# บทที่ 6
# คำสั่งควบคุมการทำงานของโปรแกรม C51

## 6.1 บทนำ

ในบทที่ผ่านมาได้ให้ความสนใจกับตัวดำเนินการซึ่งอยู่ในรูปของนิพจน์ต่างๆ สำหรับในบทนี้ได้อธิบายการใช้คำสั่งควบคุมเพื่อควบคุมการทำงานของโปรแกรม คำสั่งควบคุมทำหน้าที่ในการกำหนดโปรแกรมให้ทำงานตามเงื่อนไขที่ต้องการ ดังนั้นคำสั่งควบคุมมีความสำคัญมากในการพัฒนาโปรแกรม โดยเฉพาะในการพัฒนางานด้านไมโครคอนโทรลเลอร์ คำสั่งควบคุมการทำงานของโปรแกรม C51 ส่วนใหญ่จะยึดให้ค่าตรงตามมาตรฐาน ANSI-C แต่ก็มีบางส่วนที่แตกต่างกันบ้างทั้งนี้เพื่อให้เหมาะสมกับทรัพยากรของไมโครคอนโทรลเลอร์ โดยคำสั่งควบคุมการทำงานที่สำคัญประกอบด้วยคำสั่งกำหนดเงื่อนไขและคำสั่งวนซ้ำ ได้แก่ คำสั่ง if if/else, switch, while, do-while และคำสั่ง for โดยเนื้อหาในบทนี้จะอาศัยผังงาน (Flowchart) เป็นรูปแบบหลักในการอธิบายคำสั่งควบคุมการทำงานของโปรแกรม ซึ่งจะได้อธิบายการประยุกต์ใช้ผังงานในการเขียนขั้นตอนวิธีในหัวข้อต่อไป

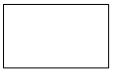
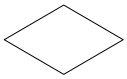
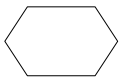
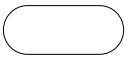
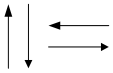
## 6.2 ผังงาน

ในการพัฒนางานด้านไมโครคอนโทรลเลอร์นั้น เครื่องมือที่นิยมใช้ในการเขียนขั้นตอนวิธี (Algorithm) เพื่อวางแผนการแก้ปัญหาได้แก่ผังงาน ผังงานก็คือขั้นตอนวิธีที่เขียนโดยใช้รูปสัญลักษณ์ที่มีเส้นเชื่อมและหัวลูกศรบอกขั้นตอนการทำงานเพื่อกำกับทิศทางการทำงาน ช่วยให้เข้าใจง่ายขึ้นและสามารถตรวจสอบความถูกต้องได้ง่าย โดยผังงานมีข้อดีดังนี้

1. ผังงานช่วยให้นักพัฒนาเข้าใจและแยกแยะปัญหาได้ง่ายขึ้น
2. ลำดับการทำงานของผังงานแสดงอย่างเป็นระบบ ทำให้สามารถหาข้อผิดพลาดของโปรแกรมและแก้ไขได้ง่าย
3. นักพัฒนาเข้าใจผังงานได้ง่ายกว่าดูจากรหัสคำสั่งของโปรแกรม
4. ผังงานไม่ได้อ้างอิงการเขียนโปรแกรมกับภาษาใดภาษาหนึ่ง ทำให้สามารถเรียนรู้และเข้าใจง่ายโดยไม่ต้องมีความรู้ด้านการเขียนโปรแกรมมาก่อน

สัญลักษณ์ผังงานสำหรับเขียนโปรแกรมที่ใช้งานหลักๆ แสดงดังตารางที่ 6.1

ตารางที่ 6.1 สัญลักษณ์ผังงาน

| ภาพสัญลักษณ์ | ความหมาย |
|---|---|
|  | แทนการประมวลผลทั่วไป (Process) |
|  | แทนการนำข้อมูลเข้า (Input) และการนำผลลัพธ์ออก (Output) |
|  | แทนกิจกรรมการตัดสินใจ (Decision) โดยเปรียบเทียบข้อมูลกับเงื่อนไข มีเส้นทางออกเป็นทางเลือก |
|  | แทนกิจกรรมเตรียมการ (Preparation) หรือกำหนดคุณลักษณะของการทำงานวนซ้ำ |
|  | แทนจุดเริ่มต้นและจุดสิ้นสุดของงาน (Terminator) |
|  | แทนทิศทางการไหลของข้อมูลหรือทิศทางการเดินของโปรแกรม (Direction) โดยหัวลูกศรเป็นตัวกำหนดทิศทาง |

ที่มา: สานนท์ เจริญฉาย, 2546

สำหรับแนวทางการเขียนขั้นตอนวิธีในแบบผังงานแสดงได้ดังตัวอย่างที่ 6.1

**ตัวอย่างที่ 6.1** ต้องการเขียนโปรแกรมเพื่อคำนวณหาพื้นที่สามเหลี่ยม

สามารถอธิบายขั้นตอนการทำงานดังนี้

*ขั้นตอนที่ 1* เริ่มต้น

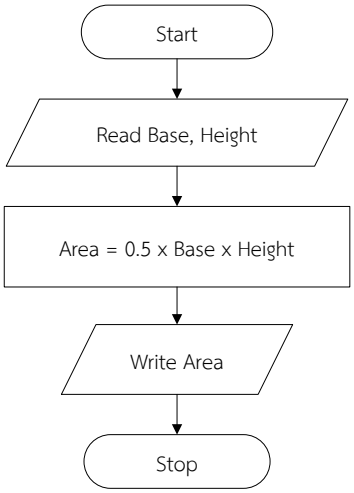
*ขั้นตอนที่ 2* รับค่า Base, Height

*ขั้นตอนที่ 3* คำนวณหาพื้นที่สามเหลี่ยม Area = 0.5 x Base x Height

*ขั้นตอนที่ 4* แสดงผลลัพธ์ Area

*ขั้นตอนที่ 5* จบการทำงาน

จากขั้นตอนการทำงานดังกล่าว สามารถนำมาเขียนขั้นตอนวิธีในการเขียนผังงานแสดงดังภาพที่ 6.1 จะเห็นว่าผังงานจะทำงานแบบเรียงลำดับจากบนลงล่าง ซึ่งจะมีคุณสมบัติการจัดลำดับการทำงานเหมือนการประมวลผลคอมพิวเตอร์ ซึ่งจะทำงานตามคำสั่งที่อยู่บรรทัดบนสุดก่อน แล้วจึงทำงานตามคำสั่งในบรรทัดต่อมาจนเสร็จสิ้นคำสั่งในบรรทัดสุดท้าย




