COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

EGCO334: Microprocessor and Interfacing

AVR C Programming

Outline

- AVR C Programming
- AVR GCC Inline Assembly

AVR C Programming

High Level Languages

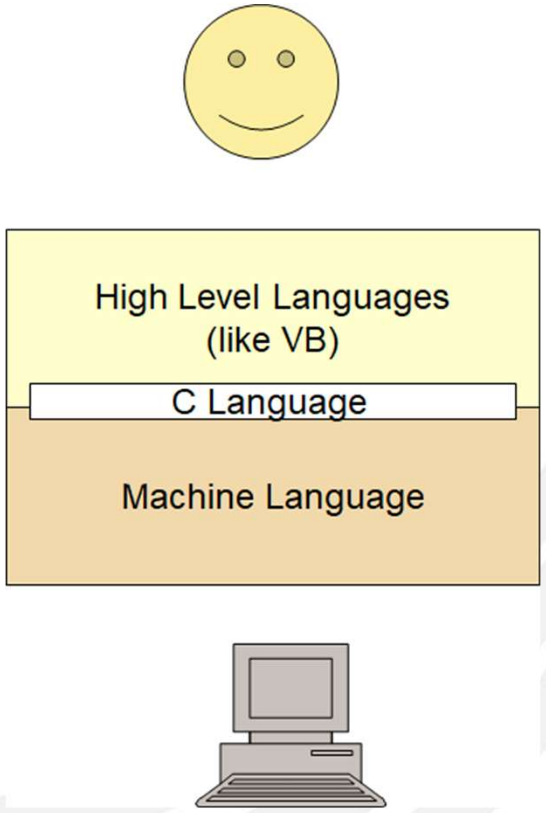
- Easy to develop and update

C Language

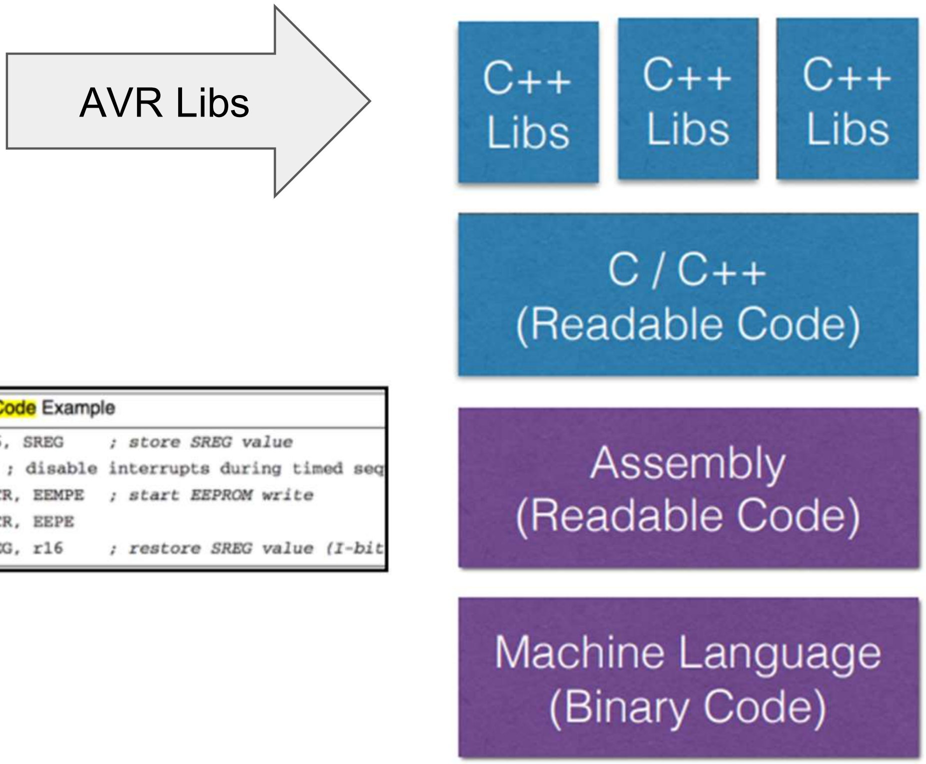
- Acceptable performance
- Easy to develop and update
- Portable

