พีชคณิตบูลีน และการลดรูปสมการ

รหัสวิชา 30127-2004 (2-3-3)
ดิจิทัลและไมโครคอนโทรลเลอร์

ภาษาซีกับไมโครคอนโทรลเลอร์

ภาษาซีกับไมโครคอนโทรลเลอร์

1. โครงสร้างของการเขียนโปรแกรมภาษาซี
2. ตัวแปร (Variable)
3. การเขียนฟังก์ชัน

3.1 ฟังก์ชันที่ไม่มีการให้ค่าเมื่อทำการเรียกใช้งานและไม่มีการคืนค่าเมื่อออกจากการทำงานของฟังก์ชัน

3.2 ฟังก์ชันที่มีการส่งผ่านค่าเมื่อทำการเรียกใช้งานและไม่มีการคืนค่าเมื่อออกจากการทำงานของฟังก์ชัน

3.3 ฟังก์ชันที่ไม่มีการส่งผ่านค่าเมื่อทำการเรียกใช้งานและมีการคืนค่าข้อมูลเมื่อออกจากการทำงานของฟังก์ชัน

3.4 ฟังก์ชันที่มีการส่งผ่านค่าเมื่อทำการเรียกใช้งานและมีการคืนค่าข้อมูลเมื่อออกจากการทำงานของฟังก์ชัน

ภาษาซีกับไมโครคอนโทรลเลอร์

4. ตัวดำเนินการ และนิพจน์คณิตศาสตร์
   - 4.1 ตัวดำเนินการนิพจน์ทางคณิตศาสตร์ (Arithmetic Operation)
   - 4.2 ตัวดำเนินการเปรียบเทียบ (Comparative Operation)
   - 4.3 ตัวดำเนินการทางตรรกะ (Logical Operation)
   - 4.4 ตัวดำเนินการกำหนดค่า
5. คำสั่งพื้นฐานในเขียนโปรแกรมภาษาซี
   - 5.1 คำสั่งการตรวจสอบเงื่อนไข
   - 5.2 คำสั่งวนรอบการทำงาน

ภาษาซีกับไมโครคอนโทรลเลอร์

6. การใช้งานขาไมโครคอนโทรลเลอร์ด้วยภาษาซี

6.1 การใช้งานขาไมโครคอนโทรลเลอร์ตระกูล MCS-51,PIC16F และAVR เป็นขาสัญญาณเอาต์พุตด้วยภาษาซี

6.2 การใช้งานขาไมโครคอนโทรลเลอร์ตระกูล MCS-51,PIC16F และAVR เป็นขาสัญญาณอินพุตด้วยภาษาซี

7. การใช้งานซอฟต์แวร์เพื่อเขียนโปรแกรมภาษาซีของไมโครคอนโทรลเลอร์

ภาษาซีกับไมโครคอนโทรลเลอร์

1. โครงสร้างของการเขียนโปรแกรมภาษาซี

การเขียนโปรแกรมภาษาซีจะมีรูปแบบของโครงสร้างในการเขียนโปรแกรมอยู่ 2 รูปแบบใหญ่ๆ ดังนี้

1. โครงสร้างของการเขียนโปรแกรมภาษาซีรูปแบบที่ 1

```
#include <headerfile.h>          //Preprocessor Directives
void  function()                 //Subroutine Function
{
    .....;
}
void main()                      //Main Function
{
   .....;
}
```


ภาษาซีกับไมโครคอนโทรลเลอร์

2. โครงสร้างของการเขียนโปรแกรมภาษาซีรูปแบบที่ 2

```
#include <headerfile.h>   //Preprocessor Directives
void function();          // Function Prototype
void main()               //Main Function
{
    .....;
}
void  function()          //Function Body
{
    .....;
}
```

ภาษาซีกับไมโครคอนโทรลเลอร์

2. ตัวแปร (Variable)

กฎในการตั้งชื่อตัวแปร

1. ต้องขึ้นต้นด้วยตัวอักษร ตัวต่อไปจะเป็นตัวอักษรหรือตัวเลขก็ได้
2. ห้ามใช้สัญลักษณ์อื่นใด ยกเว้น $ และขีดล่าง
3. ตัวอักษรพิมพ์เล็ก และพิมพ์ใหญ่มีความหมายต่างกัน
4. ห้ามเว้นวรรคระหว่างตัวแปร
5. ห้ามตั้งชื่อซ้ำกับคำสงวน

รูปแบบของการประกาศตัวแปร

type name;

โดย type = ชนิดของข้อมูล

name = ชื่อของตัวแปร

ภาษาซีกับไมโครคอนโทรลเลอร์

ตารางที่ 1.1 ตารางชนิดข้อมูลของตัวแปร

ชนิดตัวแปร	จำนวนบิต	ค่าข้อมูลที่เก็บได้
char	8	-128 ถึง 127
unsigned char	8	0 ถึง 255
int	16	-32768 ถึง 32767
unsigned int	16	0 ถึง 65535
long	32	-2147483648 ถึง 2147483648
unsigned long	32	0 ถึง 4294967925
float	32	3.4E-38 ถึง 3.4E+38
double	64	1.7E-308 ถึง 1.7E+308
bit (MCS-51)	1	0 ถึง 1