ภาพที่ 6.1 ตัวอย่างผังงานเพื่อคำนวณหาพื้นที่สามเหลี่ยม

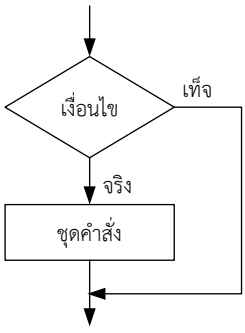
## 6.3 คำสั่งกำหนดเงื่อนไข

คำสั่งกำหนดเงื่อนไขทำหน้าที่ควบคุมทางเลือกการทำงานของโปรแกรม โดยจะเป็นไปในรูปแบบมีทางเลือกอย่างใดอย่างหนึ่งในการทำงานขึ้นอยู่กับเงื่อนไขที่ได้กำหนดไว้ มีรายละเอียดดังนี้

### 6.3.1 การเปรียบเทียบด้วยคำสั่ง if

เป็นคำสั่งที่ใช้ในการตรวจสอบเงื่อนไข (condition) ว่าเป็นจริงหรือเท็จ มีรูปแบบและผังงานแสดงในภาพที่ 6.2

```
if (<conditional expression>)
<statement-list (s)>;
```




ภาพที่ 6.2 รูปแบบและผังงานของคำสั่ง if

ที่มา: นวลนดา สงวนวงษ์ทอง, 2557

คำสั่ง if มีการทำงานตามลำดับ ดังนี้

1. ตัวแปรโปรแกรมจะทำการทดสอบนิพจน์เงื่อนไข (Condition expression) ซึ่งจะได้ผลลัพธ์เป็นจริงหรือเท็จ

2. หากผลลัพธ์เป็นจริง โปรแกรมจะทำงานในประโยคคำสั่งหรือกลุ่มของประโยคคำสั่งต่อมาของ if (statement-list(s)) ซึ่งจะอยู่ภายในช่วงวงเล็บปีกกาซึ่งอาจมี 1 คำสั่งหรือหลายคำสั่งก็ได้ แต่ถ้าคำสั่งภายในมีเพียงคำสั่งเดียวอาจไม่ต้องใส่เครื่องหมายปีกกาก็ได้

3. หากผลลัพธ์เป็นเท็จ โปรแกรมจะไม่สนใจในประโยคคำสั่งหรือกลุ่มของประโยคคำสั่งต่อมาของ if แต่จะไปทำงานในคำสั่งต่อไปจากกลุ่มคำสั่ง if

ที่ตำแหน่งนิพจน์เงื่อนไขนอกจะใช้เป็นคำสั่งเปรียบเทียบด้วยเครื่องหมายทางคณิตศาสตร์และลอจิกแล้ว ยังสามารถใช้ในการตรวจสอบค่าคงที่หรือค่าของตัวแปรใดๆ ที่นำมาเป็นเงื่อนไขได้ดังนี้

1) ถ้าค่าตัวเลขหรือตัวแปรเงื่อนไขดังกล่าวมีค่าไม่เท่ากับ 0 เงื่อนไขดังกล่าวจะเป็นจริงจะมีการกระทำประโยคคำสั่งหรือกลุ่มของประโยคคำสั่งต่อมาของคำสั่ง if นั้นๆ

2) ถ้าค่าเงื่อนไขดังกล่าวมีค่าเท่ากับ 0 จะถือว่าเงื่อนไขดังกล่าวเป็นเท็จ จะไม่มีการกระทำคำสั่งภายในประโยคคำสั่งของคำสั่ง if นั้นๆ

การทำงานของคำสั่ง if แสดงดังตัวอย่างที่ 6.2-6.3

**ตัวอย่างที่ 6.2** โปรแกรมการทำงานของคำสั่ง if

```
 1 #include <REGX52.H>
 2 void main (void)
 3 {
 4 unsigned char a, b;
 5 a = 5;
 6 b = 3;
 7   if(a >= 5)
 8   {
 9   a = a+10;
10   b = 7;
11   }
12 }
```

**ผลลัพธ์ของการรันโปรแกรม**

| Name | Location/Value |
|---|---|
| MAIN | C:0x0800 |
| a | 15 |
| b | 7 |

*บรรทัดที่ 5* ตัวแปร a = 5

*บรรทัดที่ 6* ตัวแปร b = 3

บรรทัดที่ 7-11 คำสั่ง if เพื่อตรวจสอบเงื่อนไขว่า a มากกว่าหรือเท่ากับ 5 หรือไม่ จาก โปรแกรมเงื่อนไขเป็นจริงให้กระทำกลุ่มของประโยคคำสั่งต่อมา

**ตัวอย่างที่ 6.3** โปรแกรมการทำงานของคำสั่ง if

```
1  #include <REGX52.H>
2  void main (void)
3  {
4  unsigned char a;
5  bit c;
6  a = 8;
7  c = 0;
8    if(5)
9    {
10   a++;
11   c = ~c;
12   }
13   if(-3)
14   {
15   a++;
16   }
17   if(0)
18   {
19   a = a + 10;
20   c = ~c;
21   }
22 }
```

