บทที่ 3
ความรู้พื้นฐานที่เกี่ยวข้องกับการใช้งาน PIC BASIC PRO

3.1 ไมโครคอนโทรลเลอร์กับ PIC BASIC PRO

1) Pic BASIC Pro ได้กำหนดความถี่ OSC ของไมโครคอนโทรลเลอร์ที่ใช้ไว้ที่ 4 MHz ซึ่งในความถี่ของคริสตอลตามนี้ จะทำให้ค่าหน่วงเวลาตามคำสั่ง PAUSE ตรงกับค่าเวลาจริง เช่น PAUSE 1000 จะมีค่าเท่ากับ 1 วินาทีจริง และจะทำให้ค่า Baud rate ของคำสั่ง SERIN และ SEROUT มีค่าตรงตามที่ระบุไว้ด้วย ถ้าหากเราต้องการให้ไมโครคอนโทรลเลอร์ทำงานเร็วขึ้น โดยเพิ่มความถี่ OSC ให้สูงขึ้นกว่า 4 MHz เช่น 10 MHz จะทำให้ค่าพารามิเตอร์ตามคำสั่งดังกล่าวไม่ตรงตามที่ระบุไว้ ดังนั้นถ้าต้องการเปลี่ยนค่าคริสตอล OSC แตกต่างไปจาก 4 MHz จะต้องใช้คำสั่ง DEFINE กำหนดค่า OSC ไว้ที่หัวโปรแกรมด้วย ดังนี้

```
DEFINE  OSC  10
```

การใช้คำสั่ง DEFINE OSC จะทำให้ Pic BASIC Pro ได้ชดเชยคาบเวลาของคำสั่งต่อไปนี้ให้ถูกต้อง ได้แก่ COUNT, DEBUG, DEBUGIN, DTMFOUT, FREQOUT, HPWM, HSERIN, HSEROUT, I2CREAD, I2CWRITE, LCDOUT, OWIN, OWOUT, PAUSE, PAUSEUS, SERIN, SERIN2, SEROUT, SEROUT2, SHIFTIN, SHIFTOUT, SOUND, XIN และ XOUT

2) การใช้หน่วยความจำ RAM ภายในตัวไมโครคอนโทรลเลอร์ เมื่อเรากำหนดตัวแปรหรือค่าคงที่เพื่อใช้งานในโปรแกรม Pic BASIC Pro จะกำหนดตำแหน่งของหน่วยความจำ RAM ที่อยู่ในตัว MCU ของเบอร์ที่กำหนดโดยอัตโนมัติ โดยเริ่มตั้งแต่ตำแหน่ง Address เริ่มต้น ซึ่ง MCU ของแต่ละเบอร์จะมีตำแหน่งเริ่มต้นไม่เท่ากัน เช่น ไมโครคอนโทรลเลอร์ เบอร์ PIC16F84 จะมีตำแหน่ง RAM เริ่มต้นที่ $0C และเบอร์ PIC16C74 จะมีตำแหน่งที่ $20 เป็นต้น ดังนั้น ในขั้นตอน COMPILE โปรแกรมต้องระบุเบอร์ของไมโครคอนโทรลเลอร์ให้ถูกต้องกับที่จะใช้งานด้วย

ในอีกทางหนึ่ง หากเราทราบตำแหน่งของ Address ของ RAM ใน MCU เราสามารถกำหนด BANK และตำแหน่งของ RAM ทางตัวแปรได้ตามต้องการของเราได้ดังนี้ เช่น

```
Penny   VAR    WORD   BANK0
Nickel  VAR    BYTE   BANK1
Disp    VAR    BYTE   $20
```

3) ข้อควรระวังเกี่ยวกับการใช้ PORTA ของ PIC ไมโครคอนโทรลเลอร์ PIC ไมโครคอนโทรลเลอร์บางรุ่น เช่น PIC16F62X หรือ PIC16C62X เช่น (16C 620, 621, 622 16F627 และ 16F628) เบอร์เหล่านี้ PORTA ได้กำหนดขาให้ใช้งานได้ทั้งดิจิตอลและอนาล็อก คือ ทุกขาจะมีวงจร Analog Comparator เมื่อเริ่มใช้งาน (Start up) PORTA จะถูกกำหนดให้เป็น analog mode เพื่อที่จะเปลี่ยน Pin ของ PortA เหล่านี้เป็นดิจิตอลต้องกำหนดคำสั่งต่อไปนี้บนหัวของโปรแกรม คือ

```
CMCON = 7
```

