Inside the Handy Cricket
Fred Martin
fredm@handyboard.com

Overview

• Origins and Status of Cricket Technology
• Cricket Hardware
• Cricket Firmware and Compiler
• Cricket Bus
Origins and Status

- Developed by Brian Silverman and Fred Martin at MIT Media Laboratory
- Have license to all code, designs up to Aug 2000
- Through Gleason Research, distributing Handy Crickets & more
- Publishing APIs
- MIT continuing work as well
The Handy Cricket

2 Bus Ports
Piezo Beeper
2 Sensor Inputs
IR Comms
2 DC Motor Outputs
Power Switch
4Kbyte Memory
Run/Stop Button & LED

Cricket Software

- Virtual Machine resident in Cricket Chip (written in PIC assembly)
- Logo compiler running on workstation (written in Java)
Cricket Logo Compiler

- Translates user Logo into VM bytecodes
- Interactive “Command Center”
- Buffer for editing and defining procedures

Cricket Logo VM Primitives

<table>
<thead>
<tr>
<th>16-Bit Numerics</th>
<th>Control Structures</th>
<th>Internal</th>
<th>Timing/Sound</th>
</tr>
</thead>
<tbody>
<tr>
<td>addition +</td>
<td>if-then</td>
<td>push byte on stack</td>
<td></td>
</tr>
<tr>
<td>subtraction -</td>
<td>if-then-else</td>
<td>push word on stack</td>
<td></td>
</tr>
<tr>
<td>multiplication *</td>
<td>ifelse</td>
<td>start code block</td>
<td></td>
</tr>
<tr>
<td>division /</td>
<td>wait until true</td>
<td>end code block</td>
<td></td>
</tr>
<tr>
<td>remainder %</td>
<td>wait until</td>
<td>with return value</td>
<td></td>
</tr>
<tr>
<td>equality =</td>
<td>counted loop</td>
<td>define procedure</td>
<td></td>
</tr>
<tr>
<td>greater-than &gt;</td>
<td>repeat</td>
<td>retrieve proc arg</td>
<td></td>
</tr>
<tr>
<td>less-than &lt;</td>
<td>infinite loop</td>
<td>code end marker</td>
<td></td>
</tr>
<tr>
<td>bitwise AND and</td>
<td>return value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bitwise OR or</td>
<td>output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bitwise XOR xor</td>
<td>terminate procedure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bitwise NOT not</td>
<td>terminate all proc's</td>
<td></td>
<td></td>
</tr>
<tr>
<td>random # random</td>
<td>launch daemon</td>
<td></td>
<td></td>
</tr>
<tr>
<td>high-byte, low-byte</td>
<td>when</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor/Motor</td>
<td>stop daemon</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sensora</td>
<td>whenon</td>
<td></td>
<td></td>
</tr>
<tr>
<td>switcha</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sensorb</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>switchb</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a, b, c, d, cd, abod, on off</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rd thisway thatway</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>setpower brake</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Data Structures

- 16 RAM-based global vars
- single-dimensional arrays
- data capture buffer
- memory peek and poke

Internal

- push byte on stack
- push word on stack
- start code block
- end code block
- with return value
- define procedure
- retrieve proc arg
- code end marker

Timing/Sound

- onfor <duration>
- wait <duration>
- bstop
- note <pitch> <dur>
- reset
- timer
### Compiler Examples

<table>
<thead>
<tr>
<th>This Logo source....</th>
<th>compiles to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>beep</td>
<td>12 &lt;bEEP&gt; 0 &lt;end&gt;</td>
</tr>
<tr>
<td>a, onfor 100</td>
<td>46 &lt;a&gt;, 1 100 &lt;push 100&gt; 50 &lt;onfor&gt; 0 &lt;end&gt;</td>
</tr>
<tr>
<td>send 3 + 4</td>
<td>1 3 &lt;push 3&gt; 1 4 &lt;push 4&gt; 23 &lt;plus&gt; 19 &lt;end&gt; 0 &lt;end&gt;</td>
</tr>
<tr>
<td>repeat 10 [onfor 20 beep]</td>
<td>1 10 &lt;push 10&gt; 3 &lt;start block&gt; 5 &lt;block size&gt; 1 20 &lt;push 20&gt; 50 &lt;onfor&gt; 12 &lt;beep&gt; 4 &lt;end block&gt; 9 &lt;repeat&gt; 0 &lt;end&gt;</td>
</tr>
</tbody>
</table>

### Cricket Bus System

- “One-Wire” comms with 1-master, multiple-slaves protocol
- Send and receive can be easily implemented with any micro
- Bundle device-specific driver code with HW in device itself
- Allow arbitrary expansion capability
Handy Cricket Bus Jack

- Unregulated power output
- Bus signal
- Logic power output
- Gnd

top view, looking down at Cricket’s Bus receptacle

Bus Signal

- Long start bit allows receiver to sync
- Data bits just 10 µs long
- 9th bit indicates command or data
4-Digit LED Display

4-Digit Bus Protocol

<table>
<thead>
<tr>
<th>Command Device Class</th>
<th>Data Device ID</th>
<th>Data Value</th>
<th>Data Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-digit display</td>
<td>0 = any</td>
<td>HIGH byte of num to display</td>
<td>LOW byte of num to display</td>
</tr>
<tr>
<td>$10</td>
<td>1 = display #1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 = display #2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

bsend $110          bsend 0          bsend high-byte in  bsend low-byte in
Logo Driver Code for Display

to display :n
  bsend $110
  bsend 0
  bsend high-byte :n
  bsend low-byte :n
end

“Gama Dome” Block Diagram

Sharp IR Distance front  Sharp IR Distance side  Photocell front differential  Photocell front magnitude

Cricket "W" WALL FOLLOWER CODE

Cricket "L" LIGHT FOLLOWER CODE

IR Comm Link

Cricket Bus

Gama-1 PID Motor Controller  Gama-1 PID Motor Controller

left motor  right motor

Photocell side magnitude
Concluding Remarks

- Cricket system is Cricket + Software + Bus Devices
- Modular and expandable tool that allows iterative design
- Open for others’ contributions and adaptations
- Not just for roboticists