ภาษาซีกับไมโครคอนโทรลเลอร์

การเขียนโปรแกรมภาษาซีด้วย Complier AvrGCC ชนิดของข้อมูลที่นิยมใช้งานสามารถกำหนดได้อีก 3 รูปแบบ คือ

1. uint8_t คือการประกาศตัวแปรแบบ unsigned char
2. uint16_t คือการประกาศตัวแปรแบบ unsigned int
3. uint32_t คือการประกาศตัวแปรแบบ unsigned long

ในกรณีที่มีการประกาศตัวแปรเพื่อใช้งานแบบพิเศษ จะต้องมีข้อความกำกับขึ้นต้นก่อนการประกาศตัวแปร ซึ่งมักจะพบเจอในการเขียนโปรแกรมของภาษาซีสำหรับไมโครคอนโทรลเลอร์ประมาณ 3 รูปแบบได้แก่

1. การระบุคำ extern โดยมีรูปแบบ extern type name;
2. การระบุคำ static โดยมีรูปแบบ static type name;
3. การระบุคำ volatile โดยมีรูปแบบ volatile type name.

ภาษาซีกับไมโครคอนโทรลเลอร์

ชนิดของตัวแปร

ตัวแปรจะแบ่งออกเป็น 2 ชนิดใหญ่ ๆ ได้แก่

1. Global variable คือตัวแปรที่ประกาศไว้ภายนอกฟังก์ชันทุกฟังกชัน ซึ่งฟังก์ชันอื่น ๆ สามารถเรียกใช้งานตัวแปรแบบนี้ได้

2. Local variable คือ ตัวแปรที่ประกาศภายในฟังก์ชันหลัก หรือฟังก์ชันย่อย การใช้งานจะสามารถใช้งานได้เฉพาะฟังก์ชันที่ประกาศไว้เท่านั้น

ภาษาซีกับไมโครคอนโทรลเลอร์

3. การเขียนฟังก์ชัน

การเขียนโปรแกรมฟังก์ชันย่อยในภาษาซีจะมีรูปแบบและองค์ประกอบในการเขียนดังนี้

```
function-type function-name (Argument Variable)
 {
      type variable;
      statement instruction;
      return value;
 }
```


ภาษาซีกับไมโครคอนโทรลเลอร์

3.1 ฟังก์ชันที่ไม่มีการให้ค่าเมื่อทำการเรียกใช้งานและไม่มีการคืนค่าเมื่อออกจากการทำงานของฟังก์ชัน

รูปแบบในการเขียน

```
void function-name( )
{
     type variable;
     statement instruction;
}
```

ภาษาซีกับไมโครคอนโทรลเลอร์

3.2 ฟังก์ชันที่มีการส่งผ่านค่าเมื่อทำการเรียกใช้งานและไม่มีการคืนค่าเมื่อออกจากการทำงานของฟังก์ชัน

รูปแบบในการเขียน

```
void  function-name(Argument Variable)
{
    type variable;
    statement instruction;
}
```


ภาษาซีกับไมโครคอนโทรลเลอร์

3.3 ฟังก์ชันที่ไม่มีการส่งผ่านค่าเมื่อทำการเรียกใช้งานและมีการคืนค่าข้อมูลเมื่อออกจากการทำงานของฟังก์ชัน

รูปแบบในการเขียน

```
function-type  function-name( )
    {
        type variable;
        statement instruction;
        return value;
    }
หรือ
function-type  function-name (void)
    {
        type variable;
        statement instruction;
        return value;
    }
```


ภาษาซีกับไมโครคอนโทรลเลอร์

3.4 ฟังก์ชันที่มีการส่งผ่านค่าเมื่อทำการเรียกใช้งานและมีการคืนค่าข้อมูลเมื่อออกจากการทำงานของฟังก์ชัน

รูปแบบในการเขียน

```
function-type  function-name( Argument Variable)
 {
    type variable;
    statement instruction;
    return value;
 }
```


ภาษาซีกับไมโครคอนโทรลเลอร์

4. ตัวดำเนินการ และนิพจน์คณิตศาสตร์

4.1 ตัวดำเนินการนิพจน์ทางคณิตศาสตร์ (Arithmetic Operation)

ตัวดำเนินการ	ความหมาย	ตัวอย่าง
+	บวก (addition)	x + y
-	ลบ(subtraction)	x - y
*	คูณ(multiplication)	x * y
/	หาร(division)	x / y
%	หารเอาผลลัพธ์เฉพาะเศษ	x % y
++	เพิ่มค่าครั้งละ 1	x++
--	ลดค่าครั้งละ 1	x--

รูปที่ 1.1 รูปตารางแสดงตัวดำเนินการนิพจน์ทางคณิตศาสตร์ของภาษาซี

ภาษาซีกับไมโครคอนโทรลเลอร์

4.2 ตัวดำเนินการเปรียบเทียบ (Comparative Operation)

ตัวดำเนินการ	ความหมาย	ตัวอย่าง
>	มากกว่า	x > y
<	น้อยกว่า	x < y
>=	มากกว่าหรือเท่ากับ	x >= y
<=	น้อยกว่าหรือเท่ากับ	x <= y
==	เท่ากับ	x == y
!=	ไม่เท่ากับ	x != y

