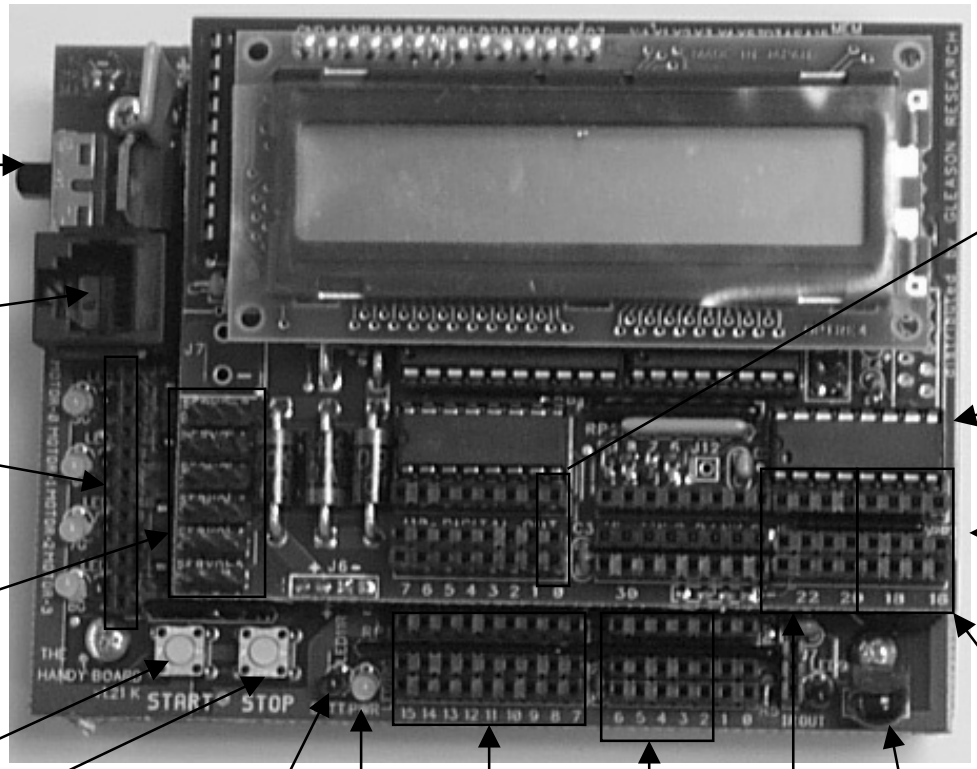
Get Started with IC Environment

- Download IC from www.kipr.org/ic
- Double click on installer
- Start the IC application
- Click on the picture of the Handy Board
- Click on the **Connect Later** button
- Click on the **New** button (upper left corner)
- Type in the program that prints out a text string
- Save the file using the **Save** button (name it whatever you want -- just remember where you put it).

The Handy Board



off
Power switch
on

download connector

Motor Ports (3 pins ea)

Servo ports
B|R|Y wires

start and stop buttons

lo-batt
led

power led
off in download mode

digital
ports
7-15

analog
ports
2-6

analog
ports
20-23

IR receiver

floating analog
ports 16-19

knob (under
exp board)