PIC ไมโครคอนโทรลเลอร์บางเบอร์มี PORTA เป็นขาที่รับสัญญาณอนาล็อกได้ เช่น เบอร์ PIC16C7XX, PIC16F87X และ PIC12C67X PIC เบอร์เหล่านี้ เมื่อ Start up portA จะถูกกำหนดให้ใช้งานเป็น analog mode ดังนั้น หากต้องการ PortA ใช้งานเป็น Digital ต้องกำหนดคำสั่งต่อไปนี้บนหัวโปรแกรม

ADCON1 = 7

แต่ถ้าหากต้องการ Set ให้ PORTA รับสัญญาณดิจิตอล ต้องใช้คำสั่งต่อไปนี้ก่อนคือ

ADCON1 = 0

* ที่ PORTA ขา 4 ตามโครงสร้างจะเป็นแบบ open-drain output ดังนั้น ถ้ากำหนดให้ขานี้เป็น output เพื่อส่งลอจิก 1 ออกมาถ้าจะทำให้ขานี้มีสภาพลอย (Float) แทนที่จะเป็น 1 (หรือ high) เพื่อที่จะแก้ปัญหานี้ ต้องต่อตัวต้านทาน Pull-up เข้ากับไฟ 5 V ตัวต้านทานที่มาต่อควรมีค่าระหว่าง 1 k – 33 k

4) PIC Microcontroller บางเบอร์สามารถกำหนดโปรแกรมตัวชิพแบบ Low-voltage Programming ได้ เช่น PIC16F627, 628, 873, 874, 876 และ 877 โดยการกำหนดให้ขาใดขาหนึ่งที่ PORTB เป็นขาควบคุมสำหรับการโปรแกรมแบบนี้ ดังนั้น เพื่อที่จะป้องกันปัญหาการโปรแกรมซ้อนกันที่โปรแกรมไว้แล้วจะต้องทำให้ขานี้ลง Ground หรือ Pull-Low ขณะที่เราโปรแกรมด้วยวิธี High-voltage Programming

5) PIC Microcontroller ถูกกำหนดให้ขาทุกขาเป็น input ขณะทีเปิดไฟ (power-up) ถ้าต้องการขาเป็น Output ต้องใช้คำสั่งกำหนดให้ขาเหล่านั้นเป็น output ก่อนใช้งาน รายละเอียดนอกเหนือจากนี้ให้ศึกษาเพิ่มเติมจาก Datasheet

3.2 ระบบตัวเลขและการจัดการข้อมูลตัวเลข

ระบบตัวเลขเป็นพื้นฐานที่สำคัญมากในการเรียนรู้ไมโครคอนโทรลเลอร์ เนื่องจากข้อมูลที่ใช้ในการประมวลผลจริง ๆ ล้วนแล้วแต่อยู่ในรูปของรหัสตัวเลขทั้งสิ้น ซึ่งก็มีทั้งเลขฐานสอง, ฐานสิบ และฐานสิบหก ดังนั้น ผู้ใช้งานไมโครคอนโทรลเลอร์ต้องให้ความสำคัญและแม่นยำในเรื่องของทฤษฎีระบบตัวเลขนี้

3.3 ระบบตัวเลขฐานสอง

ในระบบตัวเลขฐานสองนี้มีตัวเลขเพียง 2 ตัว คือ “0” และ “1” ซึ่งสามารถใช้แทนสถานะต่ำ-สูง, เปิด-ปิด, ไม่ต่อ-ต่อ, ดับ-ติด เป็นต้น แต่ถ้าหากนำตัวเลขฐานสองมากกว่า 1 หลักมาพิจารณา เช่น 2 หลัก จะทำให้เกิดจำนวนของการเปลี่ยนแปลงที่ไม่ซ้ำกัน 4 ค่า หากแทนด้วยการติด-ดับของหลอดไฟ จะได้ ดับ-ดับ, ดับ-ติด, ติด-ดับ และ ติด-ติด ถ้ามี 3 หลักก็จะเกิดการเปลี่ยนแปลง 8 สถานะ จึงสามารถสรุปเป็นสมการคณิตศาสตร์และความสัมพันธ์ของจำนวนหลักและสถานะของการเปลี่ยนแปลงได้ดังนี้