รูปที่ 1.2 รูปตารางแสดงตัวดำเนินการเปรียบเทียบของภาษาซี

ภาษาซีกับไมโครคอนโทรลเลอร์

4.3 ตัวดำเนินการทางตรรกะ (Logical Operation)

ตัวดำเนินการ	ความหมาย	ตัวอย่าง
&&	และ(and)	mark>=80&&mark<=100
||	หรือ(or)	score<0 || score>100
!	ไม่(not)	!x &&!y

รูปที่ 1.3 รูปตารางแสดงตัวดำเนินการทางตรรกะที่ใช้งานร่วมกับชุดคำสั่ง if(),for(),while() และ do-while()

ตารางที่ 1.2 ตารางตัวดำเนินการทางตรรกะของภาษาซีเพื่อใช้ในการเปลี่ยนแปลงค่าข้อมูล

ตัวดำเนินการ	ความหมาย	ตัวอย่าง
&	And Data	A & 0xff
|	Or Data	A | 0x03
! หรือ ~	Not Data	!A หรือ ~A
^	Xor Data	A ^ 0xff

ภาษาซีกับไมโครคอนโทรลเลอร์

4.4 ตัวดำเนินการกำหนดค่า

ตัวดำเนินการ	ความหมาย	ตัวอย่าง
=	กำหนดค่าให้เท่ากับ	x = y
+=	การเพิ่มค่า	x += y มีผลเท่ากับ x = x + y
-=	การลบค่า	x -= y มีผลเท่ากับ x = x - y
*=	การคูณ	x *= y มีผลเท่ากับ x = x * y
/=	หารได้ผลลัพธ์จำนวนเต็ม	x /= y มีผลเท่ากับ x = x / y
%=	การหารได้ผลลัพธ์เศษ	x %= y มีผลเท่ากับ x = x % y
&=	ดำเนินการ	x &= y มีผลเท่ากับ x = x & y
|=	ดำเนินการ	x |= y มีผลเท่ากับ x = x | y
^=	ดำเนินการ	x ^= y มีผลเท่ากับ x = x ^ y
<<=	การเลื่อนบิตไปทางซ้าย	x <<=2 มีผลเท่ากับ x = x <<2
>>=	การเลื่อนบิตไปทางขวา	x >>=2 มีผลเท่ากับ x = x >>2

รูปตารางที่ 1.4 รูปตารางแสดงการใช้งานตัวดำเนินการกำหนดค่าของภาษาซีในรูปแบบต่าง ๆ

ภาษาซีกับไมโครคอนโทรลเลอร์

5. คำสั่งพื้นฐานในเขียนโปรแกรมภาษาซี

5.1 คำสั่งการตรวจสอบเงื่อนไข

5.1.1 กลุ่มคำสั่งตรวจสอบเงื่อนไขในรูปแบบคำสั่ง if()

5.1.1.1 คำสั่งตรวจสอบเงื่อนไข 1 ทางเลือก

```
รูปแบบ  if ( เงื่อนไข )
        {
            ประโยคคำสั่งเมื่อเงื่อนไขเป็นจริง;
        }
```

5.1.1.2 คำสั่งตรวจสอบเงื่อนไข 2 ทางเลือก

```
รูปแบบ  if ( เงื่อนไข )
        {
            ประโยคคำสั่งเมื่อเงื่อนไขเป็นจริง;
        } else {
            ประโยคคำสั่งเมื่อเงื่อนไขเป็นเท็จ;
        }
```


ภาษาซีกับไมโครคอนโทรลเลอร์

5.1.1.3 คำสั่งตรวจสอบเงื่อนไขหลายทางเลือกโดยใช้ if

```
รูปแบบ  if ( เงื่อนไข 1 )
        {
            ประโยคคำสั่งเมื่อเงื่อนไขที่ 1 เป็นจริง;
        } else if ( เงื่อนไขที่ 2) {
            ประโยคคำสั่งเมื่อเงื่อนไข 2 เป็นจริง;
        } else if ( เงื่อนไขที่ n) {
            ประโยคคำสั่งเมื่อเงื่อนไข n เป็นจริง;
        } else {
            ประโยคคำสั่งเมื่อเงื่อนไขทั้งหมดเป็นเท็จ;
        }
```


ภาษาซีกับไมโครคอนโทรลเลอร์

5.1.2 กลุ่มคำสั่งตรวจสอบเงื่อนไขในรูปแบบคำสั่ง switch-case

```
รูปแบบ  switch ( ตัวแปร )
        {
            case 1 : ประโยคคำสั่งเมื่อตัวแปรมีค่าเท่ากับ 1 ;
                     break;
            case 2 : ประโยคคำสั่งเมื่อตัวแปรมีค่าเท่ากับ 2 ;
                     break;
            case n : ประโยคคำสั่งเมื่อตัวแปรมีค่าเท่ากับ n ;
                     break;
            default : break;
        }
```