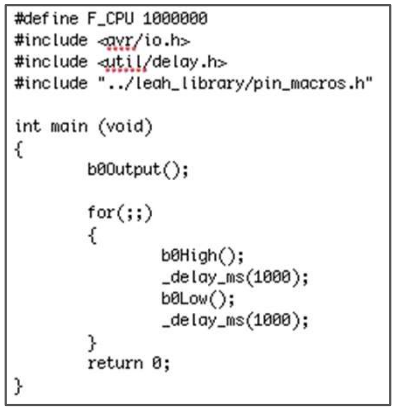
Low Level Languages

- High performance
- Not portable

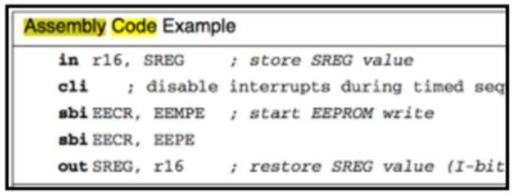


AVR C Programming



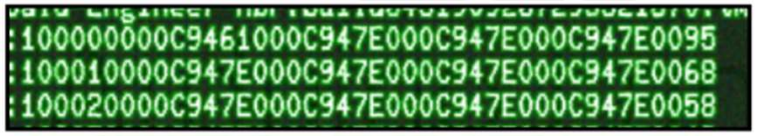


```
#define F_CPU 1000000
#include <avr/io.h>
#include <util/delay.h>
#include "../leah_library/pin_macros.h"

int main (void)
{
        b0Output();

        for(;;)
        {
                b0High();
                _delay_ms(1000);
                b0Low();
                _delay_ms(1000);
        }
        return 0;
}
```



Assembly Code Example

```
in r16, SREG     ; store SREG value
cli     ; disable interrupts during timed seq
sbi EECR, EEMPE  ; start EEPROM write
sbi EECR, EEPE
out SREG, r16    ; restore SREG value (I-bit
```



```
:100000000C9461000C947E000C947E000C947E0095
:100010000C947E000C947E000C947E000C947E0068
:100020000C947E000C947E000C947E000C947E0058
```

COMPUTER ENGINEERING MAHIDOL UNIVERSITY

AVR C Programming

Regular C programming

Write a program that calculate the sum of {1,3,…,13,15}

```
int main ()
{
    unsigned int sum;

    for (int i = 1; i <= 15; i+=2)
        sum += i;

    while (1);
    return 0;
}
```

COMPUTER ENGINEERING MAHIDOL UNIVERSITY

AVR C Programming

AVR C program to send value 0xAA to PORTD

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

int main ()
{
   DDRD = 0xFF;
   PORTD = 0xAA;

