ภาษาซีสำหรับ
AVR Microcontroller

รศ.ณรงค์ บวบทอง
ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
มหาวิทยาลัยธรรมศาสตร์

วัตถุประสงค์

- เพื่อให้มีความเข้าใจเกี่ยวกับภาษา ซี
- เพื่อให้เขียนโปรแกรมภาษา ซี สำหรับไมโครคอนโทรลเลอร์ได้

ความรู้เบื้องต้นของภาษา C

- พ.ศ.2515 เดนนิส รีตชี่ (Dennis Ritchie) ได้พัฒนาภาษา C ขึ้น
- พ.ศ. 2531 ได้รูปแบบมาตราฐานของภาษา C ที่เรียกว่า ANSI-C

ตัวแปลภาษา C สำหรับ AVR Microcontroller

WinAVR™ (pronounced "whenever") is a suite of executable, open source software development tools for the Atmel AVR series of RISC microprocessors hosted on the Windows platform. It includes the GNU GCC compiler for C and C++.

รูปแบบโครงสร้างภาษา C

1. **Header File**

```
#include
```

2. **ประกาศค่าคงที่และตัวแปรชนิดโกลบอล**

```
#define
```

3. **ฟังก์ชัน**

```
function
```

4. **ฟังก์ชัน main**

```
main()
{

}
```


การประกาศ Header File หรือพรีโพรเซสเซอร์ไดเร็คทีฟ(Preprocessor Directive)

```
#include
```

```
#define
```

```
function
```

```
main()
{

}
```

ในส่วนนี้จะเป็นการแจ้งให้คอมโพล์เลอร์ทราบถึงไฟล์ ต่างๆ ที่เก็บ ชุดคำสั่งที่ซีพียูสามารถจะเรียกใช้ได้ เป็นชุดคำสั่งที่ได้เตรียมให้ซีพียูสามารถเรียกใช้ได้ภายในโปรแกรม โดยก่อนที่จะมีการคอมไพล์โปรแกรมให้เป็นภาษาเครื่องนั้นคอมโพล์เลอร์จะแปลคำสั่งในส่วนนี้ก่อน เช่น

```
// ชุดคำสั่งมาตรฐานอินพุท เอาท์พุท
#include <stdio.h>

//ชุดคำสั่งอินพุทเอาท์พุทพอร์ทของ AVR
#include <avr/io.h>
```


ส่วน Preprocessor

คอมโพล์เลอร์ ใช้ส่วนนี้เป็นบอกให้รู้ว่าในไมโครคอนโทรลเลอร์เบอร์นั้นมีอะไรบ้าง เช่นรีจิสเตอร์และบิตควบคุมต่างๆ ของคอนโทรลเลอร์นั้น
การประกาศใช้เครื่องหมายชาร์ป (#) และไดเร็กทีฟ include ร่วมกับชื่อไฟล์ที่บอกรายละเอียดของคอนโทรลเลอร์

```
#include <AT89X051.h>
```
เป็นไฟล์ Preprocessor ของตระกูล AT89C1051 และ AT89C2051

```
#include <avr/io.h>
```
ชุดคำสั่งอินพุทเอาท์พุทพอร์ทของ AVR

ส่วนไลบรารี (Library)

ไลบรารี่เป็นไฟล์ที่มีฟังก์ชั่นบรรจุอยู่ภายใน ไฟล์นี้มีนามสกุลเป็น .h มาพร้อมกับตัวแปล สามารถตรวจสอบได้จากไฟล์ Help
การประกาศใช้ไดเร็กทีฟ #include แล้วต่อด้วยชื่อไฟล์

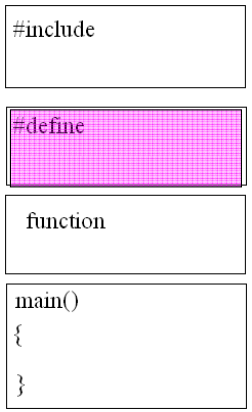
#include <stdio.h>
ประกอบด้วยฟังก์ชั่นที่ใช้ติดต่อกับพอร์ทอนุกรม ตัวอย่างฟังก์ชั่นได้แก่ printf _getkey getchar gets putchar puts scanf sprintf sscanf ungetchar vprintf และ vsprintf