ภาษาซีกับไมโครคอนโทรลเลอร์

5.2 คำสั่งวนรอบการทำงาน

5.2.1 คำสั่ง for()

```
รูปแบบ  for ( ให้ค่าเริ่มต้นแก่ตัวแปร ; ตรวจสอบเงื่อนไข ; เพิ่มค่าหรือลดค่าตัวแปร )
        {
            ประโยคคำสั่งเมื่อเงื่อนไขเป็นจริง;
        }
```

5.2.2 คำสั่ง while()

```
รูปแบบ  while ( เงื่อนไข )
        {
            ประโยคคำสั่งเมื่อเงื่อนไขเป็นจริง;
        }
```


ภาษาซีกับไมโครคอนโทรลเลอร์

5.2.3 คำสั่ง do-while()

```
รูปแบบ   do {
             ประโยคคำสั่งก่อนที่จะทำการตรวจสอบเงื่อนไข;
         } while ( เงื่อนไข );
```

6. การใช้งานขาไมโครคอนโทรลเลอร์ด้วยภาษาซี

การใช้งานขาของไมโครคอนโทรลเลอร์ตระกูล MCS-51,PIC16F และ AVR ในลักษณะขาสัญญาณดิจิทัลจะมีการใช้งาน 2 รูปแบบคือ

6.1 การใช้งานขาไมโครคอนโทรลเลอร์ตระกูล MCS-51,PIC16F และAVR เป็นขาสัญญาณเอาต์พุตด้วยภาษาซี

6.2 การใช้งานขาไมโครคอนโทรลเลอร์ตระกูล MCS-51,PIC16F และ AVR เป็นขาสัญญาณอินพุตด้วยภาษาซี

ภาษาซีกับไมโครคอนโทรลเลอร์

การเขียนโปรแกรมภาษาซีเพื่อกำหนด และใช้งานพอร์ตอินพุตเอาต์พุตของไมโครคอนโทรลเลอร์ตระกูล MCS-51 , PIC16F และ AVR

ตัวอย่างการ include หัวไฟล์

MCS51-51	PIC16F	AVR
#include <reg51.h> /*สำหรับไมโครคอนโทรลเลอร์ที่อ้างอิงไมโครโพรเซสเซอร์ 8051 */ //หรือ #include <reg52.h> /*สำหรับไมโครคอนโทรลเลอร์ที่อ้างอิงไมโครโพรเซสเซอร์ 8052 */	#include <xc.h> /*เป็นการเรียกไฟล์ IO มาตรฐานของ PIC ซึ่งจะจบที่บรรทัดนี้เนื่องตอนเริ่มต้นใช้งาน IDE ได้กำหนดเบอร์ไมโครคอนโทรลเลอร์ตระกูล PIC เรียบร้อยแล้ว*/	#include <avr/io.h> /*เป็นการเรียกไฟล์ IO มาตรฐานของ AVR ซึ่งจะจบที่ บรรทัดนี้ หรือเพิ่มการ include ไปอีก 1 บรรทัดเพื่อระบุเบอร์ของ AVR*/ #include <avr/iom32.h> /*เป็นการเรียกไฟล์ IO ของเบอร์ ATMEGA32*/

ภาษาซีกับไมโครคอนโทรลเลอร์

ตัวอย่างการประกาศตัวแปรที่อ้างถึงขาไมโครคอนโทรลเลอร์

MCS-51	PIC16F	AVR

sbit sw1 = P1^0;
/*กำหนดตัวแปรชื่อ sw1 เพื่อใช้ในอ้างถึงข้อมูลที่ขา P1.0*/
sbit led1 = P1^1;
/*กำหนดตัวแปรชื่อ led1 เพื่อใช้ในการอ้างถึงข้อมูลที่ขา P1.1*/
 	
หมายเหตุ MCS51 ไม่จำเป็นต้องกำหนด Direction ของขาก็สามารถใช้งานเป็นขาอินพุตหรือเอาต์พุตได้
	#define sw1 RB0
	/*กำหนดตัวแปรชื่อ sw1 เพื่อใช้ในอ้างถึงข้อมูลที่ขา RB0 ที่ต้องการให้ขา RB0 ทำหน้าที่เป็นอินพุต*/
	#define led1 RB1
	/*กำหนดตัวแปรชื่อ led1 เพื่อใช้ในการอ้างถึงข้อมูลที่ขา RB1ที่ต้องการให้ขา RB1 ทำหน้าที่เป็นเอาต์พุต*/
	/*เมื่อประกาศตัวแปรแล้วในส่วนฟังก์ชัน main จะต้องทำการกำหนด Direction ให้ขาพอร์ตเหล่านี้ให้ทำหน้าที่เป็นขาอินพุตหรือเอาต์พุตโดยใช้คำสั่ง TRISx=yy โดย x คือชื่อพอร์ต และ yy คือค่าข้อมูลที่กำหนดคุณสมบัติของขาพอร์ต*/
		#define sw1 PINB0
		/*กำหนดตัวแปรชื่อ sw1 เพื่อใช้ในอ้างถึงข้อมูลที่ขา PB0 ที่ต้องการให้ขา PB0 ทำหน้าที่เป็นอินพุต*/
		#define led1 PB1
		/*กำหนดตัวแปรชื่อ led1 เพื่อใช้ในการอ้างถึงข้อมูลที่ขา PB.1ที่ต้องการให้ขา PB.1 ทำหน้าที่เป็นเอาต์พุต*/
		/*เมื่อประกาศตัวแปรแล้วในส่วนฟังก์ชัน main จะต้องทำการกำหนด Direction ให้ขาพอร์ตเหล่านี้ให้ทำหน้าที่เป็นขาอินพุตหรือเอาต์พุตโดยใช้คำสั่ง DDRx=yy โดย x คือชื่อพอร์ต และ yy คือค่าข้อมูลที่กำหนดคุณสมบัติของขาพอร์ต*/