   while (1);
   return 0;
}
```

COMPUTER ENGINEERING MAHIDOL UNIVERSITY

AVR C Programming

Data Types

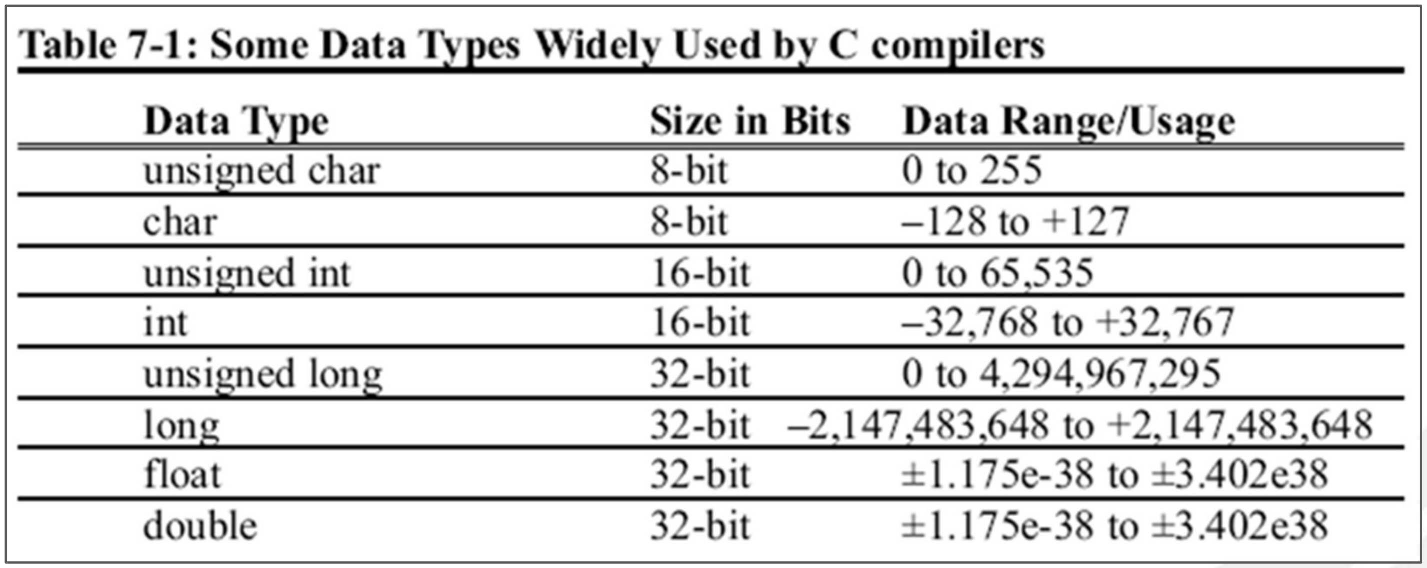


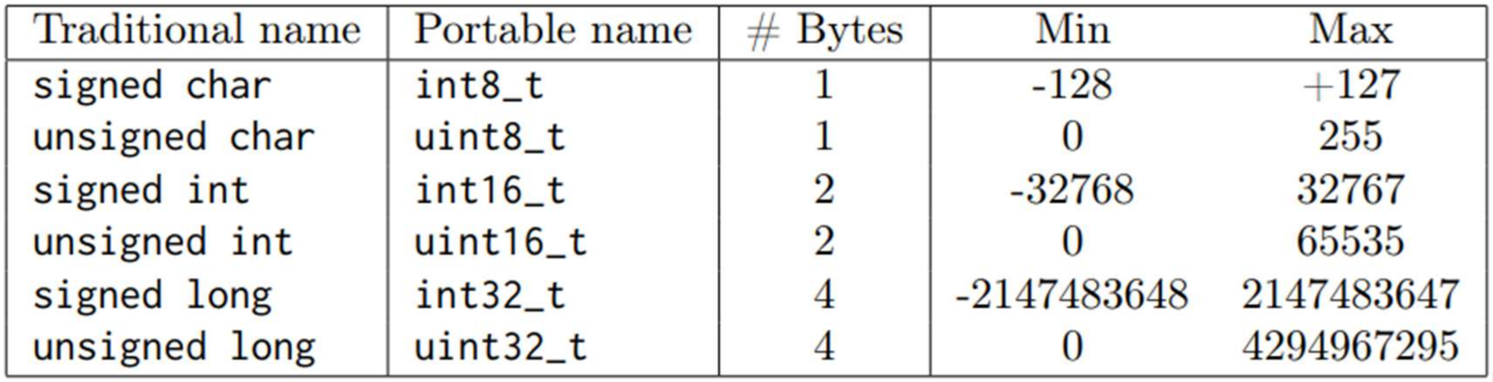
Table 7-1: Some Data Types Widely Used by C compilers

Data Type	Size in Bits	Data Range/Usage
unsigned char	8-bit	0 to 255
char	8-bit	–128 to +127
unsigned int	16-bit	0 to 65,535
int	16-bit	–32,768 to +32,767
unsigned long	32-bit	0 to 4,294,967,295
long	32-bit	–2,147,483,648 to +2,147,483,648
float	32-bit	±1.175e-38 to ±3.402e38
double	32-bit	±1.175e-38 to ±3.402e38

COMPUTER ENGINEERING MAHIDOL UNIVERSITY

