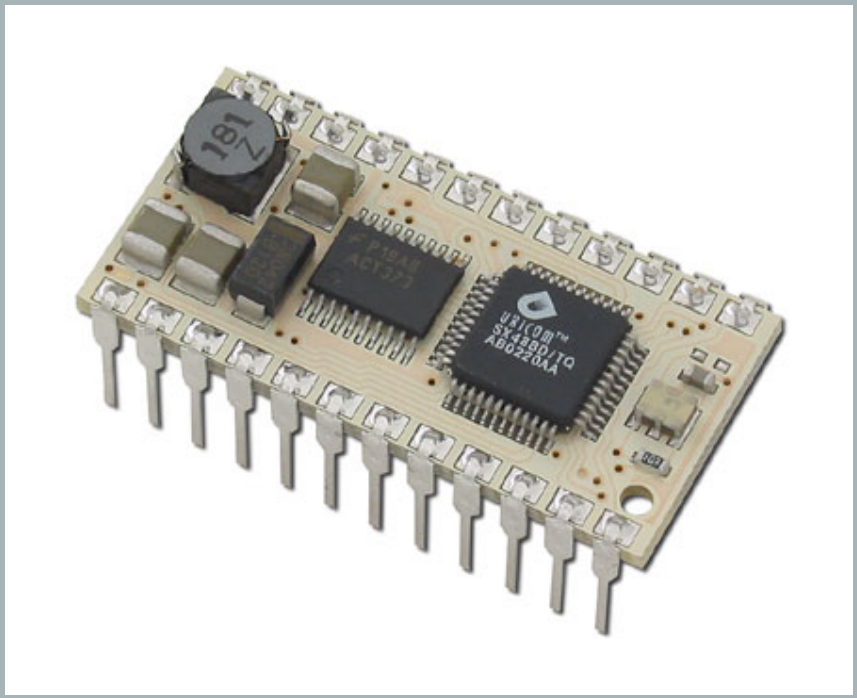
Embedded Java: Chips, Javelins, and Robots



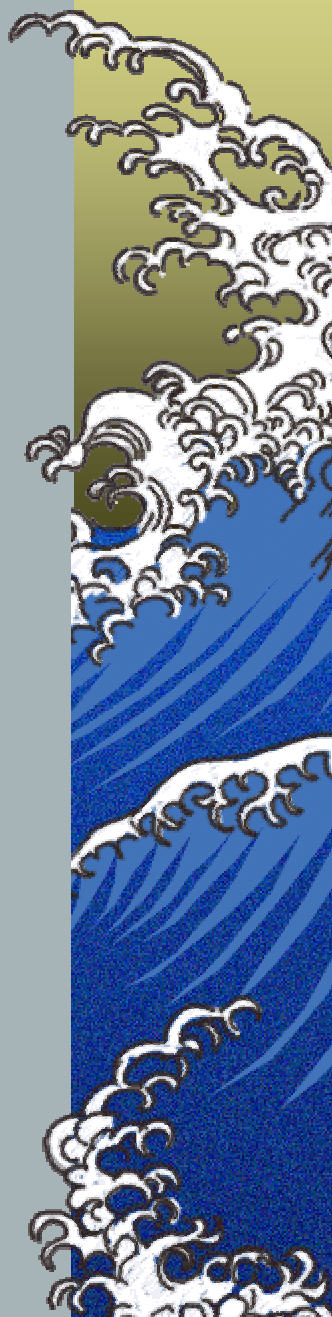
Jeffrey Peacock

jeffp@JeffreyPeacock.com

Introduction

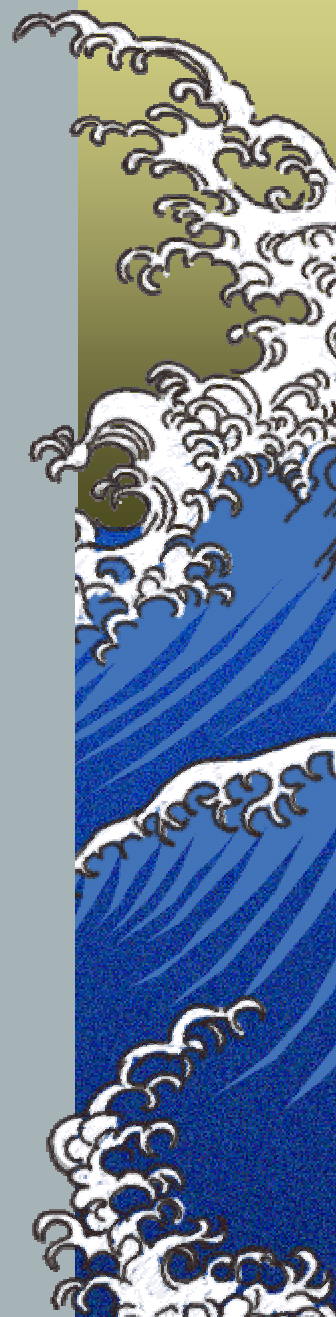
Why embedded Java?