$$\text{จำนวนของการเปลี่ยนแปลง} = 2^{\text{จำนวนหลัก}}$$

ถ้ามี 2 หลักจะได้จำนวนของการเปลี่ยนแปลง $2^2 = 4$

ถ้ามี 3 หลักจะได้จำนวนของการเปลี่ยนแปลง $2^3 = 8$

ถ้ามี 4 หลักจะได้จำนวนของการเปลี่ยนแปลง $2^4 = 16$

3.4 การนับจำนวนของระบบเลขฐานสอง

เนื่องจากเลขฐานสองมีจำนวนตัวเลขเพียง 2 ตัวคือ 0 และ 1 เมื่อมีการนับจำนวนเกิดขึ้น จึงต้องมีการเพิ่มจำนวนหลักขึ้น เพื่อให้เห็นการเปลี่ยนแปลงอย่างชัดเจนจะใช้เลขฐานสิบเป็นตัวเปรียบเทียบดังนี้

เลขฐานสอง	เลขฐานสิบ
00	0
01	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10

3.5 ตัวแปรของเลขฐานสอง (bit variables)

เลขฐานสองถูกนำมาใช้งานมากขึ้นจาก 1 หลัก เป็น 2, 3 จนถึง 8 หลัก ทำให้เกิดตัวแปรแบบใหม่ ๆ ขึ้น ดังนี้ คือ

(1) **บิต(bit)** หมายถึง หนึ่งหลักของเลขฐานสอง(binary digit) จะมีเพียงเลข 0 กับ 1

(2) **นิบเบิล (nibble)** หมายถึง ตัวเลขฐานสองจำนวน 4 หลักหรือเท่ากับ 4 บิต ทำให้เกิดจำนวนตัวเลขที่แตกต่างกัน 16 ค่า คือ 0000 ถึง 1111 ในระบบเลขฐานสอง หรือ 0-15 ในระบบเลขฐานสิบ

(3) **ไบต์ (byte)** หมายถึง ตัวเลขฐานสองจำนวน 8 หลักหรือเท่ากับ 8 บิต ไบต์มีความ เพราะในไมโครคอนโทรลเลอร์มักจะประมวลข้อมูลตัวเลขฐานสองทีละ 8 บิตหรือ 1 ไบต์ ใน 1 ไบต์จะเกิดจำนวนตัวเลขที่แตกต่างกัน 256 ค่าคือ 00000000-11111111 ในระบบเลขฐานสอง หรือ 0-255 ในระบบเลขฐานสิบ

(4) **เวิร์ด (word)** เป็นหน่วยของกลุ่มข้อมูลเลขฐานสองขนาด 16 บิต หรือ 2 ไบท์ทำให้มีจำนวนเลขมีค่าแตกต่างกัน 2^{16} ค่า หรือ 65,536 ค่าในฐานสิบ คือมีจำนวนเลขตั้งแต่ 0-65,535 ในการคำนวณทางคณิตศาสตร์ของเบสิกแสตมป์ 2SX จะอ้างอิงถึงจำนวนข้อมูลในระดับเวิร์ดเป็นหลัก

(5) **LSB : Least Significant Bit** หรือบิตนัยสำคัญต่ำสุด หมายถึง บิตที่อยู่ในตำแหน่งขวาสุดของเลขฐานสอง ซึ่งมีค่าน้ำหนักประจำหลักต่ำสุด คือ 20 ถ้าหากบิตสุดท้ายเป็น “1” ค่าของหลักสูตรท้ายจะเท่ากับ $1X\ 2^0 = 1\ X\ 1 = 1$ แต่ถ้าบิตสุดท้ายนี้เป็น “0” ค่าของหลักสุดท้ายจะเท่ากับ $0\ X\ 2^0 = 0\ X\ 1 = 0$

(6) การกำหนดชื่อหลักของเลขฐานสอง บิตที่อยู่ขวามือสุดจะถูกเรียกว่า บิตศูนย์ (bit 0 : b_0) หรือ บิต LSB บิตถัดมาเรียกว่า บิตหนึ่ง (bit 1 : b_1) ไล่ไปทางซ้ายเรื่อย ๆ สามารถสรุปชื่อหลักทั้ง 8 บิตของเลขฐานสองได้ดังนี้

b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0 โดยตัวเลขแสดงตำแหน่ง 0-7 ต้องเขียนเป็นตัวห้อยเสมอ และเพื่อความสะดวกในการเขียน จึงขอเขียนเลขแสดงหลักในระดับเดียวกันเป็น b7 b6 b5 b4 b3 b2 b1 b0