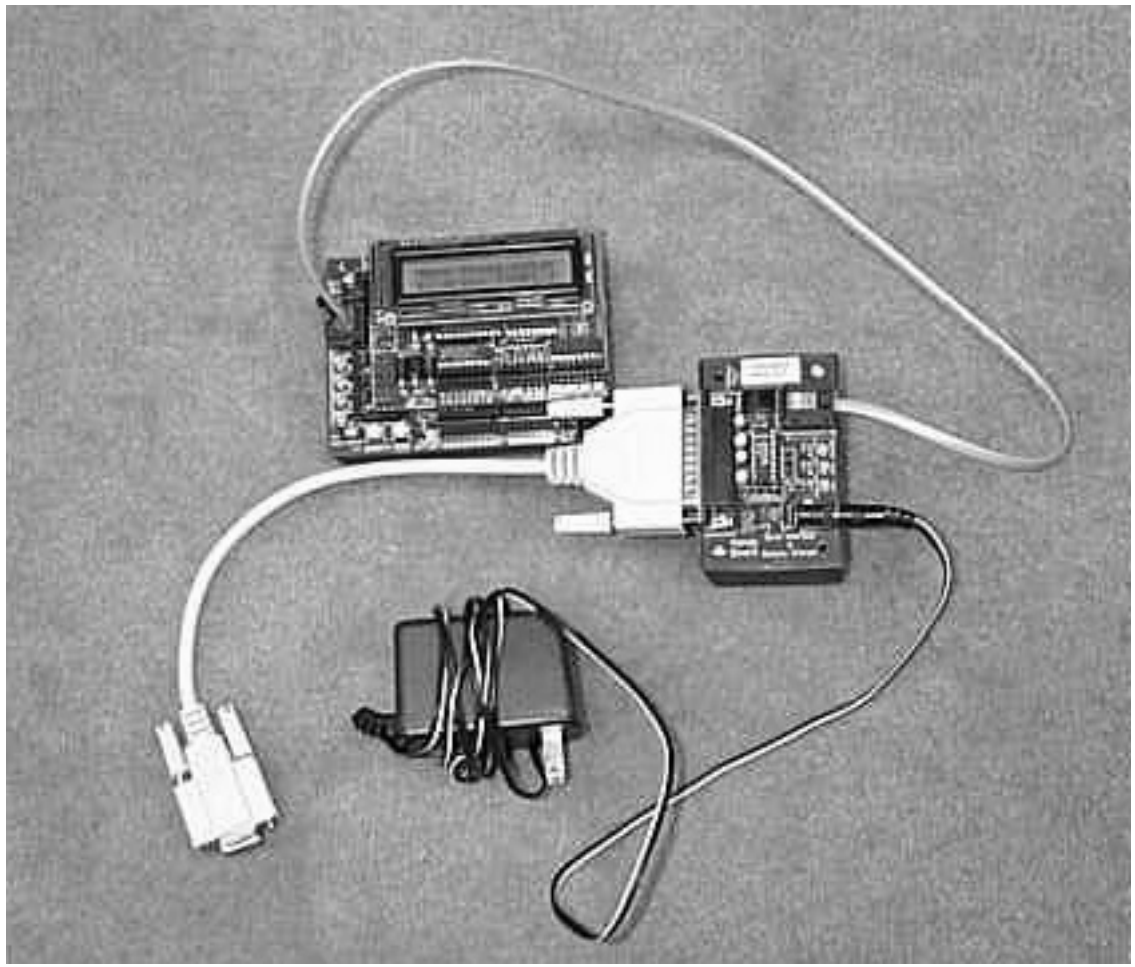
Charge port
(under exp board)

sonar red port

Handy Board Checklist

- Handy Board
  - Display
  - Expansion Board
  - Main Board
  - Battery Box
- HB to Interface cable (phone cord)
- Interface Board
- Computer to Interface cable (25 pin to 9 pin)
- AC Adapter (note: ground center, +12 outside)

Handy Board Setup



Download the Firmware

- Make sure your Handy Board is connected to your personal computer via serial port cable, serial interface box and 4-pin modular cable
- Select **Download Firmware** from the ***Settings*** menu
- Select the appropriate serial port
- Click on **Download Firmware** button
- Follow the onscreen directions

Interacting with IC

- Click on the **Interaction** tab
- Just type into area at bottom of IC window
- Simple expressions

```
2+2;
```

- Making noise

```
beep();
```

- Printing to the LCD screen

```
printf("I'm printing!!\n");
```

Download a Program

- Select the tab with your program's name and click download
- To run your program (**`main`** function), turn the Handy Board off and then turn it back on
- To turn your HB on without running your program, hold down the start button while sliding the switch