ภาษาซีกับไมโครคอนโทรลเลอร์

ตัวอย่างการกำหนดคุณสมบัติขาพอร์ตของไมโครคอนโทรลเลอร์ให้ทำหน้าที่เป็นอินพุตหรือเอาต์พุต

MCS-51	PIC16F	AVR
MCS51 ไม่จำเป็นต้องกำหนด Direction ของขาก็สามารถใช้งานเป็นขาอินพุตหรือเอาต์พุตได้	TRISB = 0b00000001; /*ค่าที่กำหนดให้รีจิสเตอร์ TRISB เป็นข้อมูลเลขฐานสองโดยเรียงจากซ้ายมาขวาเป็นค่าข้อมูลที่กำหนดการทำงานของขา RB7-RB0 ซึ่งค่า 0 หมายถึงให้ทำงานเป็นขาเอาต์พุต และค่า 1 หมายถึงให้ทำงานเป็นขาอินพุต ดังนั้นจากคำสั่งขา RB7-RB1 จึงทำงานเป็นขาเอาต์พุต ส่วน RB0 ทำงานเป็นขาอินพุต*/	DDRB = 0b00000001; /*ค่าที่กำหนดให้รีจิสเตอร์ DDRB เป็นข้อมูลเลขฐานสองโดยเรียงจากซ้ายมาขวาเป็นค่าข้อมูลที่กำหนดการทำงานของขา PB7-PB0 ซึ่งค่า 0 หมายถึงให้ทำงานเป็นขาอินพุต และค่า 1 หมายถึงให้ทำงานเป็นขาเอาต์พุต ดังนั้นจากคำสั่งขา PB7-PB1 จึงทำงานเป็นขาอินพุตส่วน PB0 ทำงานเป็นขา เอาต์พุต*/

ภาษาซีกับไมโครคอนโทรลเลอร์

ตัวอย่างการส่งค่าข้อมูลออกที่ขาพอร์ตในรูปแบบข้อมูลแบบบิตผ่านตัวแปรที่กำหนด

MCS-51	PIC16F	AVR

led1 = 1;
/*เป็นการส่งค่าข้อมูล 1 หรือให้แรงดัน VCC ออกไปยังขาพอร์ตของไมโครคอนโทรลเลอร์ที่ถูกอ้างถึงด้วยตัวแปร led1*/
 	
led1 = 0;
/*เป็นการส่งค่าข้อมูล 0 หรือให้แรงดัน GND ออกไปยังขาพอร์ตของไมโครคอนโทรลเลอร์ที่ถูกอ้างถึงด้วยตัวแปร led1*/
	led1 = 1;
	/*เป็นการส่งค่าข้อมูล 1 หรือให้แรงดัน VDD ออกไปยังขาพอร์ตของไมโครคอนโทรลเลอร์ที่ถูกอ้างถึงด้วยตัวแปร led1*/
	
	led1 = 0;
	/*เป็นการส่งค่าข้อมูล 0 หรือให้แรงดัน VSS ออกไปยังขาพอร์ตของไมโครคอนโทรลเลอร์ที่ถูกอ้างถึงด้วยตัวแปร led1*/
		PORTB |= _BV(led1);
		หรือ
		PORTB |= (1<<led1);
		/*เป็นการส่งค่าข้อมูล 1 หรือให้แรงดัน VCC ออกไปยังขาพอร์ตของไมโครคอนโทรลเลอร์ที่ถูกอ้างถึงด้วยตัวแปร led1*/
		PORTB &= ~_BV(led1);
		หรือ
		PORTB &= ~(1<<led1);
		/*เป็นการส่งค่าข้อมูล 0 หรือให้แรงดัน GND ออกไปยังขาพอร์ตของไมโครคอนโทรลเลอร์ที่ถูกอ้างถึงด้วยตัวแปร led1*/

ภาษาซีกับไมโครคอนโทรลเลอร์

ตัวอย่างการอ่านค่าข้อมูลจากขาพอร์ตในรูปแบบข้อมูลแบบบิตมาเก็บไว้ในตัวแปรที่กำหนด

MCS-51	PIC16F	AVR
bit dat; dat = sw1; /*เป็นนำค่าข้อมูลขนาด 1 บิตที่ขาพอร์ตที่อ้างถึงด้วยตัวแปร sw1 มาเก็บไว้ที่ตัวแปร dat*/	unsigned char dat; dat = sw1; /*เป็นนำค่าข้อมูลขนาด 1 บิตที่ขาพอร์ตที่อ้างถึงด้วยตัวแปร sw1 มาเก็บไว้ที่ตัวแปร dat โดยตัวแปร dat จะเก็บค่าข้อมูลขนาด 1 byte */	unsigned char dat; dat = PINx & (1<<sw1); /*เป็นนำค่าข้อมูลขนาด 1 บิตที่ขาพอร์ตที่อ้างถึงด้วยตัวแปร sw1 มาเก็บไว้ที่ตัวแปร dat โดยตัวแปร dat จะเก็บค่าข้อมูลขนาด 1 byte*/

