The Handyboard Architecture

Handyboard with Extender Interfaces
The Handyboard Libraries

The Interactive-C library for the Handyboard is lib_hb and it contains a variety of functions to make it easier to program the Handyboard without writing in assembler. These functions can be categorized as follows:

- Timer management
- System interrupt control
- Multitasking support
- Motor controls
- Sensor inputs
- Utilities
Timer Management Functions

- `void reset_system_time()`: Set the system time to zero.
- `long mseconds()`: Get the current time in milliseconds.
- `float seconds()`: Get the current time in seconds.
- `void sleep(float sec)`: Wait `sec` seconds.
- `void msleep(long msec)`: Wait `msec` milliseconds.
- `void beep()`: Sound the beeper.
- `void beeper_on()`: Enable the beeper.
- `void beeper_off()`: Disable the beeper.
- `void tone(float f, float l)`: Sound frequency `f` for time `l`.
- `void set_beeper_pitch(float f)`: Set the beeper pitch to `f`.
# System Interrupt Control

These functions control certain capabilities that depend on system interrupts to operate. The 5 capabilities are:

<table>
<thead>
<tr>
<th>Capability</th>
<th>Default Status</th>
<th>CPU %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulse Width Modulation</td>
<td>ON</td>
<td>3</td>
</tr>
<tr>
<td>Infrared decoding</td>
<td>ON</td>
<td>11</td>
</tr>
<tr>
<td>LCD Printing</td>
<td>ON</td>
<td>8</td>
</tr>
<tr>
<td>Quad Shaft Decoding</td>
<td>OFF</td>
<td>5</td>
</tr>
<tr>
<td>IR Transmission</td>
<td>OFF</td>
<td>1</td>
</tr>
</tbody>
</table>
PWM and LCD Controls

<table>
<thead>
<tr>
<th>PWM</th>
<th>void system_pwm_on()</th>
<th>Speed bytes are used to control motors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>void system_pwm_off()</td>
<td>Motors run at full speed</td>
</tr>
<tr>
<td>LCD</td>
<td>void system_print_on()</td>
<td>Turn on printing to LCD</td>
</tr>
<tr>
<td></td>
<td>void system_print_off()</td>
<td>Turn off printing to LCD</td>
</tr>
</tbody>
</table>
Infrared Control Functions

The infrared functions are in the sony-ir.icb which must be downloaded. It can be placed in the lib hb.lis file for convenience, but r22_ir.lis cannot be present. The older IR functionality has been obsoleted.

int sony_init (1)  Enable the sony IR driver.
int sony_init (0)  Disable the sony IR driver.
int ir_data (int d)  Get the last by received by the IR driver or zero.
Quad Shaft Encoding

Quadrature shaft encoding is used to make exact determinations of the direction and amount of travel of a wheel or possibly some other object. The quad shaft encoding functions are in encoder.c.

```c
void enable_encoder (int n)    // Enable encoder n
void disable_encoder (int n)   // Disable encoder n
void reset_encoder (int n)     // Reset encoder n
int read_encoder (int n)       // Read encoder n
```
## Multitasking Support

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void hog_processor ()</td>
<td>Gives the caller 256 clock ticks before swapping</td>
</tr>
<tr>
<td>int start_process (void *f(), int t, int s)</td>
<td>Start a process that calls f, runs for t ms per invocation and has stack size s.</td>
</tr>
<tr>
<td>int kill_process (int i)</td>
<td>Destroy process i returning 1 if it fails. Called by a process to call to be swapped out.</td>
</tr>
</tbody>
</table>
Motor Control

Motors are numbered 0 - 3.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void fd (int n)</td>
<td>Run motor n at full speed in the green direction</td>
</tr>
<tr>
<td>void bk (int n)</td>
<td>Run motor n at full speed in the red direction</td>
</tr>
<tr>
<td>void motor (int n, int s)</td>
<td>Start motor n at speed s, where s is -100 - +100, - is backward, 0 is stop</td>
</tr>
<tr>
<td>void off (int n)</td>
<td>Turn off motor n</td>
</tr>
<tr>
<td>void alloff ()</td>
<td>Turn off all motors</td>
</tr>
<tr>
<td>void ao ()</td>
<td>Synonym for alloff</td>
</tr>
</tbody>
</table>
Servo Motor Control

The servo motor functions are in servo.c.

- `void servo_on ()` Enables servo control.
- `void servo_off ()` Disables servo control.
- `int servo (int perios)` Sets the length of the servo control pulse in half-seconds (1400 - 4860).
- `int servo_rad (float angle)` Sets the servo angle in radians.
- `int servo_deg (float angle)` Sets the servo angle in degrees.
Sensor Inputs

Sensor values are 1 for true and 0 for false and are assumed to be active low - a low value is true.

- `int stop_button ()` Returns the value of the stop button.
- `int start_button ()` Returns the value of the start button.
- `int digital (int n)` Returns the value of the sensor in sensor port p.
- `int analog (int n)` Returns the value of the sensor in sensor port p in the range 0 - 255.
int stop_press ()  Waits for the stop button to be pressed and released.
int start_press ()  Waits for the start button to be pressed and released.
int knob ()       Returns the position of a knob as the value from 0 - 255.
int random (int mod)  Returns a random number in [0 - 32767].
IC Utilities

Interactive-C provides non-standard C functions that are valuable in working with the Handyboard. Some of the multi-tasking functions fall into this category as well.

int peek (int l)  
Returns the byte at location l.

int peekword (int l)  
Returns the word at location l.

int poke (int l, int b)  
Stores byte b at location l.

int pokeword (int l, int w)  
Stores word w at location l.

int bit_set (int l, int m)  
Sets bits in location l that are set in m.

int bit_clear (int l, int m)  
Clears bits in location l that are set in m.
Trig Functions

float f (float a)  Trig functions include sin, cos, tan, atan, sqrt, log, log10, exp and exp10.

\[ a \uparrow b \]  Raises \( a \) to the \( b \)th power where \( b \) and the result are floats.