- *Consistent language features.*
- *Reusability.*
- *Write once. Port everywhere.*
- *If you've got to work on limited hardware then you might as well have some creature comforts.*



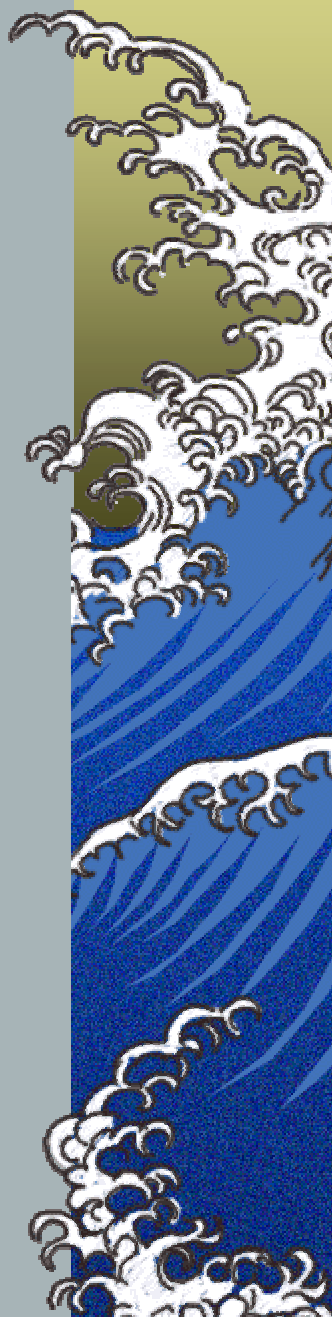
1st the Basic Stamp

- *Parallax, released in 1992*
- *Single 24-pin DIP, BASIC language*
- *~ the size of a commemorative stamp*
- *Very popular for education, electronics and robotics*
- *Parallax claims even "aerospace subsystem designs"*



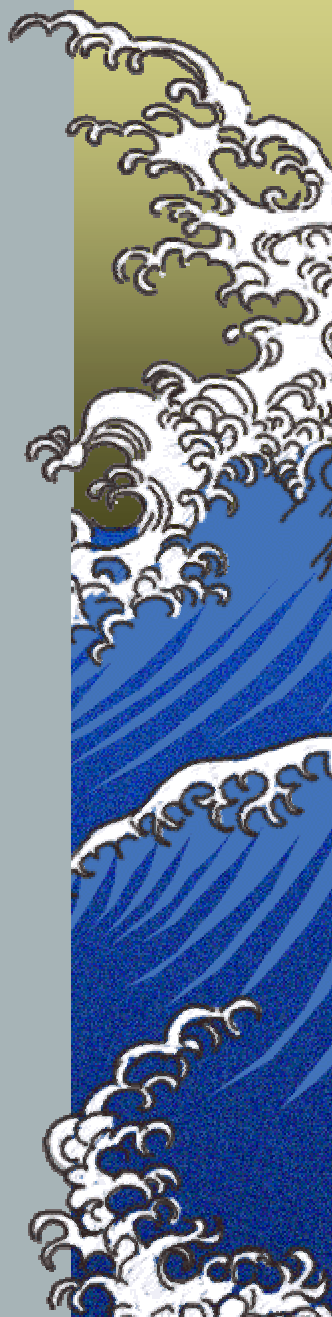
Javelin Features

- *Single 24-pin DIP, BS pin compatible*
- *Java 1.2 Subset*
- *Ubicom SX48AC MCU @ 25 MHz Turbo (~8,500 instructions/sec.)*
- *32K RAM/32K EEPROM (Program) Memory*
- *16 I/O Pins*



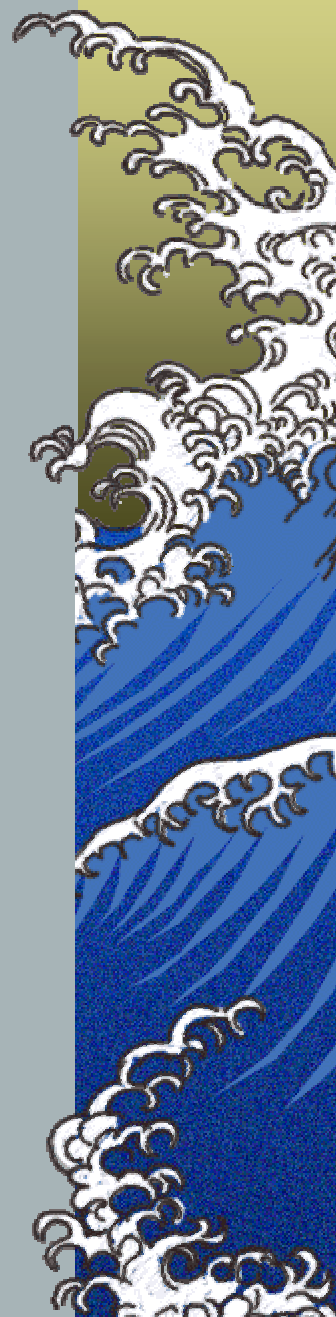
Javelin Features (cont.)