AVR C Programming

Data Types



Traditional name	Portable name	# Bytes	Min	Max
`signed char`	`int8_t`	1	-128	+127
`unsigned char`	`uint8_t`	1	0	255
`signed int`	`int16_t`	2	-32768	32767
`unsigned int`	`uint16_t`	2	0	65535
`signed long`	`int32_t`	4	-2147483648	2147483647
`unsigned long`	`uint32_t`	4	0	4294967295

AVR C Programming

AVR C Programming

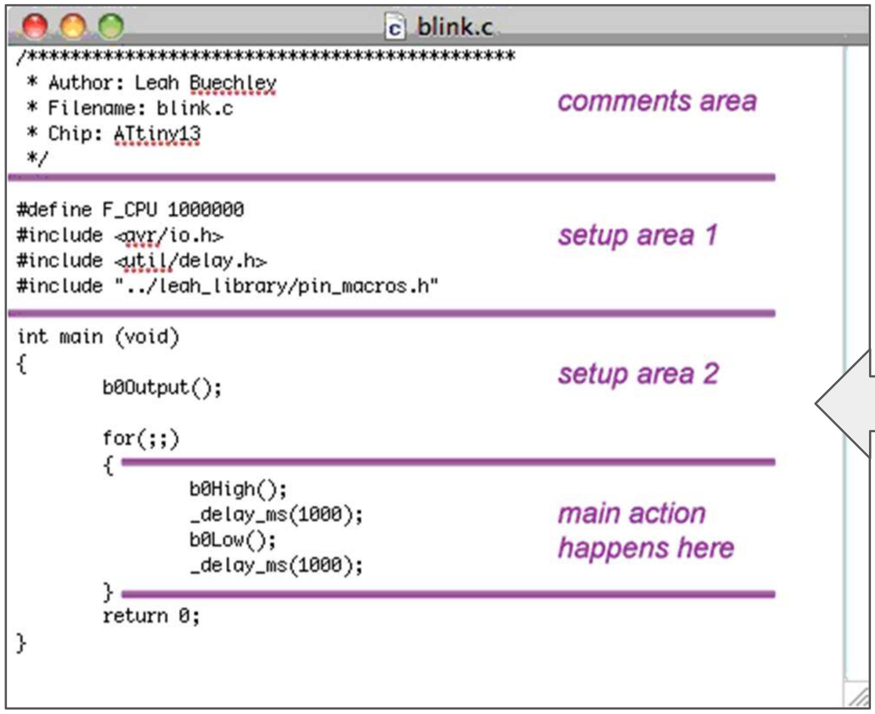
Everything is same as regular C programming style.

Except the coding structure

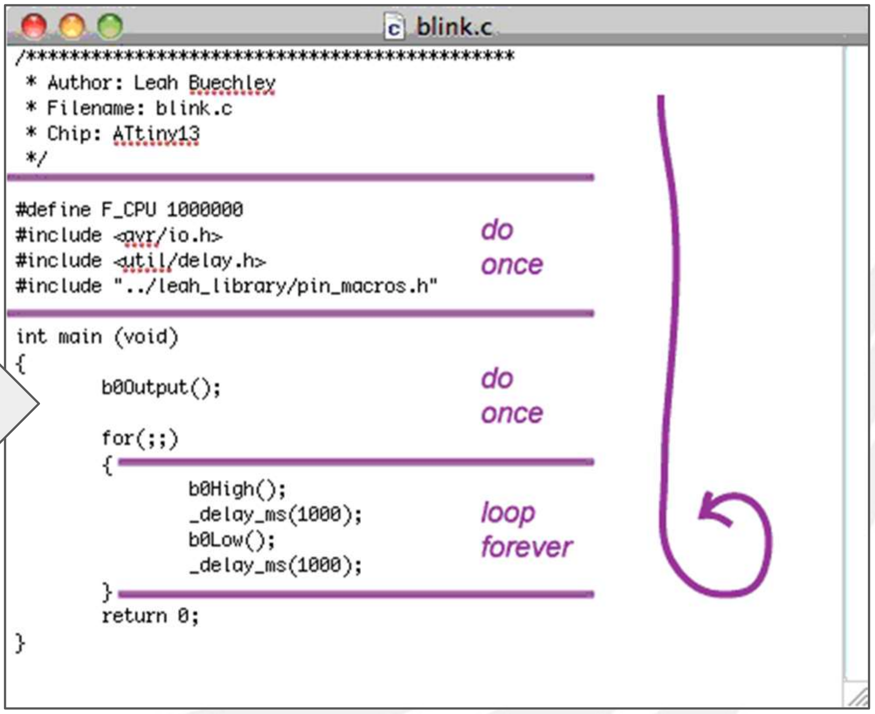
COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

AVR C Programming

Coding Structure