Data Types

- **`int`** (16 bit integer number in IC)

  +/-32,767
- **`long`** (32 bit long integer number)

  +/-2,147,483,647
- **`float`** (32 bit floating point number)

  e.g. 3.1416
- and others (see IC documentation)

Integer Arithmetic

- + addition

  X + Y means X plus Y
- - subtraction

  X - Y means X minus Y
- * multiplication

  X * Y means X multiplied by Y
- / division

  X / Y means X divided by Y with the decimal truncated
- % modulus (remainder)

  X % Y means the remainder of X divided by Y

Floating Point Arithmetic

- +,-,* operate the same as with integers
- / yields a floating point number
- % is not defined with floating point numbers

IC Interaction Environment

- You can type C statements (and C blocks) in the interaction window
- Pressing return will send that code to the IC interpreter which executes the code on the Handy Board
- This is a quick way to test things
- Try it!

Useful Function: **`sleep(`***seconds*`);`**

- **`sleep()`** delays the function's execution for an amount of time equal to the number of seconds (expressed as a float) given as an argument

C Function: `printf();`

- Takes a quoted string and prints it out
  - %d is used for an integer
  - %f is used for a float
  - \n starts next character back at upper left corner (or new line on terminal)
  - commas separate all arguments after the quoted string

Exercise

- Write a program that prints out the answer to a math problem on the LCD for each of the math operations +,-,*,/,%
  - **`printf(“13+5=%d\n”, 13+5);`**
  - **`printf(“2.0*1.5=%f\n”, 2.0*1.5);`**
- Put a three second delay between each print statement
  - **`sleep(3.0);`**

More IC Library Functions

- **`start_button();`**
- **`stop_button();`**
- **`knob();`**
- **`beep();`**

New Features of IC 4.0

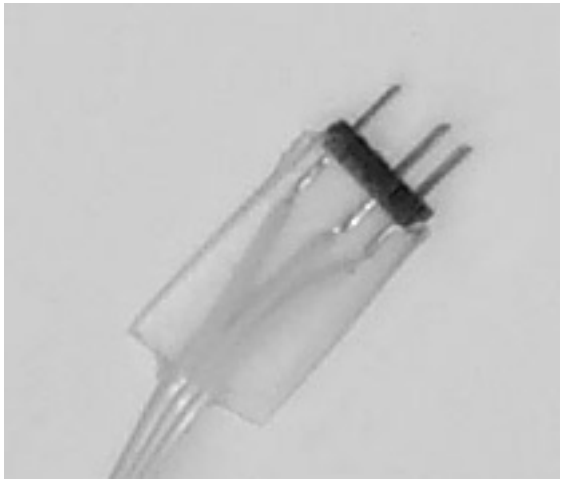
- Tools menu
  - List functions
  - List global variables (e.g., **`servo0`**)
  - List loaded files
- Settings
  - Huge font
  - Download firmware

Sensor Types

- Digital:
  - Return a 0 or a 1
  - Switches or bumpers are an example
    (open: 0, or closed: 1)
- Analog:
  - Sensor returns a continuum of values
  - Processor digitizes results (8 bits give values of 0-255)
  - e.g. light sensors

Handy Board Sensors

- Knob
- Start and Stop Buttons
- 4 light sensors: analog
- 2 IR reflectance sensors: analog
- 2 Optical rangefinder: floating analog
- 6 assorted touch sensors: digital
- 2 Slotted encoder sensors: digital
- 1 Sonar rangefinder: (special ports)
- All detachable Handy Board sensors have keyed connector



Built-in Sensors: Knob

- Knob: the knob is a potentiometer which is treated like an analog sensor
  - The access function **knob( )** returns 0-255
  - Can be used for adjusting values during runtime

Built-in Sensors: Start

- Start Button:
  - Access function **`start_button()`** returns 1 while pressed, 0 otherwise
  - Holding down Start while powering the board will bypass executing main (e.g., if your program immediately causes the robot to move, and you want to download new code, this is very useful)

Built-in Sensors: Stop

- Stop Button:
  - Access function **`stop_button()`** returns 1 while pressed, 0 otherwise
  - Holding down Stop while powering the board will put the board in download mode. This is necessary for loading the firmware.