**ผลลัพธ์ของการรันโปรแกรม**

| Name | Location/Value |
|---|---|
| MAIN | C:0x0800 |
| a | 10 |
| c | 1 |

*บรรทัดที่ 5* ตัวแปร a = 8

*บรรทัดที่ 6* ตัวแปร c = 0

*บรรทัดที่ 8-12* คำสั่ง if เพื่อตรวจสอบเงื่อนไขว่า a มีค่าไม่เท่ากับ 0 หรือไม่ จาก โปรแกรมเงื่อนไขเป็นจริงให้กระทำกลุ่มของประโยคคำสั่งต่อมา

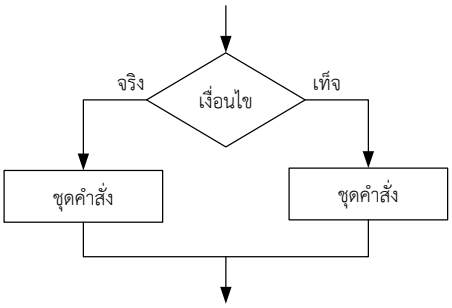
*บรรทัดที่ 13-16* คำสั่ง if เพื่อตรวจสอบเงื่อนไขว่า a มีค่าไม่เท่ากับ 0 หรือไม่ จาก โปรแกรมเงื่อนไขเป็นจริงให้กระทำกลุ่มของประโยคคำสั่งต่อมา

*บรรทัดที่ 17-21* คำสั่ง if เพื่อตรวจสอบเงื่อนไขว่า a มีค่าไม่เท่ากับ 0 หรือไม่ จาก โปรแกรมเงื่อนไขเป็นเท็จจึงไม่มีการกระทำคำสั่งภายในคำสั่ง if นี้

### 6.3.2 การเปรียบเทียบด้วยคำสั่ง if/else

เป็นคำสั่งที่ใช้ในการตรวจสอบเงื่อนไข โดยกำหนดให้ทำงานอย่างหนึ่งเมื่อเงื่อนไขเป็นจริง และทำงานอีกอย่างหนึ่งเมื่อเงื่อนไขเป็นเท็จ มีรูปแบบและผังงานแสดงในภาพที่ 6.3

```
if (<conditional expression>)
        <statement-list1 (s)>;
else
        <statement-list2 (s)>;
```




ภาพที่ 6.3 รูปแบบและผังงานของคำสั่ง if/else

ที่มา: นวลนดา สงวนวงษ์ทอง, 2557

การทำงานของคำสั่ง if/else จะทำตามลำดับ ดังนี้

1. ตัวแปรโปรแกรมจะทำการทดสอบนิพจน์เงื่อนไขซึ่งจะได้ผลลัพธ์เป็นจริงหรือเท็จ
2. หากผลลัพธ์เป็นจริง โปรแกรมจะทำงานในประโยคคำสั่งที่ 1 หรือกลุ่มของประโยคคำสั่งที่ 1 (statement-list1(s))
3. หากผลลัพธ์เป็นเท็จ โปรแกรมจะทำงานในประโยคคำสั่งที่ 2 หรือกลุ่มของประโยคคำสั่งที่ 2 (statement-list2 (s))

การทำงานของคำสั่ง if แสดงดังตัวอย่างที่ 6.4-6.5

**ตัวอย่างที่ 6.4** โปรแกรมการทำงานของคำสั่ง if

```
#include <REGX52.H>
void main (void)
{
unsigned char z = 10;
  if(z >= 5)
  {
  z += 15;
  z = z%10;
  }
  else
  {
  z += 17;
  z = z%4;
  }
}
```

**ผลลัพธ์ของการรันโปรแกรม**

| Name | Location/Value |
|---|---|
| MAIN | C:0x0800 |
| z | 5 |

*บรรทัดที่ 4* ตัวแปร z = 10

*บรรทัดที่ 5* คำสั่ง if เพื่อตรวจสอบเงื่อนไขว่า z มากกว่าหรือเท่ากับ 5 หรือไม่ เนื่องจากโปรแกรมเงื่อนไขเป็นจริงให้กระทำกลุ่มของประโยคคำสั่งต่อมา

*บรรทัดที่ 7* z = z +15 ผลลัพธ์ z = 25

*บรรทัดที่ 8* z = z%10 ผลลัพธ์ z = 5

*บรรทัดที่ 10-14* เนื่องจากโปรแกรมเงื่อนไขเป็นจริง จึงข้ามการกระทำกลุ่มของประโยคคำสั่ง else

**ตัวอย่างที่ 6.5** โปรแกรมการทำงานของคำสั่ง if

```
1  #include <REGX52.H>
2  void main (void)
3  {
4  unsigned char z = 10;
5    if(z < 5)
6    {
7    z += 15;
8    z = z%10;
9    }
10   else
11   {
12   z += 17;
13   z = z%4;
14   }
15 }
```

**ผลลัพธ์ของการรันโปรแกรม**

| Name | Location/Value |
|---|---|
| MAIN | C:0x0800 |
| z | 0x03 |

*บรรทัดที่ 4* ตัวแปร z = 10

*บรรทัดที่ 5* คำสั่ง if เพื่อตรวจสอบเงื่อนไขว่า z มากกว่าหรือเท่ากับ 5 หรือไม่ เนื่องจากโปรแกรมเงื่อนไขเป็นเท็จ ให้กระทำกลุ่มของประโยคคำสั่ง else

*บรรทัดที่ 6-9* เนื่องจากโปรแกรมเงื่อนไขเป็นเท็จ จึงข้ามการกระทำกลุ่มของประโยคนี้

