
27.         if(!input(INF_R)) output_low(LED2);
28.         else output_high(LED2);
29.     }
30. }

```

### อธิบายการทำงานของโปรแกรม

1. บรรทัดที่ 13-16 กำหนดชื่อขาพอร์ตที่ใช้เรียก sensorINF ด้านซ้ายและขวาและ LED
2. บรรทัดที่ 25 คำสั่ง if() ตรวจสอบสัญญาณ sensorINF ด้านซ้าย เป็น low(0) หรือไม่ ถ้าใช่ให้ LED ด้านซ้ายติด
3. บรรทัดที่ 26 ถ้าไม่ใช่ให้ LED ซ้ายดับ



4. บรรทัดที่ 27 คำสั่ง if() ตรวจสอบสัญญาณ sensorINF ด้านขวา เป็น low(0) หรือไม่ ถ้าใช่ให้ LED ด้านขวาดิจ
5. บรรทัดที่ 28 ถ้าไม่ใช่ให้ LED ซ้าย ดับ

### ตัวอย่างโปรแกรมที่ 7 การใช้งานโมดูล PWM

#### EX\_7PWM

```

1.  /******
2.  *Work File : EX_7PWM
3.  *Target   : AP114 PIC ROBOD V2
4.  *Compiler : CCS C Compiler
5.  *Copyright : Advance Project Group Co.,ltd
6.  *Device   : PIC16F886
7.  *****/
8.  #include <16f886.h>           // Standard Header file for the PIC16F886 device
9.  #define CLOCK_SP 8000000     // Clock Speed(Hz)
10. // Device Specification
11. #fuses INTRC_IO,PUT,NOLVP,NOWDT,NOPROTECT,NOBROWNOUT,MCLR
12. #use delay (clock=CLOCK_SP)  // Use built-in function: delay_ms() & delay_us()
13. void go()
14. {
15. // Control motor right
16.     output_high(PIN_C2);
17.     output_high(PIN_C3);
18.     output_low(PIN_C4);
19.     output_low(PIN_C5);
20. // Control motor left
21.     output_low(PIN_A4);
22.     output_low(PIN_A5);
23.     output_high(PIN_A6);
24.     output_high(PIN_A7);
25. }
26. void main(void)
27. {
28.     unsigned int8 duty,value=0;
29. // Set port input = 1 , output = 0,
30.     set_tris_c(0b00000000);           // Set RC0-7 = output
31. // Set interrupt ccp2
32.     enable_interrupts(GLOBAL);
33.     enable_interrupts(INT_CCP2);      //Pin RC1/CCP2
34.     setup_ccp2(CCP_PWM);             // Setup CCP Module
35. /* Setup Timer2 Table datasheet pic16f886
36.     PR2   = 0x65 or 101
37.     Prescale = 16
38.     Fpwm  = 1.22kHz
39. */

```

