ภาษาซีสำหรับ
AVR Microcontroller

รศ.ฉรงค์ บวบทอง
ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
มหาวิทยาลัยธรรมศาสตร์

ทึ่ม

ทึ่มเข้าสอบทีไรใช้ลอกเพื่อน
ครูขู่เตือนจะไล่ออกถ้าจับได้
สอบปลายปีทึ่มจึงไม่ลอกใคร
ทำไม่ได้กระดาษเปล่าเอาส่งคืน

พวกเพื่อนเพื่อนที่ให้ทึ่มได้ลอก
สอบเสร็จบอกกับทึ่มว่าอย่าขมขื่น

ขนาดกูดูหนังสือมาทั้งคืน
กูยังยื่นกระดาษเปล่าเอาส่งครู

ทึ่มโมโหตัดพ้อแล้วต่อว่า
กูอุตสาห์ไม่ลอกมึงตามคำขู่
มึงดันส่งกระดาษเปล่าเอาเหมือนกู
เดี๋ยวคุณครูเขารู้หรอกว่าลอกกัน

วัตถุประสงค์

- เพื่อให้มีความเข้าใจเกี่ยวกับภาษา ซี
- เพื่อให้เขียนโปรแกรมภาษา ซี สำหรับไมโครคอนโทรลเลอร์ได้

3

ความรู้เบื้องต้นของภาษา C

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

4

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++.

WINAVR

5

รูปแบบโครงสร้างภาษา C

```
#include
```

```
#define
```

```
function
```

```
main()
{

}
```

การประกาศ Header File หรือพรีโพรเซสเซอร์ไดเร็คทีฟ(Preprocessor Directive) ในส่วนนี้จะเป็นการแจ้งให้คอมโพล์เลอร์ทราบถึงไฟล์ ต่างๆ ที่เก็บ ชุดคำสั่งที่ซีพียูสามารถจะเรียกใช้ได้ เป็นชุดคำสั่งที่ได้เตรียมให้ซีพียูสามารถเรียกใช้ได้ภายในโปรแกรม โดยก่อนที่จะมีการคอมไพล์โปรแกรมให้เป็นภาษาเครื่องนั้นช คอมโพล์เลอร์จะแปลคำสั่งในส่วนนี้ก่อน เช่น

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

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

6

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

7

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

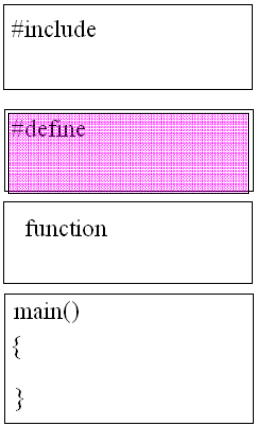
#include <math.h>
ประกอบด้วยฟังก์ชั่นที่ใช้คำนวณทางคณิตศาสตร์

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

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

8

รูปแบบโครงสร้างภาษา C



ส่วนประกาศค่าคงที่และตัวแปรชนิดโกลบอล

การกำหนดค่าคงที่ และชื่อแทน (Definitions)
เป็นการกำหนดชื่อแทนให้กับค่าคงที่ หรือชุดคำสั่ง ต่างๆ
#define max_time 10
#define cpu_clk 8000000

การประกาศค่าตัวแปร (Declarations)
เป็นการประกาศใช้งานตัวแปรด้วยการกำหนดตัวแปร และชนิดข้อมูลของตังวแปร เช่น

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

9

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");
    }
}
```

10

ส่วนค่าคงที่ของโปรแกรม

การประกาศค่าคงที่ในภาษา C ใช้ไดเร็กทีฟ #defined แล้วตามด้วย ค่าคงที่ดังนี้

#define setValue 99 กำหนดค่า setValue ให้เป็น 99

หรือใช้แทนการเขียนคำสั่งเพื่อให้สื่อความหมายได้ชัดเจน

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

11

รูปแบบโครงสร้างภาษา C

```
#include

#define

function

main()
{

}
```

ส่วนของการสร้างฟังก์ชัน

เป็นส่วนที่ผู้ใช้งานสามารถสร้างฟังก์ชัน หรือโปรแกรมย่อยเพื่อการใช้งานต่างๆ ได้เองตามที่ต้องการ โดยปกติจะเขียนไว้ก่อนฟังก์ชั่น main ( ) เพื่อให้ฟังก์ชั่น main ( ) สามารถเรียกใช้งานฟังก์ชั่นต่างๆ ที่เราสร้างขึ้นมาได้

12

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
{
}
```