#include <math.h>
ประกอบด้วยฟังก์ชั่นที่ใช้คำนวณทางคณิตศาสตร์

#include <string.h>
ประกอบด้วยฟังก์ชั่นที่ใช้จัดการเกี่ยวกับสตริง (String) และบัฟเฟอร์ (Buffer)

#include < intrins.h >
ประกอบด้วยฟังก์ชั่นเกี่ยวกับการหมุนบิต และการตรวจสอบสถานะของเลขแบบ Floating-point

ส่วนประกาศค่าคงที่และตัวแปรชนิดโกลบอล



การกำหนดค่าคงที่ และชื่อแทน (Definitions)
เป็นการกำหนดชื่อแทนให้กับค่าคงที่ หรือชุดคำสั่ง ต่างๆ

```
#define max_time 10
#define cpu_clk 8000000

#define SW0_CLOSE (PINB & 0b00000001) == 0
#define SW0_OPEN (PINB & 0b00000001)==1
```

การประกาศค่าตัวแปร (Declarations)
เป็นการประกาศใช้งานตัวแปรด้วยการกำหนดตัวแปร และชนิดข้อมูลของตัวแปร เช่น

```
unsigned char i,j; // ประกาศค่าตัวแปร i และ j เป็น
                   // ชนิด unsigned char
int max = 10;      //ประกาศค่าตัวแปร max เป็น
                   // ชนิด int พร้อมทั้งกำหนดค่าเริ่มต้น
```


ส่วนการใช้งานร่วม (Global Declarations)

ตัวแปรแบบใช้งานร่วมหรือ
ตัวแปรแบบโกลบอล
หมายถึงตัวแปรที่สามารถเรียกใช้ได้ทุกที ภายในโปรแกรม
การประกาศให้ตัวแปรเป็นแบบโกลบอล ให้ประกาศไว้นอกขอบเขตของส่วนโปรแกรมใดๆ

ตัวแปรแบบโลคัล (Local)
ตัวแปรใช้ได้เฉพาะภายในส่วนของโปรแกรมใดโปรแกรมหนึ่ง
การประกาศให้ประกาศไว้ภายในโปรแกรมที่ต้องการใช้งาน

```
/*-------------------------------------------*/
#include <avr/io.h>
#include <stdio.h>
unsigned int i;
Static void delay_1s(void)
{
  ...
}
/*------MAIN C function  -------------------*/
void main (void)
{
    char c;
    USART_Init(51);
    while (1)
    {
     delay_1s();
     fprintf (&uart_str"\nYou program\n");
    }
}
```


ส่วนค่าคงที่ของโปรแกรม

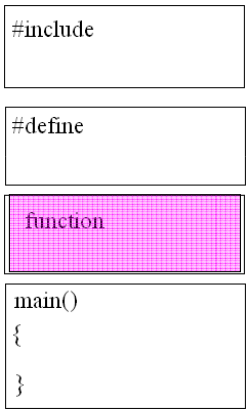
การประกาศค่าคงที่ในภาษา C ใช้ไดเร็กทีฟ #defined แล้วตามด้วยค่าคงที่ดังนี้

```
#define setValue 99          กำหนดค่า setValue ให้เป็น 99
```

หรือใช้แทนการเขียนคำสั่งเพื่อให้สื่อความหมายได้ชัดเจน

```
#define LED_ON PORTD |= 0b00000001
#define LED_OFF PORTD &= 0b11111110
```