```

40.     setup_timer_2(T2_DIV_BY_16, 102, 1);
41.     set_timer2(0);
42.     go();
43.     value = 80;
44.     while (TRUE)
45.     {
46.         duty = (value/100.0)*102;
47.         set_pwm2_duty(duty);
48.         delay_ms(50);
49.     }
50. }

```

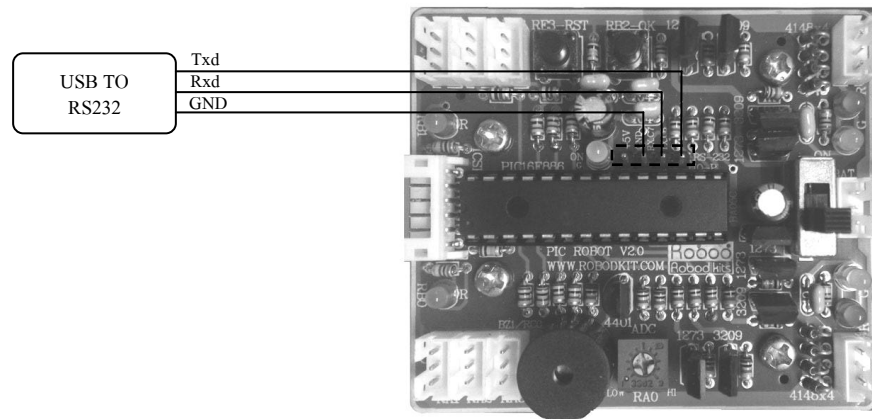
### อธิบายการทำงานของโปรแกรม

1. บรรทัดที่ 28 สร้างตัวแปรชื่อ duty และ value ขนาด 8 บิต
2. บรรทัดที่ 32 เปิดใช้งานฟังก์ชันอินเทอร์รัปต์โดยรวม (อินเทอร์รัปต์โดยรวมต้องเปิดใช้งานทุกครั้งเมื่อต้องการใช้งานอินเทอร์รัปต์) หลังจากนั้นถึงจะเปิดใช้งานอินเทอร์รัปต์ตัวที่ต้องการได้
3. บรรทัดที่ 32 เปิดใช้งานฟังก์ชันอินเทอร์รัปต์โมดูล INT\_CCP2
4. บรรทัดที่ 34 คัดตั้งการใช้งานโมดูล CCP2 ในโหมด PWM
5. บรรทัดที่ 36-38 แสดงค่าที่ใช้กำหนดตามตาราง datasheet ของ pic16f886
6. บรรทัดที่ 40 กำหนดค่าของสัญญาณ PWM ผ่านโมดูล ไทมเมอร์ 2 ด้วยฟังก์ชัน setup\_timer\_2() T2\_DIV\_BY\_16 คือค่าของ Prescale = 16 ส่วนค่า 102 คือค่าของ PR2+1 และสุดท้ายคือค่า postscale จะมีค่าอยู่ในช่วง 1-16 เป็นการกำหนดค่าการรีเซตก่อนการเกิดอินเทอร์รัปต์
7. บรรทัดที่ 41 เปิดใช้งานฟังก์ชัน set\_timer2() เริ่มต้นการนับค่าที่ 0
8. บรรทัดที่ 42-43 เรียกใช้ฟังก์ชัน go() เพื่อสั่งให้มอเตอร์เดินหน้า และกำหนดค่า value = 80
9. บรรทัดที่ 46 นำค่าที่กำหนดมาคำนวณและนำไปเก็บไว้ที่ duty ค่า 100.0 คือค่าที่นำไปหารเพื่อให้ออกมาเป็นเปอร์เซ็นต์(%) ค่า 102 ได้มาจากค่า PR2+1
10. บรรทัดที่ 47-48 เรียกใช้ฟังก์ชัน set\_pwm2\_duty() และนำค่าที่ได้ไปกำหนดความกว้างของสัญญาณ PWM และทำการหน่วงเวลา 0.5 ด้วยฟังก์ชันหน่วงเวลา delay\_ms()

### ใบงาน

1. ให้ทดลองเปลี่ยนค่าใน value ในช่วง 0-100 สังเกตการทำงานของมอเตอร์
2. ให้ทดลองเขียนโปรแกรมให้ มอเตอร์ทำงานที่ความเร็วสูงสุดแล้วลดมาต่ำสุดแล้วจากต่ำสุดไปหาเร็วสุด

ตัวอย่างโปรแกรมที่ 8 การใช้งานโมดูล UART



รูปแสดงการต่อ USB เข้าที่ RS232

EX\_801.c USART ก็ับการรับส่งข้อมูลตัวเลขเพื่อนำไปคำนวณแล้วแสดงผลออกทางหน้าจอ

```

1.  /******
2.  *Work File : EX_801.c USART
3.  *Target   : AP114 PIC ROBOD V2
4.  *Compiler : CCS C Compiler
5.  *Copyring : Advance Project Group Co.,ltd
6.  *Device   : PIC16F886
7.  *****/
8.  #include <16f886.h> // Standard Header file for the PIC16F886 device
9.  #define CLOCK_SP 8000000 // Clock Speed(Hz)
10. // Device Specification
11. #fuses INTRC_IO,PUT,NOLVP,NOWDT,NOPROTECT,NOBROWNOUT,MCLR
12. #use delay (clock=CLOCK_SP) // Use built-in function: delay_ms() & delay_us()
13. #use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7) //Serial port
14. #include <stdlib.h>
15. #include "input.c"
16. void main(void)
17. {
18.     long a,b,result;
19.     char opr;
20.     while (TRUE)
21.     {
22.         printf("\r\nEnter the first number: ");
23.         a = get_long();
24.         do
25.         {
26.             printf("\r\nEnter the operator(+,-,*,/): ");
27.             opr = getc();
28.         }
29.         while(!isamoung(opr,"+-*/"));
30.         printf("%c",opr);
31.         printf("\r\nEnter the second number: ");

```

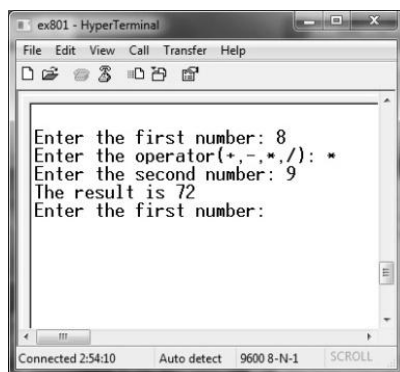
```