ภาษาซีกับไมโครคอนโทรลเลอร์

ตัวอย่างการอ่านค่าข้อมูลจากขาพอร์ตในรูปแบบข้อมูลแบบบิตเพื่อเปรียบเทียบกับค่าข้อมูลที่กำหนด

MCS-51	PIC16F	AVR

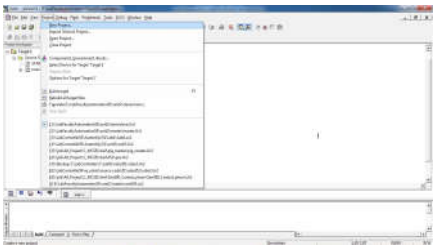
If(sw1 == 0)
{
led1 = 1;
}else{
led1 = 0;
}
/*เป็นการตรวจสอบข้อมูลที่ขาของพอร์ตผ่านตัวแปร sw1 ว่ามีค่าเท่ากับ 0 หรือไม่ ถ้าใช่ให้ทำคำสั่งส่งข้อมูลออกขาพอร์ตผ่านตัวแปร led1 มีค่าเป็น 1 แต่ถ้าไม่ใช่ให้ส่งค่าข้อมูล 0 ออกไปที่ขาพอร์ตผ่านตัวแปร led1*/
	If(sw1 == 0)
	{
	led1 = 1;
	}else{
	led1 = 0;
	}
	/*เป็นการตรวจสอบข้อมูลที่ขาของพอร์ตผ่านตัวแปร sw1 ว่ามีค่าเท่ากับ 0 หรือไม่ ถ้าใช่ให้ทำคำสั่งส่งข้อมูลออกขาพอร์ตผ่านตัวแปร led1 มีค่าเป็น 1 แต่ถ้าไม่ใช่ให้ส่งค่าข้อมูล 0 ออกไปที่ขาพอร์ตผ่านตัวแปร led1*/
		If((PINx & (1<<sw1)) == 0)
		{
		PORTx |= _BV(led1);
		}else{
		PORTx &= ~_BV(led1);
		}
		/*เป็นการตรวจสอบข้อมูลที่ขาของพอร์ตผ่านตัวแปร sw1 ว่ามีค่าเท่ากับ 0 หรือไม่ ถ้าใช่ให้ทำคำสั่งส่งข้อมูลออกขาพอร์ตผ่านตัวแปร led1 มีค่าเป็น 1 แต่ถ้าไม่ใช่ให้ส่งค่าข้อมูล 0 ออกไปที่ขาพอร์ตผ่านตัวแปร led1 โดย x ให้เติมด้วยชื่อของพอร์ตที่เรากำหนดเป็นอินพุตหรือเอาต์พุต*/

ภาษาซีกับไมโครคอนโทรลเลอร์

7. การใช้งานซอฟต์แวร์เพื่อเขียนโปรแกรมภาษาซีของไมโครคอนโทรลเลอร์

7.1 การใช้งานโปรแกรม Keil uVision3 สำหรับเขียนโปรแกรมภาษาซีของไมโครคอนโทรลเลอร์ตระกูล MCS51

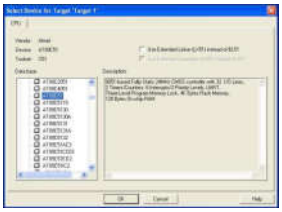
รูปที่ 1.5 รูปแสดงไอคอนของโปรแกรม Keil uVision3



รูปที่ 1.6 รูปแสดงโปรแกรม Keil uVision3

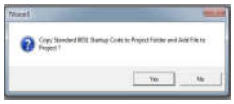
ภาษาซีกับไมโครคอนโทรลเลอร์

รูปที่ 1.7 รูปแสดงหน้าต่าง Create New Project

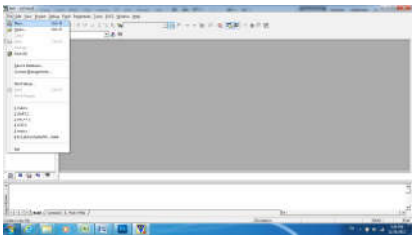


รูปที่ 1.8 รูปแสดงหน้าต่าง Select Device for Target 'Target1'