Light Sensors

- Analog sensor, four in kit
- Connect to ports 2-6 or 20-23
- Access with function **`analog(`***port#***`)`**
- Low values indicate bright light
- High values indicate low light
- Sensor is somewhat directional and can be made more so using black paper or tape or an opaque straw or lego to shade extraneous light. Sensor can be attenuated by placing paper in front.

IR Reflectance Sensor “Top Hat”

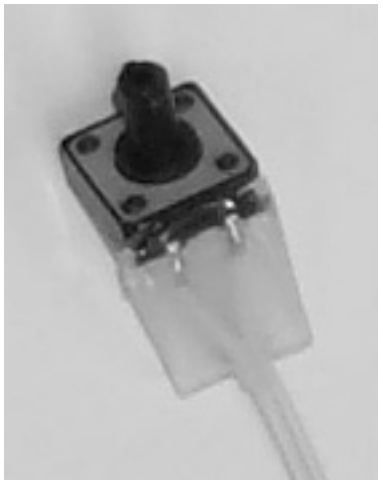
- Analog sensor, two in kit
- Connect to ports 2-6 or 20-23
- Access with function **`analog(`***port#***`)`**
- Low values indicate bright light, light color, or close proximity
- High values indicate low light, dark color, or distance of several inches
- Sensor has a reflectance range of about 3 inches

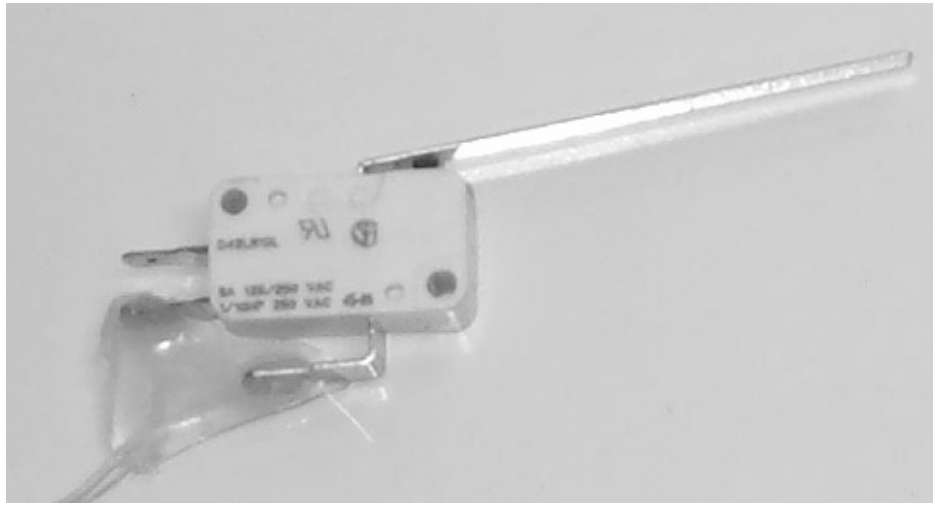
Optical Rangefinder “ET”

- Floating analog sensor, two in kit
- Connect to ports 16-19
- Access with function **`analog(`***port#***`)`**
- Low values indicate large distance
- High values indicate distance approaching ~4 inches
- Range is 4-30 inches. Result is approx $1/d^2$. Objects closer than 4 inches will appear to be far away.