(7) **MSB : Most Significant Bit** หรือบิตนัยสำคัญสูงสุด หมายถึงบิตที่อยู่ในตำแหน่งซ้ายมือสุดของเลขฐานสองจำนวนนั้น ๆ หากเลขฐานสองมีจำนวน 8 บิต บิต MSB คือบิต 7 (bit 7 : b_7) มีค่าน้ำหนักประจำหลักเท่ากับ 2^7 หรือ 126 แต่ถ้าหากจำนวนบิตมีน้อยกว่านั้น เช่น 6 บิต, 5 บิต หรือ 4 บิต บิต MSB จะมีค่าน้ำหนักประจำหลักเปลี่ยนเป็น 2^5, 2^4 และ 2^3 ตามลำดับ

3.6 ค่าน้ำหนักประจำหลักและการแปลงเลขฐาน

เลขฐานสิบ มีค่าน้ำหนักประจำหลัก โดยคิดจากจำนวนสิบยกกำลัง โดยในหลักหน่วยมีค่าน้ำหนักประจำหลักเป็น 10^0 หรือ 1 หลักสิบมีค่าน้ำหนักประจำหลักเป็น 10^1 หรือ 10 ในหลักร้อยมีค่าน้ำหนักประจำหลักเป็น 10^2 หรือ 100 เป็นต้น

เลขฐานสอง มีค่าน้ำหนักประจำหลักโดย คิดจากจำนวนสองยกกำลัง โดยในหลักขวาสุดคือ บิต 0 หรือบิต LSB มีค่าน้ำหนักประจำหลักเป็น 2^0 หรือเท่ากับ 1 หลักถัดมาคือบิต 1 มีค่าน้ำหนักเป็น 2^1 หรือ 2 ถัดมาเป็นบิต 2 มีค่าน้ำหนักเป็น 2^2 หรือ 4 เมื่อพิจารณาที่เลขฐานสอง 8 บิต

บิต	ค่าน้ำหนักประจำหลัก	เลขฐานสิบ
0	2^0	1
1	2^1	2
2	2^2	4
3	2^3	8
4	2^4	16
5	2^5	32
6	2^6	64
7	2^7	128

จากค่าน้ำหนักประจำหลักจึงสามารถแปลงเลขฐานสองเป็นฐานสิบ หรือแปลงฐานสิบเป็นฐานสองได้ โดยจะเริ่มต้นที่การ แปลงเลขฐานสองเป็นฐานสิบก่อน

ตัวอย่าง จงแปลงเลขฐานสอง 1011 เป็นฐานสิบ

(1) กำหนดค่าน้ำหนักประจำหลัก

หลัก	b^3	b^2	b^1	b^0
ค่าน้ำหนักประจำหลักคือ	2^3	2^2	2^1	2^0
เลขฐานสอง	1	0	1	1

(2) นำค่าน้ำหนักประจำหลักคูณกับค่าของเลขฐานสองในบิตนั้น แล้วนำผลคูณของทุกหลักมารวมกัน

เลขฐานสิบ $= (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$

= (1 X 8) + (0 X 4) + (1 X 2) + (1 X 1)

= 8+0+2+1 = 11

3.7 การแปลงเลขฐานสิบเป็นเลขฐานสอง

ใช้วิธีการหารเลขฐานสิบจำนวนนั้นด้วย 2 แล้วเก็บค่าของเศษที่ได้จากการหารเป็นเลขฐานสองในแต่ละหลัก เศษที่ได้จากการหารครั้งแรกอาจจะเป็น “0” หรือ “1” จะเป็นหลักที่มีนัยสำคัญต่ำสุดหรือบิต LSB หรือบิต 0 (b0) เศษตัวสุดท้ายจะเป็นเลขฐานสองหลักที่มีนัยสำคัญสูงสุดหรือบิต MSB

ตัวอย่าง จงแปลงเลขฐานสิบ 13 เป็นเลขฐานสอง

(1) หาร 13 ด้วย 2 ได้ 6 เศษ 1 เศษที่ได้จะเป็นบิตศูนย์หรือบิต LSB นั่นคือบิต LSB = 1