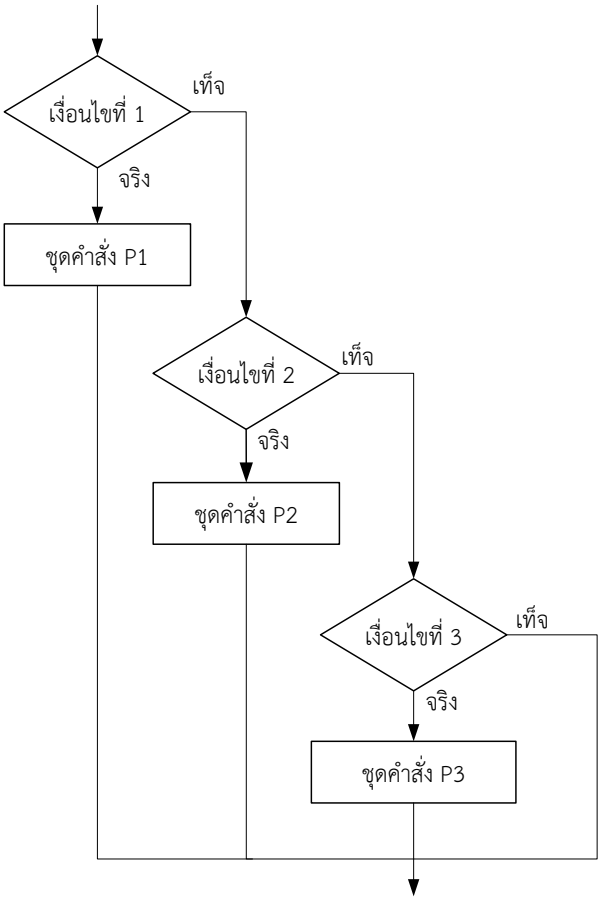
*บรรทัดที่ 12* z = z+17 ผลลัพธ์ z = 27

*บรรทัดที่ 13* z = z%4 ผลลัพธ์ z = 3

### 6.3.3 การเปรียบเทียบด้วยคำสั่ง if/else if

คำสั่งนี้เป็นคำสั่งที่ใช้ในการตรวจสอบเงื่อนไขหลายเงื่อนไขโดยเพิ่มการทำงานจากคำสั่ง if/else อย่างไรก็ตามในการทำงานถ้าทดสอบเงื่อนไขแรกเป็นเท็จ ก็จะทำการทดสอบเงื่อนไขที่สองต่อมา หากเงื่อนไขที่สองเป็นจริงก็จะทำงานในชุดคำสั่ง แต่หากเงื่อนไขที่สองเป็นเท็จก็จะทำการทดสอบเงื่อนไขต่อไปจนครบ คำสั่งนี้มีรูปแบบและผังงานแสดงในภาพที่ 6.4

```
if (<conditional expression>)
        <statement-list1 (s)>;
else if
        <statement-list2 (s)>;
else
        <statement-list3 (s)>;
```




ภาพที่ 6.4 รูปแบบและผังงานของคำสั่ง if/else if

ที่มา: นวลนดา สงวนวงษ์ทอง, 2557

การทำงานของคำสั่ง if/else if จะทำตามลำดับ ดังนี้

1. ตัวแปรโปรแกรมจะทำการทดสอบนิพจน์เงื่อนไขที่ 1 ซึ่งจะได้ผลลัพธ์เป็นจริงหรือเท็จ

2. หากผลลัพธ์เป็นจริง โปรแกรมจะทำงานในประโยคคำสั่งที่ 1 หรือกลุ่มของประโยคคำสั่งที่ 1 แล้วเสร็จสิ้นการทำงานของคำสั่ง

3. หากผลลัพธ์เป็นเท็จ โปรแกรมจะทำการทดสอบนิพจน์เงื่อนไขที่ 2 ซึ่งจะได้ผลลัพธ์เป็นจริงหรือเท็จ

4. หากผลลัพธ์เป็นจริง โปรแกรมจะทำงานในประโยคคำสั่งที่ 2 หรือกลุ่มของประโยคคำสั่งที่ 2 แล้วเสร็จสิ้นการทำงานของคำสั่ง

5. หากผลลัพธ์เป็นเท็จ โปรแกรมจะทำการทดสอบในลักษณะข้างต้นและทำเช่นนี้เรื่อยๆ จนจบชุดคำสั่ง

การทำงานของคำสั่ง if/else if แสดงดังตัวอย่างที่ 6.6

**ตัวอย่างที่ 6.6** โปรแกรมการทำงานของคำสั่ง if/else

```
 1 #include <REGX52.H>
 2 void main (void)
 3 {
 4   unsigned char z = 10;
 5   if(z < 5)
 6   {
 7   z += 15;
 8   z = z%10;
 9   }
10   else if (z > 10)
11   {
12   z += 17;
13   z = z%4;
14   }
15   else
16   {
17   z += 20;
18   z = z%4;
19   }
20 }
```

**ผลลัพธ์ของการรันโปรแกรม**

| Name | Location/Value |
|---|---|
| MAIN | C:0x0800 |
| z | 2 |

*บรรทัดที่ 4* ตัวแปร z = 10

*บรรทัดที่ 5* คำสั่ง if เพื่อตรวจสอบเงื่อนไขว่า z น้อยกว่า 5 หรือไม่ เนื่องจากโปรแกรมเงื่อนไขเป็นเท็จ ให้กระทำกลุ่มของประโยคคำสั่งเงื่อนไขต่อไป

*บรรทัดที่ 10* คำสั่ง else if เพื่อตรวจสอบเงื่อนไขว่า z มากกว่า 10 หรือไม่ เนื่องจากโปรแกรมเงื่อนไขเป็นเท็จ ให้กระทำกลุ่มของประโยคคำสั่งเงื่อนไขต่อไป

*บรรทัดที่ 15* คำสั่ง else เพื่อตรวจสอบเงื่อนไขกรณีที่ z ไม่ตรงกับเงื่อนไขใด ให้กระทำกลุ่มของประโยคคำสั่งนี้