- *Virtual Peripherals (VPs) that take care of serial communication, pulse width modulation and tracking time in the background.*
- *Serial communication is buffered as a background process.*



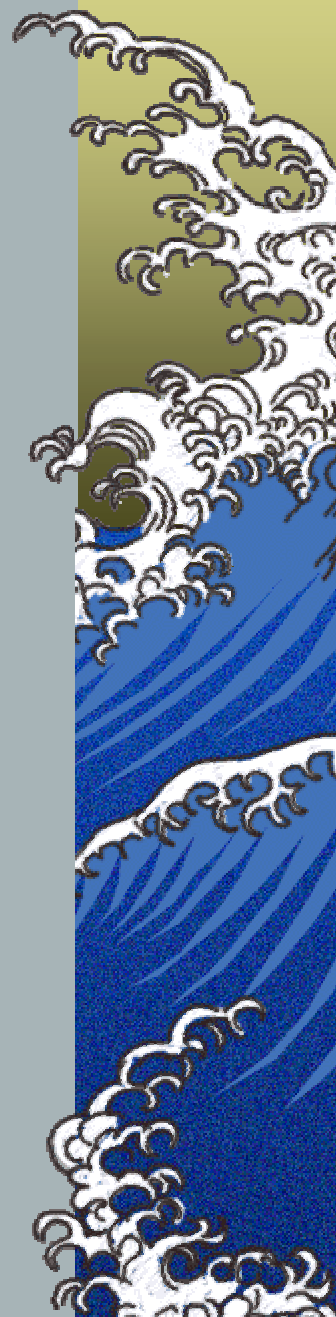
Javelin IDE

- *Runs on PC host*
- *Connects to Javelin via serial cable*
- *Integrated in-circuit debugging*
- *Single-step, run, stop, reset*
- *Multiple breakpoints*
- *Inspection of all variables and objects, both static and dynamically allocated*



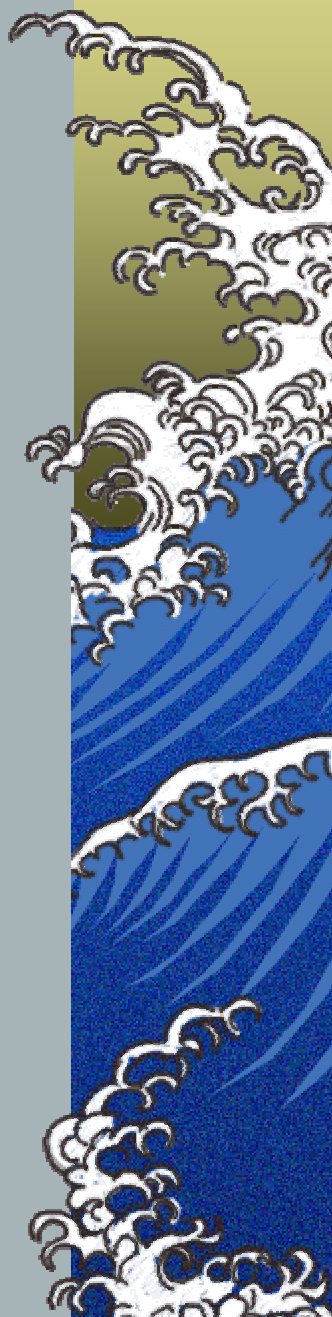
Javelin IDE (cont.)

- *Stack backtrace*
- *Memory usage*
- *I/O pin states*
- *Built-in bi-directional serial message terminal for System.out.println() and*
- *Terminal.getChar() type debugging*