ส่วนของการสร้างฟังก์ชัน



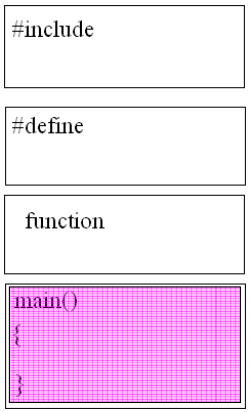
เป็นส่วนที่ผู้ใช้งานสามารถสร้างฟังก์ชัน หรือโปรแกรมย่อยเพื่อการใช้งานต่างๆ ได้เองตามที่ต้องการ โดยปกติ จะเขียนไว้ก่อนฟังก์ชั่น main ( ) เพื่อให้ฟังก์ชั่น main ( ) สามารถเรียกใช้งานฟังก์ชั่นต่างๆ ที่เราสร้างขึ้นมาได้

ฟังก์ชั่น (Function)

- โปรแกรมหนึ่งๆ จะมีกี่ฟังก์ชั่นก็ได้
- ต้องมีฟังก์ชั่นหลักอยู่หนึ่งฟังก์ชั่น ใช้เป็นฟังก์ชั่นสำหรับเริ่มการทำงานของโปรแกรม เรียกฟังก์ชั่นนี้ว่า main
- ส่วนของคำสั่งของฟังก์ชั่นจะอยู่ภายในวงเล็บปีกกา { }และสามารถกำหนดให้มีการส่งผ่านตัวแปรได้ หรืออาร์กิวเมนต์ของฟังก์ชั่น (Function Arguments) ถ้าไม่ต้องการให้ส่งตัวแปรเข้าจะใช้คำว่า void ไว้ภายในวงเล็บและถ้าไม่มีการส่งค่าคืน จะใส่คำว่า void ไว้หน้าชื่อฟังก์ชั่น ตามมาตราฐาน ANSI C มีได้สูงสุด 31 ตัว ตัวอย่างเช่น

```
void   func_abcd(void)
{
.....
}
void delay_ms(int t)  //ฟังก์ชั่นนี้มี อาร์กิวเมนต์หรือพารามิเตอร์ส่งให้ฟังก์ชั่น 1 ตัว
{
}
```


ฟังก์ชัน main ( )



สำหรับฟังก์ชั่นนี้ถือเป็นฟังก์ชั่นหลักของโปรแกรมภาษา C ที่จำเป็นต้องมีเสมอและจะมีได้เพียงฟังก์ชั่นเดียวเท่านั้น หน้าที่สำคัญของฟังก์ชั่นนี้คือ เป็นจุดเริ่มต้นการทำงานของโปรแกรมนั้นเอง ดังนั้นในการเขียนโปรแกรมจึงเริ่มต้นที่ฟังก์ชั่นนี้เสมอ

ส่วนหมายเหตุ (Comment)

- เป็นส่วนที่ให้ผู้เขียนโปรแกรมใช้อธิบายความหมายของโปรแกรมหรือคำสั่งที่เขียน เพื่อให้ง่ายต่อการทำความเข้าใจ ส่วนนี้ต้องอยู่ในเครื่องหมาย /* ..... */ หรืออยู่หลังเครื่องหมาย //

```
/*--------------------------------------------------------------------------------
Set serial port for 9600 baud at 11.0592 MHz.  Note that we use Timer 1
for the baud rate generator.
--------------------------------------------------------------------------------*/

/* -------------------------  Main Function --------------------------------*/


// PS2 I/P Program
```


ข้อกำหนดอื่นๆ

- คำสั่งส่วนใหญ่ต้องเขียนด้วยอักษรตัวพิมพ์เล็ก
- ตัวแปรสามารถตั้งชื่อเป็นตัวพิมพ์เล็กหรือใหญ่ก็ได้แต่ต้องเรียกใช้ให้ถูกตามตัวพิมพ์ และต้องไม่ตรงกับคำสงวนของตัวคอมไพเลอร์ที่ใช้
- ทุกคำสั่งต้องมีเครื่องหมาย ; แสดงการจบคำสั่ง และสามารถเขียน 1 คำสั่งใน 1 บรรทัดหรือจะเขียนต่อกันยาวก็ได้ แต่ก็จะทำ าให้แก้ไขโปรแกรมได้ยาก
- คำสั่งใดๆในโปรแกรมสามารถมีเลเบลได้ถ้าต้องการ และต้องมีเครื่องหมาย : ตามท้ายด้วย

