สิ่งที่เราควรทำความรู้จักเมื่อมีความจำเป็นต้องใช้ภาษาซีพูดคุยกับ MICROCONTROLLER

ภาษาซีถือเป็นสุดยอดแห่งวิวัฒนาการที่จะนำมาใช้เป็นสื่อกลางระหว่างมนุษย์กับmicrocontroller เนื่องจากว่ามันมีความสามารถ

ทำให้microcontrollerรับรู้และเข้าใจภาษาของมนุษย์ได้ ดังนั้นผู้ที่มีความสนใจในการพัฒนาเทคโนโลยีที่ใช้ microcontroller

เป็นหัวใจในการควบคุมการทำงานจะต้องทำความเข้าใจรูปแบบและไวยากรณ์ของภาษาซีในเบื้องต้นดังนี้

1 ชนิด และ การประกาศตัวแปร

2 กลุ่มคำสั่ง

- การเคลื่อนย้ายข้อมูล
- การ shift และ rotate ข้อมูล
- ตัวกระทำทางคณิตศาสตร์
- ตัวกระทำทางลอจิก
- ตัวกระทำทางตรรก
- เงื่อนไขและการตัดสินใจ
- รูปแบบการวนลูบ

3 รูปแบบการสร้างงาน

- การสร้างฟังชั่นหลัก
- การสร้างฟังชั่นย่อย
- การเรียกใช้ฟังชั่นภายใน

ชนิด และ การประการตัวแปร

ทำไมในระบบคอมพิวเตอร์ต้องมีตัวแปร และ ตัวแปรคืออะไร ? ที่ต้องให้คำจำกัดความว่า " ตัวแปร "น่าจะด้วยสาเหตุที่ว่าตัวมันเอง

มีความเปลี่ยแปรง(ข้อมูล)อยู่ตลอดเวลาตามสภาวะการทำงานของ microcontroller เปรียบได้กับ จิต ของมนุษย์ที่แปรเปลี่ยนอยู่ตลอด

เวลาตามสิ่งที่มากระทบทางผัสสะ โดยที่ microcontroller อาศัยพื้นที่ของหน่วยความจำข้อมูล (data memory)ในการประกาศตัวแปร

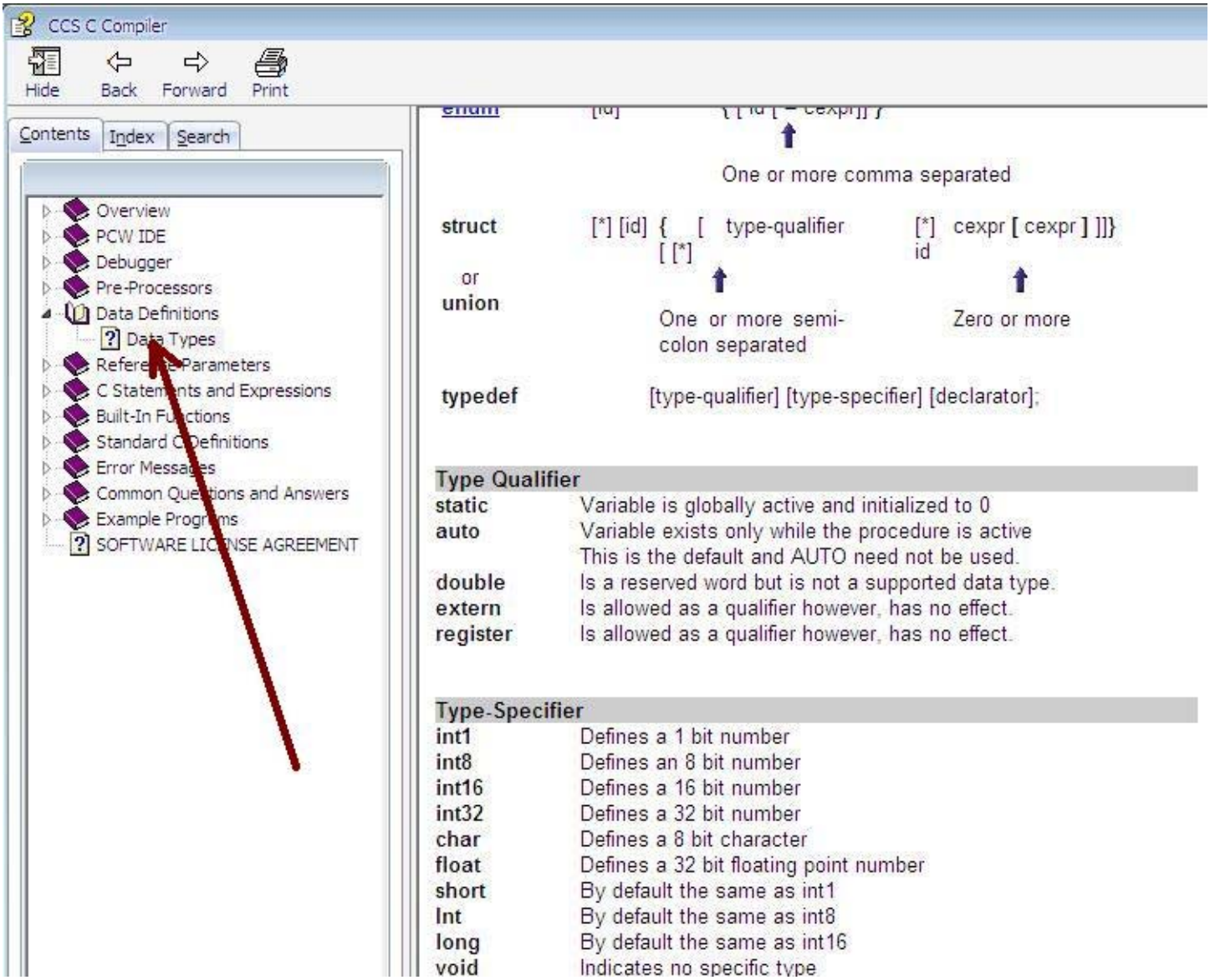
เพราะจะนั้น microcontroller ตัวไหนที่มีหน่วยความจำข้อมูลมากก็จะมีความคล่องตัวในการทำงานมากขึ้นตามไปด้วย เปรียบได้กับคนที่มีมันสมองเยอะ ก็น่าจะมีความสามารถในการทำงานที่สลับซับซ้อนมากขึ้น(เป็นความคิดส่วนตัว)

ชนิดตัวของตัวแปร

ชนิดตัวแปร	ใช้หน่วยความจำข้อมูล	ขนาดของข้อมูล

int1	1 bit	0-1
int8	8 bit	0-255 00H-FFH
int16	16bit	0-65535 0000H – FFFFH

ให้ดูข้อมูลจาก compiler น่าจะดีกว่านะครับ



จากข้อมูลดังกล่าวทำให้เรารู้ว่าเราจะประกาศตัวแปรแบบไหนให้พิจราณาจากข้อมูลที่มีการเปลี่ยนแปรงบนตัวมันนั่นเอง (เปรียบได้กับเราจะซื้อรถเพื่อใช้ในครอบครัวเอาไว้ไปไหนมาไหนสัก 3-4 คน ก็ไมควรซื้อรถบัส หรือ รถจักรยายยนต์)

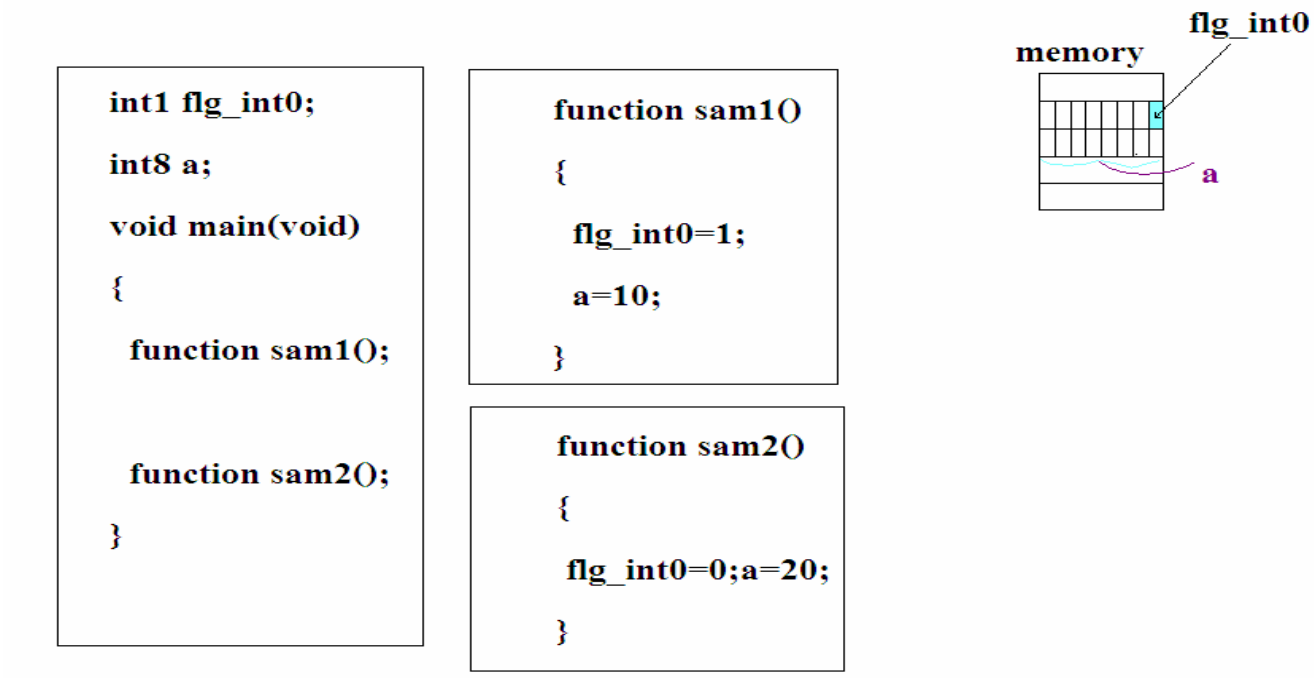
การประกาศตัวแปร

เรื่องของการประกาศตัวแปรก็เป็นเรื่องที่ต้องใส่ใจเหมือนกันเพราะถ้าเกิดเราประกาศไม่ถูกทีถูกทางแล้วอาจจะมีปัญหาหรือนำมาใช้งานไม่ได้ ซึ่งในส่วนนี้เราจะมองถึงตำแหน่งการประกาศ และ ขอบเขตการใช้งานเป็นหลัก