```
/*******************************************
 * Author: Leah Buechley
 * Filename: blink.c
 * Chip: ATtiny13
 */

#define F_CPU 1000000
#include <avr/io.h>
#include <util/delay.h>
#include "../leah_library/pin_macros.h"

int main (void)
{
        b0Output();

        for(;;)
        {
                b0High();
                _delay_ms(1000);
                b0Low();
                _delay_ms(1000);
        }
        return 0;
}
```



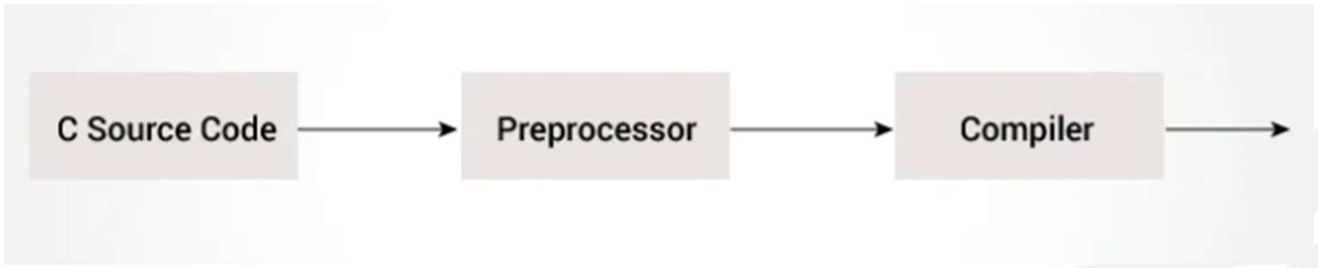
```
/*******************************************
 * Author: Leah Buechley
 * Filename: blink.c
 * Chip: ATtiny13
 */

#define F_CPU 1000000
#include <avr/io.h>
#include <util/delay.h>
#include "../leah_library/pin_macros.h"

int main (void)
{
        b0Output();

        for(;;)
        {
                b0High();
                _delay_ms(1000);
                b0Low();
                _delay_ms(1000);
        }
        return 0;
}
```

AVR C Programming

Preprocessor

The C **preprocessor** is a macro processor that is used automatically by the C compiler to transform your program before actual compilation.

It is called a macro processor because it allows you to define macros, which are brief abbreviations for longer constructs.



COMPUTER ENGINEERING MAHIDOL UNIVERSITY

AVR C Programming

AVR Preprocessor

#assert	Assertions
#cpu	Assertions
#define	**Macros with Arguments**
#elif	**The `#elif' Directive**
#else	**The `#else' Directive**
#error	The `#error' and `#warning' Directives
#ident	Miscellaneous Preprocessing Directives

***Assertions** are a more systematic alternative to macros in writing conditionals to test what sort of computer or system the compiled program will run on

COMPUTER ENGINEERING MAHIDOL UNIVERSITY

AVR C Programming

AVR Preprocessor

#if	**Syntax of Conditionals**
#ifdef	**Conditionals and Macros**
#ifndef	**Conditionals and Macros**
#import	Once-Only Include Files
#include	**The `#include' Directive**
#include_next	Inheritance and Header Files
#line	Combining Source Files

COMPUTER ENGINEERING MAHIDOL UNIVERSITY

AVR C Programming

AVR Preprocessor

#machine	Assertions
#pragma	Miscellaneous Preprocessing Directives
#pragma once	Once-Only Include Files
#system	Assertions
#unassert	Assertions
#warning	The `#error' and `#warning' Directives

AVR C Programming

AVR Preprocessor

#define: is a macro that requires arguments, you write a `#define' directive with a list of argument names in parentheses after the name of the macro.

Example

- #define PI 3.14
- #define circleArea(r) (3.1415*(r)*(r))
- #define min(X, Y) ((X) < (Y) ? (X) : (Y))

AVR C Programming

AVR Preprocessor

#if, #else, #elif : expresses the condition for preprocessing (logically similar as using in C programming).

Kindly note, **#endif** is not required to be used for ending **#if** condition

Syntax

```
#if expression
        Text if true
#elif expression
        Text if true
#else
        Text if true
#endif
```

AVR C Programming

AVR Preprocessor

#if, #else, #elif

Example