ข้อระบุอื่นๆ

- นิพจน์ (Expressions)
  คือการกระทำระหว่างตัวดำเนินการ (Operators) กับตัวกระทำ (Operands) เพื่อให้เกิดค่าใดค่าหนึ่ง
  i = j+10;
- สเตทเมนต์ (Statements)
  หมายถึงคำสั่งเพื่อให้เกิดการทำการตามความต้องการ

```
while(i<10)
{
      a = i+3;
      i = i+1
}
```

ตัวอย่าง

```
/*-------------------------------------------*/
#include <avr/io.h>
#include <stdio.h>
unsigned int i;
Static void delay_1s(void)
{
  ...
}
/*------MAIN C function  ------------------*/
void main (void)
{
    char c;
    USART_Init(51);
    while (1)
    {
     delay_1s();
     fprintf (&uart_str"\nYou program\n");
    }
 }
```

การประกาศ Preprocessor ชุดคำสั่งอินพุท เอาท์พุทพอร์ทของ AVR

การประกาศไลบรารี่ stdio สำหรับฟังก์ชั่นการใช้งานพอร์ทอนุกรม และ ไลบรารี่ ctype

การเขียนฟังก์ชั่น

ส่วนหลักของโปรแกรม ไม่มีการส่งข้อมูลเข้าและไม่มีข้อมูลออก

เรียกใช้ฟังก์ชั่น delay_1s

ฟังก์ชั่น printf เป็นการส่งข้อมูลออกทางพอร์ทอนุกรม

ค่าคงที่และตัวแปร

ชนิดของตัวแปร

ชนิด (Type)	ชื่อแทน	ขนาดความกว้าง		ช่วงของค่าที่เก็บ(Range)
		Bits	Bytes	
char		8	1	-128 to +127
signed char	int8_t	8	1	-128 to +127
unsigned char	uint8_t	8	1	0 to 255
int		16	2	-32768 to +32767
signed int	int16_t	16	2	-32768 to +32767
unsigned int	uint16_t	16	2	0 to 65535
long		32	4	-2147483648 to 2147483647
signed long int	int32_t	32	4	-2147483648 to 2147483647
unsigned long int	uint32_t	32	4	0 to 4294967296
signed long long int	int64_t	64	8	
unsigned long long int	uint64_t	64	8	0 to 2^64
float		64	8	3.4x10-38 to 3.4x1038

ตัวดำเนินการ

ตัวดำ าเนินการทางคณิตศาสตร์

เครื่องหมาย	ความหมาย
+	บวก
-	ลบ
*	คูณ
/	หาร
%	หารแบบใช้เศษของการหาร(Mod)

ตัวดำเนินการ

ตัวดำเนินการที่ใช้ในการแปลงค่า

เครื่องหมาย	ความหมาย
=	นำค่าทางขวามาให้ทางซ้าย
++	เพิ่มค่าขึ้น 1
--	ลดค่าลง 1
+=	เพิ่มค่าเท่ากับค่าทางขวา
-=	ลดค่าเท่ากับค่าทางขวา
*=	นำตัวเองคูณกับค่าทางขวา
/=	นำตัวเองหารกับค่าทางขวา
%=	นำตัวเองหารกับค่าทางขวาเอาเศษ
&=	นำตัวเองมาAND กับค่าทางขวา
|=	นำตัวเองมา OR กับค่าทางขวา
^=	นำตัวเองมา XORกับค่าทางขวา
<<=	นำตัวเองมาเลื่อนบิตไปทางซ้ายจำนวนครั้งเท่ากับค่าทางขวา
>>=	นำตัวเองมาเลื่อนบิตไปทางขวาจำ นวนครั้งเท่ากับค่าทางขวา

