Input/Output Systems

Processor needs to communicate with other devices:

• Receive signals from sensors
• Send commands to actuators
• Or both (e.g., disks, audio, video devices)
I/O Systems

Communication can happen in a variety of ways:

- Binary parallel signal
- Analog
- Serial signals
An Example:
SICK Laser Range Finder

- Laser is scanned horizontally
- Using phase information, can infer the distance to the nearest obstacle
- Resolution: ~.5 degrees, 1 cm
- Can handle full 180 degrees at 20 Hz
Serial Communication

• Communicate a set of bytes using a single signal line

• We do this by sending one bit at a time:
  – The value of the first bit determines the state of a signal line for a specified period of time
  – Then, the value of the 2\textsuperscript{nd} bit is used
  – Etc.
Serial Communication

The sender and receiver must have some way of agreeing on when a specific bit is being sent

• Typically, each side has a clock to tell it when to write/read a bit

• In some cases, the sender will also send a clock signal (on a separate line)

• In other cases, the sender/receiver will first synchronize their clocks before transfer begins
Asynchronous Serial Communication

• The sender and receiver have their own clocks, which they do not share
• This reduces the number of signal lines
• Bidirectional transmission, but the two halves do not need to be synchronized in time

But: we still need some way to agree that data is valid. How?
Asynchronous Serial Communication

How can the two sides agree that the data is valid?

• Must both be operating at essentially the same transmit/receive frequency

• A data byte is prefaced with a bit of information that tells the receiver that data is coming

• The receiver uses the arrival time of this start bit to synchronize its clock
A Typical Data Frame

The start bit indicates that a byte is coming
A Typical Data Frame

The stop bits allow the receiver to immediately check whether this is a valid frame

- If not, the byte is thrown away
Data Frame Handling

Most of the time, we do not personally deal with the data frame level. Instead, we rely on:

• **Hardware solutions: Universal Asynchronous Receiver Transmitter (UART)**
  – Very common in computing devices
• **Software solutions in libraries**
One Standard: RS232-C

Defines a logic encoding standard:

- “High” is encoded with a voltage of -5 to -15 (-12 to -13V is typical)
- “Low” is encoded with a voltage of 5 to 15 (12 to 13V is typical)
RS232 on the Mega2560

Our mega 8 has FOUR Universal, Asynchronous serial Receiver/Transmitters (UARTs):

• Each handles all of the bit-level manipulation
• You only have to interact with it on the byte level
• Uses 0V and 5V to encode “lows” and “highs”
  – Must convert if talking to a true RS232C device (+/- 13V)
Mega2560 UART C Interface

OUlib support:

```c
fp = serial_init_buffered(1, 9600, 40, 40)
    Initialize port one for a transmission rate of 9600 bits per second (input and output buffers are both 40 characters long)
```

```c
serial_buffered_input_waiting(fp)
    Is there a character in the buffer?
```

See the Atmel HOWTO: examples_2560/serial
Mega2560 UART C Interface

Lib C support (standard C):

`char fgetc(fp)` : receive a character

`fputc('a', fp)` : put a character out to the port

`fputs("foobar", fp)` : put a string out to the port

`fprintf(fp, "foobar %d %s", 45, "baz")` : put a formatted string out to the port
Summary: Using OUlib + LibC

• At the top of your source file:
  
  ```
  #include "oulib_serial_buffered.h"
  ```

• Initialization (in your main() function):
  
  ```
  fp = serial_init_buffered(1, 9600, 40, 40)
  sei();
  ```

• Getting a character:
  
  ```
  char c;
  C = fgetc(fp);
  ```

• Sending a character:
  
  ```
  fputc('f', fp);
  ```
Character Representation

- A “char” is just an 8-bit number
- In some cases, we just interpret it differently.
- But: we can still perform mathematical operations on it
### Character Representation: ASCII