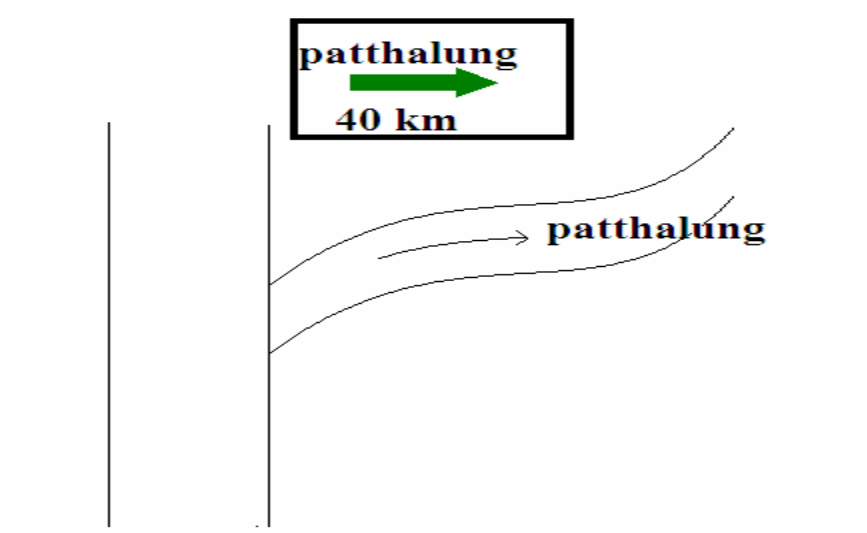
ตัวอย่างการประกาศตัวแปร แบบที่ 1

```
#include "D:\Program Files\PICC\testcommand\re.h"

Int1  flg_int0;
Int8  a;
void main()
{
  setup_adc_ports(NO_ANALOGS);
  setup_adc(ADC_OFF);
  setup_psp(PSP_DISABLED);
  setup_spi(FALSE);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
  setup_timer_1(T1_DISABLED);
  setup_timer_2(T2_DISABLED,0,1);

}
```

ตัวแปร flg_int0 และ ตัวแปร a ตำแหน่งประกาศอยู่เหนือฟังชั่นหลักเราจะเรียกตัวแปรแบบนี้ว่าเป็นตัวแปรแบบ Global เนื่องมาจากว่าทุกฟังชั่นในโปรเจ็กหรืองานนี้จะรู้จักรตัวแปรนี้ทั้งหมด ถ้ามองไปที่การใช้พื้นที่หน่วยความจำก็จะพูดได้ว่า ทุก ๆ ฟังชั่นที่มีการใช้ตัวแปร flg_int0 หรือ ตัวแปร a ก็จะใช้พื้นที่หน่วยความจำข้อมูลในส่วนเดียวกัน หรือ พูดได้อีกนัยหนึ่งว่าพื้นที่หน่วยความจำส่วนนี้ถูกจองไว้สำหรับตัวแปร flg_int0 และ ตัวแปร a อย่างถาวร



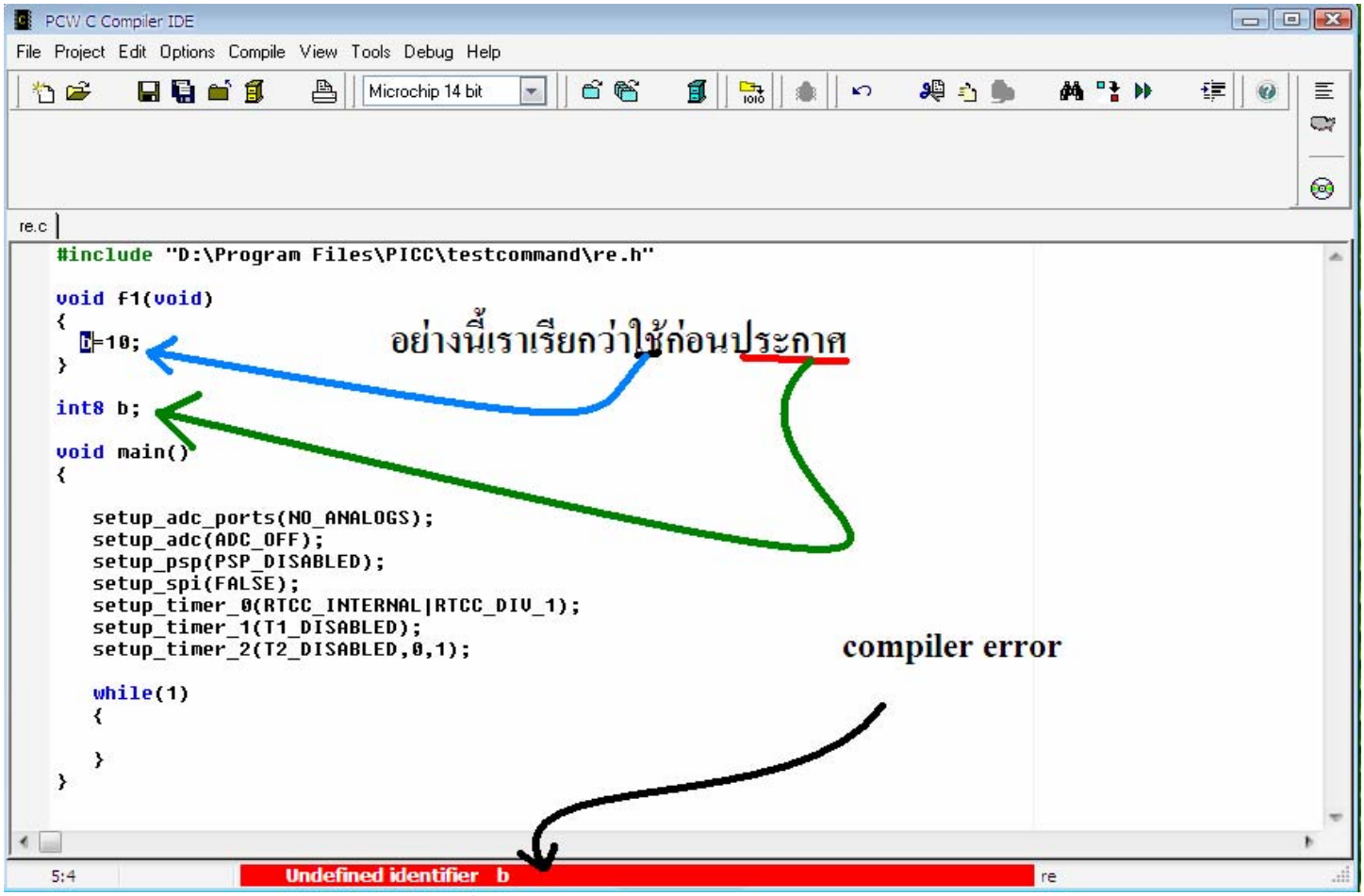
คลาวนี้จะกล่าวถึงตำแหน่งการประกาศ



รูปที่ 3 ตำแหน่งป้ายบอกทางที่แย่มาก ๆ

จากรูปที่ 3 พอจะนำมาเทียบเคียงได้กับตำแหน่งการประกาศตัวแปร นั่นก็คือว่าถ้าเราวางตำแหน่งการประกาศที่ไม่ถูกต้องแล้วตัวแปรตัวนั้นก็จะนำมาใช้งานไม่ได้ ดั่งเช่นจากรูปที่ป้ายไปพัทลุงไปวางไว้เลยตำแหน่งทางแยกแบบนี้คงจะเป็นปัญหาให้กับนักเดินทางอย่างแน่นอน