13

รูปแบบโครงสร้างภาษา C

```
#include

#define

function

main()
{
}
```

ฟังก์ชัน main ( )

สำหรับฟังก์ชั่นนี้ถือเป็นฟังก์ชั่นหลักของโปรแกรมภาษา C ที่จำเป็นต้องมีเสมอและจะมีได้เพียงฟังก์ชั่นเดียวเท่านั้น หน้าที่สำคัญของฟังก์ชั่นนี้คือ เป็นจุดเริ่มต้นการทำงานของโปรแกรมนั่นเอง ดังนั้นในการเขียนโปรแกรมจึงเริ่มต้นที่ฟังก์ชั่นนี้เสมอ

14

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
```

15

ข้อกำหนดอื่นๆ

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

16

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 ชุดคำสั่งอินพุทเอาท์พุทพอร์ทของ 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

19

ตัวดำเนินการ

ตัวดำ าเนินการทางคณิตศาสตร์

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

20

ตัวดำเนินการ

ตัวดำเนินการที่ใช้ในการแปลงค่า

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

21

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)

22

อะเรย์และพอยน์เตอร์ (Array and Pointer)

ตัวแปร์อะเรย์

ตัวแปรอะเรย์ คือกลุ่มของตัวแปร เช่นกลุ่มของตัวแปร char กลุ่มของตัวแปร int สมาชิกของกลุ่มตัวแปรนี้มีจำนวนจำกัด เฉพาะเจาะจง เมื่อกำหนดจำนวนแล้วไม่สามารถลดหรือเพิ่มได้ ดังนั้นการกำหนดอะเรย์สำหรับไมโครคอนโทรลเลอร์ต้องคำนึงถึงหน่วยความจำที่มีใช้งานด้วย อะเรย์นี้สามารถกำหนดให้เป็นแบบหลายๆมิติได้ ขึ้นอยู่กับว่าตัวแปรภาษาย่อมให้มีได้เท่าไร

รูปแบบการกำหนดอะเรย์

แบบ 1 มิติ ชนิดของตัวแปร ชื่อตัวแปร[n]
แบบ 2 มิติ ชนิดของตัวแปร ชื่อตัวแปร[n][m]
โดย n และ m หมายถึงจำนวนสมาชิก

23

ตัวอย่างตัวแปรและการกำหนดค่าให้กับตัวแปร

```
void main( void )
{
        unsigned char x, y[3], z[2][3];
        x = 4;
        y[0] = 1;
        y[1] = 2;
        y[2] = 3;

        z[0][0] = 10;
        z[0][1] = 11;
        z[0][2] = 12;
        z[1][0] = 13;
        z[1][1] = 14;
        z[1][2] = 15;
}
```

- x เป็นตัวแปร
- y เป็นตัวแปร 1 มิติ
- z เป็นตัวแปร 2 มิติ

24

ตัวแปรแบบพอยเตอร์

ตัวแปรพอยเตอร์หรือตัวแปรตัวชี้ เป็นตัวแปรที่มีความสำคัญมากในภาษาซี เพราะว่าให้ความยืดหยุ่นในการใช้งานได้ดี คล้ายกับภาษาแอสเซมบลี้ ตัวแปรแบบพอยเตอร์มีหน้าที่เก็บตำแหน่งของตัวแปรอื่นๆ ว่าอยู่ที่ใดในหน่วยความจำ

รูปแบบการกำหนดตัวแปรพอยเตอร์

ชนิดของตัวแปร ***ชื่อตัวแปร**

25

ตัวอย่างการใช้ตัวแปรแบบพอยเตอร์