<table>
<thead>
<tr>
<th>Binary</th>
<th>Dec</th>
<th>Hex</th>
<th>Glyph</th>
<th>Binary</th>
<th>Dec</th>
<th>Hex</th>
<th>Glyph</th>
<th>Binary</th>
<th>Dec</th>
<th>Hex</th>
<th>Glyph</th>
</tr>
</thead>
<tbody>
<tr>
<td>001 0000</td>
<td>16</td>
<td>10</td>
<td>a</td>
<td>100 0000</td>
<td>64</td>
<td>40</td>
<td>@</td>
<td>110 0000</td>
<td>96</td>
<td>60</td>
<td>`</td>
</tr>
<tr>
<td>001 0001</td>
<td>17</td>
<td>11</td>
<td>b</td>
<td>100 0010</td>
<td>65</td>
<td>41</td>
<td>A</td>
<td>110 0001</td>
<td>97</td>
<td>61</td>
<td>a</td>
</tr>
<tr>
<td>001 0010</td>
<td>18</td>
<td>12</td>
<td>c</td>
<td>100 0010</td>
<td>66</td>
<td>42</td>
<td>B</td>
<td>110 0010</td>
<td>98</td>
<td>62</td>
<td>b</td>
</tr>
<tr>
<td>001 0011</td>
<td>19</td>
<td>13</td>
<td>d</td>
<td>100 0010</td>
<td>67</td>
<td>43</td>
<td>C</td>
<td>110 0011</td>
<td>99</td>
<td>63</td>
<td>c</td>
</tr>
<tr>
<td>001 0100</td>
<td>20</td>
<td>14</td>
<td>e</td>
<td>100 0100</td>
<td>68</td>
<td>44</td>
<td>D</td>
<td>110 0100</td>
<td>100</td>
<td>64</td>
<td>d</td>
</tr>
<tr>
<td>001 0101</td>
<td>21</td>
<td>15</td>
<td>f</td>
<td>100 0101</td>
<td>69</td>
<td>45</td>
<td>E</td>
<td>110 0101</td>
<td>101</td>
<td>65</td>
<td>e</td>
</tr>
<tr>
<td>001 0110</td>
<td>22</td>
<td>16</td>
<td>g</td>
<td>100 0110</td>
<td>70</td>
<td>46</td>
<td>F</td>
<td>110 0110</td>
<td>102</td>
<td>66</td>
<td>f</td>
</tr>
<tr>
<td>001 0111</td>
<td>23</td>
<td>17</td>
<td>h</td>
<td>100 0111</td>
<td>71</td>
<td>47</td>
<td>G</td>
<td>110 0111</td>
<td>103</td>
<td>67</td>
<td>g</td>
</tr>
<tr>
<td>001 1000</td>
<td>24</td>
<td>18</td>
<td>i</td>
<td>100 1000</td>
<td>72</td>
<td>48</td>
<td>H</td>
<td>110 1000</td>
<td>104</td>
<td>68</td>
<td>h</td>
</tr>
<tr>
<td>001 1001</td>
<td>25</td>
<td>19</td>
<td>j</td>
<td>100 1001</td>
<td>73</td>
<td>49</td>
<td>I</td>
<td>110 1001</td>
<td>105</td>
<td>69</td>
<td>i</td>
</tr>
<tr>
<td>001 1010</td>
<td>26</td>
<td>20</td>
<td>k</td>
<td>100 1010</td>
<td>74</td>
<td>4A</td>
<td>J</td>
<td>110 1010</td>
<td>106</td>
<td>6A</td>
<td>j</td>
</tr>
<tr>
<td>001 1011</td>
<td>27</td>
<td>21</td>
<td>l</td>
<td>100 1011</td>
<td>75</td>
<td>4B</td>
<td>K</td>
<td>110 1011</td>
<td>107</td>
<td>6B</td>
<td>k</td>
</tr>
<tr>
<td>001 1100</td>
<td>28</td>
<td>22</td>
<td>m</td>
<td>100 1100</td>
<td>76</td>
<td>4C</td>
<td>L</td>
<td>110 1100</td>
<td>108</td>
<td>6C</td>
<td>l</td>
</tr>
<tr>
<td>001 1101</td>
<td>29</td>
<td>23</td>
<td>n</td>
<td>100 1101</td>
<td>77</td>
<td>4D</td>
<td>M</td>
<td>110 1101</td>
<td>109</td>
<td>6D</td>
<td>m</td>
</tr>
<tr>
<td>001 1110</td>
<td>30</td>
<td>24</td>
<td>o</td>
<td>100 1110</td>
<td>78</td>
<td>4E</td>
<td>N</td>
<td>110 1110</td>
<td>110</td>
<td>6E</td>
<td>n</td>
</tr>
<tr>
<td>001 1111</td>
<td>31</td>
<td>25</td>
<td>p</td>
<td>100 1111</td>
<td>79</td>
<td>4F</td>
<td>O</td>
<td>110 1111</td>
<td>111</td>
<td>6F</td>
<td>o</td>
</tr>
<tr>
<td>010 0000</td>
<td>32</td>
<td>26</td>
<td>q</td>
<td>101 0000</td>
<td>80</td>
<td>50</td>
<td>P</td>
<td>111 0000</td>
<td>112</td>
<td>70</td>
<td>p</td>
</tr>
<tr>
<td>010 0001</td>
<td>33</td>
<td>27</td>
<td>r</td>
<td>101 0001</td>
<td>81</td>
<td>51</td>
<td>Q</td>