```
#if X == 1
      ...
#elif X == 2
      ...
#else /* X != 2 and X != 1*/
      ...
#endif /* X != 2 and X != 1*/
```

COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

AVR C Programming

AVR Preprocessor

#ifdef, #ifndef:

- #ifdef name → is equivalent to `#if defined (name)'.

- #ifndef name → is equivalent to `#if ! defined (name)'.

AVR C Programming

AVR Preprocessor

#include

- #include <file>: searches for a file named file in a list of directories specified by you, then in a standard list of system directories

- #include “file” : searches for a file named file first in the current directory

AVR C Programming

AVR Preprocessor

#include

- #include <file>: searches for a file named file in a list of directories specified by you, then in a standard list of system directories

- #include “file” : searches for a file named file first in the current directory

AVR C Programming

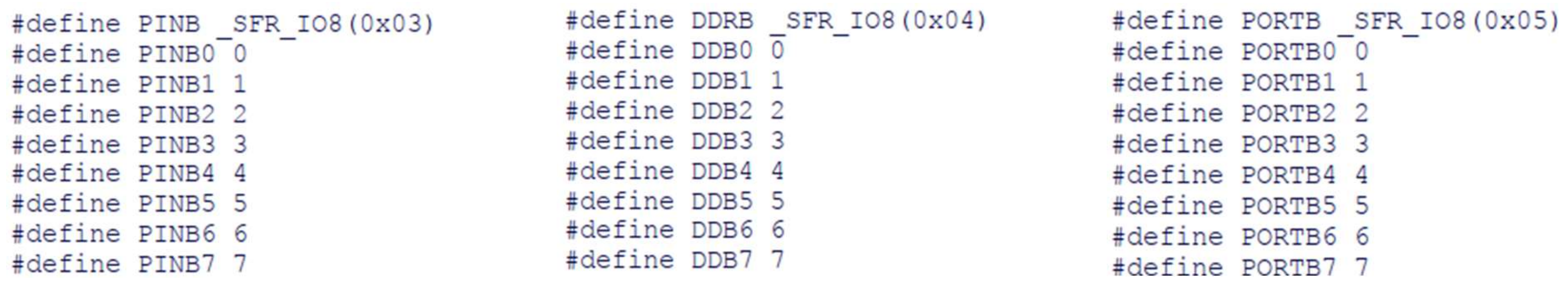
Libraries

- avr/ioXXXX.h
  This header file includes the appropriate IO definitions for the device
- util/delay.h
  The convenience (busy wait) functions where actual time values can be specified rather than a number of cycles to wait for
- avr/interrupt.h
  Interrupt handling functions
- regular C library
  - string.h
  - string.h
  - math.h
  - stlib.h

AVR C Programming

Libraries

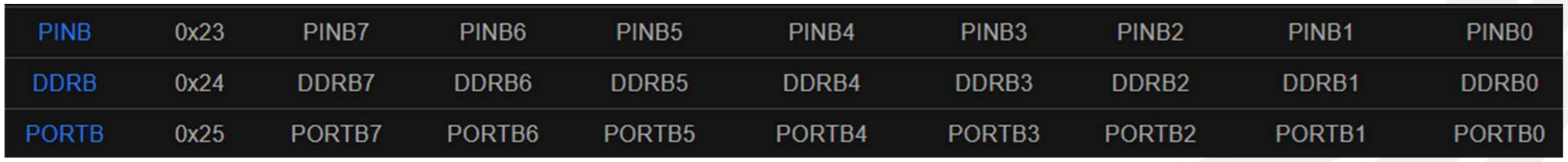
avr/ioXXXX.h



```
#define PINB _SFR_IO8(0x03)
#define PINB0 0
#define PINB1 1
#define PINB2 2
#define PINB3 3
#define PINB4 4
#define PINB5 5
#define PINB6 6
#define PINB7 7
```

```
#define DDRB _SFR_IO8(0x04)
#define DDB0 0
#define DDB1 1
#define DDB2 2
#define DDB3 3
#define DDB4 4
#define DDB5 5
#define DDB6 6
#define DDB7 7
```