ตัวอย่างการวางตำแหน่งการประกาศตัวแปรที่ไม่ถูกต้อง



รูปที่ 4 ตำแหน่งประกาศตัวแปรไม่ถูกต้อง

กลุ่มคำสั่ง

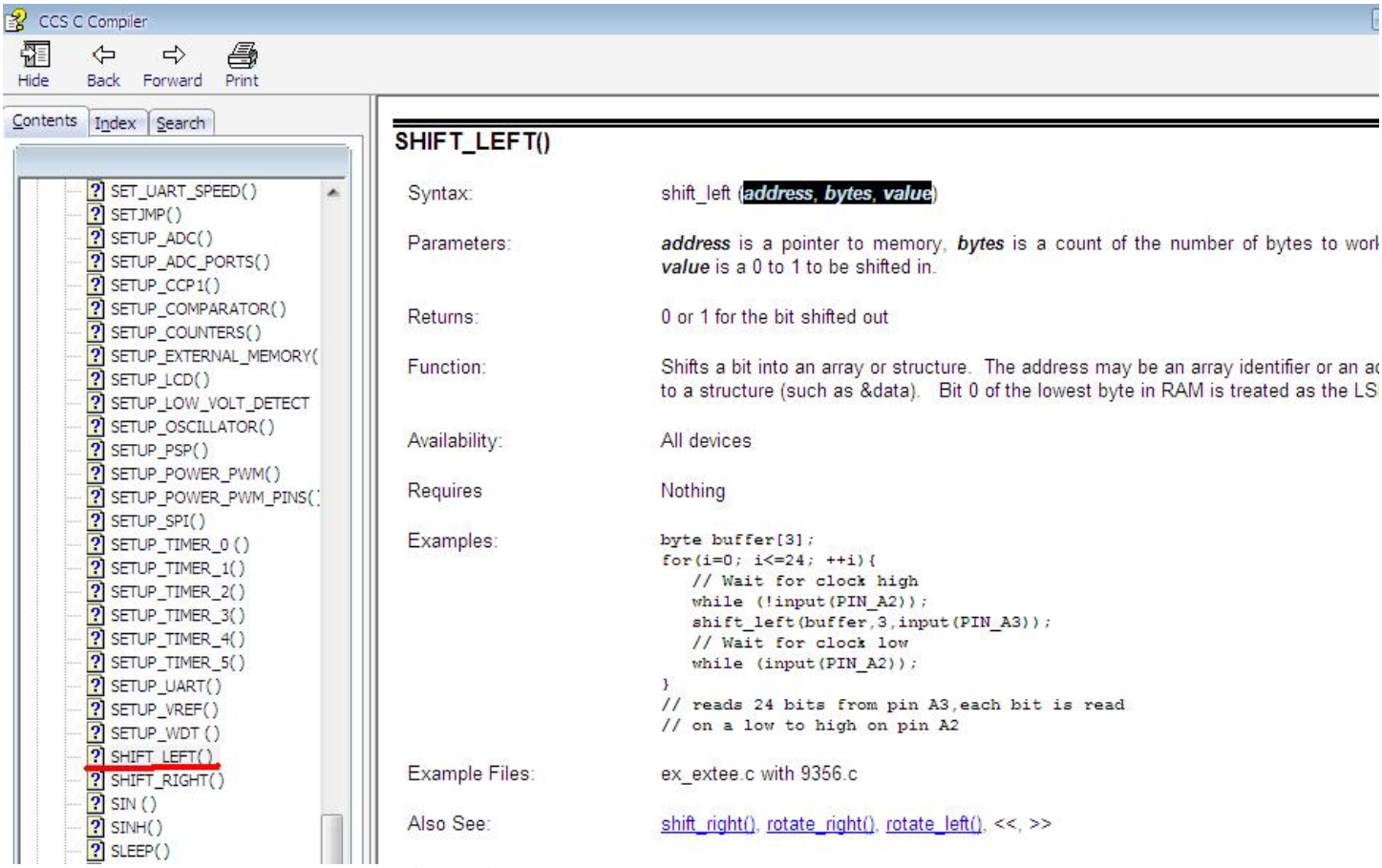
การเคลื่อยย้ายข้อมูล

คำสั่งที่ใช้ในการเคลื่อยย้ายข้อมูลถือได้ว่าเป็นคำสั่งที่ขาดไม่ได้เลยในระบบคอมพิวเตอร์เพราะระบบคอมพิวเตอร์เป็นระบบของการประเมินผลและระบบเคลื่อนย้ายข้อมูลเป็นหลัก

ตัวอย่างคำส่งที่ใช้ในการเคลื่อนย้ายข้อมูล

รูปที่ 5 การเคลื่อนย้ายข้อมูล

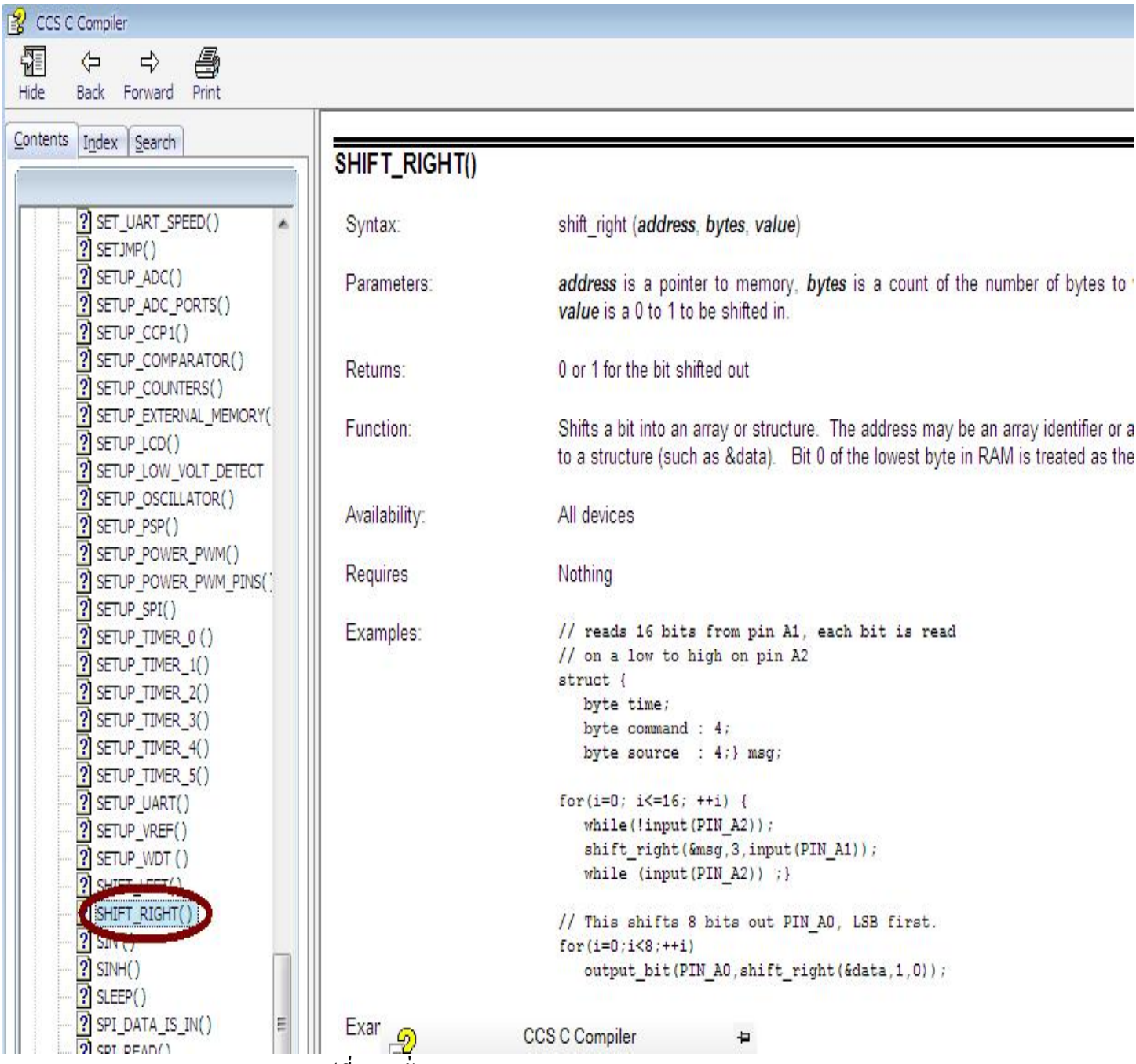
คำสั่ง shift left(*address, bytes, value*)



รูปที่ 6 คำสั่ง shift_left(***address***, ***bytes***, ***value***)

จากรูปที่ 6 เป็นรูปแบบการการใช้ฟังชั่นภายในในกระบวนการเคลื่อนข้อมูลไปทางซ้าย (shift left)จำนวน 3 byte หรือ 24 bit โดยรับข้อมูลเข้าจาก pin_a3 แล้วนำไปเก็บไว้ในตัวแปร buffer