(2) หาร 6 ด้วย 2 ได้ 3 เศษ 0 เศษที่ได้จะเป็นบิตหนึ่ง ซึ่งก็คือ 0

(3) หาร 3 ด้วย 2 ได้ 1 เศษ 1 เศษที่ได้จะเป็นบิตสอง ซึ่งก็คือ 1

(4) หาร 1 ด้วย 2 ได้ 0 เศษ 1 เศษที่ได้จะเป็นบิตสามและเป็นบิต MSB ซึ่งก็คือ 1

ดังนั้นจะได้เลขฐานสองเท่ากับ 1101

3.8 เครื่องหมายของเลขฐานสอง

เลขฐานสอง จะมีทั้งค่าตัวเลขที่เป็นบวกและลบเช่นเดียวกับเลขฐานอื่น ๆ โดยจะใช้บิต MSB เป็นตัวกำหนดเครื่องหมายของเลขฐานสอง ถ้ากำหนดบิต MSB เป็น “0” เลขจำนวนนั้นจะมีค่าเป็นบวก และหากกำหนดบิต MSB เป็น “1” เลขจำนวนนั้นจะมีค่าเป็นลบ

จากตารางเป็นการแสดงค่าของจำนวนเลขฐานสอง เมื่อคิดเครื่องหมายและไม่คิดเครื่องหมาย โดยได้ทำการแปลงเป็นเลขฐานสิบเปรียบเทียบเพื่อให้เห็นความแตกต่างอย่างชัดเจน

เลขฐานสอง	**เลขฐานสิบ**	
	คิดเครื่องหมาย	ไม่คิดเครื่องหมาย
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	-8	8
1001	-7	9
1010	-6	10
1011	-5	11

1100	-4	12
1101	-3	13
1110	-2	14
1111	-1	15

กรณีคิดเครื่องหมาย เมื่อนับถอยหลังจาก 0000 ก็จะเป็น 1111 นั่นคือเกิดการถอยหลังหนึ่งจำนวน หรือ –1 นับถอยหลังต่อไปจะเป็น 1110 ซึ่งก็คือ –2 เมื่อเป็นเช่นนี้การแปลงเลขฐานสองที่คิดเครื่องหมายเป็นฐานสิบจึงไม่สามารถใช้วิธีการแปลงแบบเดิมได้ แต่ก็พอมีเทคนิคในการพิจารณา โดยใช้หลักเกณฑ์ค่าน้ำหนักประจำหลัก ยกตัวอย่าง เลขฐานสอง 1000 เลข 1 ที่อยู่หน้าสุด มีค่าน้ำหนักประจำหลักเท่ากับ 2^3 หรือ 8 จากการกำหนดว่า ถ้าคิดเครื่องหมาย เมื่อบิต MSB เป็น “1” จะต้องเป็นค่าลบ ดังนั้นจึงเป็น –8 ส่วนอีก 3 หลักที่เหลือจะเป็นเลขบวกจึงกลายเป็น –8+0 = -8 มาพิจารณาที่เลขฐานสอง 1101 บิตแรกเป็นลบเท่ากับ –8 ส่วน 3 บิตหลังเป็นบวกมีค่า +5 จึงได้ –8+5 = -3 เป็นต้น

3.9 การบวกเลขฐานสอง

การบวกเลขฐานสอง 1 หลัก มีกฎการบวกดังนี้

(i) 0+0=0

(ii) 0+1=1

(iii) 1+0=0

(iv) 1+1=0 ตัวทด =1

ถ้าเป็นการบวกเลขฐานสอง 2 หลัก มีตัวอย่างผลการบวกดังนี้

(i)

1 0	1 1	1 1
0 1	0 1	1 1
1 1	1 0 0	1 1 0

ในกรณีที่บวกเลขฐานสองโดยคิดตัวทด จะมีรูปแบบการบวกดังนี้

ตัวทดจากหลักก่อน		ตัวตั้ง		ตัวบวก		ผลบวก	ตัวทด
0	+	0	+	0	=	0	0
1	+	0	+	0	=	1	0
0	+	1	+	0	=	1	0
1	+	1	+	0	=	0	1
0	+	1	+	1	=	0	1
1	+	1	+	1	=	1	1

3.10 การลบเลขฐานสอง

มีกฎการลบ 4 กฎดังนี้

(i) 0-0 = 0

(ii) 1-0 = 1

(iii) 1-1 = 0

(iv) 10-1 = 1

ในกรณี (iv) เกิดขึ้นเนื่องจากมีการยืมหลักต่อไป ซึ่งสามารถอธิบายได้เห็นชัดเจนขึ้นด้วยตัวอย่างการลบเลขฐานสอง 3 หลัก 110-101 ดังนี้