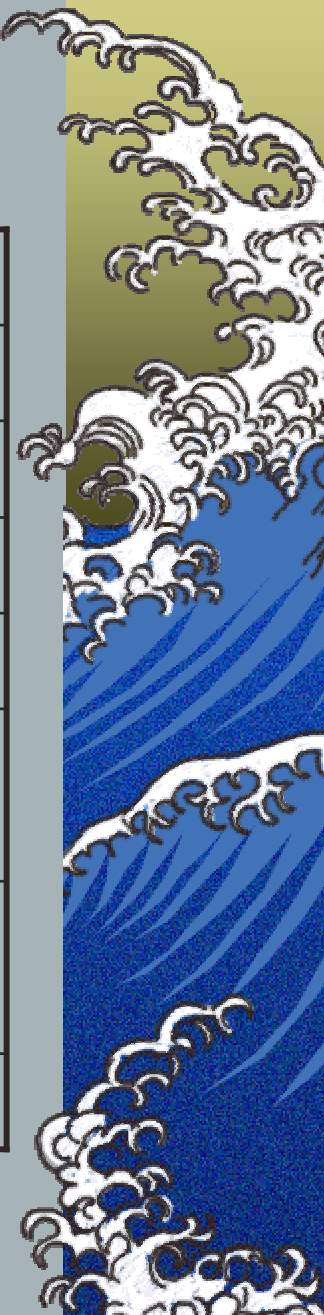
Javelin Java

- *Summary of Differences:*
  - *Subset of Primitive Data Types*
  - *No Interfaces*
  - *Subset of Java Libraries*
  - *Single Thread*
  - *No Garbage Collection*
  - *Strings are ASCII*
  - *One Dimensional Arrays*



Primitive Data Types

Type	***Supported***
`boolean`	*Yes*
`byte`	*Yes*
`char`	*Yes*
`short`	*Yes*
`int`	*16-bit* *(32-bit library available)*
`float`	*No* *(6DP library available)*
`double`, `long`	*No (for now, BigDecimal?)*

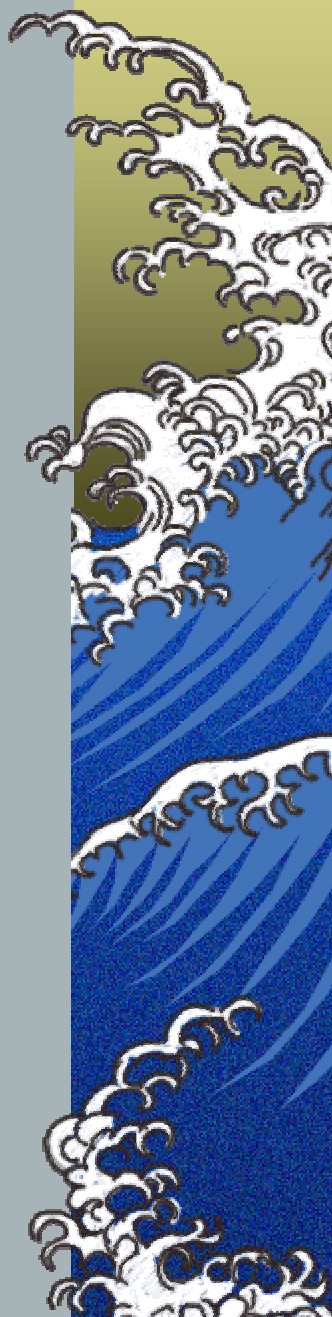


Subset of Java Libraries

- *java.lang*
  - *Primitive wrapper classes,* `Class`, `Object`, `System`, `Throwable, Exception`*'s*
  - `System` *is almost completely stripped*
- *java.io*

  Only `PrintStream`

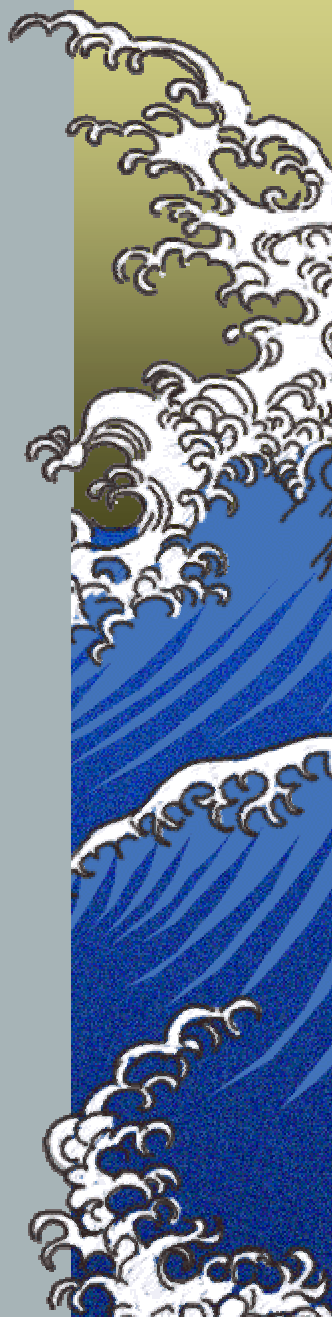
  `Serializable` *a pacifier for compilers*
- *java.util*
  - *Just can't live without* `Random`



Single Thread

Some relief:

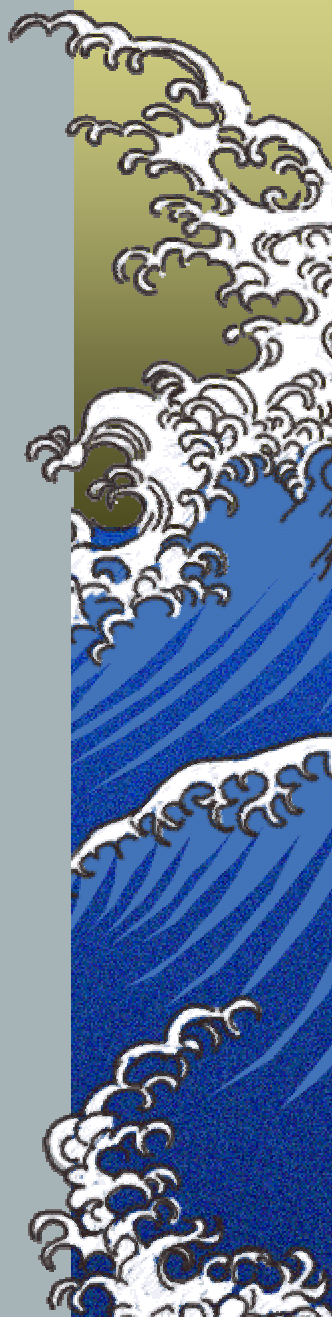
- *Virtual Processes*
- `stamp.core.Timer`
- *Co-operative RTOS classes*
  - `stamp.util.os.Task`
  - `stamp.util.os.TaskManager`



Virtual Processes

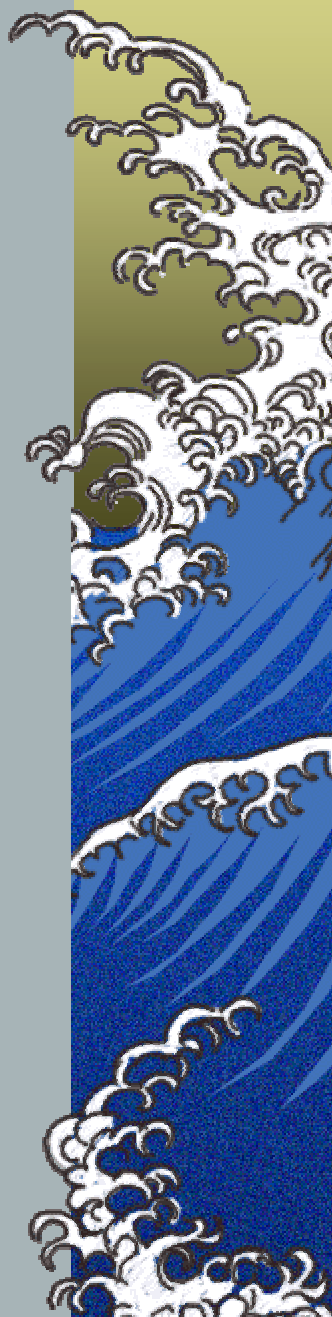
- ***Foreground***
  - Pulse count
  - Pulse width measurement
  - Pulse generation
  - RC Timer
  - SPI master
- ***Background***
  - UART

    Full duplex, HW flow control, buffered
  - PWM
  - 32-bit Timer
  - 1-bit DAC
  - Delta/Sigma ADC



No GC

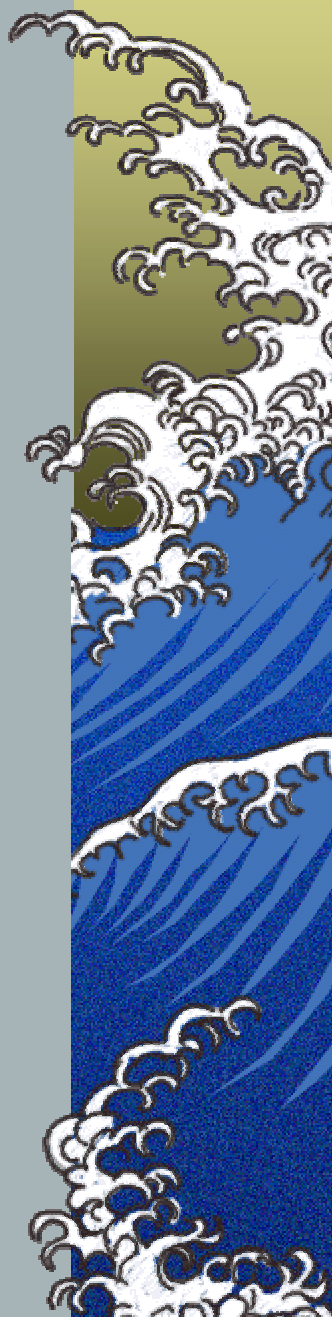
- *A nod to RT embedded system perf(?)*
- *Requires some adjustment for the standard Java programmer*
  - *Use* **`static`** *variables whenever possible.*
  - *Use the* **`StringBuffer`** *not* **`String`**
  - **`String`***s are reusable*
    `public void setCharArray(char[] a)`
  - ***Reuse, reuse, reuse!!***



