Bahan Ajar

Chapter 1/5

Materi Pembelajaran

Matakuliah :

PEMROGRAMAN TERSTRUKTUR

Kode Matakuliah : SKO 21411

Prodi : SISTEM KOMPUTER

Dosen Pengampu Matakuliah:

Bayu Nugroho, S.Kom., M.Eng

Tables of Content

The C Preprocessor and Bitwise Operations

- Preprocessor Directives
- Parameterized Macros

Preprocessor Directives

The ANSI C specification details the duties of the C preprocessor. It is the function of the C preprocessor to process the defined directives supported by the C compiler. Table lists the preprocessor directives that Arduino C can translate.

Arduino C Preprocessor Directives

Directive	Action
#define NAME value	Ascribes the identifier NAME to the constant value.
#undef NAME	Removes NAME from the list of defined constants
#line lineNumberValue "filename.ino"	Allows the compiler to refer to any line numbers in the file named filename.ino to be referenced as line lineNumberValue from this point on by the compiler. Normally used in debugging. This is not in the Arduino C reference material, but the compiler recognizes it

Arduino C Preprocessor Directives

Directive	Action
#if definedConstant expression operand	Conditional compilation. Example: #if LED == 12 #define VOLTS 5 #endif This is not in the Arduino C reference material, but the compiler recognizes it..
#if defined NAME // statement(s) #endif	Allows for conditional compilation of statements if NAME is defined. The statement block ends with #endif. This is not in the Arduino C reference material, but the compiler recognizes it
#if !defined NAME // statement(s) #endif	Same as #if defined, but processes statement block only if NAME is not defined. This is not in the Arduino C reference material, but the compiler recognizes it.

Arduino C Preprocessor Directives

Directive	Action
#ifdef	Same as #if defined. This is not in the Arduino C reference material, but the compiler recognizes it.
#if defined NAME // statement(s) #endif	Allows for conditional compilation of statements if NAME is defined. The statement block ends with #endif. This is not in the Arduino C reference material, but the compiler recognizes it
#ifndef	Same as #if !defined. This is not in the Arduino C reference material, but the compiler recognizes it.
#else	Can be used with #if like an if-else statement but to control compiled statements. Example: #if defined ATMEGA2560 #define BUFFER 64 #else #define BUFFER 32 #endif

Arduino C Preprocessor Directives

Directive	Action
#elif	Used with #if for cascading #if's
#include "filename.xxx"	Opens the file named filename.xxx and reads the contents of the file into the program source code. Usually, if double quotes surround the file name, then the search for the file is in the currently active directory. If angle brackets are used (<filename.xxx>), then the search begins in some implementation-defined manner. This is not in the Arduino C reference material, but the compiler recognizes it.

#undef

The #undef is used to turn off a previously-defined #define preprocessor directive. For example:

```
#ifdef DEBUG
        Serial.print("The counter value is: ");
        Serial.println(myCounter);
#endif
```

This is a technique (called scaffolding) that you have used before to toggle debugging code into the program. If the source contains:

```
#define DEBUG 1
```

#undef

```
#define DEBUG 1
        // A whole bunch of program lines
        // that still need to be debugged
#undef DEBUG
ReadSensorCounter() {
        // code for the debugged function
}
```

When the preprocessor sees the #undef directive, it removes DEBUG from its list of #defines. This has the effect of removing the Serial() calls from the (now debugged) ReadSensorCounter() function.

#line

The #line directive is used most often while debugging a program. The syntax is:

```
#line lineNumberValue "filename.ino"
```

where lineNumberValue is the line number you want to the compiler to use from that point on in the source code file name filename.ino. Therefore:

```
#line 100 "C:\Temp\myCode.ino"
```

#if, Conditional Directives

There are a number of conditional directives, and they are very similar, so we can discuss them as a group. First, the expression:

```
#if definedConstant expression operand
        // Statement(s)
#endif
```

might be written as:

```
#if BOARD == ATMEGA168
#define MAXEEPROM 1024
#endif
```

Bitwise Operators

Arduino C provides you with four bitwise logic operators: AND, OR, XOR, and NOT. The first three operators are binary operators and, hence, require two operands in the expression. The bitwise NOT operator is a unary operator and uses only a single operand. Bitwise operations can only be performed on integer data (i.e., no floating point data).