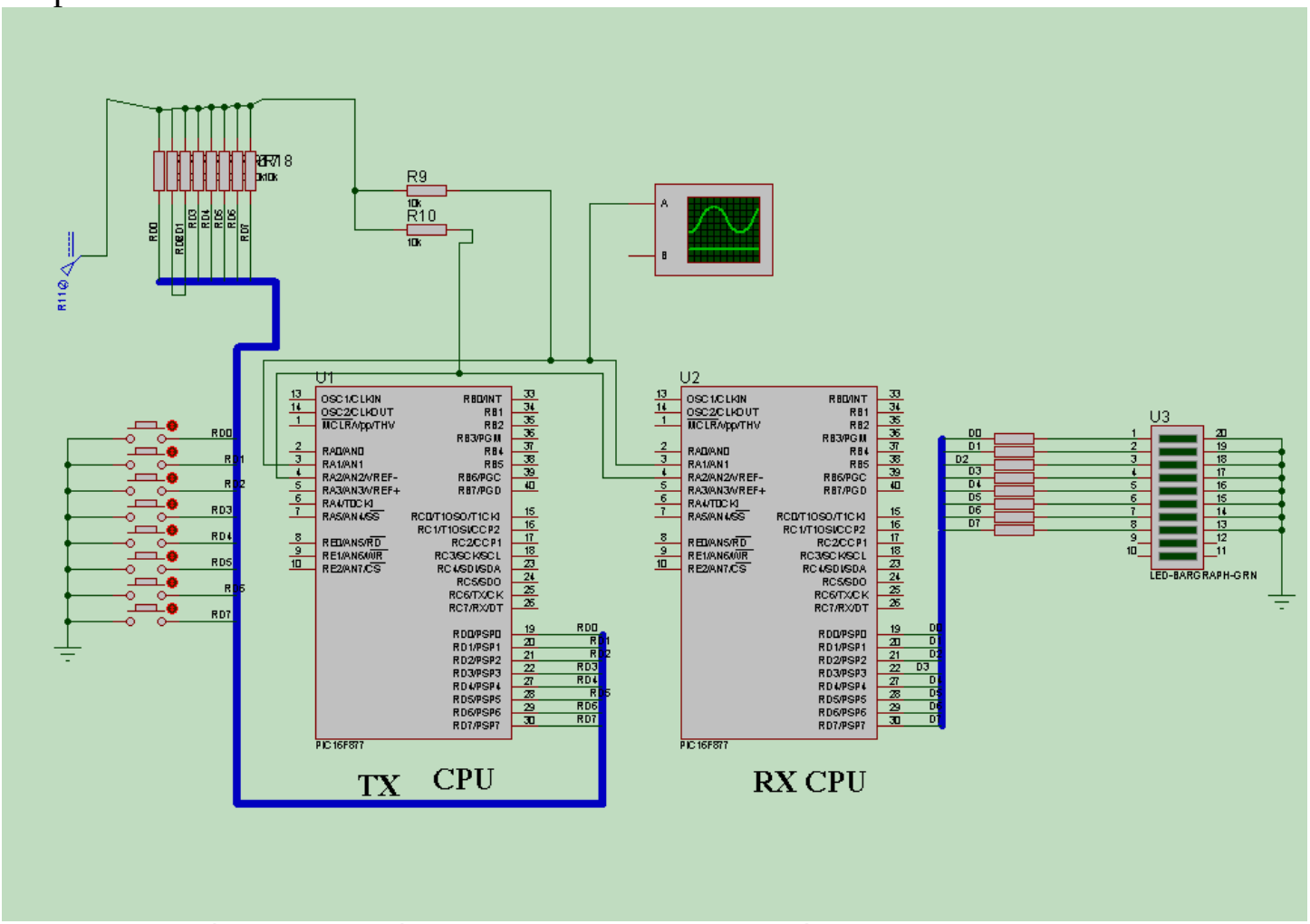
คำสั่ง shift_right (***address***, ***bytes***, ***value***)



รูปที่ 7 คำสั่ง shift_right (***address***, ***bytes***, ***value***)

จากรูปที่ 7 เป็นรูปแบบการการใช้ฟังชั่นภายในในกระบวนการเคลื่อนข้อมูลไปทางขวา (shift right)จำนวน 3 byte หรือ 24 bit โดยนำข้อมูลออ(serial out)ทาง pin_a1

ตัวอย่างโปรแกรมการประยุกต์ใช้ฟังชั่น shift left และ shift right ในการสื่อสารระหว่าง cpu 2 ตัว ผ่านทาง i/o port



รูปที่ 8 ประยุกต์ใช้ฟังชั่น shift_lift และ shift_right สื่อสารข้อมูลระหว่าง cpu

SOURCE CODE TX CPU

```
 #include <16F877.h>
#use delay(clock=20000000)
#fuses XT,NOWDT
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7)
int8 b;
int8 i;
void main()
{

  setup_adc_ports(NO_ANALOGS);
  setup_adc(ADC_OFF);
  setup_psp(PSP_DISABLED);
  setup_spi(FALSE);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
```

```
  setup_timer_1(T1_DISABLED);
  setup_timer_2(T2_DISABLED,0,1);

  output_low(pin_a2);
  output_low(pin_a1);
  while(1)
  {
    B=input_d();
    for(i=0; i<=7; ++i)
    {
      //shift_right(&b,1,input(PIN_A1));
      output_bit(PIN_A1,shift_right(&B,1,0));
      output_high(PIN_A2);
      delay_ms(100);
      output_low(pin_a2);

    }
  }
}
```

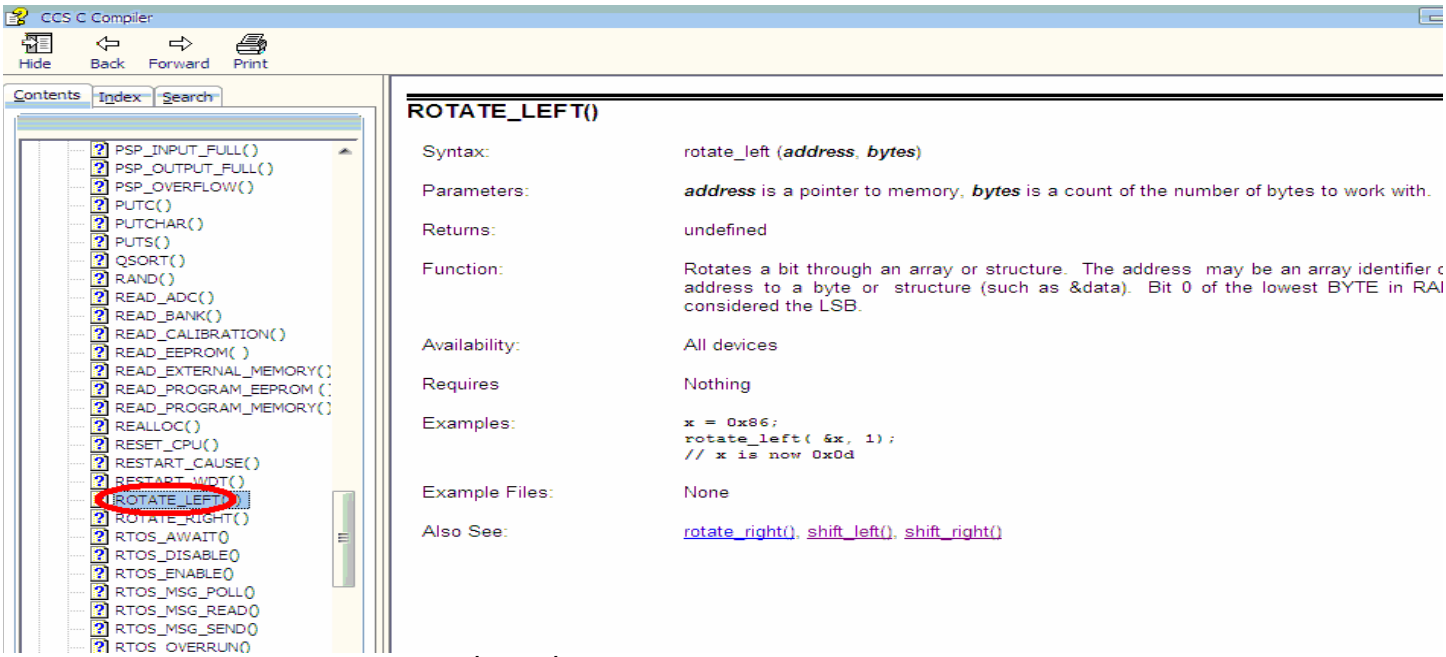
SOURCE CODE RX CPU

```
      // cpu recive data
#include <16F877.h>
#use delay(clock=20000000)
#fuses XT,NOWDT
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7)
int8 b;
int8 i;
void main()
{
  setup_adc_ports(NO_ANALOGS);
  setup_adc(ADC_OFF);
  setup_psp(PSP_DISABLED);
  setup_spi(FALSE);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
  setup_timer_1(T1_DISABLED);
  setup_timer_2(T2_DISABLED,0,1);
  output_high(pin_a2);// define pin_a2 is input port
  output_high(pin_a1);// define pin_a1 is input port
```

```
  while(1)
  {
          for(i=0; i<=7; ++i)
          {
           // Wait for clock high
            while (!input(PIN_A2));
            shift_left(&b,1,input(PIN_A1));
            // Wait for clock low
            while (input(PIN_A2));
           }

          output_d(b);
  }
}
```