*บรรทัดที่ 17* z = z+20 ผลลัพธ์ z = 30

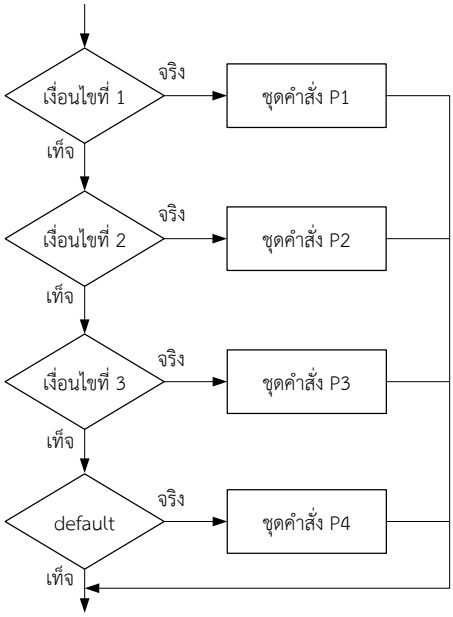
*บรรทัดที่ 8* z = z%4 ผลลัพธ์ z = 2

### 6.3.4 การเปรียบเทียบด้วยคำสั่ง switch

คำสั่งนี้เป็นคำสั่งที่ใช้ในการเลือกทำงานอย่างใดอย่างหนึ่งจากหลายๆ ทางเลือก โดยการทำงานจะทำการทดสอบนิพจน์ในคำสั่ง switch และนำผลลัพธ์ที่ได้จากการทดสอบมาเปรียบเทียบกับ label ในแต่ละกรณี หากไม่พบจะไปทำงานที่ default ซึ่งหลังจากทำงานในคำสั่งในกรณีใดเสร็จเรียบร้อยจะนิยมใส่คำสั่ง break ตามหลังในแต่ละกรณีเพื่อออกจากคำสั่ง switch ซึ่งมีรูปแบบและผังงานแสดงในภาพที่ 6.5

```
switch (<expression>)
{
    <case case-label-1:    statement-list1;>
                           break;
    <case case-label-2:    statement-list2;>
                           break;
    ...                    ...
    ...                    ...
    <case case-label-n:    statement-listn;>
                           break;
    <default:              statement-list0;>
}
```

ภาพที่ 6.5 รูปแบบและผังงานของคำสั่ง switch




ภาพที่ 6.5 รูปแบบและผังงานของคำสั่ง switch (ต่อ)

ที่มา: นวลนดา สงวนวงษ์ทอง, 2557

การทำงานของคำสั่ง switch แสดงดังตัวอย่างที่ 6.7-6.8

**ตัวอย่างที่ 6.7** โปรแกรมการทำงานของคำสั่ง switch

```
#include <REGX52.H>
void main (void)
{
  unsigned char key = 0x05;
  unsigned char a = 0x41;
  switch (key)
  {
    case 0x01:  a = a & 0xFF;
      break;
    case 0x05:  a = a & 0xF0;
      break;
    default:    a = a & 0x0F;
      break;
  }
}
```

**ผลลัพธ์ของการรันโปรแกรม**

| Name | Location/Value |
|---|---|
| MAIN | C:0x0800 |
| key | 0x05 |
| a | 0x40 '@' |

*บรรทัดที่ 4* ตัวแปร key = 0x05

*บรรทัดที่ 5* ตัวแปร a = 0x41

*บรรทัดที่ 6* คำสั่ง switch การทำงานจะนำค่าตัวแปร key มาเปรียบเทียบกรณีต่าง ๆ

*บรรทัดที่ 10* ค่าตัวแปร key ตรงกับ 0x05 ดังนั้นโปรแกรมจะทำตามคำสั่ง
a = a & 0xF0

*บรรทัดที่ 11* คำสั่ง break หยุดการทำงานและออกจากคำสั่ง switch

**ตัวอย่างที่ 6.8** โปรแกรมการทำงานของคำสั่ง switch

```
 1 #include <REGX52.H>
 2 void main (void)
 3 {
 4   unsigned char key = 'c';
 5   unsigned char a = 0x41;
 6   switch (key)
 7   {
 8     case 'a':   a = a & 0xFF;
 9       break;
10     case 'b':   a = a & 0xF0;
11       break;
12     default:    a = a & 0x0F;
13       break;
14   }
15 }
```

**ผลลัพธ์ของการรันโปรแกรม**

| Name | Location/Value |
|---|---|
| MAIN | C:0x0800 |
| key | 0x63 'c' |
| a | 0x01 |

*บรรทัดที่ 4* ตัวแปร key สำหรับเก็บอักขระ c

*บรรทัดที่ 5* ตัวแปร a = 0x41

*บรรทัดที่ 6* คำสั่ง switch การทำงานจะนำค่าตัวแปร key มาเปรียบเทียบกรณีต่าง ๆ

*บรรทัดที่ 12* ค่าตัวแปร key ไม่ตรงกับกรณีใดๆ ดังนั้นโปรแกรมจะกระโดดไปทำคำสั่ง
default นั่นคือ a = a & 0xF0

*บรรทัดที่ 13* คำสั่ง break หยุดการทำงานและออกจากคำสั่ง switch

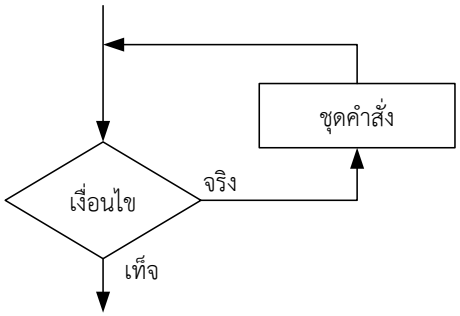
## 6.4 คำสั่งวนซ้ำ

คำสั่งให้โปรแกรมทำงานแบบวนรอบใช้ในกรณีที่โปรแกรมมีการทำงานซ้ำมากกว่า 1 ครั้ง ซึ่งคำสั่งวนซ้ำที่สำคัญได้แก่ คำสั่ง while, do-while และ for