32.         b = get_long();
33.         switch(opr)
34.         {
35.             case '+': result = a+b; break;
36.             case '-': result = a-b; break;
37.             case '*': result = a*b; break;
38.             case '/': result = a/b; break;
39.         }
40.         printf("\n\nThe result is %lu ",result);
41.     }
42. }

```

### อธิบายการทำงานของโปรแกรม

- บรรทัดที่ 13-15 กำหนดการใช้งานไดเรกทีฟ #rs232() กำหนดการใช้งานพอร์ตนุกรม stdlib.h จะเป็น library ที่เก็บชุดคำสั่งที่เกี่ยวข้องกับ rs232 และ ไฟล์ INPUT.C เป็นไฟล์ที่ประกอบไปด้วยฟังก์ชันการรับข้อมูลผ่าน rs232 i/o
- บรรทัดที่ 18-19 เป็นการประกาศตัวแปรในการรับค่ามาจาก keyboard
- บรรทัดที่ 22 เป็นคำสั่งในการแสดงผลออกทางหน้าจอผ่าน โปรแกรม Hyper Terminal คำว่า Enter the first number: เพื่อให้เราป้อนค่าตัวเลข 0-9 จากนั้นกด Enter จากนั้นหน้าจอก็จะแสดง Enter the operator(+, -, \*, /): เพื่อให้เราป้อนเครื่องหมายในการคำนวณเช่น + - \* / จากนั้นกด Enter จากนั้นหน้าจอก็จะแสดง nEnter the second number: เพื่อให้เราใส่ค่าตัวเลขในลำดับต่อไปเพื่อทำการคำนวณ จากนั้นกด Enter หน้าจอก็จะแสดงผลลัพธ์ออกมา



### รูปแสดงหน้าต่าง Hyper Terminal

#### วิธีการตั้งค่าการใช้งาน Hyper Terminal

เมื่อเปิดหน้าต่าง Hyper Terminal ขึ้นมาจะแสดงหน้าต่าง Connection Description ในช่อง NAME ให้ตั้งชื่อตามต้องการ แล้วกด OK จะขึ้นหน้าต่าง Connect To ในช่อง Connect using ให้เลือกคอมพอร์ตที่ใช้ติดต่อกับ microcontroller ช่องอื่น ๆ ปกติแล้วได้ไว้ แล้วกด OK จากนั้นจะขึ้นหน้าต่าง com.... Properties ในช่อง Bits per second เลือก 9600 ในช่อง Flow control เลือก Xon / Xoff เพื่อให้สามารถรับค่าจากคีย์บอร์ดได้ จากนั้นกด OK จากนั้นก็จะเข้าสู่หน้าต่าง

### การใช้งาน

#### EX\_802.c UART Interrupt กับการควบคุมการ ติดดับ ของ LED โดยการรับค่าจากคีย์บอร์ด

- ```

1.  /******
2.  *Work File : EX_802.c UART Interrupt
3.  *Target   : AP114 PIC ROBOD V2
4.  *Compiler : CCS C Compiler
5.  *Copyring : Advance Project Group Co.,ltd
6.  *Device   : PIC16F886
7.  *****/
8.  #include <16f886.h> // Standard Header file for the PIC16F886 device

```

```

9. #define CLOCK_SP 8000000 // Clock Speed(Hz)
10. // Device Specification
11. #fuses INTRC_IO,PUT,NOLVP,NOWDT,NOPROTECT,NOBROWNOUT,MCLR
12. #use delay (clock=CLOCK_SP) // Use built-in function: delay_ms() & delay_us()
13. #use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)//Rerial port
14. #include <stdlib.h>
15. #include "input.c"
16. #define Led1 PIN_B0 // Port LED1
17. #define Led2 PIN_B1 // Port LED2
18. char ch;
19. short int RX=0;
20. //RS232 receive data
21. #INT_RDA
22. void RxD_ISR() //RxD_ISR()
23. {
24.     ch = getc();
25.     RX = 1;
26. }
27. void main(void)
28. {
29.     set_tris_b(0b00111100); // Set RB0-RB1,RB6-RB7 = output, RB2-RB5 = input
30.     output_high(PIN_B0);
31.     output_high(PIN_B1);
32.     enable_interrupts(GLOBAL);
33.     enable_interrupts(INT_RDA);
34.     printf("\n\nNumber 0 is off LED or Number 1 is on LED");
35.     while(TRUE)
36.     {
37.         if(RX)
38.         {
39.             if(ch=='1')
40.             {
41.                 printf("\n\nLED ON is ");
42.                 putc(ch);
43.                 RX = 0;
44.                 output_low(PIN_B0);
45.                 delay_ms(100);
46.             }
47.             else if(ch=='0')
48.             {
49.                 printf("\n\nLED OFF is ");
50.                 putc(ch);
51.                 RX = 0;
52.                 output_high(PIN_B0);
53.                 delay_ms(100);
54.             }

```

