Bit-Wise Operators
Questions?
Bit-Wise Operators

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

\[ C = A \& B; \]
If A and B are bytes, what does this code mean?

```c
C = A & B;
```

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

0 1 0 1 1 1 1 0

1 0 0 1 1 0 1 1

? C = A & B
Bit-Wise AND

\[ 0 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \ 0 \]
\[ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \]

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

\[
\begin{array}{cccccc}
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
\hline
0 & \text{A} & \text{B} & \text{C} = \text{A} \& \text{B}
\end{array}
\]
Bit-Wise AND

\[ 01011110 \quad \text{A} \]

\[ 10011011 \quad \text{B} \]

\[ 00011010 \quad C = A \& B \]
Logical AND

\[ A = 01011110 \]
\[ B = 10011011 \]
\[ C = A \&\& B \]
Logical AND

0 1 0 1 1 1 1 0   A

1 0 0 1 1 0 1 1   B

???

C = A && B

true
Logical AND

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

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

\[
\begin{array}{ccccccccc}
\textbf{C} = A \&\& B \\
\end{array}
\]
Logical AND

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

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

\[
A \text{ true} \quad B \text{ true} \quad C = A \&\& B \text{ true}
\]
Logical AND

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

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

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 \textit{false}
• A non-zero value is interpreted as \textit{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:

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?

\[ A = A \& 0xFFB; \]

or

\[ A = A \& \sim 4; \]
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 = (PINE & 0x30) >> 4;
```
Example (cont)

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

PINE: E7 E6 E5 E4 E3 E2 E1 E0
PINE & 0x30: 0 0 E5 E4 0 0 0 0

() >> 4:
Example (cont)

PINE: E7 E6 E5 E4 E3 E2 E1 E0
PINE&0x30: 0 0 E5 E4 0 0 0 0 0
() >> 4: 0 0 0 0 0 0 0 0 E5 E4
Next Time

Motor control