### 6.4.1 คำสั่ง while

เป็นคำสั่งให้ทำงานวนซ้ำด้วยการตรวจสอบเงื่อนไขก่อนการทำซ้ำ ถ้าเงื่อนไขเป็นจริงจะทำตามชุดคำสั่งที่กำหนดให้และกลับไปทดสอบเงื่อนไขใหม่ถ้าเงื่อนไขเป็นเท็จก็จะออกจากการทำซ้ำทันที มีรูปแบบและผังงานแสดงได้ดังภาพที่ 6.6 โดยสามารถแสดงการทำงานของคำสั่ง while ได้ดังตัวอย่างที่ 6.9

```
while (<conditional expression>)
<statement-list (s)>;
```



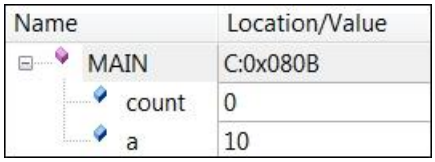

ภาพที่ 6.6 รูปแบบและผังงานของคำสั่ง while

ที่มา: นวลนดา สงวนวงษ์ทอง, 2557

**ตัวอย่างที่ 6.9** โปรแกรมการทำงานของคำสั่ง while

```
1 #include <REGX52.H>
2 void main (void)
3 {
4   unsigned char count = 10;
5   unsigned char a = 0;
6   while (count > 0)
7   {
8   count--;
9   a++;
10  }
11 }
```

**ผลลัพธ์ของการรันโปรแกรม**




*บรรทัดที่ 4* ตัวแปร count = 10

*บรรทัดที่ 5* ตัวแปร a = 0

*บรรทัดที่ 6* คำสั่ง while การทำงานจะตรวจสอบเงื่อนไขว่า count มากกว่า 0 หรือไม่

*บรรทัดที่ 7-10* กรณีเงื่อนไขเป็นจริง โปรแกรมจะทำการลดค่า count ครั้งละ 1 และเพิ่มค่า a ครั้งละ 1 เช่นกันจนกว่าเงื่อนไขจะเป็นเท็จจึงจะหลุดออกจากคำสั่ง while

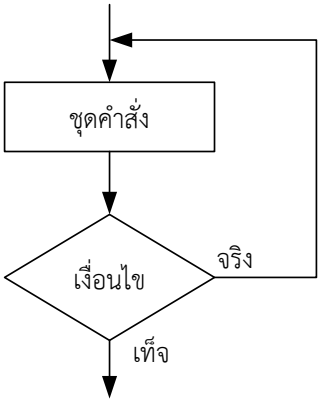
ในส่วนของเงื่อนไขของคำสั่ง while นั้นสามารถตรวจสอบค่าคงที่ได้ด้วย โดยถ้าค่าคงที่ไม่เท่ากับศูนย์จะทำซ้ำ แต่ถ้าค่าเท่ากับศูนย์จะไม่ทำซ้ำ ตัวอย่างเช่น ถ้าหากต้องการให้ค่าลอจิกทางพอร์ต P1 ของไมโครคอนโทรลเลอร์ AT89C52 ทุกบิตมีค่าลอจิกสลับกันไปมาจะเขียนโปรแกรมควบคุมด้วยคำสั่ง while ได้ดังนี้

```
while (1)
{
P1 = 0x55;
P1 = 0xAA;
}
```

### 6.4.2 คำสั่ง do-while

เป็นคำสั่งที่มีการทำงานแบบวนซ้ำโดยการตรวจสอบเงื่อนไขเหมือนกับคำสั่ง while แต่แตกต่างกันตรงที่จะมีการกระทำคำสั่งหรือชุดคำสั่งก่อน 1 ครั้ง แล้วจึงตรวจสอบเงื่อนไขในส่วนของ while มีรูปแบบและผังงานแสดงได้ดังภาพที่ 6.7 โดยสามารถแสดงการทำงานของคำสั่ง do-while ได้ดังตัวอย่างที่ 6.10

```
do {
    <statement-list (s)>;
   }
  while (<conditional expression>);
```




ภาพที่ 6.7 รูปแบบและผังงานของคำสั่ง do-while

ที่มา: นวลนดา สงวนวงษ์ทอง, 2557

**ตัวอย่างที่ 6.10** โปรแกรมการทำงานของคำสั่ง do-while

```
1  #include <REGX52.H>
2  void main (void)
3  {
4    unsigned char a = 0;
5    do
6    {
7    a = a+1;
8    } while (a==0);
9  }
```

**ผลลัพธ์ของการรันโปรแกรม**

| Name | Location/Value |
|---|---|
| MAIN | C:0x000F |
| a | 1 |

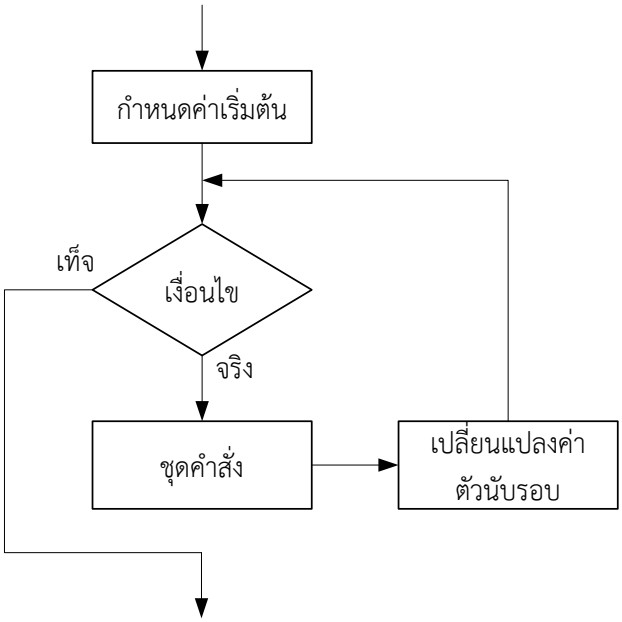
*บรรทัดที่ 4* ตัวแปร a = 0

*บรรทัดที่ 6-8* คำสั่ง do-while การทำงานจะทำงานในคำสั่ง do ก่อน นั่นคือ a = a+1 จากนั้นตรวจสอบเงื่อนไขว่า a = 0 หรือไม่ ในที่นี้จากโปรแกรมพบว่าเงื่อนไขเป็นเท็จ ดังนั้นโปรแกรมจะหลุดออกจากการวนซ้ำ

