# Kinetrix V1.0 - Complete Syntax Guide

## Table of Contents



---

## Introduction

Kinetrix is a high-level, compiled robotics programming language designed specifically for Raspberry Pi and Arduino-compatible platforms. It abstracts hardware complexity while maintaining direct control over GPIO pins, sensors, motors, and communication protocols.

### Language Philosophy

- **Readability**: Natural English-like syntax
- **Simplicity**: Minimal boilerplate code
- **Hardware-focused**: Built-in support for robotics peripherals
- **Compiled**: Translates to optimized C/Arduino code

### Compilation

```
kinetrix robot.kx
```

This generates `Kinetrix_Arduino.ino` which can be uploaded to Arduino-compatible boards.

---

## Basic Program Structure

Every Kinetrix program begins with a `program` block:

```
program robot_name {
    # Your code here
}
```

### Minimal Program Example

```
program hello_world {
    print "Hello from Kinetrix!"
}
```

---

## Data Types and Variables

Kinetrix uses dynamic typing with a single unified numeric type (float).

### Variable Declaration

**Syntax:**

```
make var variable_name = value
```

**Examples:**

```
make var speed = 100
make var sensor_value = 0
make var distance = 45.5
make var counter = 0
```

### Variable Assignment

After declaration, update variables using their name:

```
speed = 200
```

### Variable Modification

```
change variable_name by amount
```

**Example:**

```
make var count = 0
change count by 1      # Increments count by 1
change count by -5     # Decrements count by 5
```

---

## Control Structures

### 1. Conditional Statements (if)

**Basic Syntax:**

```
if condition {
    # code block
}
```

**With Else:**

```
if condition {
    # code when true
} else {
    # code when false
}
```

**Pin State Check:**

```
if pin 7 is high {
    print "Button pressed"
}
```

**Comparison Examples:**

```
if speed > 100 {
    print "Too fast!"
}

if temperature < 20 {
    print "Too cold"
}

if distance = 50 {
    print "Target reached"
}
```

**Logical Operators:**

```
if speed > 50 and distance < 100 {
    print "Approaching target"
}

if sensor1 > 500 or sensor2 > 500 {
    print "High reading detected"
}
```

### 2. Forever Loop

Runs indefinitely:

```
forever {
    turn on 13
    wait 1000
    turn off 13
```

```
    wait 1000
}
```

### 3. Repeat Loop

Execute a fixed number of times:

**Syntax:**

```
repeat count {
    # code block
}
```

**Examples:**

```
repeat 10 {
    print "Iteration"
    wait 100
}

repeat 5 {
    turn on 13
    wait 200
    turn off 13
    wait 200
}
```

### 4. While Loop

Execute while condition is true:

**Syntax:**

```
while condition {
    # code block
}
```

**Example:**

```
make var counter = 0
while counter < 10 {
    print counter
    change counter by 1
    wait 100
}
```

### 5. Break Statement

Exit a loop early:

```
make var i = 0
forever {
    if i > 100 {
        break
    }
    change i by 1
}
```

---

## Functions

### Function Definition

#### Without Parameters:

```
def function_name {
    # code block
}
```

#### With Parameters:

```
def function_name(parameter) {
    # code block
}
```

### Calling Functions

#### Without Parameters:

```
function_name
```

#### With Parameters:

```
function_name(value)
```

### Return Statement

```
def calculate_area(radius) {
    make var area = 3.14 * radius * radius
    return area
}
```

### Complete Function Examples

```
# Function without parameters
def blink_led {
    turn on 13
    wait 500
    turn off 13
    wait 500
```

```
}

# Function with parameter
def set_motor_speed(speed) {
    set pin 3 to speed
    set pin 5 to speed
}

# Function with calculation and return
def convert_to_celsius(fahrenheit) {
    make var celsius = (fahrenheit - 32) * 5 / 9
    return celsius
}

# Using functions
program main {
    blink_led
    set_motor_speed(150)
    make var temp = convert_to_celsius(98.6)
    print temp
}
```

---

## GPIO Operations

### Digital Output

**Turn Pin On (HIGH):**

```
turn on pin_number
```

**Turn Pin Off (LOW):**

```
turn off pin_number
```

**Examples:**

```
turn on 13        # LED on
turn off 13       # LED off
turn on 7         # Any digital pin
```

### Analog Output (PWM)

**Syntax:**

```
set pin pin_number to value
```

Where value ranges from 0 (0V) to 255 (5V)

**Examples:**

```
set pin 3 to 128      # 50% duty cycle
set pin 5 to 255      # Full power
set pin 6 to 0        # Off
```

### Servo Control

**Syntax:**

```
servo pin pin_number set angle
```

Where angle ranges from 0 to 180 degrees

**Examples:**

```
servo pin 9 set 0       # Minimum position
servo pin 9 set 90      # Center position
servo pin 9 set 180     # Maximum position
```

---

## Sensor Input

### Digital Input

**Pin State Check:**

```
if pin pin_number is high {
    # Pin is HIGH (5V)
}

if pin pin_number is low {
    # Pin is LOW (0V)
}
```

**Example:**

```
if pin 2 is high {
    print "Button pressed"
    turn on 13
}
```

### Analog Input

**Syntax:**

```
make var variable_name = read analog pin_number
```

Returns value from 0 to 1023

**Example:**

```
make var light_level = read analog 0
print light_level
```

```
if light_level < 300 {
    print "It's dark"
    turn on 13
}
```

