ATMEGA328/ARDUINO – C_LANGUAGE – BITWISE
https://github.com/teaksoon/lmaewapm

The C-Language Keywords and Symbols

Keywords

MEMORY

01.void
02.char
03.int
04.short
05.long
06.float
07.double
08.signed
09.unsigned
10.struct
11.union
12.enum
13.const
14.volatile
15.auto
16.extern
17.static
18.register
19.typedef
20.sizeof

CONTROL

21.return
22.if
23.else
24.switch
25.case
26.default
27.while
28.do
29.for
30.break
31.continue
32.goto

Symbols

CONTROL

```
#
< >
//
/* */
( )
{ }
;
,
“
‘
=
[ ]
:
?
.
\
```

MEMORY

```
&
*
```

LOGIC

```
==
!=
<
>
<=
>=

&&
||

!
```

MATH

```
*
%
/
+
-
```

BIT OP

```
|
&
^
~
<<
>>
```

BITWISE OPERATIONS

These symbols allow us to work on individual bits in a BYTE
Another special feature in C-Language which allows us to deal directly with the CPU Memory

ATMEGA328/ARDUINO - C_LANGUAGE - BITWISE
https://github.com/teaksoon/lmaewapm

BITWISE OPERATIONS

Bitwise operations allow us to manipulate individual bits within the 8-BIT of a BYTE

BIT positions in the BYTE is matched against BIT positions of another BYTE

BITWISE AND &

- To get 1, both must be 1

```
  11111111
& 11111111
= 11111111

  11111111
& 00000000
= 00000000

  10101010
& 00001111
= 00001010
```

Common usage

```
  00000001
& 00000001
= 00000001

  00000001
& 00000000
= 00000000
```

BITWISE OR |

- To get 1, One must be 1

```
  11111111
| 11111111
= 11111111

  11111111
| 00000000
= 11111111

  10101010
| 00001111
= 10101111
```

Common usage

```
  00000001
| 00000001
= 00000001

  00000001
| 00000000
= 00000001
```

BITWISE XOR ^

- To get 1, Both not same

```
  11111111
^ 11111111
= 00000000

  11111111
^ 00000000
= 11111111

  10101010
^ 00001111
= 10100101
```

Common usage

```
  00000001
^ 00000001
= 00000000

  00000001
^ 00000000
= 00000001
```

BITWISE NOT ~

- Change 1 to 0 or 0 to 1

```
~ 11111111
= 00000000

~ 00000000
= 11111111

~ 10101010
= 01010101
```

Arduino IDE|Save PROGRAM as: **c_bitwise_operator**
Enter codes below and upload. Use the Serial Monitor to see results

```
void setup() {
  Serial.begin(9600);Serial.print("\n\nSerial Monitor(9600)...\n");

  Serial.print("\n\nBITWISE OR |");
  Serial.print("\n10101010 | 00001111 = ");show_bits(0b10101010|0b00001111);

  Serial.print("\n\nBITWISE AND &");
  Serial.print("\n10101010 & 00001111 = ");show_bits(0b10101010&0b00001111);

  Serial.print("\n\nBITWISE XOR ^");
  Serial.print("\n10101010 ^ 00001111 = ");show_bits(0b10101010^0b00001111);

  Serial.print("\n\nBITWISE NOT ~");
  Serial.print("\n~10101010 = ");show_bits(~0b10101010);

}
void loop(){}

void show_bits(unsigned char data) {
  for (int i=7; i>=0; i--) {
    Serial.print((data >> i) & 1);Serial.print(" ");
  }
}
```

BITWISE SHIFT OPERATIONS

In order to retrieve an individual BIT, we need to do BIT shifting within a BYTE

LEFT SHIFT <<

Part 1: source_in_binary
Part 2: << left-shift symbol
Part 3: positions_to_move

source_in_binary << postitions_to_move

```
 1 << 0 (1 in binary=00000001, shift 0 position to the left=00000001
 1 << 1 (1 in binary=00000001, shift 1 position to the left=00000010
 1 << 2 (1 in binary=00000001, shift 2 position to the left=00000100
 1 << 3 (1 in binary=00000001, shift 3 position to the left=00001000
 1 << 4 (1 in binary=00000001, shift 4 position to the left=00010000
 1 << 5 (1 in binary=00000001, shift 5 position to the left=00100000
 1 << 6 (1 in binary=00000001, shift 6 position to the left=01000000
 1 << 7 (1 in binary=00000001, shift 7 position to the left=10000000

 1 << 1 (1 in binary=00000001, shift 1 position to the left=00000010
 2 << 1 (2 in binary=00000010, shift 1 position to the left=00000100
 4 << 1 (3 in binary=00000100, shift 1 position to the left=00001000
 8 << 1 (4 in binary=00001000, shift 1 position to the left=00010000
16 << 1 (5 in binary=00010000, shift 1 position to the left=00100000
32 << 1 (6 in binary=00100000, shift 1 position to the left=01000000
64 << 1 (8 in binary=01000000, shift 1 position to the left=10000000
```