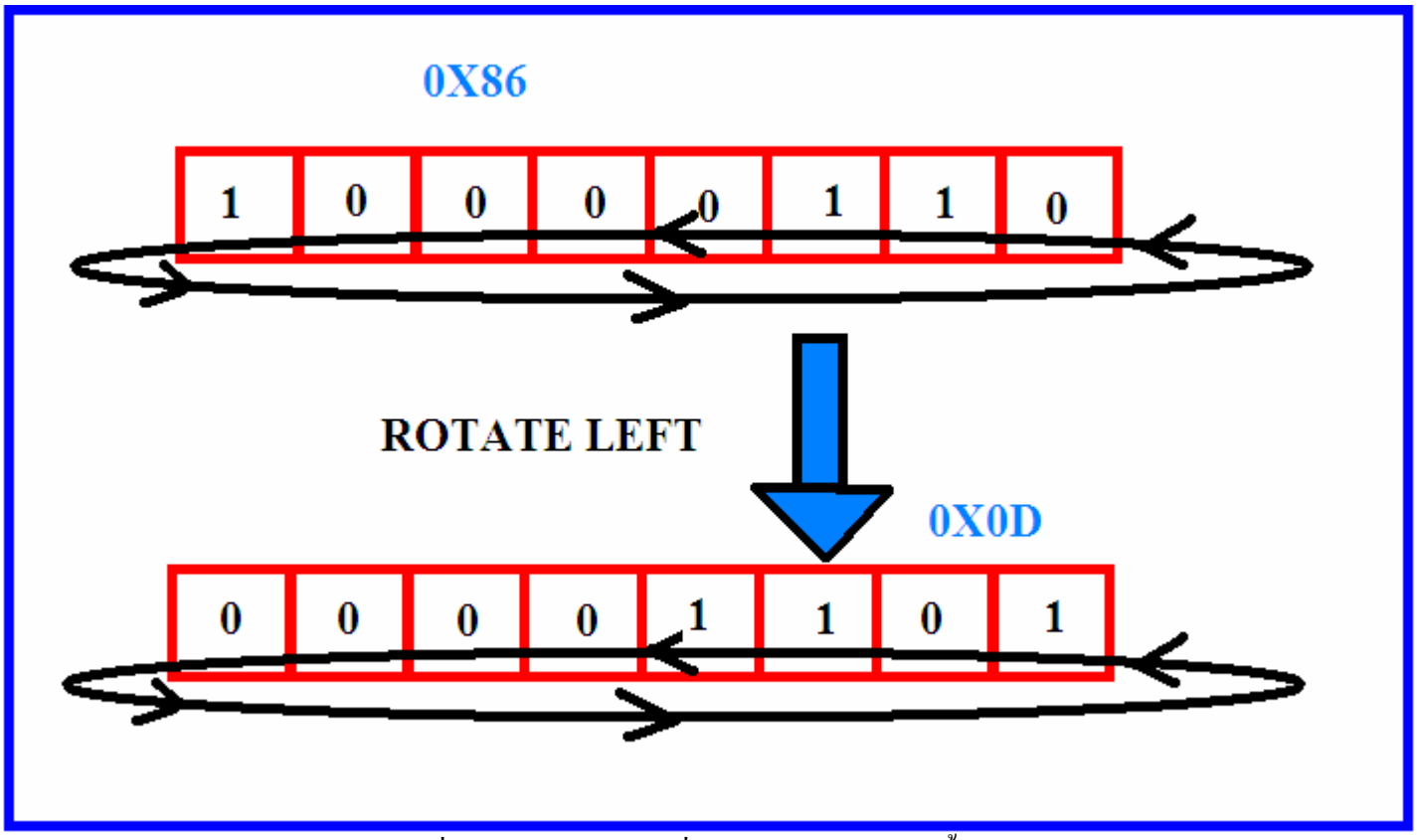
DOWNLOAD EXAMPLE PROGRAM FROM
WWW.BPCD.NET/TEACHER/ELECTRICAL/WEB/

คำสั่ง ROTATE

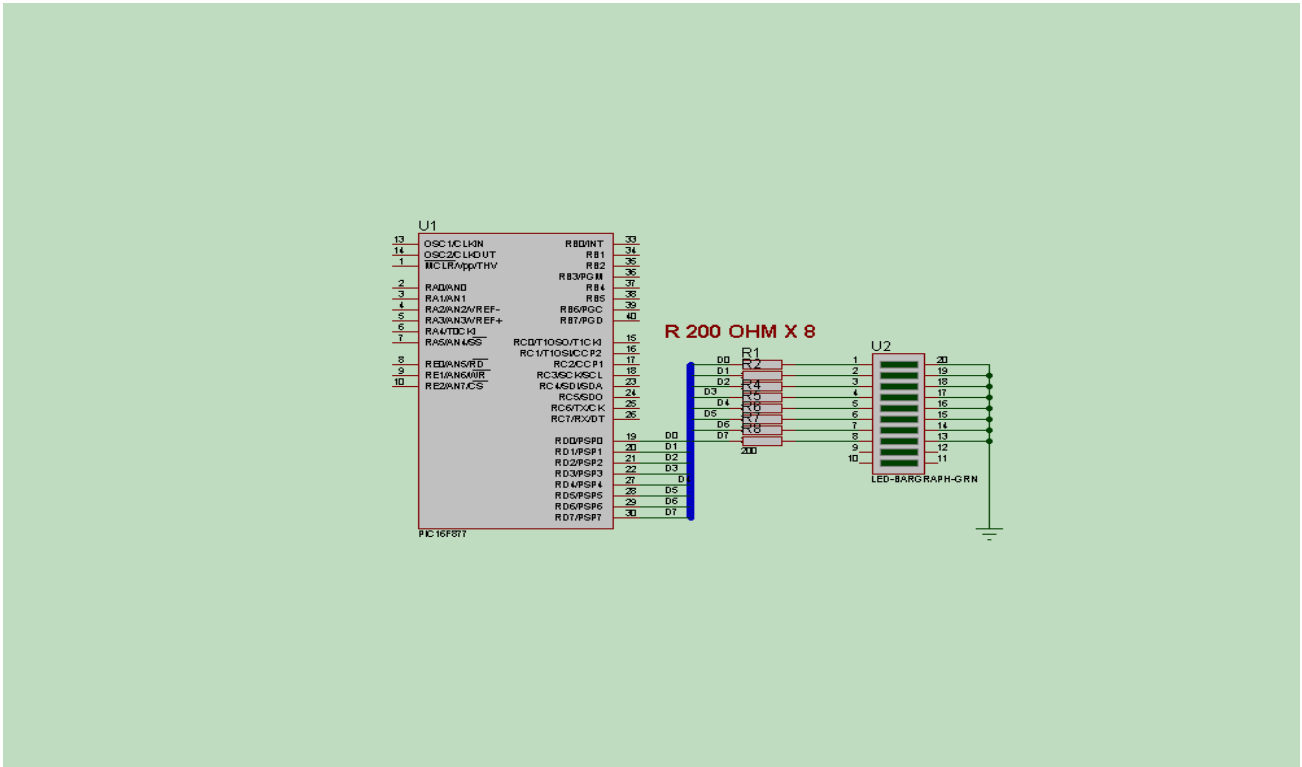


รูปที่ 9 ฟังชั่นภายใน ROTATE LEFT

ผลการเปลี่ยนแปลงของข้อมูลเมื่อถูกกระทำด้วยฟังชั่น rotate left



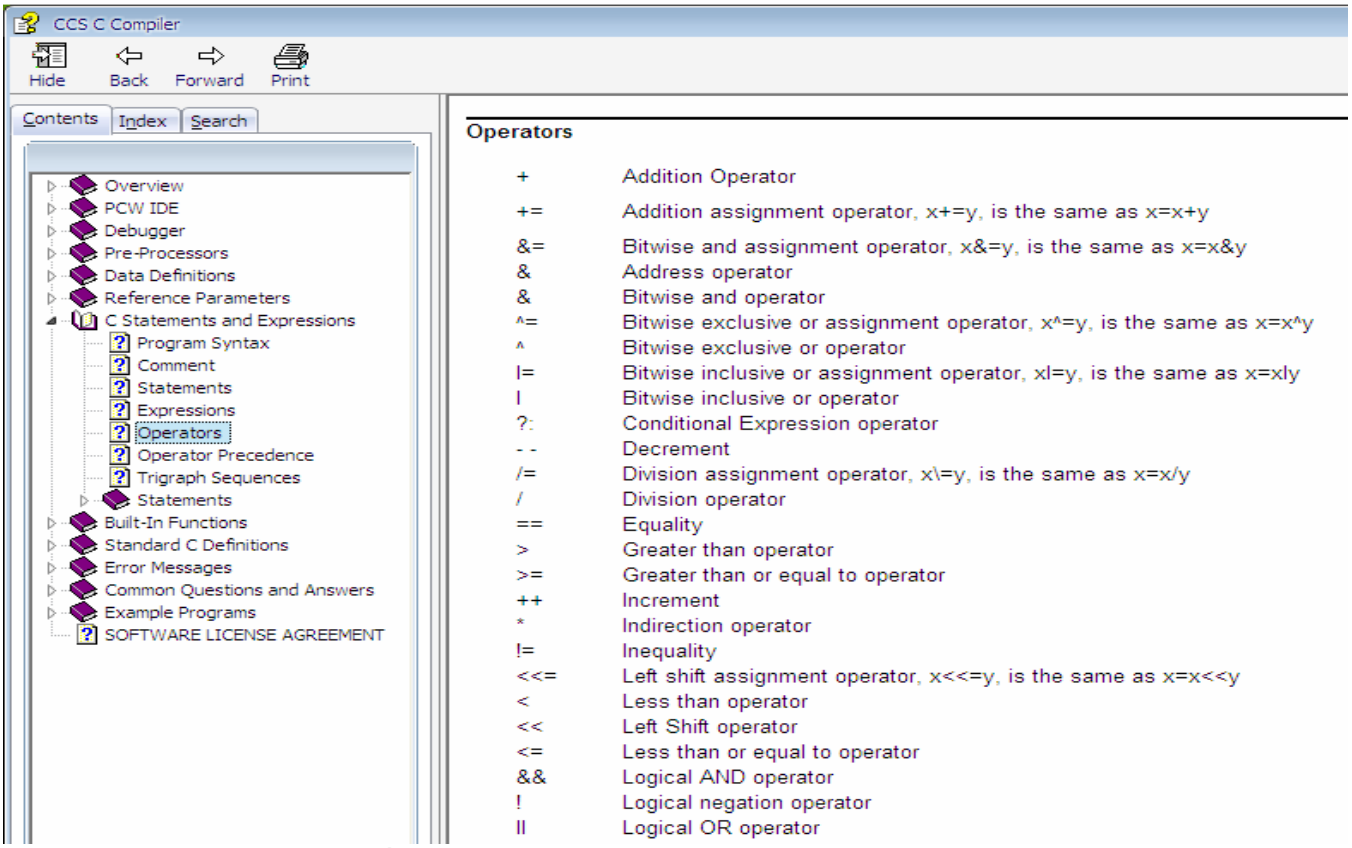
รูปที่ 10 ผลจากการใช้ฟังชั่น rotate left 1 ครั้ง