Touch Sensors

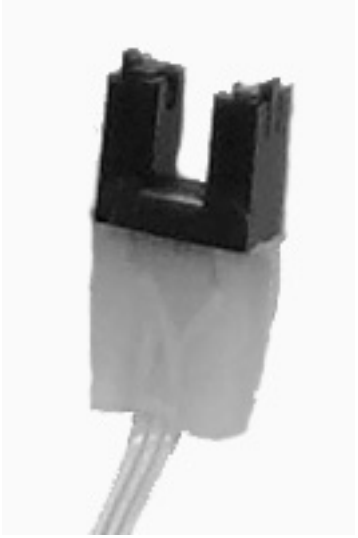
- Digital sensor
- Connect to ports 7-15
- Access with function **`digital(`***port#***`)`**
- Three types (2 of each)
- 1 indicates switch is closed
- 0 indicates switch is open
- These make good bumpers and can be used for limit switches on an actuator





Slot Sensors

- Digital sensor
- Connect to ports 7-15
- Access with function **`digital(`***port#***`)`**
- 1 indicates slot is empty
- 0 indicates slot is blocked
- These can be used much like touch sensors (if the object being touched fits in the slot)
- Special abilities when used as encoders



Using Encoders

- Use the slot sensor (Handy Board only)
- Connect to ports 7,8,12,13 (*encoder#* is 0,1,2,3)

  `enable_encoder(`*encoder#***`);`**
  - Only enable an encoder once...unless you disable it.

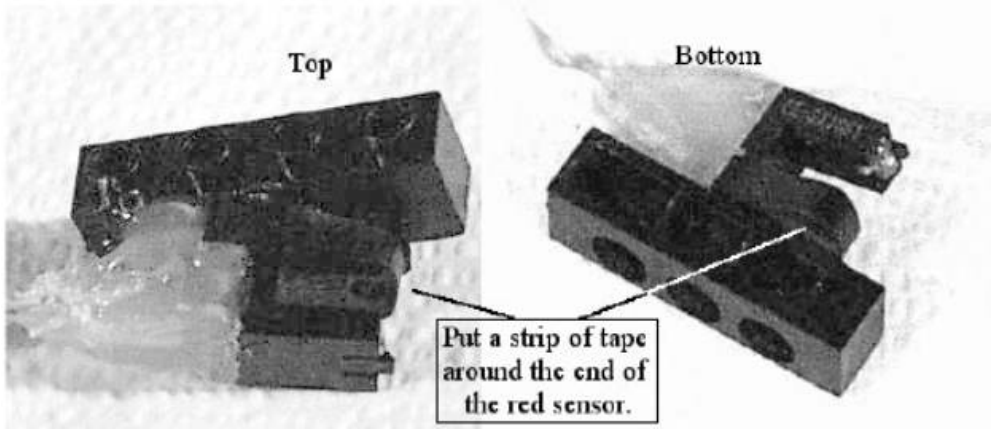
  `disable_encoder(`*encoder#***`);`**

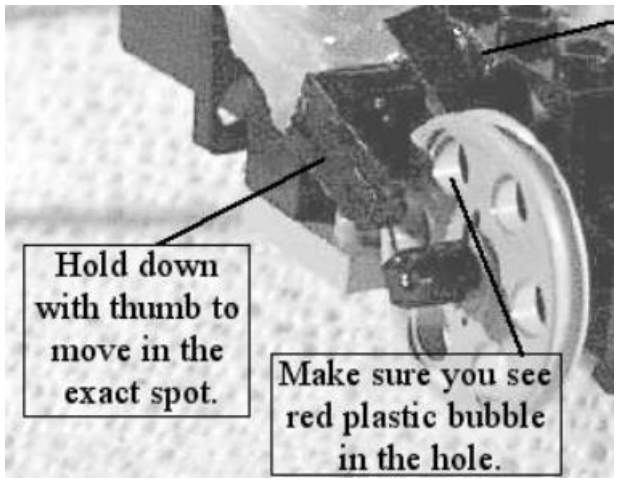
  `read_encoder(`*encoder#***`);`**
  - returns the number of transitions

  `reset_encoder(`*encoder#***`);`**
  - sets that encoder to 0
- Reflectance and slot sensors work best for encoders