```

55.         else RX = 0;
56.         printf("\nrNumber 0 or 1");
57.         }
58.     }
59. }

```

#### อธิบายการทำงานของโปรแกรม

- บรรทัดที่ 21 กำหนดการใช้งานไบนารีที่ #INT\_RDA คือการเกิดอินเตอร์รัปต์ จากการรับค่าจาก RS232
- บรรทัดที่ 22-25 ฟังก์ชัน Rx\_D\_ISR() เมื่อเกิดอินเตอร์รัปต์จาก rs232 ค่าที่ได้จะรับมาจากฟังก์ชัน getch(); และเก็บค่าไว้ที่ ch และเซตค่า RX = 1
- บรรทัดที่ 34 ส่งคำว่า Number 0 is off LED or Number 1 is on LED ออกทางหน้าจอ Hyper Terminal
- บรรทัดที่ 37-56 ตรวจสอบค่า RX ด้วยคำสั่ง if(RX) เท่ากับ 1 จะเข้าไปทำงานตรวจสอบค่า ch ด้วยคำสั่ง if(ch) เท่ากับ '1' ก็จะเข้าไป ส่งค่า LED ON is และตามด้วยค่าใน ch ก็คือ 1 ออกทางหน้าจอ พร้อมกับกำหนดค่า RX เท่ากับ 0 เพื่อป้องกันไม่ให้โปรแกรมกลับมาทำงานซ้ำอีกครั้งจนกว่าจะมีการรับค่าเข้ามาใหม่ และจากนั้นก็ สั่งให้ LED ตำแหน่ง RB0 ติด และทำการหน่วงเวลา 0.1 วินาที และส่งค่า Number 0 or 1 และออกจากโปรแกรมไปรอที่คำสั่ง if(RX) ว่าจะมีการกดค่าเข้ามาใหม่ เมื่อกดค่าใหม่มาเป็น '0' โปรแกรมก็จะเข้าไปตรวจสอบค่าด้วยคำสั่ง if(ch) แต่ค่าที่ได้ไม่เท่ากับ '1' จึงกระโดดไปที่คำสั่ง else if(ch) และตรวจสอบว่าเป็น '0' ก็จะเข้าไปทำงานในคำสั่งต่อไปคือ ส่งค่า LED OFF is และค่า ch และกำหนดค่า RX=0 พร้อมกับสั่งให้ LED ดับ ตามด้วยการหน่วงเวลา 0.1 วินาที และส่งค่า Number 0 or 1 และออกจากโปรแกรม ไปรอที่ if(RX) ทำเรากดค่าที่ไม่ใช่ 0 หรือ 1 คำสั่งจะไปอ่านที่ else แทน และกำหนดค่า RX=0 และส่งค่า Number 0 or 1 เพื่อให้เรากดค่าที่ถูกต้องต่อไป

#### สรุป

จากตัวอย่างที่กล่าวมาทั้งหมดนั้นก็จะเป็นการอธิบายการเขียน โปรแกรมในแบบเบื้องต้นเท่านั้น และจะอ้างอิงจากการใช้งานจริงและบทความที่มีในปัจจุบัน ถ้าต้องการข้อมูลที่ละเอียดกว่านี้ แนะนำหนังสือ 2 เล่มนี้ครับ เล่มที่ 1 PIC Works Examples and C Source Code และเล่มที่ 2 All About CCS C คอมไพเลอร์ แต่ CPU ที่ใช้งานในบทความจะไม่ตรงกันแต่สามารถอ้างอิงและใช้แทนกันได้ บางฟังก์ชันและบางคำสั่งอาจจะใช้ชื่อไม่เหมือนกันตรงนี้ต้องอ้างอิงจาก datasheet ของเบอร์ CPU นั้นๆ