# Handy Logo Summary

Notation: `command` performs an action  `expression` returns a 16-bit signed integer value

## Sensors

### Digital Sensors

`switch` *port* — current switch value, 0 (off) or 1 (on)
*0 -- 15*

Analog sensors may be used in `switch`:
0 -- 127 --> 1 (on); 128 -- 255 --> 0 (off)

### Analog Sensors

`sensor` *port* — current sensor value, 0 (high) -- 255 (low)
*0 -- 6*

Digital sensors may be used in `sensor`:
0 (off) --> number in range 128-255;
1 (on) --> number in range 0-127

### Timer

`timer` — current timer value (-32768 -- 32767)

`resett` — resets timer to 0

### Infrared Receiver

`ir` — IR value (0 -- 255) sent by another Handy Board or infrared remote.

## Numeric Operations

*int* { + - * / \ } *int* — usual arithmetic ops
\ is remainder

*int* { < = > } *int* — usual relational ops
result is 1 (true) or 0 (false)

{and or} *int* *int* — bitwise and/or

`not` *int* — bitwise negation

`random` *int* — psuedo-random integer between 0 and (int - 1)

## Controller

### Sequential Control

*command1* *command2* — command sequencing

`wait` *tenths* — grabs control for *tenths* tenths of seconds

`repeat` *times* `[` *action* `]` — repeat *action* *times* times

`loop [` *action* `]` — repeat *action* forever

`waituntil [` *test* `]` — evaluate *test* until it is true

`if` *test* `[` *then* `]` — perform *then* if *test* is non-zero

`ifelse` *test* `[` *then* `][` *else* `]` — perform *then* if *test* is non-zero or *else* if *test* is zero

`output` *result* — return *result* from procedure

`stop` — exit procedure

### Concurrency

`launch [` *action* `]` — perform *action* in a new control thread

`forever [` *action* `]` — perform *action* continuously in a new control thread

`every` *tenths* `[` *action* `]` — perform *action* every *tenths* tenths of a second

`when [` *test* `][` *action* `]` — perform *action* whenever *test* changes from false to true

`stoprules` — stop all threads in current family except for thread executing stoprules. New families are started by launch and START button; forever, every, and when extend current family.

## Actuators

### Beeper

`beep` — short beep

`note` *freq* *tenths* — play note at *freq* for *tenths* tenths of a second
*40 -- 120*

### Motors

*Selection:* select motor port for subsequent motor commands

`a,` `b,` `c,` `d,` `ab,` `bc,` `ad,` `abcd,`

*Power:*

`on` `off` `toggle` (on/off state) `onfor` *tenths* (grabs control)

*Direction:*

`thisway` (green LED) `thatway` (red LED) `rd` (reverse direction)

*Power Level:*

`setpower` *level* — level ranges from 0 (none) -- 8 (full)

### LCD Display

print erases, then adds text; type just adds text

`print` *int* `print "`*word* `print [`*word1 ...wordn* `]`

`type` *int* `type "`*word* `type` *word1 ...wordn* `]`

`top` — selects top line of LCD display

`bottom` — selects bottom line of LCD display

### Serial Line

say sends text over serial line for display on computer console

`say` *int* `say "`*word* `say` *word1 ...wordn* `]`

### Infrared Transmitter

`irsend` *value* — send *value* via infrared transmitter

Handy Logo was developed by Brian Silverman (Logo Computer Systems Inc.) and Fred Martin (Epistemology and Learning Group, MIT Media Lab).
As of this writing, Handy Logo is proprietary software, not for general distribution. Handy Boards may also be programmed via Interactive C (IC), which is freely distributed.
This summary was created by Franklyn Turbak (Wellesley College).