หลัก	**b0**	**b1**	**b2**
ตัวตั้ง	1	1	0
ตัวลบ	1	0	1
ผลลบ	0	0	1

ที่หลัก b0 0-1 ไม่ได้จึงต้องยืมหลักถัดไปนั่นคือ b1 ถ้าเป็นการยืมตัวของเลขฐานสิบจะยืมมาเท่ากับสิบบวกเข้ากับค่าของหลักที่ยืม ในกรณีเลขฐานสอง ค่าของการยืมเท่ากับ $2_{10} - 1_{10} = 1_{10} = 1_2$ (ตัวเลขที่ห้อยหมายถึง ฐานของเลขจำนวนนั้น เช่น 2_{10} หมายถึง ค่า 2 ของเลขฐานสิบ เป็นต้น)

เมื่อหลัก b1 ถูก b0 ยืมไป ทำให้ค่าของหลัก b1 เท่ากับ 0 ที่หลัก b1 เกิดการลบเป็น 0-0 = 0

ส่วนที่หลัก b2 เกิดการลบเป็น 1-1 = 0

ดังนั้นผลลัพธ์สุดท้ายจึงเป็น 001

การลบเลขฐานสองด้วยวิธีนี้จะไม่ยุ่งยาก หากมีจำนวนหลักหรือจำนวนบิตน้อยและค่าของตัวตั้งมากกว่าตัวลบ ในกรณีที่มีจำนวนหลักหรือจำนวนบิตมาก ๆ หรือค่าของตัวตั้งน้อยกว่าตัวลบ จะทำให้ผลของการลบเป็นจำนวนติดลบ หากใช้วิธีการลบตรง ๆ จะยุ่งยากและอาจเกิดความผิดพลาดได้จึงใช้วิธีการบวกด้วยค่าลบแทนวิธีการลบตรง ๆ

ตัวอย่าง $2_{10} - 4_{10} - = -2_{10}$ ถ้าใช้วิธีการบวกด้วยค่าลบจะเป็น $2_{10} + (-4_{10}) = -2_{10}$ วิธีการแบบนี้มีชื่ออย่างเป็นทางการว่า **วิธีการคอมพลีเมนต์ (Complement)** ซึ่งจะมีด้วยกัน 2 ขั้นตอนคือ วันคอมพลีเมนต์ (One's Complement) และทูคอมพลีเมนต์ (Two's Complement)

3.11 การลบด้วยวิธีการคอมพลีเมนต์

จะต้องใช้การคิดเครื่องหมายเลขฐานสอง (Signal Number) มาช่วย สามารถเปรียบเทียบการลบด้วยกระบวนการคอมพลีเมนต์ของเลขฐานสิบและฐานสองได้ดังนี้

(i) ในระบบเลขฐานสิบ 2 + (-4) = -2

ในระบบเลขฐานสอง 0010 + 1100 = 1110

(ii) ในระบบเลขฐานสิบ 127 + (-125) = 2

ในระบบเลขฐานสอง 01111111 + 10000011 = 1000010

วิธีการวันคอมพลีเมนต์คือ การแทนที่เลข 0 ในแต่ละหลักของตัวลบด้วย "1" และแทนที่เลข "1" ในแต่ละหลักของตัวลบด้วย "0" เช่น 0100 เมื่อทำวันคอมพลีเมนต์จะเป็น 1011

วิธีการทูคอมพลีเมนต์คือ นำค่าของตัวเลขฐานสองที่ได้จากการทำวันคอมพลีเมนต์บวกด้วย “1” ที่หลัก LSB หรือ b0 ยกตัวอย่างจากค่า 0100 เมื่อทำวันคอมพลีเมนต์เป็น 1011 ทำทูคอมพลีเมนต์ได้ 1100 ซึ่งก็คือ –4 นั่นเอง

3.12 การคูณเลขฐานสอง

กฎการคูณเลขฐานสองมีดังนี้ คือ

(i) 0 X 0 = 0

(ii) 0 X 1 = 0

(iii) 1 X 0 = 0

(iv) 1 X 1 = 1

ตัวอย่าง จงหาผลคูณ 1001 X 1011

```
      1 0 0 1
  x   1 0 1 1
-------------
      1 0 0 1
    1 0 0 1
  0 0 0 0
+ 1 0 0 1
-------------
  1 1 0 0 0 1 1
=============
```

3.13 เลขฐานสิบหก (Hexadecimal Numbers)