### 6.4.3 คำสั่ง for

เป็นคำสั่งที่กำหนดให้มีการทำงานแบบวนซ้ำด้วยการตรวจสอบจากเงื่อนไข โดยปกติจะมีการกำหนดรอบของการทำงานเป็นจำนวนที่แน่นอน มีรูปแบบและผังงานแสดงได้ดังภาพที่ 6.8

```
for (initialize; condition; incremental)
{
<statement-list (s)>;
}
```




ภาพที่ 6.8 รูปแบบและผังงานของคำสั่ง for

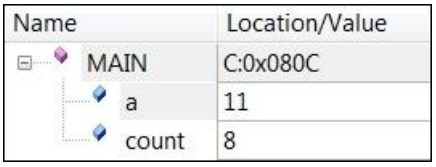
ที่มา: นวลนดา สงวนวงษ์ทอง, 2557

การทำงานของคำสั่ง for แสดงดังตัวอย่างที่ 6.11

**ตัวอย่างที่ 6.11** โปรแกรมการทำงานของคำสั่ง for

```
1 #include <REGX52.H>
2 void main (void)
3 {
4   unsigned char a = 3;
5   unsigned char count = 0;
6   for(count = 0;count < 8; count++)
7   {
8   a++;
9   }
10 }
```

**ผลลัพธ์ของการรันโปรแกรม**



| Name | Location/Value |
|---|---|
| MAIN | C:0x080C |
| a | 11 |
| count | 8 |

*บรรทัดที่ 4* ตัวแปร a = 3

*บรรทัดที่ 5* ตัวแปร count = 0

*บรรทัดที่ 6* คำสั่ง for กำหนดค่าตัวแปรเริ่มต้น count = 0 มีเงื่อนไขในการตรวจสอบตัวแปร count น้อยกว่า 8 โดยเมื่อจบการทำงานในแต่ละรอบให้เพิ่มค่าตัวแปร count 1 ค่า หากเงื่อนไขการตรวจสอบเป็นเท็จโปรแกรมจะหลุดออกจากคำสั่ง for

*บรรทัดที่ 8* เพิ่มค่าตัวแปร a ขึ้น 1 ค่า

## 6.5 คำสั่งที่เกี่ยวข้องกับการวนซ้ำ

เป็นคำสั่งที่ใช้ร่วมกับการวนซ้ำ เพื่อช่วยกำหนดเงื่อนไขการทำงานในการวนซ้ำได้มากขึ้น โดยมีคำสั่งที่เกี่ยวข้องมีดังนี้

### 6.5.1 คำสั่ง goto

เป็นคำสั่งที่ใช้ในการกระโดดไปยังตำแหน่งที่ต้องการ คำสั่ง goto จะต้องคู่กับตำแหน่ง label ที่ต้องการกระโดดไปด้วยทุกครั้ง การทำงานของคำสั่ง break แสดงดังตัวอย่างที่ 6.12

**ตัวอย่างที่ 6.12** โปรแกรมการทำงานของคำสั่ง goto

```
1  #include <REGX52.H>
2  void main (void)
3  {
4    unsigned char i;
5    for (i=0; i<10; i++)
6    {
7    if (i>5)
8    goto label;
9    }
10   label: i = 2;
11 }
```

*บรรทัดที่ 4* ประกาศตัวแปร i

*บรรทัดที่ 5* คำสั่ง for กำหนดค่าตัวแปรเริ่มต้น i = 0 มีเงื่อนไขในการตรวจสอบตัวแปร i น้อยกว่า 10 โดยเมื่อจบการทำงานในแต่ละรอบให้เพิ่มค่า count 1 ค่า

*บรรทัดที่ 7* คำสั่ง if ตรวจสอบเงื่อนไขว่า i > 5 หรือไม่

*บรรทัดที่ 8* ถ้า i > 5 คำสั่ง goto จะถูกเรียกใช้งาน โดยโปรแกรมจะกระโดดไปยังตำแหน่ง label

### 6.5.2 คำสั่ง break

เป็นคำสั่งที่ใช้ในการหยุดและออกจากการวนซ้ำ คำสั่งนี้แตกต่างจากคำสั่ง goto ตรงที่เมื่อคำสั่ง break ถูกเรียกใช้งานจะออกจากการวนซ้ำและไปทำงานในตำแหน่งต่อจากการวนซ้ำนั้น แต่คำสั่ง goto จะไปทำงานที่ตำแหน่งที่ต้องการแทน การทำงานของคำสั่ง break แสดงดังตัวอย่างที่ 6.13

**ตัวอย่างที่ 6.13** โปรแกรมการทำงานของคำสั่ง break

```
1 #include <REGX52.H>
2 void main (void)
3 {
4   unsigned char i;
5   for (i=0; i<10; i++)
6   {
7   if (i>5)
8   break;
9   }
10  i = 0;
11 }
```

*บรรทัดที่ 4* ประกาศตัวแปร i

*บรรทัดที่ 5* คำสั่ง for กำหนดค่าตัวแปรเริ่มต้น i = 0 มีเงื่อนไขในการตรวจสอบตัวแปร i น้อยกว่า 10 โดยเมื่อจบการทำงานในแต่ละรอบให้เพิ่มค่า count 1 ค่า