Mounting a slot sensor encoder

Carefully align sensor with encoder wheel





Simple Encoder Program

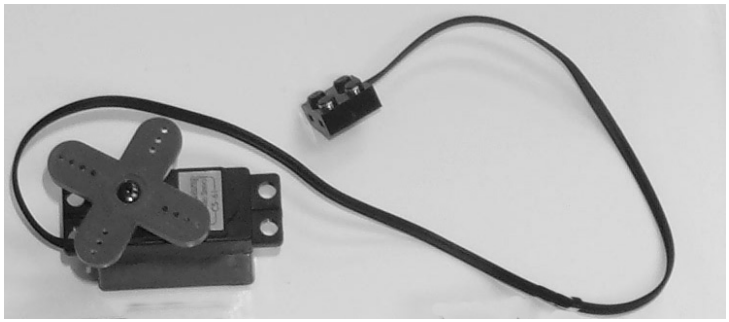
```
void main(){
   int enc1, enc0;
   enable_encoder(0); /* turn on the encoders */
   enable_encoder(1);
   while(!stop_button())
   {
      enc0=read_encoder(0);  /* read each encoder */
      enc1=read_encoder(1);  /* and show values   */
      printf("Enc0=%d Enc1=%d\n",enc0, enc1);
      sleep(0.1);  /* wait a bit and do it again */
   }
}
```

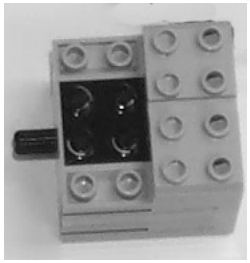
Last Word on Encoders

- Library encoder functions only work with Handy Board
  - You need to write your own if you want use an encoder on the RCX
- Every enabled encoder uses a lot of the HB's processor -- so don't enable an encoder unless you are going to use it, and *never put an enable statement inside of a loop*
- Just because you count an encoder does not mean that the robot moved that distance
  - tires slip on the ground (and tires slip on the wheels)
  - Lego's bend, gears skip, etc…

Handy Board DC Motors

- All motors are connected to HB using motor wire (4 in kit)
- 4 High speed gear motors
- 1 Micro-motor slow gear motor
- 1 High-power gear motor (modified servo)





Recharging Kit Batteries

- Handy Boards should be charged through the interface board
  - Interface board charging can be done in normal or Zap mode.
    - Normal charge lights yellow light on interface board when working, and does a trickle charge -- you can leave the board charging in this mode indefinitely. A full charge takes about 12 hours, though you can use the board earlier.
    - **ZAP fully charges in 2 hours, and will damage batteries after that. ONLY recommended while working with the robot with a dead battery or in tournament emergencies,**

Experimenting With Motors and Sensors

- Motors

```
fd(3);
bk(3);
off(3);
```

- Sensors

```
digital(7);
analog(6);
sonar();
```

Testing Your HB, Sensors & Motors

- Download **hbtest.ic**
- connect and test motors
- connect and test digital sensors
- connect and test analog sensors (be sure you have the right type of sensors in the right ports)

IC: Functions & Processes

- Functions are called sequentially
- Processes can be run simultaneously

  `start_process(`*function-call***`);`**

  – processes halt when function exits or parent process exits

  – processes can be halted by using

  `kill_process(`*process_id***`);`**
- **`hog_processor();`** locks process in CPU until it finishes or defers
- **`defer();`** causes process to give up the rest of its time slice until next time.

Example of Processes

```
void main()
{
   int pid;
   /* run my waa function   */
   /* as a parallel process */
   pid=start_process(waa());
   sleep(5.0);
   /* if waa is still going */
   /* turn waa off          */
   kill_process(pid);
   /* if tone got killed    */
   /* finish it off         */
   beeper_off();
}
```

```
void waa()
{
   float p;
   while(!stop_button())
   {  /* run tone up */
      /* over & over */
      p=200.0;
      while(p<5000.0)
      {
         tone(p,0.05);
         p=p+200.0;
      }
   }
}
```