Bit-Wise Operators
Bit-Wise Operators

If A and B are bytes, what does this code mean?

$$C = A \& B;$$
Bit-Wise Operators

If A and B are bytes, what does this code mean?

\[ C = A \& B; \]

The corresponding bits of A and B are ANDed together.
Bit-Wise AND

\[ \begin{array}{ccl}
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
\hline
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
\end{array} \]

\[ C = A \& B \]
Bit-Wise AND

\[ C = A \& B \]

\[ \begin{array}{cccccccc}
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
\end{array} \]

A

B

C = A & B
Bit-Wise AND

0 1 0 1 1 1 1 0

1 0 0 1 1 0 1 1

A
B

C = A & B
Bit-Wise AND

\[ \begin{array}{c|c}
A & 01011110 \\
B & 10011011 \\
\hline
C = A \& B & 10 \\
\end{array} \]
Bit-Wise AND

\[
\begin{align*}
01011110 & \quad A \\
10011011 & \quad B \\
00011010 & \quad C = A \& B
\end{align*}
\]
Logical AND

\[ \begin{array}{cccccc}
0 & 1 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
\hline
???
\end{array} \]

\[ C = A \land B \]
Logical AND

0 1 0 1 1 1 1 0  
1 0 0 1 1 0 1 1  

C = A && B

true
Logical AND

\[
\begin{array}{ccccccc}
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
\end{array}
\quad A
\]

\[
\begin{array}{ccccccc}
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
\end{array}
\quad B
\]

\[
\begin{array}{ccccccc}
??? \\
\end{array}
\quad C = A \&\& B
\]
Logical AND

\[
\begin{array}{cccccc}
0 & 1 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
\hline
\text{true} & \text{true} & \text{true} \\
\end{array}
\]

\[C = A \&\& B\]
Logical AND

\[
\begin{align*}
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & \quad A \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & \quad B \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \quad C = A \& \& B
\end{align*}
\]

NOTE: we are assuming an 8-bit value
Representing Logical Values

Most of the time, we represent logical values using a multi-bit value. (e.g., using 8 or 16 bits). The rules are:

• A value of zero is interpreted as false
• A non-zero value is interpreted as true
Representing Logical Values

A logical operator will give a result of **true** or **false**:

- **false** is represented with a value of zero (0)
- **true** is represented with a value of one (1)
Other Operators

<table>
<thead>
<tr>
<th>LOGICAL</th>
<th>Bit-Wise</th>
</tr>
</thead>
<tbody>
<tr>
<td>• OR:</td>
<td></td>
</tr>
<tr>
<td>• NOT:</td>
<td>!</td>
</tr>
<tr>
<td>• XOR:</td>
<td>^</td>
</tr>
<tr>
<td>• Shift left:</td>
<td>&lt;&lt;</td>
</tr>
<tr>
<td>• Shift right:</td>
<td>&gt;&gt;</td>
</tr>
</tbody>
</table>

When coding: keep this distinction straight
Putting the Bit-Wise Operators to Work: Bit Manipulation

Assume a variable A is declared as such:

```c
uint8_t A;
```

What is the code that allows us to set bit 2 of A to 1? (we start counting bits from 0)
Bit Manipulation

What is the code that allows us to set bit 2 of A to 1? (we start counting bits from 0)

\[ A = A \mid 4; \]
Bit Manipulation

What is the code that allows us to set bit 2 of A to 0?
Bit Manipulation

What is the code that allows us to set bit 2 of A to 0?

A = A & 0xFB;

or

A = A & ~4;
Bit Shifting

```c
uint8_t A = 0x5A;
uint8_t B = A << 2;
uint8_t C = A >> 5;
```

What are the values of B and C?
What mathematical operations have we performed?
Example

Suppose a sensor is connected to pins 4 and 5 of port E:
• Fill in the following code so that variable “state” will have one of the following values: 0, 1, 2, 3

```c
uint8_t state;
:
state = ????
```
Example (cont)

Suppose a sensor is connected to pins 4 and 5 of port E:
• Fill in the following code so that variable “state” will have one of the following values: 0,1,2,3

```c
uint8_t state;
:
state = (GPIOE_PDIR & 0x30) >> 4;
```
Example (with only 8 bits)

GPIOE_PDIR: E7 E6 E5 E4 E3 E2 E1 E0
GPIOE_PDIR&0x30:
Example (cont)

GPIOE_PDIR : E7  E6  E5  E4  E3  E2  E1  E0
GPIOE_PDIR&0x30: 0  0  E5  E4  0  0  0  0
() >> 4:
Example (cont)

GPIOE_PDIR : E7 E6 E5 E4 E3 E2 E1 E0
GPIOE_PDIR &0x30: 0 0 E5 E4 0 0 0 0
() >> 4: 0 0 0 0 0 0 0 E5 E4
… Back to Digital I/O