รูปที่ 11 วงจรทดลองคำสั่ง rotate_left

ตัวกระทำทางคณิตศาสตร์

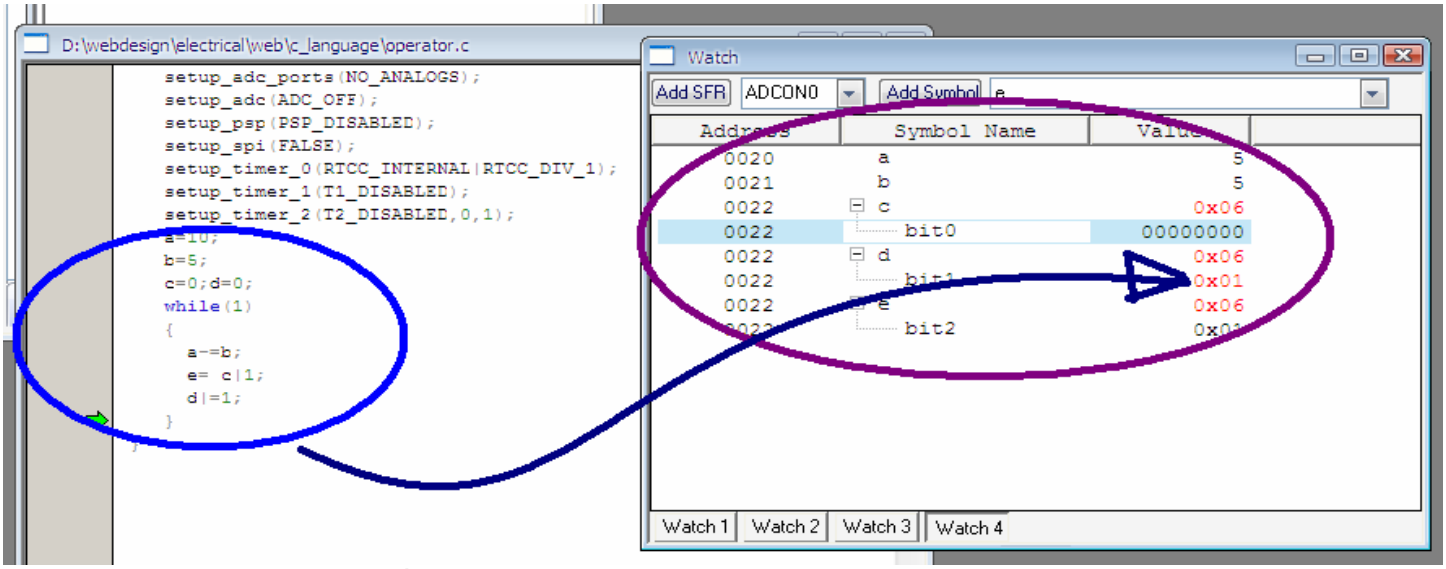
Pic c compiler ถือได้ว่าการวางรูปแบบไวยากรณ์ตาม ANSI C เพราะจะนั้นเราสามารถใช้ตำหรับตำราการใช้งานภาษาซี ได้ทั่วไป พร้อม ๆ กับดูคู่มือที่ PIC C compiler ให้มาด้วย



รูปที่ 12 ตัวกระทำทางคณิตศาสตร์ และ ตัวกระทำทางลอจิก

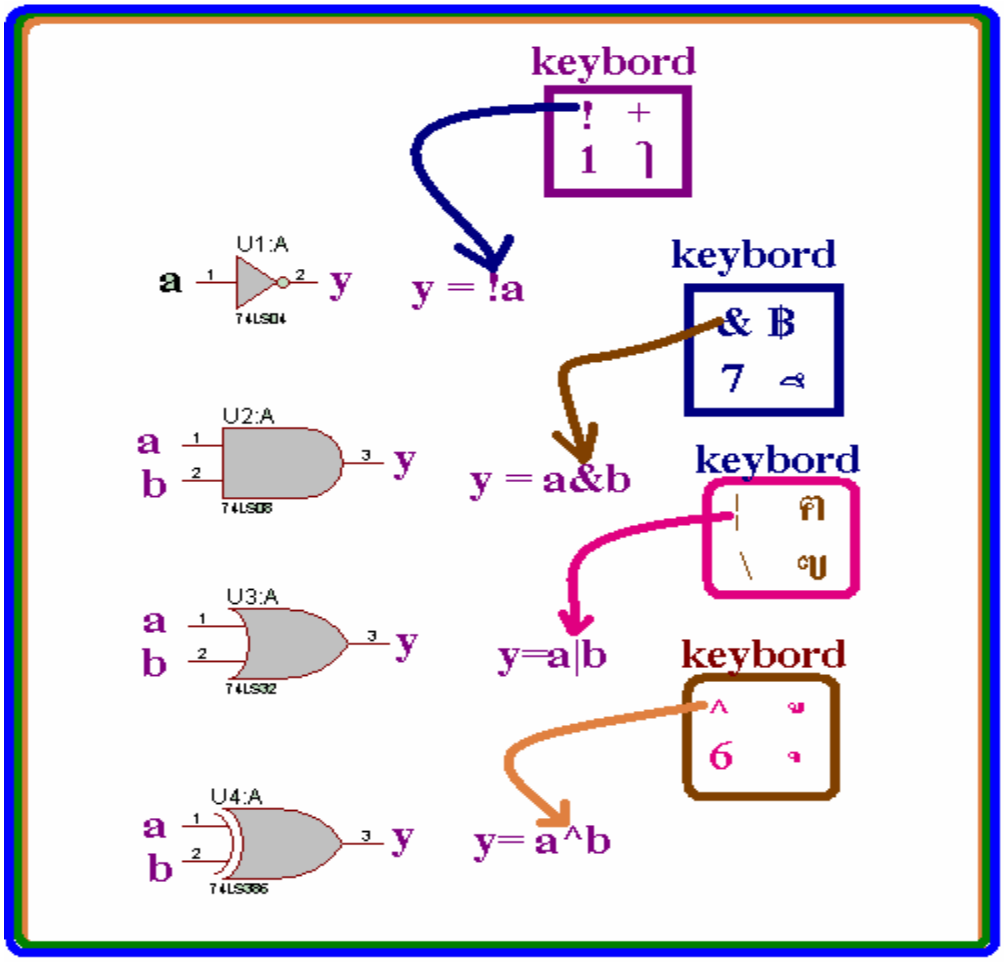
รูปที 12 เป็นคู่มือการใช้งานตัวกระทำทางคณิตศาสตร์และตัวกระทำทางลอจิก ขอแนะนำสักเล็กน้อยสำหรับผู้เริ่มต้นใช้ pic c compiler ให้ตรวจสอบการทำงานของตัวกระทำทางคณิตศาสตร์โดยใช้ software Mplab

ตัวอย่าง การดูผลของตัวกระทำทางคณิตศาสตร์

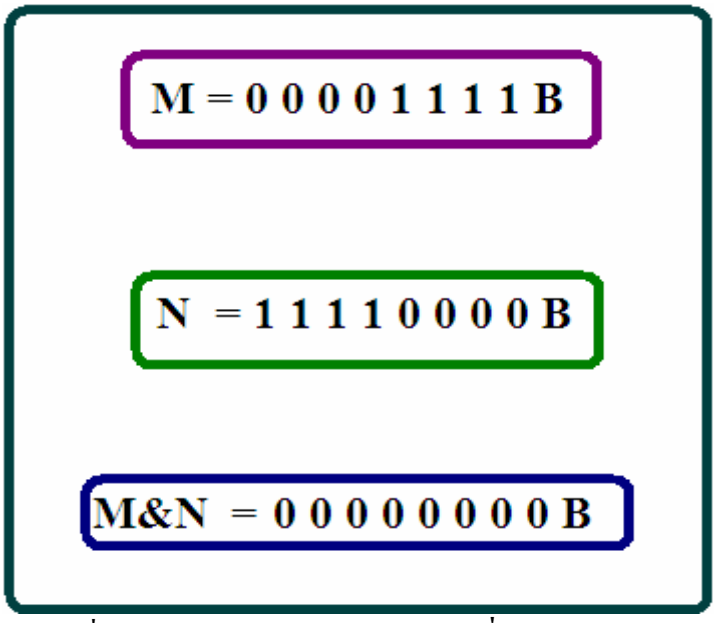


รูปที่ 13 ใช้ MPLAB ดูผลการกระทำของ operator

ตัวกระทำทางลอจิก



รูป14 ตัวกระทำทางลอจิกระดับบิทที่น่าทำความรู้จัก



รูปที่ 15 **ตัวกระทำทางลอจิกระดับไบท์ที่น่าทำความรู้จัก**

ตัวกระทำทางตรรกะ

ตัวดำเนินการทางตรรกะเป็นตัวที่ทำหน้าที่ในเรื่องของเหตุและผล จริงหรือเท็จ เพื่อเอาไว้ตัดสินใจว่าจะทำอะไรต่อไปในอนาคต เช่น ถ้าคืนนี้ฝนไม่ตกและท้องฟ้าแจ่มใสเราจะไปกรีดยาง