ตัวดำเนินการ

ตัวดำเนินการที่ใช้ในการเปรียบเทียบและตัวดำเนินการทางบิต

เครื่องหมาย	ความหมาย
==	เท่ากับ
!=	ไม่เท่ากับ
!	ตรงกันข้าม(NOT)
&&	และ (AND)
>=	มากกว่าหรือเท่ากับ
<=	น้อยกว่าหรือเท่ากับ
||	หรือ(OR)
<	น้อยกว่า
>	มากกว่า

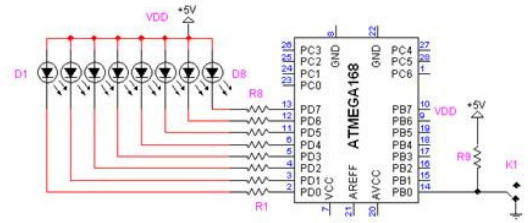
เครื่องหมาย	ความหมาย
&	AND บิต
|	OR บิต
^	XOR บิต
<<	Left shift บิต
>>	Right shift บิต
~	One' Complement (Inverse)

ตัวอย่างโปรแกรมที่ 1 กำหนดค่า และ เลื่อนข้อมูล

```
#include <avr\io.h>              // Most basic include files
#include <avr\interrupt.h>       // Add the necessary ones
#include <avr\signal.h>          // here

// Define here the global static variables
//
int a;
// ****************************************************************
// Main program
//
int main(void) {
  a = 0x80;
  while(1) {             // Infinite loop; define here the
    // my_function();     // system behaviour
    a>>=2;
  }
```

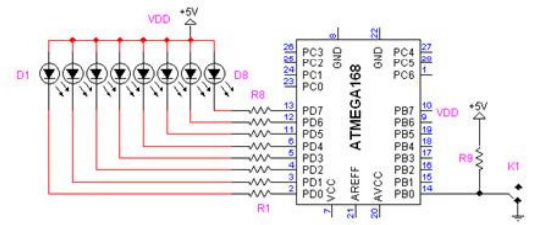

ตัวอย่างโปรแกรมที่ 2 การส่งค่าออกพอร์ทเอาท์พุท



ตัวอย่างโปรแกรมที่ 2 การส่งค่าออกพอร์ทเอาท์พุท

```
#include <avr\io.h>
#include <stdio.h>
#include <util/delay.h>
unsigned int a,i;
// ***********************************************
// Main program
int main(void)
{
      DDRD = 0xff;  //Port D as output
      DDRB = 0xfe;
      a = 1;
      i = 0;
      while(1)
      {           // Infinite loop; define here the
                  PORTD = ~a;
                  _delay_ms(300);
                  a<<=1;
                  i++;
                  if(i==8)
                  {
                        i=0;
                        a = 1;
                  }
      }
}
```


ตัวอย่างโปรแกรมที่ 3 การอ่านค่าจากพอร์ทอินพุท



ตัวอย่างโปรแกรมที่ 3 การอ่านค่าจากพอร์ทอินพุท

```
#include <avr\io.h>             // Most basic include files
#include <stdio.h>
#include <util/delay.h>
unsigned int a,i;
#define SW0_CLOSE (PINB & 0b00000001) == 0
#define SW0_OPEN (PINB & 0b00000001)==1
int main(void)
{
      DDRD = 0xff;  //Port D as output
      DDRB = 0xfe;
      a = 1;
      i = 0;
      while(1)
      {           // Infinite loop; define here the
                  PORTD = ~a;
                  _delay_ms(100);
```


ตัวอย่างโปรแกรมที่ 3 การอ่านค่าจากพอร์ทอินพุท

```
      if(SW0_CLOSE)
      {
            a<<=1;
            i++;
            if(i==8)
            {
                  i=0;
                  a = 1;
            }
      }
      else
      {
            a>>=1;
            if(i==0)
            {
                  i=7;
                  a = 128;
            }
            else
            {
                  i--;
            }
      }
  }
}
```