ARobot

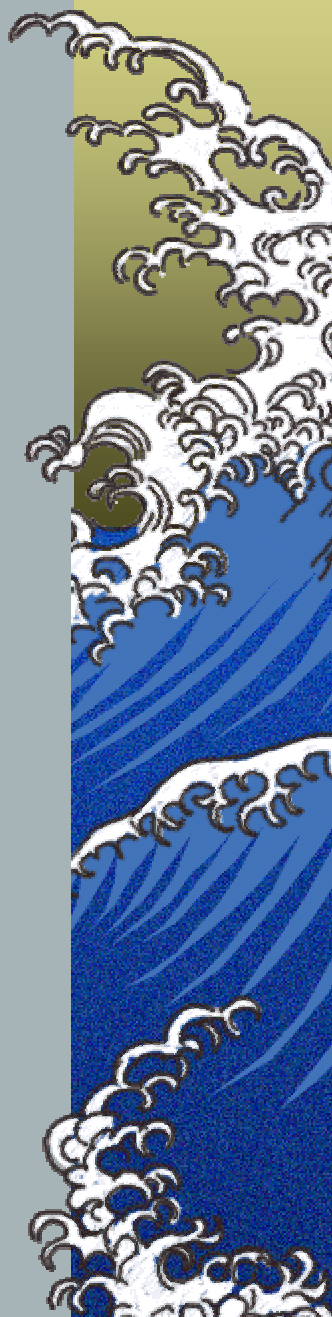
Arrick.com kit

- *Rugged aluminum frame (no plastic or wood)*
- *Dual front whisker sensors*
- *Rear wheel steering servo motor*
- *Front wheel DC gear drive motor*
- *Optical wheel encoder for distance measurement*
- *On-board coprocessor handles all motor control*



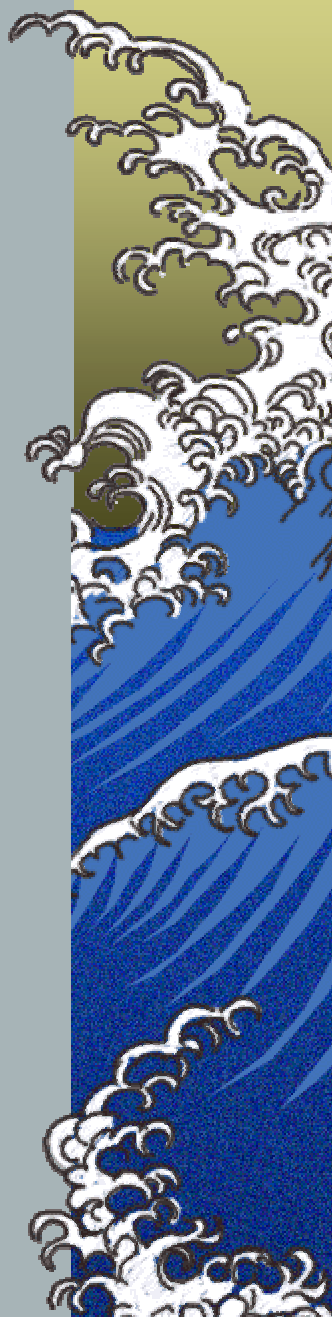
ARobot (cont.)

- *Controllable Red and Green LEDs*
- *Sound output transducer*
- *Two user defined push button switches*
- *3 User-defined RC servo control ports*
- *Serial communications port*
- *Expansion connector*
- *Socket to accept a Basic Stamp II or Javelin controller chip*