```
#define PORTB _SFR_IO8(0x05)
#define PORTB0 0
#define PORTB1 1
#define PORTB2 2
#define PORTB3 3
#define PORTB4 4
#define PORTB5 5
#define PORTB6 6
#define PORTB7 7
```



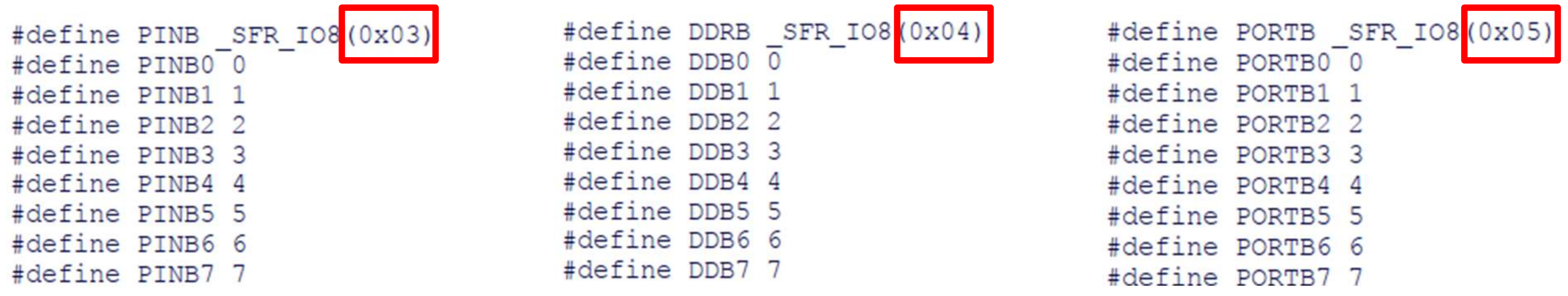
PINB	0x23	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
DDRB	0x24	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
PORTB	0x25	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0

COMPUTER ENGINEERING MAHIDOL UNIVERSITY

AVR C Programming

Libraries

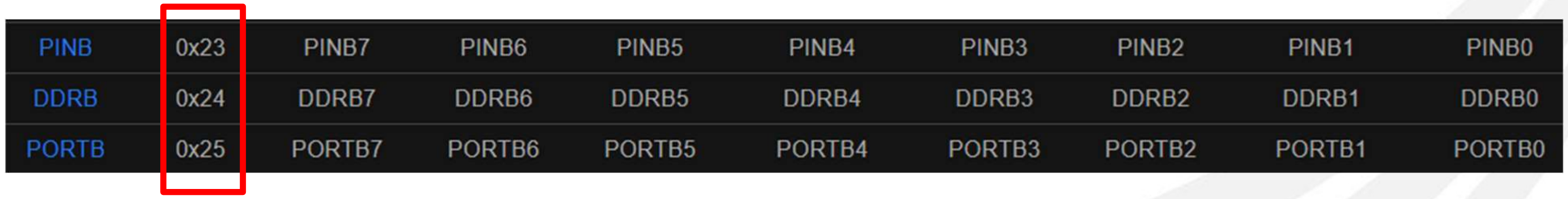
avr/ioXXXX.h



```
#define PINB _SFR_IO8(0x03)
#define PINB0 0
#define PINB1 1
#define PINB2 2
#define PINB3 3
#define PINB4 4
#define PINB5 5
#define PINB6 6
#define PINB7 7
```

```
#define DDRB _SFR_IO8(0x04)
#define DDB0 0
#define DDB1 1
#define DDB2 2
#define DDB3 3
#define DDB4 4
#define DDB5 5
#define DDB6 6
#define DDB7 7
```

```
#define PORTB _SFR_IO8(0x05)
#define PORTB0 0
#define PORTB1 1
#define PORTB2 2
#define PORTB3 3
#define PORTB4 4
#define PORTB5 5
#define PORTB6 6
#define PORTB7 7
```



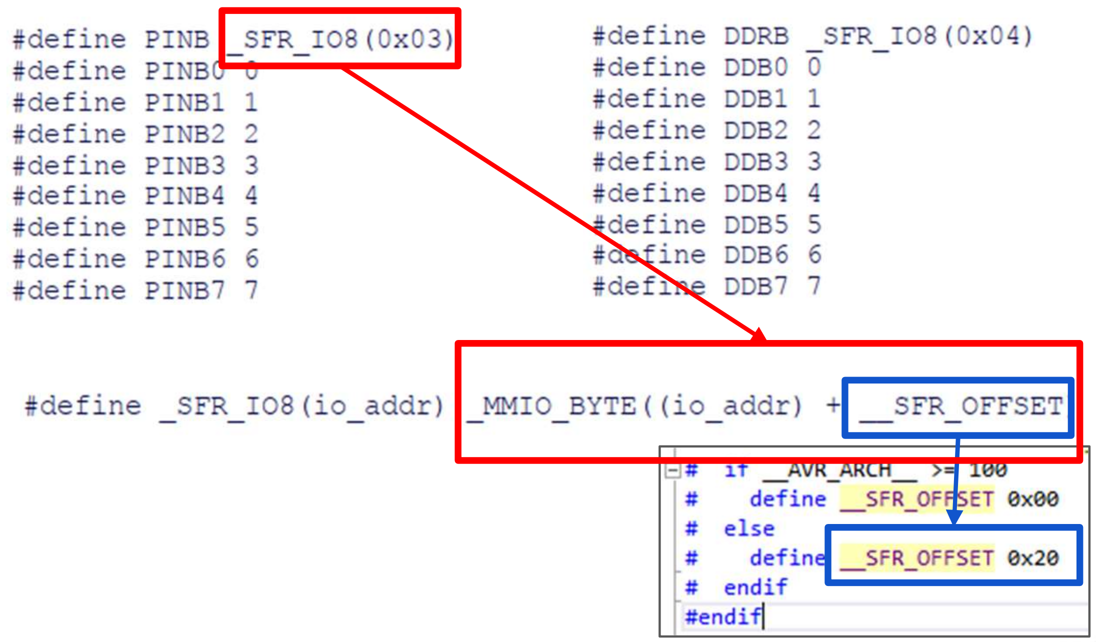
PINB	0x23	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
DDRB	0x24	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
PORTB	0x25	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0

COMPUTER ENGINEERING MAHIDOL UNIVERSITY

AVR C Programming

Libraries

avr/ioXXXX.h