แต่ถ้าไม่เป็นดั่งว่าแล้วเราจะไม่ไปกรีดยางเด็ดขาด แปรได้ว่าคืนนี้เราอาจจะกรีด หรือ ไม่ได้กรีดยางก็ได้ เนื่องจากขึ้นอยู่เหตุการณ์

ตัวดำเนินการทาง ตรรกะ มีอยู่ 3 ชนิดคือ

ตัวดำเนินการ && ใช้กับเหตุการณ์ เช่น a และ b , นายแดง และ นายดำ

a&&b , นายแดง&&นายดำ

ตัวดำเนินการ || ใช้กับเหตุการณ์ เช่น a หรือ b , นายเขียว หรือ นายขำ

a||b ,นายเขียว||นายขำ

ตัวดำเนินการ ! ใช่กับเหตุการณ์ เช่น a ไม่เท่ากับ 5 , b ไม่มากกว่า 10

a!=5 , b!>10

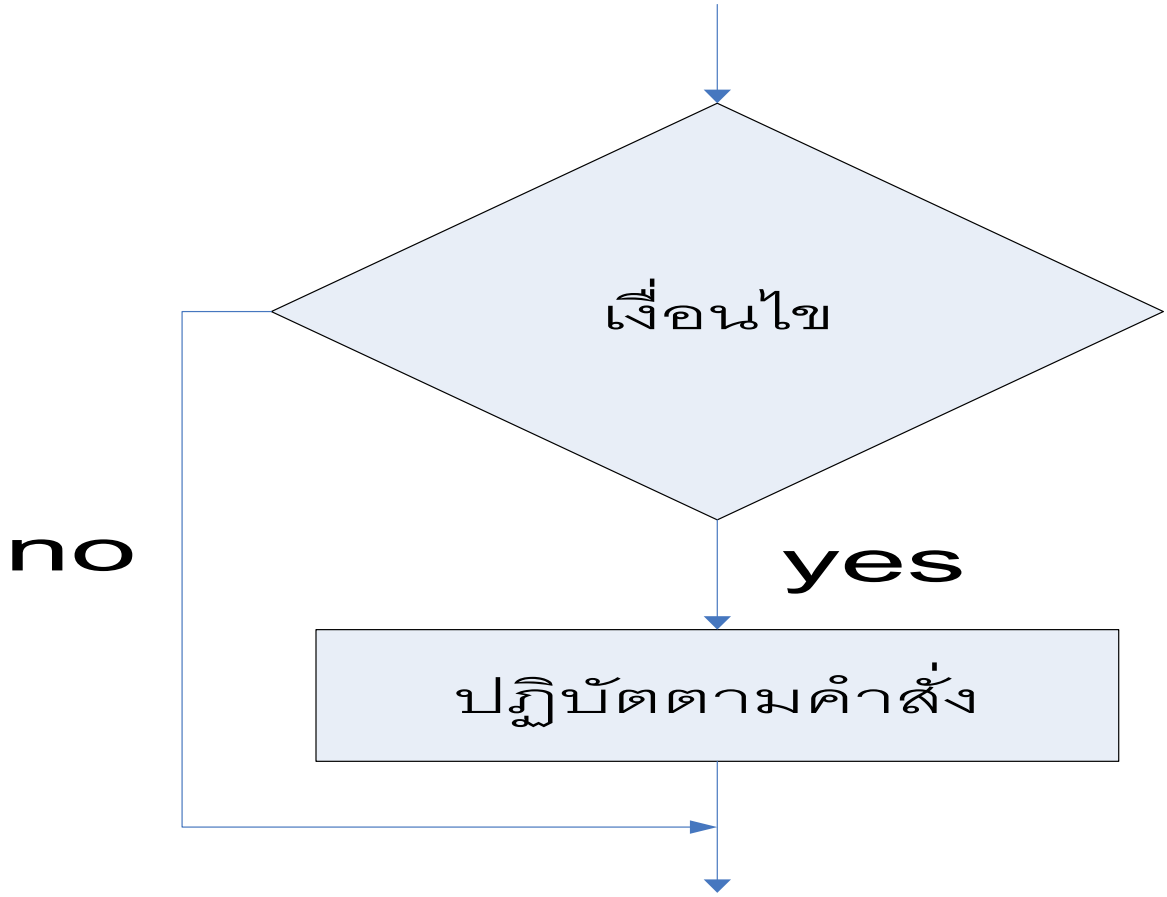
เงื่อนไข และ การตัดสินใจ

ดังที่ได้กล่าวมาแล้วว่าความเปลี่ยนแปลงทุกสิ่งทุกอย่างของสรรพสิ่งบนจักรวาล ย่อมเป็นไปตามเหตุและปัจจัย ที่ต้องกล่าวอย่างนี้เพราะยากให้นำไปเทียบเคียงกับการดำเนินชีวิตในแต่ละวินาทีนั้นมันเต็มไปด้วยเงื่อนไขต่าง ๆ มากมายซึ่งต้องนำมาพิจารณาเพื่อการตัดสินใจต่อการดำเนินชีวิตในวินาทีถัดไป ซึ่ง ในแต่ละเงื่อนไขก็จะมีทางเลือกในการตัดสินใจแตกต่างกันไปดังต่อไปนี้

เงื่อนไขในลักษณะที่ไม่มีทางเลือก

เช่น if (a>b) a=b; กรณีที่กระทำเพียงคำสั่งเดียว มีหรือไม่มี { }ก็ได้

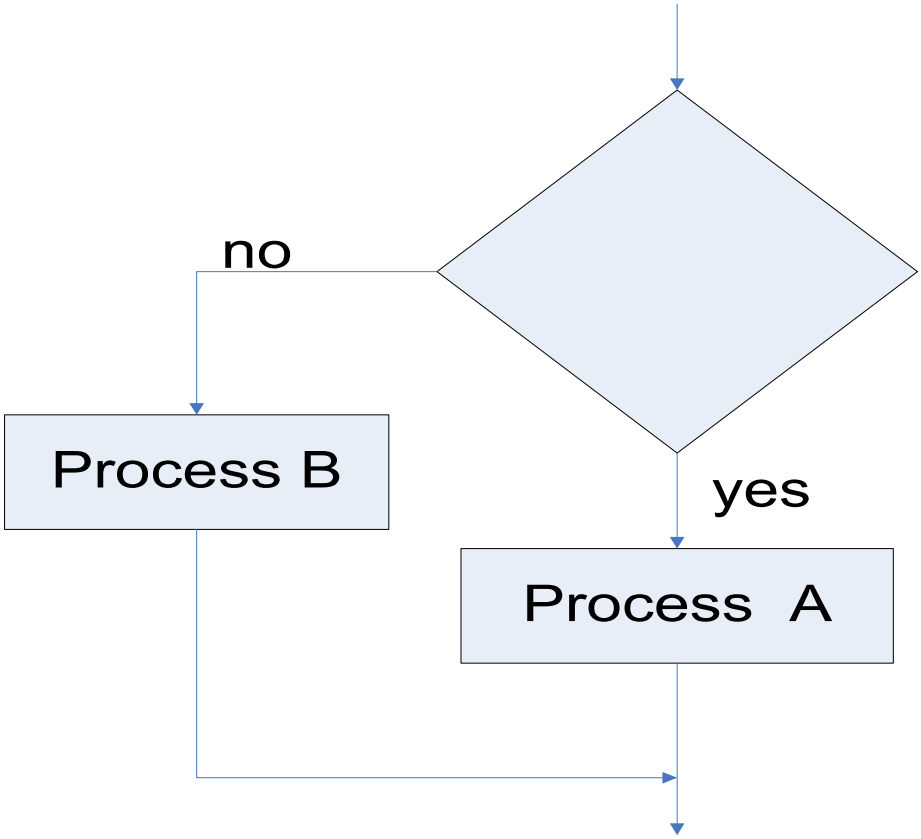
If(a>b) { a=b;b-=5;c=20;}กรณีต้องกระทำหลายคำสั่งต้องมี{ }