<td>111 0001</td>
<td>113</td>
<td>71</td>
<td>q</td>
</tr>
<tr>
<td>010 0010</td>
<td>34</td>
<td>28</td>
<td>s</td>
<td>101 0010</td>
<td>82</td>
<td>52</td>
<td>R</td>
<td>111 0010</td>
<td>114</td>
<td>72</td>
<td>r</td>
</tr>
<tr>
<td>010 0011</td>
<td>35</td>
<td>29</td>
<td>t</td>
<td>101 0011</td>
<td>83</td>
<td>53</td>
<td>S</td>
<td>111 0011</td>
<td>115</td>
<td>73</td>
<td>s</td>
</tr>
<tr>
<td>010 0100</td>
<td>36</td>
<td>30</td>
<td>u</td>
<td>101 0100</td>
<td>84</td>
<td>54</td>
<td>T</td>
<td>111 0100</td>
<td>116</td>
<td>74</td>
<td>t</td>
</tr>
<tr>
<td>010 0101</td>
<td>37</td>
<td>31</td>
<td>v</td>
<td>101 0101</td>
<td>85</td>
<td>55</td>
<td>U</td>
<td>111 0101</td>
<td>117</td>
<td>75</td>
<td>u</td>
</tr>
<tr>
<td>010 0110</td>
<td>38</td>
<td>32</td>
<td>w</td>
<td>101 0110</td>
<td>86</td>
<td>56</td>
<td>V</td>
<td>111 0110</td>
<td>118</td>
<td>76</td>
<td>v</td>
</tr>
<tr>
<td>010 0111</td>
<td>39</td>
<td>33</td>
<td>x</td>
<td>101 0111</td>
<td>87</td>
<td>57</td>
<td>W</td>
<td>111 0111</td>
<td>119</td>
<td>77</td>
<td>w</td>
</tr>
<tr>
<td>010 1000</td>
<td>40</td>
<td>34</td>
<td>y</td>
<td>101 1000</td>
<td>88</td>
<td>58</td>
<td>X</td>
<td>111 1000</td>
<td>120</td>
<td>78</td>
<td>x</td>
</tr>
<tr>
<td>010 1001</td>
<td>41</td>
<td>35</td>
<td>z</td>
<td>101 1001</td>
<td>89</td>
<td>59</td>
<td>Y</td>
<td>111 1001</td>
<td>121</td>
<td>79</td>
<td>y</td>
</tr>
<tr>
<td>010 1010</td>
<td>42</td>
<td>36</td>
<td>\</td>
<td>101 1010</td>
<td>90</td>
<td>5A</td>
<td>Z</td>
<td>111 1010</td>
<td>122</td>
<td>7A</td>
<td>z</td>
</tr>
<tr>
<td>010 1011</td>
<td>43</td>
<td>37</td>
<td>]</td>
<td>101 1011</td>
<td>91</td>
<td>5B</td>
<td></td>
<td>111 1011</td>
<td>123</td>
<td>7B</td>
<td>]</td>
</tr>
<tr>
<td>010 1100</td>
<td>44</td>
<td>38</td>
<td>^</td>
<td>101 1100</td>
<td>92</td>
<td>5C</td>
<td>\</td>
<td>111 1100</td>
<td>124</td>
<td>7C</td>
<td>\</td>
</tr>
<tr>
<td>010 1101</td>
<td>45</td>
<td>39</td>
<td>_</td>
<td>101 1101</td>
<td>93</td>
<td>5D</td>
<td></td>
<td>111 1101</td>
<td>125</td>
<td>7D</td>
<td>)</td>
</tr>
<tr>
<td>010 1110</td>
<td>46</td>
<td>3A</td>
<td>`</td>
<td>101 1110</td>
<td>94</td>
<td>5E</td>
<td>^</td>
<td>111 1110</td>
<td>126</td>
<td>7E</td>
<td>~</td>
</tr>
</tbody>
</table>
Reading a Byte from the Serial Port

```c
int c;

c=fgetc(fp);
```

Note: fgetc() “blocks” until a byte is available
• Will only return with a value once a character is available to be returned
Processing Serial Input

```c
int c;
while(1) {
    if(serial_buffered_input_waiting(fp)) {
        // A character is available for reading
        c = fgetc(fp);
        <do something with the character>
    }
    <do something else while waiting>
}

serial_buffered_input_waiting(fp) tells us whether a byte
is ready to be read
```
Mega2560 UART C Interface

Also available:

- `fscanf()`: formatted input

See the LibC documentation or the AVR C textbook
Physical Interface

Four matched pairs of transmit and receive pins (TX? and RX?)
Physical Interface

Port 0 is also connected to the USB port

See “hyperterm” on downloads page.
Physical Interface

Our compass module also speaks 0/5 V RS232

• See project 2 for details