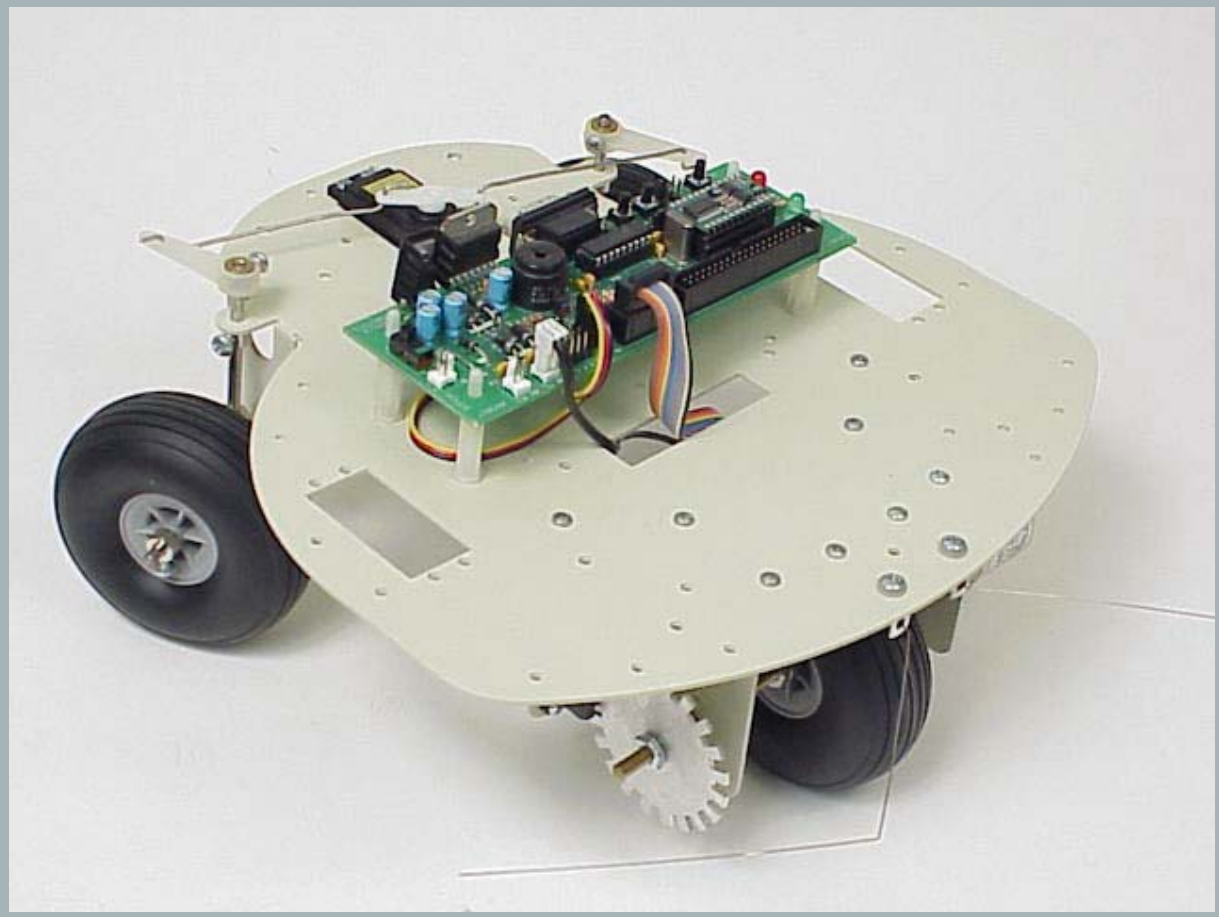


ARobot (cont.)

- *Dimensions: 10" x 10", 5" tall, 2-1/4 lbs*
- *Runs on 8 AA-cell batteries for 5 hours or more (myth?)*

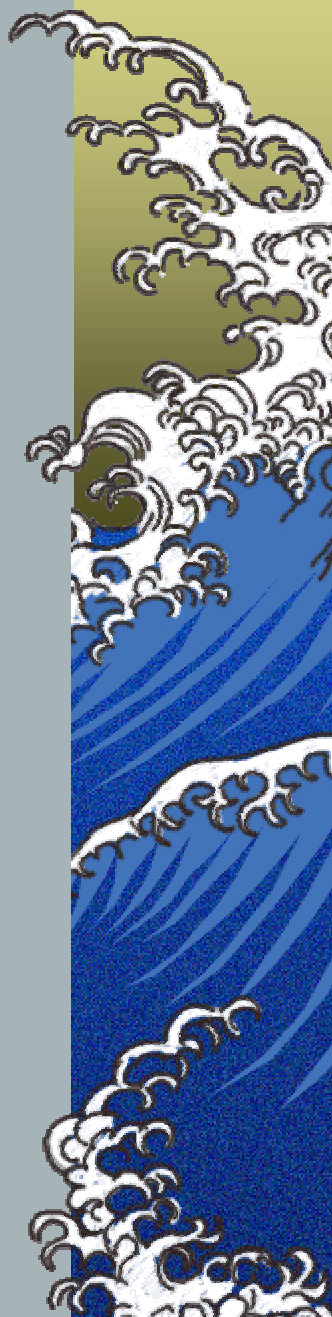


ARobot (cont.)



Robot Expansions

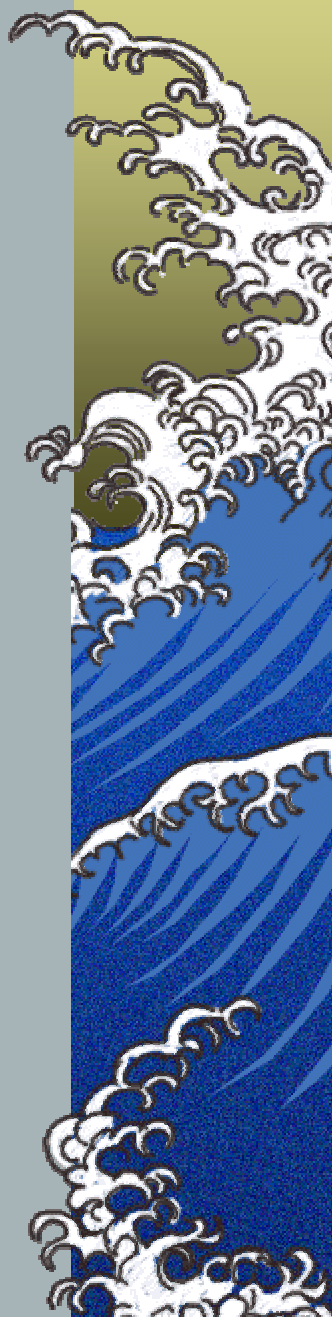
- *Expansion board*
- *Ultrasonic Range Finder, light sensor I2C*
- *64K EEPROM Memory I2C*
- *RT Clock I2C*
- *LCD Display*
- *AC adapter*