ภาษาซีกับไมโครคอนโทรลเลอร์

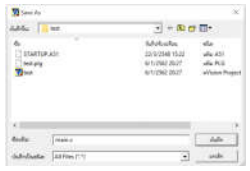


รูปที่ 1.9 รูปแสดงหน้าต่างให้เลือกสร้างไฟล์ 8051 Startup Code

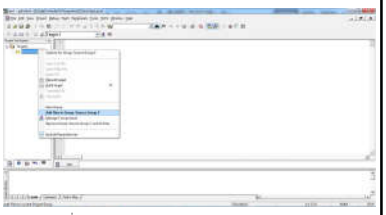


รูปที่ 1.10 รูปแสดงขั้นตอนสร้าง new file

ภาษาซีกับไมโครคอนโทรลเลอร์



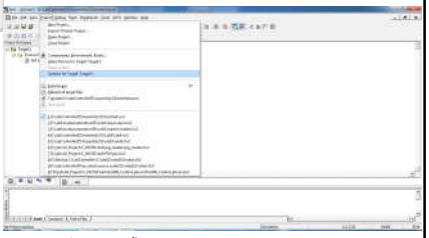
รูปที่ 1.11 แสดงรูปหน้าต่างการตั้งชื่อไฟล์นามสกุล .c



รูปที่ 1.12 แสดงหน้าต่างการเลือกการเลือก Add File to Group "Source Group 1"

ภาษาซีกับไมโครคอนโทรลเลอร์

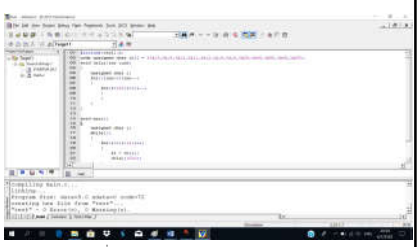
รูปที่ 1.13 แสดงการเพิ่มไฟล์นามสกุล .c เข้าสู่ Source Group 1



รูปที่ 1.14 แสดงขั้นตอนการเลือกเมนู Options for Target 'Target 1'

ภาษาซีกับไมโครคอนโทรลเลอร์

รูปที่ 1.15 รูปแสดงการเลือกเมนู Create HEX File

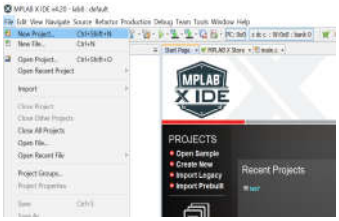


รูปที่ 1.16 รูปแสดงการตัวอย่างการเขียนโปรแกรมภาษาซีและ Build target ได้ 0 Error

ภาษาซีกับไมโครคอนโทรลเลอร์

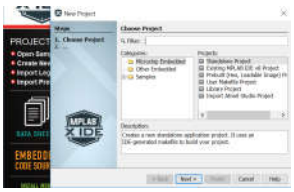
7.2 การใช้งานโปรแกรม MPLAB X สำหรับเขียนโปรแกรมภาษาซีของไมโครคอนโทรลเลอร์ตระกูล PIC16F