*บรรทัดที่ 7* คำสั่ง if ตรวจสอบเงื่อนไขว่า i > 5 หรือไม่

*บรรทัดที่ 8* ถ้า i > 5 คำสั่ง break จะถูกเรียกใช้งาน โดยโปรแกรมจะกระโดดออกจากการวนซ้ำทันที

### 6.5.3 คำสั่ง continue

เป็นคำสั่งการทำงานที่ตรงข้ามกับคำสั่ง break การทำงานของคำสั่ง continue จะบังคับให้โปรแกรมกระโดดข้ามการทำงานไปในรอบต่อไป การทำงานของคำสั่ง break แสดงดังตัวอย่างที่ 6.14

**ตัวอย่างที่ 6.14** โปรแกรมการทำงานของคำสั่ง continue

```
 1 #include <REGX52.H>
 2 void main (void)
 3 {
 4   unsigned char a,i;
 5   for (i=0; i<10; i++)
 6   {
 7   if (i==5)
 8   continue;
 9   a = i;
10   }
11 }
```

*บรรทัดที่ 4* ประกาศตัวแปร a และ i

*บรรทัดที่ 5* คำสั่ง for กำหนดค่าตัวแปรเริ่มต้น i = 0 มีเงื่อนไขในการตรวจสอบตัวแปร i น้อยกว่า 10 โดยเมื่อจบการทำงานในแต่ละรอบให้เพิ่มค่า count 1 ค่า

*บรรทัดที่ 7* คำสั่ง if ตรวจสอบเงื่อนไขว่า i เท่ากับ 5 หรือไม่

*บรรทัดที่ 8* ถ้า i เท่ากับ 5 คำสั่ง continue จะถูกเรียกใช้งาน โดยโปรแกรมจะกระโดดข้ามการทำงานไปในรอบต่อไปทันที

## 6.6 สรุป

ในบทนี้ได้นำเสนอเนื้อหาเกี่ยวกับการใช้คำสั่งควบคุมเพื่อควบคุมการทำงานของโปรแกรม C51 โดยคำสั่งควบคุมการทำงานที่สำคัญประกอบด้วยคำสั่งกำหนดเงื่อนไขและคำสั่งวนซ้ำ ได้แก่ คำสั่ง if if/else, switch, while, do-while และคำสั่ง for เพื่อใช้เป็นความรู้พื้นฐานในการพัฒนาระบบควบคุมไมโครคอนโทรลเลอร์ด้วยภาษาซีต่อไป

## 6.7 แบบฝึกหัดท้ายบท

แบบฝึกหัดมีทั้งหมด 6 ข้อ ให้นักศึกษาทำแบบฝึกหัดทุกข้อ

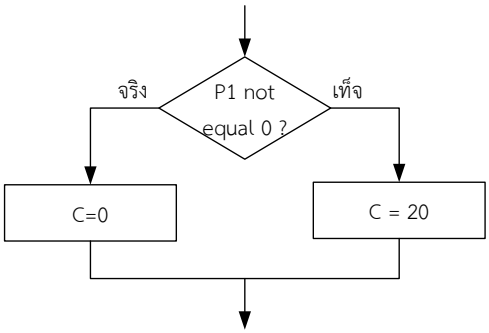
1. จงอธิบายการทำงานของคำสั่งต่อไปนี้พร้อมยกตัวอย่างประกอบการนำไปใช้งาน
   1.1) คำสั่ง if
   1.2) คำสั่ง if/else

1.3) คำสั่ง switch

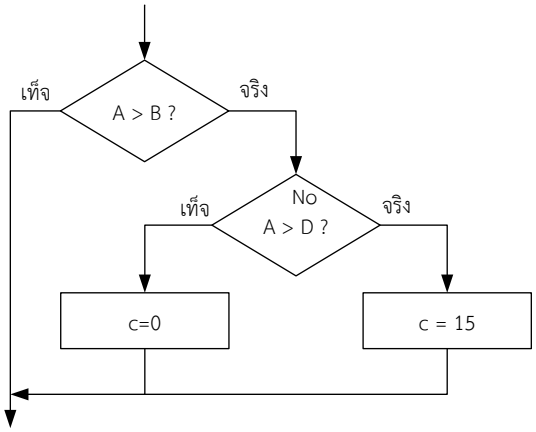
1.4) คำสั่ง while

1.5) คำสั่ง do-while

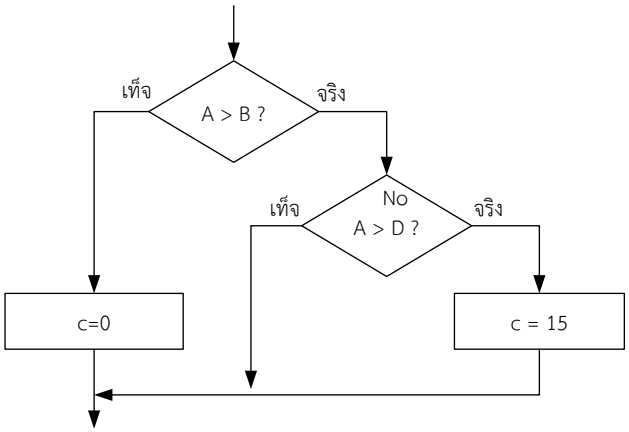
2. จงอธิบายการทำงานพร้อมทั้งเขียนโปรแกรม C51 จากผังงาน ดังนี้




3. จงอธิบายการทำงานพร้อมทั้งเขียนโปรแกรม C51 จากผังงาน ดังนี้




4. จงอธิบายการทำงานพร้อมทั้งเขียนโปรแกรม C51 จากผังงาน ดังนี้



5. จงอธิบายความแตกต่างของโปรแกรมจากผังงานในข้อ 3 และข้อ 4
6. จงอธิบายการทำงานพร้อมทั้งเขียนโปรแกรม C51 จากผังงาน ดังนี้