Future Expansions

Goal is to explore and record

- *More ultrasonic RF's, complete coverage*
- *More environmental sensing: temp. humidity, sound, etc.*
- *More memory for recording data*
- *Electronic compass & GPS*



Challenges

The GOOD …

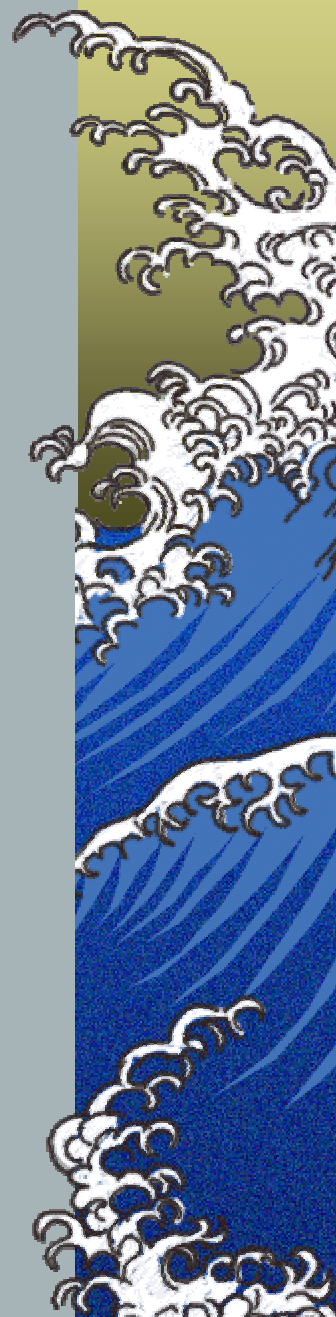
Get to interact with the physical world!!!

The BAD …

HW-Debugging can be extra frustrating

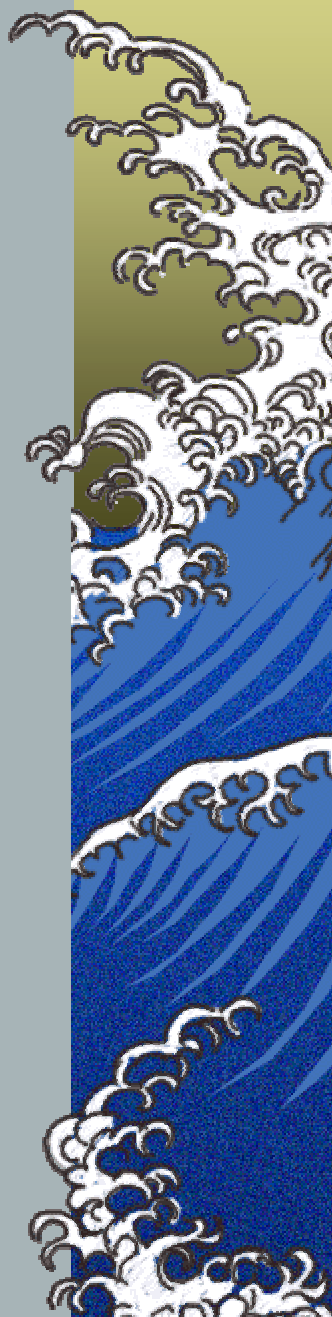
& The UGLY…

Most programming examples need to be converted from Basic



Where to Get More Information

- *Parallax: www.Parallax.com*
  - ***www.parallax.com/dl/docs/prod/stamps/stampscomparison.pdf***
- *Javelin Stamp:*
  - *www.parallax.com/javelin/index.asp*
  - *Javelin manual **plus** errata*
- *www.Arrick.com*
  - *"Robot Building for Dummies"*



Cool Robots

- *Mars Exploration Rover Mission*
  - ***http://marsrovers.jpl.nasa.gov/home/***
- *Mars Time (using Java)*
  - ***http://www.giss.nasa.gov/tools/mars24/***
- *iRobot*
  - *http://www.irobot.com*
  - *Roomba – commercial vacuum cleaner*
  - *PackBot – unmanned tactical*