ระบบตัวเลขอีกฐานหนึ่งที่มีบทบาทในปัจจุบันนั่นคือ เลขฐานสิบหก ซึ่งจะมีตัวเลขด้วยกัน 16 ตัว เริ่มตั้งแต่ 0 ถึง F สัญลักษณ์นำหน้าตัวเลขแสดงฐานในกรณีใช้กับเบสิกแสตมป์ 2SX ใช้ตัวอักษร $ สำหรับเลขฐานสิบหก ในขณะที่เลขฐานสองใช้ % และฐานสิบไม่มีตัวเลขใด ๆ นำหน้าและต่อท้าย

ต่อไปนี้จะเป็นการแสดงความสัมพันธ์ของเลขฐานสอง, ฐานสิบ และฐานสิบหก (กรณีไม่คิดเครื่องหมาย)

เลขฐานสอง	เลขฐานสิบ	เลขฐานสิบหก
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6

0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F
10000	16	10

3.14 การแปลงเลขฐานสองเป็นฐานสิบหก

เลขฐานสิบหก 1 หลัก เมื่อแทนด้วยเลขฐานสองจะต้องใช้ 4 หลัก ดังนั้น ในการแปลงฐานจะต้องจัดกลุ่มเลขฐานสองเป็นกลุ่มละ 4 หลัก ดังตัวอย่าง จงแปลงเลขฐานสอง 101011100011 เป็นเลขฐานสิบหก

เลขฐานสอง 101011100011B

จัดกลุ่มละ 4 หลัก 1010 1110 0011

แปลงเป็นเลขฐานสิบหก A B C

ผลลัพธ์เท่ากับ $AE3

ส่วนการแปลงเลขฐานสิบหกเป็นฐานสองก็ใช้หลักการเดียวกัน โดยแบ่งเลขฐานสิบหกแต่ละหลักออกจากกัน แล้วจึงแปลงเป็นเลขฐานสองทีละหลัก ดังตัวอย่าง

เลขฐานสิบหก BC75H

แยกแต่ละหลักออกจากกัน B C 7 5

แปลงเป็นเลขฐานสอง 1011 1100 0111 0101

ผลลัพธ์เท่ากับ 1011110001110101

3.15 การแปลงเลขฐานสิบเป็นฐานสิบหก

วิธีการที่ง่ายที่สุดคือ แปลงเลขฐานสิบเป็นฐานสองก่อน จากนั้นจึงจัดกลุ่มของเลขฐานสองแล้วแปลงเป็นเลขฐานสิบหก ดังตัวอย่าง

เลขฐานสิบ 302

แปลงเป็นเลขฐานสอง 100101110B

แบ่งกลุ่มละ 4 หลักจากท้าย 1 0010 1110

แปลงเป็นเลขฐานสิบหก 1 2 E

ผลลัพธ์เท่ากับ $12E

3.16 การแปลงเลขฐานสิบหกเป็นฐานสิบ

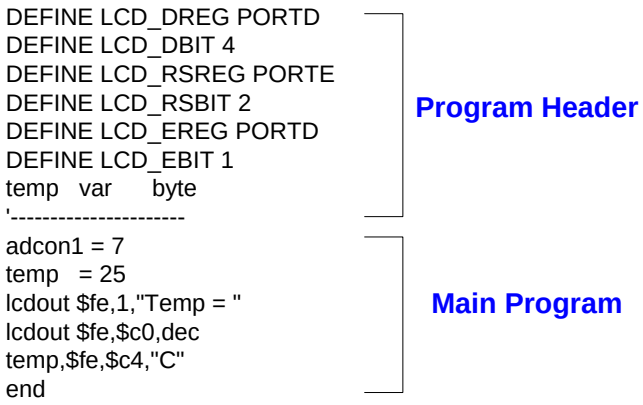
การแบ่งเลขฐานสิหกแต่ละหลักออกจากกัน แล้วแปลงเป็นฐานสิ บ จากนั้นคูณด้วยค่าน้ำหนักประจำหลัก โดยในหลักซ้ายสุดจะมีค่าน้ำหนักประจำหลักเท่ากับ 16^0 ซึ่งก็คือ 1 หลักถัดมาทางขวามือจะมีค่าน้ำหนักประจำหลักเท่ากับ 16^1 หรือ 16 หลักถัดมามีค่าน้ำหนักประจำหลักเท่ากับ 16^3 หรือ 256 ไล่เรียงเช่นนี้จนครบทุกหลัก จากนั้นนำผลคูณในแต่ละหลักมารวมกัน ก็จะได้ค่าของเลขฐานสิบในที่สุด ดังตัวอย่าง