Arduino IDE|Save PROGRAM as: **c_bitwise_left_shift**
Enter codes below and upload. Use the Serial Monitor to see results

```
void setup() {
  Serial.begin(9600);Serial.print("\n\nSerial Monitor(9600)...\n");
  unsigned char source_number = 1;

  Serial.print("\nLEFT SHIFT <<");
  Serial.print("\nsource_number << position to move\n");
  Serial.print("\n 1 << 0 = "); show_bits(source_number << 0);
  Serial.print("\n 1 << 1 = "); show_bits(source_number << 1);
  Serial.print("\n 1 << 2 = "); show_bits(source_number << 2);
  Serial.print("\n 1 << 3 = "); show_bits(source_number << 3);
  Serial.print("\n 1 << 4 = "); show_bits(source_number << 4);
  Serial.print("\n 1 << 5 = "); show_bits(source_number << 5);
  Serial.print("\n 1 << 6 = "); show_bits(source_number << 6);
  Serial.print("\n 1 << 7 = "); show_bits(source_number << 7);
  Serial.print("\n");
  Serial.print("\n 1 << 1 = "); show_bits( 1 << 1);
  Serial.print("\n 2 << 1 = "); show_bits( 2 << 1);
  Serial.print("\n 4 << 1 = "); show_bits( 4 << 1);
  Serial.print("\n 8 << 1 = "); show_bits( 8 << 1);
  Serial.print("\n16 << 1 = "); show_bits(16 << 1);
  Serial.print("\n32 << 1 = "); show_bits(32 << 1);
  Serial.print("\n64 << 1 = "); show_bits(64 << 1);
}
void loop(){}

void show_bits(unsigned char data) {
  for (int i=7; i>=0; i--) {
    Serial.print((data >> i) & 1);Serial.print(" ");
  }
}
```

When shifting left, the BIT on the LEFT will be removed

BITWISE SHIFT OPERATIONS

In order to retrieve an individual BIT, we need to do BIT shifting within a BYTE

RIGHT SHIFT >>

Part 1: source_in_binary
Part 2: >> right-shift symbol
Part 3: positions_to_move

source_in_binary << postitions_to_move

```
128 >> 0 (128 in binary=10000000, shift 0 position to the right=10000000
128 >> 1 (128 in binary=10000000, shift 1 position to the right=01000000
128 >> 2 (128 in binary=10000000, shift 2 position to the right=00100000
128 >> 3 (128 in binary=10000000, shift 3 position to the right=00010000
128 >> 4 (128 in binary=10000000, shift 4 position to the right=00001000
128 >> 5 (128 in binary=10000000, shift 5 position to the right=00000100
128 >> 6 (128 in binary=10000000, shift 6 position to the right=00000010
128 >> 7 (128 in binary=10000000, shift 7 position to the right=00000001

128 >> 1 (128 in binary=10000000, shift 1 position to the right=01000000
 64 >> 1 ( 64 in binary=01000000, shift 1 position to the right=00100000
 32 >> 1 ( 32 in binary=00100000, shift 1 position to the right=00010000
 16 >> 1 ( 16 in binary=00010000, shift 1 position to the right=00001000
  8 >> 1 (  8 in binary=00001000, shift 1 position to the right=00000100
  4 >> 1 (  4 in binary=00000100, shift 1 position to the right=00000010
  2 >> 1 (  2 in binary=00000010, shift 1 position to the right=00000001
```

Arduino IDE|Save PROGRAM as: **c_bitwise_right_shift**
Enter codes below and upload. Use the Serial Monitor to see results

```
void setup() {
  Serial.begin(9600);Serial.print("\n\nSerial Monitor(9600)...\n");
  unsigned char source_number = 128;

  Serial.print("\nRIGHT SHIFT >>");
  Serial.print("\nsource_number >> position to move\n");
  Serial.print("\n 128 >> 0 = "); show_bits(source_number >> 0);
  Serial.print("\n 128 >> 1 = "); show_bits(source_number >> 1);
  Serial.print("\n 128 >> 2 = "); show_bits(source_number >> 2);
  Serial.print("\n 128 >> 3 = "); show_bits(source_number >> 3);
  Serial.print("\n 128 >> 4 = "); show_bits(source_number >> 4);
  Serial.print("\n 128 >> 5 = "); show_bits(source_number >> 5);
  Serial.print("\n 128 >> 6 = "); show_bits(source_number >> 6);
  Serial.print("\n 128 >> 7 = "); show_bits(source_number >> 7);
  Serial.print("\n");
  Serial.print("\n128 >> 1 = "); show_bits(128 >> 1);
  Serial.print("\n 64 >> 1 = "); show_bits( 64 >> 1);
  Serial.print("\n 32 >> 1 = "); show_bits( 32 >> 1);
  Serial.print("\n 16 >> 1 = "); show_bits( 16 >> 1);
  Serial.print("\n  8 >> 1 = "); show_bits(  8 >> 1);
  Serial.print("\n  4 >> 1 = "); show_bits(  4 >> 1);
  Serial.print("\n  2 >> 1 = "); show_bits(  2 >> 1);
}
void loop(){}

void show_bits(unsigned char data) {
  for (int i=7; i>=0; i--) {
    Serial.print((data >> i) & 1);Serial.print(" ");
  }
}
```

When shifting right, the BIT on the RIGHT will be removed