Bitwise AND

In code, the bitwise AND might look like:

```
byte a = 10; // 00001010
byte b = 6; // 00000110
byte c = a & b; // 00000010 = c
```

Bitwise OR

In code, a bitwise OR fragment might be written as:

```
byte a = 10; // 00001010
byte b = 6; // 00000110
byte c = a | b; // 00001110
```

Bitwise Exclusive OR (XOR)

Using our code fragment:

```
byte a = 10; // 00001010
byte b = 6; // 00000110
byte c = a ^ b; // 00001100
```

Bitwise NOT (~)

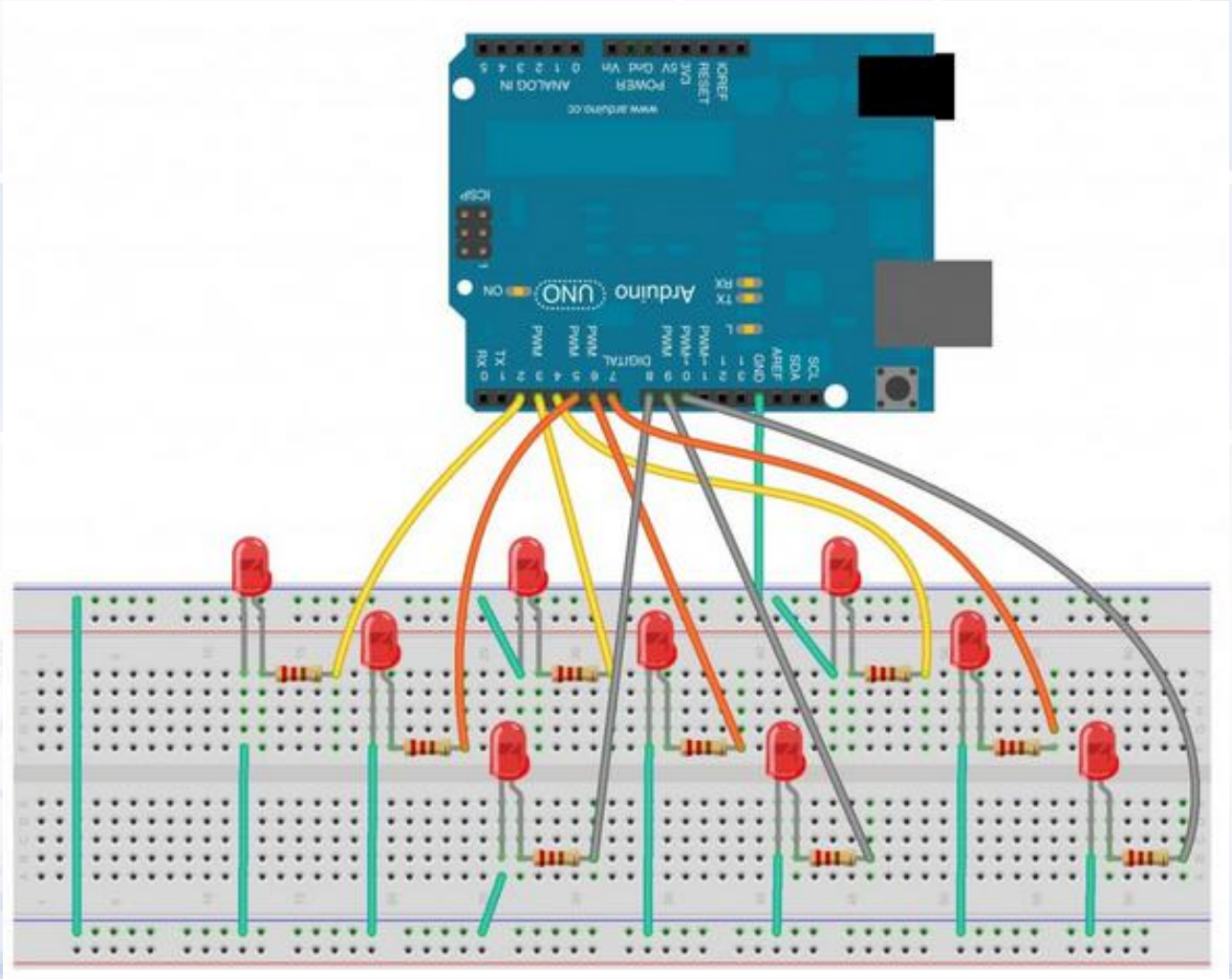
That is, all 0 bits become 1s and all 1 bits become 0s. For example:

```
byte a = 1; // 00000001
byte c = ~a ; // 11111110
```

Tugas Mandiri (teori):

1. If you have an integer value k and wish to multiply it by 2 and assign the result into variable j, then what statement would you use?
2. What types of data would you consider using for bitwise operations?

Tugas Mandiri (prakt):



Tugas Mandiri (prakt):

```
//the matrix will hold the pin assignments for the led
int pinMatrix[3][3] = {
  {2,  3,  4  },
  {5,  6,  7  },
  {8,  9, 10  }
};
void setup() {
  //use a nested for loop to initialize all the pins
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      pinMode(pinMatrix[i][j], OUTPUT);
    }//close for i
  }//close for j
} //close setup()
void loop() {
  //this nested for loop will turn each LED on and off in sequence
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      digitalWrite(pinMatrix[i][j], HIGH);
      delay(100);
      digitalWrite(pinMatrix[i][j], LOW);
    }//close for i
  }//close for j
} //close loop()
```

Analisa kondisi Led saat Sketch Program ini di jalankan

end