```
#define PINB _SFR_IO8(0x03)
#define PINB0 0
#define PINB1 1
#define PINB2 2
#define PINB3 3
#define PINB4 4
#define PINB5 5
#define PINB6 6
#define PINB7 7
```

```
#define DDRB _SFR_IO8(0x04)
#define DDB0 0
#define DDB1 1
#define DDB2 2
#define DDB3 3
#define DDB4 4
#define DDB5 5
#define DDB6 6
#define DDB7 7
```

```
#define PORTB _SFR_IO8(0x05)
#define PORTB0 0
#define PORTB1 1
#define PORTB2 2
#define PORTB3 3
#define PORTB4 4
#define PORTB5 5
#define PORTB6 6
#define PORTB7 7
```

```
#define _SFR_IO8(io_addr) _MMIO_BYTE((io_addr) + __SFR_OFFSET
```

```
#  if __AVR_ARCH__ >= 100
#    define __SFR_OFFSET 0x00
#  else
#    define __SFR_OFFSET 0x20
#  endif
#endif
```

COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

AVR C Programming

Libraries

avr/ioXXXX.h

```
#define PINB _SFR_IO8(0x03)
#define PINB0 0
#define PINB1 1
#define PINB2 2
#define PINB3 3
#define PINB4 4
#define PINB5 5
#define PINB6 6
#define PINB7 7
```

```
#define DDRB _SFR_IO8(0x04)
#define DDB0 0
#define DDB1 1
#define DDB2 2
#define DDB3 3
#define DDB4 4
#define DDB5 5
#define DDB6 6
#define DDB7 7
```

```
#define PORTB _SFR_IO8(0x05)
#define PORTB0 0
#define PORTB1 1
#define PORTB2 2
#define PORTB3 3
#define PORTB4 4
#define PORTB5 5
#define PORTB6 6
#define PORTB7 7
```

```
#define _SFR_IO8(io_addr) _MMIO_BYTE((io_addr) + __SFR_OFFSET)
```

```
#define _MMIO_BYTE(mem_addr) (*(volatile uint8_t *)(mem_addr))
```

COMPUTER ENGINEERING MAHIDOL UNIVERSITY

AVR C Programming

Libraries

avr/ioXXXX.h

Therefore, if we want to set bit 5 high, we can now just say

```
PORTB = PORTB | 0x20; // or more typically: PORTB |= 0x20;
```

COMPUTER ENGINEERING MAHIDOL UNIVERSITY

AVR C Programming

Libraries

util/delay.h

The functions available allow the specification of microsecond, and millisecond delays directly, using the application-supplied macro F_CPU as the CPU clock frequency (in Hertz).

#define F_CPU   1000000UL   *//The macro F_CPU specifies the CPU frequency to be considered by the delay macros*

void _delay_ms   (double __ms)   *//Perform a delay of milliseconds (The maximal possible delay is **262.14 ms / F_CPU** in MHz)*

void _delay_us   (double __us)   *//Perform a delay of microseconds (The maximal possible delay is **768 us / F_CPU** in MHz)*

AVR C Programming

Libraries

util/delay.h

Example

```
#define F_CPU 1000000UL
     _delay_ms(50);                     // 50ms delay
     PORTB &= ~(1 << PB0);              // LED off
          _delay_ms(50);                     // 50ms delay
          PORTB |=  (1 << PB0);              // LED on
```

COMPUTER ENGINEERING MAHIDOL UNIVERSITY

Exercise 8-1

Question 1

Write AVR C program read pins 1 and 0 of PORTB and send ASCII character to PORTD according to the following table

pin1	pin0	send
0	0	‘1’
0	1	‘2’
1	0	‘3’
1	1	‘4’

Exercise 8-1

Question 2

Write AVR C program that check the value of PORTB.7 every 100ms. If it is 1, make bit 4 of PORTB input, otherwise, change pin 4 of PORTB to output