เลขฐานสิบหก	A	E	3
แปลงเป็นเลขฐานสิบในแต่ละหลัก	10	14	3
ค่าน้ำหนักประจำหลัก	$256(16^2)$	$16(16^1)$	$1(16^0)$
ผลคูณ	2560	224	3

รวมผลคูณเท่ากับ 2560+224+3 = 2787

3.17 รูปแบบการเขียนโปรแกรมภาษา PIC BASIC PRO COMPILER

การเขียนโปรแกรมภาษา PIC BASIC PRO COMPILER แทบไม่แตกต่างกันกับโปรแกรม Quick BASIC หรือ Turbo BASIC หรือแม้กระทั่ง Visual BASIC ที่ใช้กันปัจจุบัน จะแตกต่างกันที่มีคำสั่งที่เกี่ยวกับการทำงานของไมโครคอนโทรลเลอร์เพิ่มเติมเข้ามา ถ้าเราทำความเข้าใจคำสั่งที่เพิ่มเข้ามานี้ดีแล้ว จะทำให้การใช้งานเขียนโปรแกรมง่ายขึ้นมาก องค์ประกอบของโปรแกรมประกอบด้วยส่วนสำคัญใหญ่ ๆ ได้แก่ ส่วนหัวของโปรแกรม (Header) เป็นส่วนที่ต้องกำหนดการผนวกไฟโปรแกรมที่ต้องทำงานประมวลผลร่วมกัน หากมีการกำหนดค่านิยาม (DEFINE) หากต้องใช้ การกำหนดค่าตัวแปร (Variables) ที่ต้องใช้ และค่าเริ่มต้นการทำงาน ส่วนตัวโปรแกรมหลัก (Main Program) เป็นส่วนที่เริ่มการประมวลผลคำสั่ง และต้องจบปิดท้ายด้วยคำสั่ง END เสมอ และสุดท้ายคือ ส่วนโปรแกรมย่อย (Subroutine) เป็นส่วนที่โปรแกรมหลักเรียกใช้จึงจะได้ทำ เมื่อทำเสร็จก็จะต้องส่งกลับคืนไปยังโปรแกรมหลักด้วยคำสั่ง RETURN เสมอ โปรแกรมย่อยเหล่านี้จะต้องวางต่อท้ายจากคำสั่ง END ของโปรแกรมหลักเป็นต้นไป ตามตัวอย่าง รูปที่ 40 และ 41



รูปที่ 40 แสดงส่วนประกอบของโปรแกรม PIC BASIC PRO COMPILER

```
DEFINE LCD_DREG PORTD
DEFINE LCD_DBIT 4
DEFINE LCD_RSREG PORTE
DEFINE LCD_RSBIT 2
DEFINE LCD_EREG PORTD
DEFINE LCD_EBIT 1
S1            VAR   PORTA.3
S2            VAR   PORTA.4
spk       var portb.5
heater  var portb.4
temp    var byte
      TRISA     =    %111111
      ADCON1 = 7
      temp   = 25
      lcdout $fe,1,"Temp = "
      lcdout $fe,$c0,dec temp,$fe,$c4,"C"
LOOP:   IF  (S1 =  0) AND (S2 = 1)  THEN
        pause 50
        temp  =  temp+1
        gosub click
        gosub display
        gosub action
        if temp = 80 then temp = 79
      idle1: if (S1 =  0) AND (S2 = 1)  THEN idle1
      endif
                    IF  (S1 =  1) AND (S2 = 0)
THEN
                      pause 50
        if temp = 20  then  temp=21
        temp  =  temp - 1
        gosub click
        gosub display
        gosub action
      idle2: if (S1 =  1) AND (S2 = 0)  THEN idle2
      endif
                    PAUSE  50
                    GOTO     LOOP
      END
'----------------- End of Main Program -----------
'
'------- Subroutine  Start  Here ----------------
click:
      freqout spk,5,2000
      return

display:
      lcdout $fe,$c0,dec temp
      return

action:
      if temp > 25 then
         low  heater
      else
         high  heater
      endif
      return
'
'------- End of Subroutine -----------------------
```

Program Header

Main Program

Subroutines

รูปที่ 41 แสดงรูปแบบการเขียนโปรแกรมที่มีโปรแกรมหลักและโปรแกรมย่อย