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

A = 01011110
B = 10011011

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

A

B

C = A & B

0 1 0 1 1 1 1 0
1 0 0 1 1 0 1 1
Bit-Wise AND

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

\[
\begin{align*}
A &= 01011110 \\
B &= 10011011 \\
C &= A \& B = 10
\end{align*}
\]
Bit-Wise AND

\[
\begin{array}{c}
A = 01011110 \\
B = 10011011 \\
C = A \& B = 00011010
\end{array}
\]
Logical AND

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

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

A = 0 1 0 1 1 1 0
B = 1 0 0 1 1 0 1 1
C = A && B

C = 1 0 0 1 1 0 1 1

true
Logical AND

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

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

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

\[ \begin{array}{cccccccc}
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
\end{array} \quad A \quad \text{true} \]

\[ \begin{array}{cccccccc}
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
\end{array} \quad B \quad \text{true} \]

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

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

A

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

B

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

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 \textit{true} or \textit{false}:

- \textit{false} is represented with a value of zero (0)
- \textit{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><strong>OR:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>NOT:</strong></td>
<td>!</td>
</tr>
<tr>
<td><strong>XOR:</strong></td>
<td>^</td>
</tr>
<tr>
<td><strong>Shift left:</strong></td>
<td>&lt;&lt;</td>
</tr>
<tr>
<td><strong>Shift right:</strong></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)

• All other bits of variable A must be unchanged!
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?

\[ A = A \& 0\text{xFB}; \]

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 = (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

Andrew H. Fagg: Embedded Real-Time Systems: Diodes
… Back to Digital I/O