รูปที่ 1.17 แสดงรูปไอคอนของโปรแกรม MPLAB X IDE V4.20

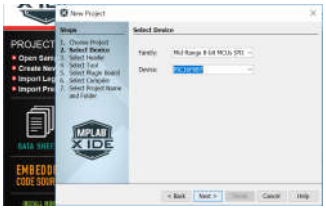


รูปที่ 1.18 แสดงวิธีการสร้าง New project

ภาษาซีกับไมโครคอนโทรลเลอร์

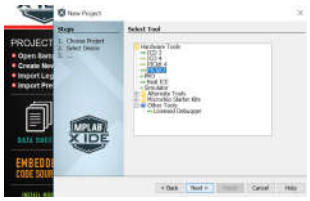


รูปที่ 1.19 แสดงหน้าต่าง New project Choose Project

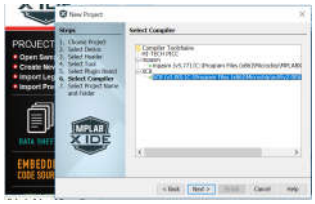


รูปที่ 1.20 แสดงหน้าต่าง Select Device

ภาษาซีกับไมโครคอนโทรลเลอร์



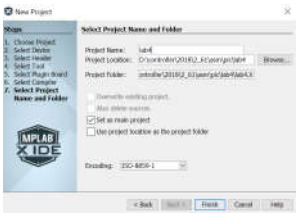
รูปที่ 1.21 แสดงหน้าต่าง Select Tool



รูปที่ 1.22 หน้าต่างแสดง Select Compiler

ภาษาซีกับไมโครคอนโทรลเลอร์

รูปที่ 1.23 แสดงหน้าต่าง Select Project Location

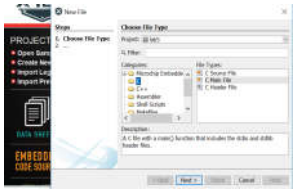


รูปที่ 1.24 แสดงหน้าต่าง Select Project Name and Folder

ภาษาซีกับไมโครคอนโทรลเลอร์

รูปที่ 1.25 แสดงการเตรียมการสร้างไฟล์ main.c

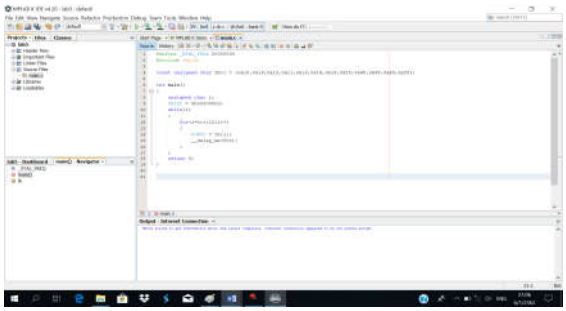
ภาษาซีกับไมโครคอนโทรลเลอร์



รูปที่ 1.26 หน้าต่างแสดงให้เลือกชนิดของไฟล์ที่ต้องการสร้าง

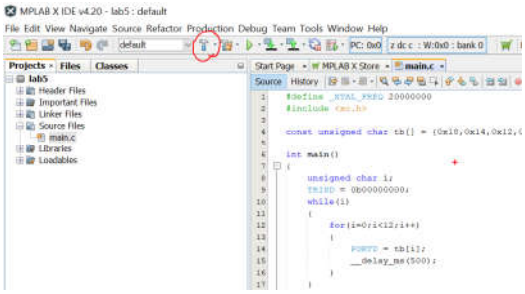
รูปที่ 1.27 รูปหน้าต่างแสดงการสร้างไฟล์ main.c

ภาษาซีกับไมโครคอนโทรลเลอร์



รูปที่ 1.28 รูปแสดงการตัวอย่างการเขียนโปรแกรมภาษาซีของ
ไมโครคอนโทรลเลอร์ PIC16F887

ภาษาซีกับไมโครคอนโทรลเลอร์



รูปที่ 1.29 รูปแสดงตำแหน่งของเครื่องมือ Build Main Project

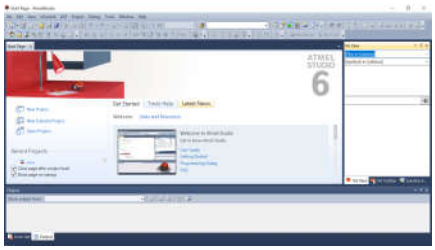
ภาษาซีกับไมโครคอนโทรลเลอร์

รูปที่ 1.30 รูปแสดงการ Build Main Project แล้ว
หน้าต่าง Output แสดงการ Build Success

ภาษาซีกับไมโครคอนโทรลเลอร์

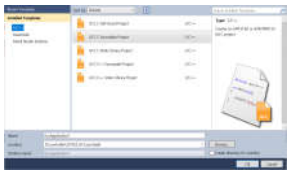
7.3 การใช้งานโปรแกรม AVR Studio 6.2 สำหรับเขียนโปรแกรม ภาษาซีของไมโครคอนโทรลเลอร์ตระกูล AVR

รูปที่ 1.31 รูปแสดงไอคอนของ
โปรแกรม AVR Studio 6.2

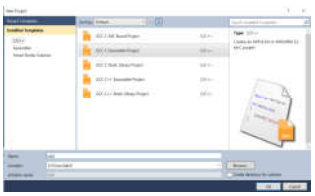


รูปที่ 1.32 รูปภาพแสดงหน้าต่างของโปรแกรม AVR Studio 6.2

ภาษาซีกับไมโครคอนโทรลเลอร์

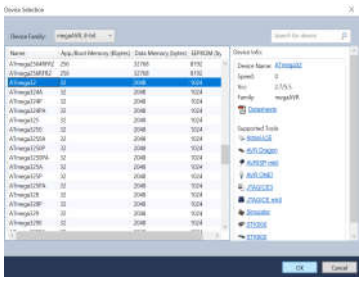


รูปที่ 1.33 แสดงรูปหน้าต่างของ New Project



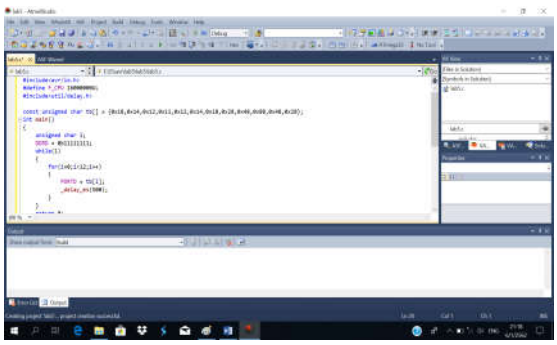
รูปที่ 1.34 แสดงรูปหน้าต่างของ New Project ที่ตั้งชื่อในช่อง
name และเลือก location เรียบร้อยแล้ว

ภาษาซีกับไมโครคอนโทรลเลอร์



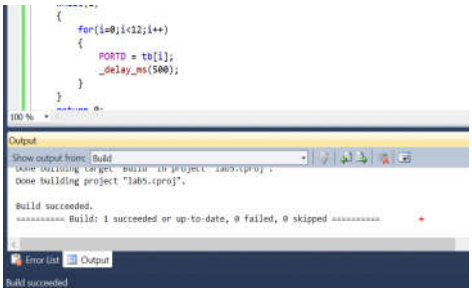
รูปที่ 1.35 รูปหน้าต่างแสดงการเลือกเบอร์ไมโครคอนโทรลเลอร์

ภาษาซีกับไมโครคอนโทรลเลอร์



รูปที่ 1.36 รูปแสดงตัวอย่างการเขียนโปรแกรมภาษาซีของไมโครคอนโทรลเลอร์ AVR

ภาษาซีกับไมโครคอนโทรลเลอร์



รูปที่ 1.37 รูปแสดงการ Build แล้วไม่เกิดข้อผิดพลาด