```
void main( void )
{
        unsigned char *ptr1;
        unsigned char *ptr2;
        unsigned char a;
        ptr1 = 0x40;
        ptr2 = 0x50;
        a = *ptr1;
        a = a+5;
        *ptr2 = a;
}
```

เป็นการกำหนดพอยเตอร์ขึ้น 2 ตัว เพื่อใช้ชี้ตำแหน่งของหน่วยความจำข้อมูลภายใน

ตำแหน่งที่ 40H

และตำแหน่งที่ 50H

และระบุตัวแปร a เป็นแบบ Unsigned char

การอ่านข้อมูลจากหน่วยความจำมาใว้ที่ a ก็จะอ่านที่ละ 8 บิต

เท่านั้น แต่ถ้าระบุ a เป็นตัวแปรแบบอื่นเช่น char ก่านอ่านก็จะทำที่ละ 16 บิต ดังนั้นการกำหนดชนิดตัวแปรนี้ต้องระมัดระวังให้ดี

26

ตัวอย่างการใช้ตัวแปรแบบพอยเตอร์ 2

```
unsigned char c;
#include <avr/io.h>

int main(void)
{
        unsigned char *ptr1;
        unsigned char *ptr2;
        unsigned char a,c,i;
        unsigned char mydata[10];
        DDRD = 0x00;
        PORTD = 0XFF;
        DDRB = 0XFF;
        c = PIND;
        ptr1 = &mydata;          // =  start address of mydata
        ptr2 = &mydata+1;        // =  end address of mydata + 1
        for(i=1;i<11;i++)
        {
                mydata[i-1] = i;
        }
        *ptr1 = c;
        a = *ptr1;
        a = a+5;
        *ptr2 = a;
        c = *ptr2;
        PORTB = c;
}
```

27

คำสั่งควบคุมต่างๆในภาษา C

- If
- if-else
- Switch
- For
- While
- do-while
- GOTO

28

คำสั่ง

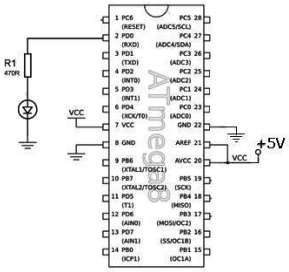
break
หยุดการทำงานของลูป และออกจากลูป

continue;
ตรงข้ามกับคำสั่ง break การทำงานของคำสั่ง continune จะบังคับให้โปรแกรมกระโดดข้ามไปทำงานในรอบต่อไปเลยโดยไม่ต้องทำคำสั่งที่อยู่ถัดไป

```
for (i=0;i<10;i++)
{
    if(i==5)
    {
        continune;
    }
    printf("%d\n\r",i);
}
```

ตัวอย่างโปรแกรมไฟกระพริบที่ พอร์ท D บิต 0

```
#define F_CPU 1000000UL  /* 1 MHz CPU clock */
#include <stdio.h>
#include <util/delay.h>
#include <avr/io.h>
#define LED_ON PORTD |= 0b00000001
#define LED_OFF PORTD &= 0b11111110
int main(void)
{
    DDRD = 1;
    while(1)
    {
        LED_ON;
        _delay_ms(500);
        LED_OFF;
        _delay_ms(500);
    }
}
```



30

Intel HEX file format

Intel HEX file format (*.HEX) เป็นไฟล์รหัส ASCII ซึ่งสามารถใช้โปรแกรม EDITOR ทั่วไปเรียกขึ้นมาดูได้

:**BC AAAA TT HHH…HH CC**

- : เครื่องหมายโคลอน ใช้เป็นตัวเริ่มต้นข้อมูล
- BC จำนวนไบท์ของข้อมูลในเรคคอร์ด มีค่าเป็นเลขฐาน 16 (HEX) (เป็น 00 ถ้าเป็นเรคคอร์ดสิ้นสุดไฟล์)
- AAAA เป็นแอดเดรสข้อมูลไบท์แรกในเรคคอร์ด
- TT แสดงชนิดของเรคคอร์ด
  = 00 ถ้าเป็นเรคคอร์ดข้อมูล
  = 01 ถ้าเป็นเรคคอร์ดสิ้นสุดไฟล์
- HH ข้อมูล 1 ไบท์ จำนวนตามที่ระบุไว้ ใน BC
- CC ค่าของ CHECKSUM ซึ่งเป็นค่า two's complement ของผลบวกของข้อมูลทุกไบท์ในเรคคอร์ด

ตัวอย่างไฟล์เฮกซ์

```
:10 0000 00 0C 94 2A 00 0C 94 3F 00 0C 94 3F 00 0C 94 3F 00 89
:100010000C943F000C943F000C943F000C943F0064
:100020000C943F000C943F000C943F000C943F0054
:00000001FF
```

จบ

33