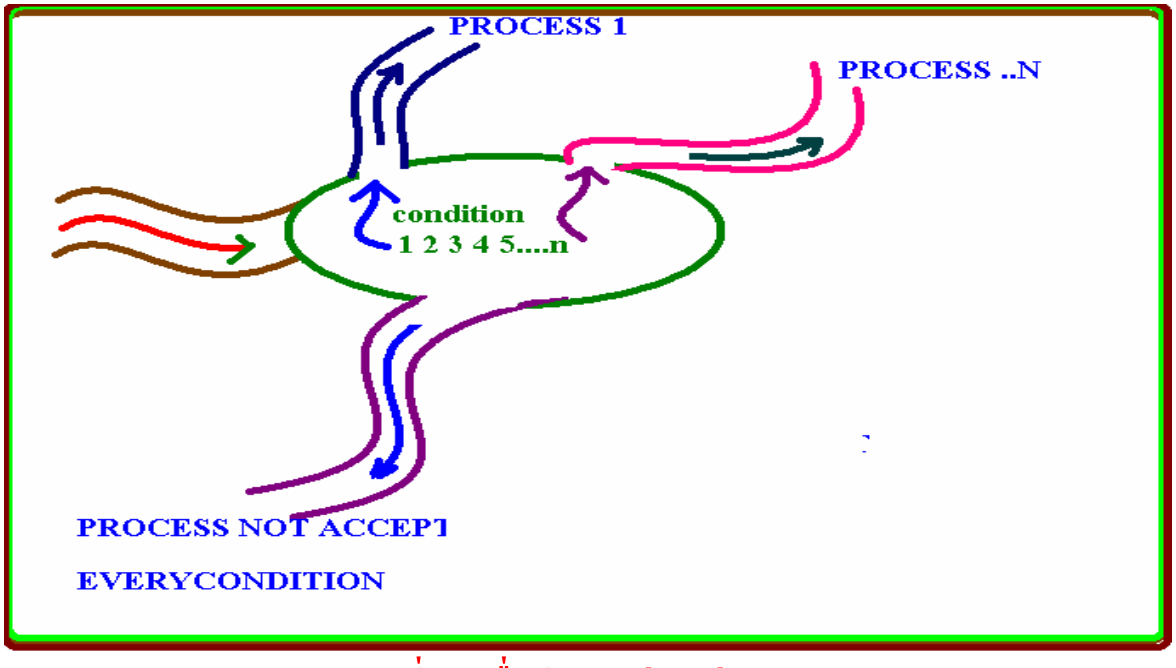
รูปที่ 16 เงื่อนไขที่ไม่มีทางเลือก

<u>เงื่อนไขเมื่อมีทางเลือกเพียง 2 ทาง (ไม่หัว ก็ ก้อย)</u>

```
If( a==b)
{
    A=0; Process A
 }
 Else
{
   A=5;Process B
 }
```

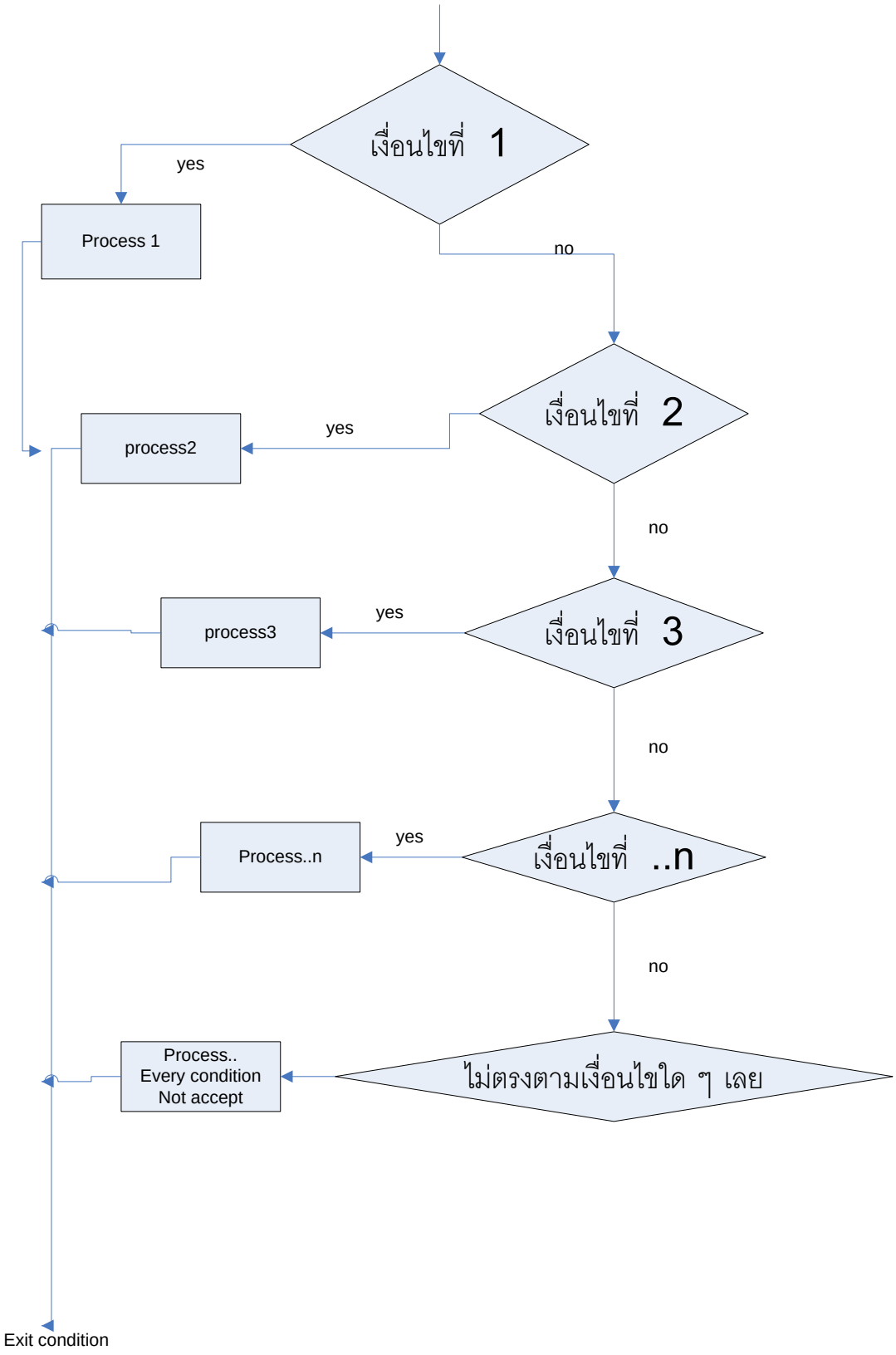


รูปที่ 17 เงื่อนไขที่มีทางเลือก 2 ทาง



รูปที่ 18 เมื่อมีมากกว่า 2 ช่องทาง

การใช if เมื่อมีเงื่อนไขที่มีทางเลือกมากกว่า 2 ทาง



รูปที่ 19 ทางเลือกมากกว่า 2 เงื่อนไข

ไวยากรณ์ซีตามรูปที่ 19

```
If (condition 1)
 {
   Process 1
 }
Else if(condition 2)
{
   Process 2
 }
Else if(condition 3)
{
    Process 3
 }
Else if(condition…n)
{
     Process…n
}
Else
{
       Process  every condition not accept
 }
```

รูปแบบการวนลูป

การทำงานบางครั้งเรามีความจำเป็นต้องทำซ้ำแล้วซ้ำอีกกว่างานจะบรรลุจุดประสงค์ตามความต้องการ ตัวอย่างเช่น การตอกตะปูในงานก่อสร้างต้องตอกหลายครั้งต่อตะปูหนึ่งตัว การกรีดยางต้องดึงมีดหลายครั้งต่อยางหนึ่งต้น การหมุนพวงมาลัยในการบังคับรถต้องหมุนหลาย ๆ รอบในการกลับรถ เป็น ต้น

ในทางโปรแกรมในรูปแบบของภาษาซีก็จะมีรูปแบบการทำงานในลักษณะการวนวนลูปอยู่สองสามรูปแบบด้วยกันคือ

<u>วนลูปโดยใช้ for(ค่าเริ่มต้น; เงื่อนไข ; การเปลี่ยนแปลงค่าในแต่ละลูป)</u>

ตัวอย่าง 1.1

```
for(x=0 ; x<10 ; x++)
{
 Process  a;
}
```

จากตัวอย่างข้างตัน เราใช้ x เป็นตัวแปร และ ให้ค่าเริ่มต้น เป็น 0 โดยมีเงื่อนไขในการวนลูปคือ x < 10 และในการวนลบแต่ละครั้ง ค่าในตัวแปร x จะเพิ่มขึ้นทีละหนึ่งด้วยคำสั่ง _x++ จากตัวอย่าง process a จะถูกทำซ้ำ ๆ 10 ครั้ง ตามเงื่อนไขการเปลี่ยนแปลงค่าของตัวแปร x ระหว่าง 0-9

<u>วนลูปโดยใช้ while(เงื่อนไข)</u>

ตัวอย่าง 1.2

```
x=0;
while( x <  10 )
{
  Process  b;
 X++;
}
```

จากตัวอย่าง 1.2 จากตัวอย่าง process b จะถูกทำซ้ำ ๆ 10 ครั้ง ตามเงื่อนไขการเปลี่ยนแปลงค่าของตัวแปร x ระหว่าง 0-9

<u>วนลูปโดยใช้ do…….while()</u>

ตัวอย่าง 1.3

```
Do
{
 Process c;
 X++;
}
 While(x<10)
```

จากตัวอย่าง 1.3 จากตัวอย่าง process c จะถูกทำซ้ำ ๆ 10 ครั้ง ตามเงื่อนไขการเปลี่ยนแปลงค่าของตัวแปร x ระหว่าง 0-9 จะเห็นว่าความแตกแตกต่างระหว่างตัวอย่าง 1.2 กับ 1.3 คือ เช็คเงื่อนไขก่อนทำงาน กับ ทำงานก่อนเช็คเงื่อนไข