### Pulse Input (Ultrasonic Sensors)

**Syntax:**

```
make var duration = read pulse pin pin_number
```

Returns pulse duration in microseconds

**Distance Calculation Example:**

```
def measure_distance {
    make var duration = read pulse pin 7
    make var distance = duration / 58    # Convert to centimeters
    print "Distance (cm):"
    print distance
    return distance
}
```

### Serial Input

**Syntax:**

```
make var value = read serial
```

Reads integer from serial port

**Example:**

```
program serial_control {
    forever {
        make var command = read serial
        if command = 1 {
            turn on 13
        }
        if command = 0 {
            turn off 13
        }
    }
}
```

---

## Motor Control

### PWM Motor Speed Control

```
# Set left motor speed
set pin 3 to 200

# Set right motor speed
set pin 5 to 200

# Stop motors
set pin 3 to 0
set pin 5 to 0
```

### Direction Control with H-Bridge

```
def forward {
    turn on 2      # Left motor forward
    turn off 3     # Left motor backward OFF
    turn on 4      # Right motor forward
    turn off 5     # Right motor backward OFF
}

def backward {
    turn off 2
    turn on 3
    turn off 4
    turn on 5
}

def turn_left {
    turn off 2     # Left motor off
    turn on 4      # Right motor on
}

def turn_right {
    turn on 2      # Left motor on
    turn off 4     # Right motor off
}

def stop {
    turn off 2
    turn off 3
    turn off 4
    turn off 5
}
```

## Audio Output

### Tone Generation

**Syntax:**

```
tone pin pin_number freq frequency
```

**Stop Tone:**

```
notone pin pin_number
```

**Examples:**

```
# Play middle C (261 Hz)
tone pin 8 freq 261
wait 1000
notone pin 8

# Play melody
def play_melody {
    tone pin 8 freq 261    # C
    wait 500
    tone pin 8 freq 294    # D
    wait 500
    tone pin 8 freq 329    # E
    wait 500
    notone pin 8
}
```

### Musical Notes Reference

| Note | Frequency (Hz) |
|---|---|
| C | 261 |
| D | 294 |
| E | 329 |
| F | 349 |
| G | 392 |
| A | 440 |
| B | 494 |

## Arrays

### Array Declaration

**Syntax:**

```
make array array_name size length
```

**Example:**

```
make array readings size 10
make array sensor_data size 5
```

### Array Access and Modification

**Set Array Element:**

```
set array array_name at index to value
```

**Get Array Element:**

```
make var value = array_name[index]
```

**Complete Example:**

```
program array_demo {
    # Create array of 5 elements
    make array temperatures size 5

    # Store values
    set array temperatures at 0 to 23.5
    set array temperatures at 1 to 24.1
    set array temperatures at 2 to 22.8
    set array temperatures at 3 to 25.0
    set array temperatures at 4 to 23.2

    # Read and process
    make var i = 0
    repeat 5 {
        make var temp = temperatures[i]
        print temp
        change i by 1
        wait 100
    }
}
```

---

## I2C Communication

Kinetrix provides built-in support for I2C protocol, essential for communicating with sensors like gyroscopes, accelerometers, and OLED displays.

### I2C Bus Initialization

```
i2c begin
```

### Start Communication with Device

**Syntax:**

```
i2c start device_address
```

**Example:**

```
i2c start 0x68    # MPU6050 gyroscope address
```

### Send Data to Device

**Syntax:**

```
i2c send data_byte
```

**Example:**

```
i2c send 0x6B     # Send register address
```

### Stop Transmission

```
i2c stop
```

### Read Data from Device

**Syntax:**

```
make var data = read i2c device_address
```

**Example:**

```
make var accel_data = read i2c 0x68
```

### Complete I2C Example - MPU6050 Gyroscope

```
def init_mpu6050 {
    i2c begin
    i2c start 0x68
    i2c send 0x6B       # PWR_MGMT_1 register
    i2c send 0          # Wake up MPU6050
    i2c stop
    print "MPU6050 initialized"
```

```
}

def read_accel_x {
    i2c start 0x68
    i2c send 0x3B       # ACCEL_XOUT_H register
    i2c stop

    make var data = read i2c 0x68
    return data
}

program drone_control {
    init_mpu6050

    forever {
        make var accel = read_accel_x
        print "Accel X:"
        print accel
        wait 100
    }
}
```

---

## Mathematical Operations

### Basic Arithmetic

```
make var sum = 10 + 5          # Addition
make var diff = 10 - 5         # Subtraction
make var product = 10 * 5      # Multiplication
make var quotient = 10 / 5     # Division
```

### Comparison Operators

```
if x > 10 { }     # Greater than
if x < 10 { }     # Less than
if x = 10 { }     # Equal to
```

### Logical Operators

```
if x > 5 and y < 10 { }    # AND
if x > 5 or y < 10 { }     # OR
```

### Trigonometric Functions

```
make var sine_val = sin(angle)
make var cosine_val = cos(angle)
```

```
make var tangent_val = tan(angle)
```

### Inverse Trigonometric Functions

```
make var acos_val = acos(value)
make var asin_val = asin(value)
make var atan_val = atan(value)
make var atan2_val = atan2(y, x)    # Two-argument arctangent
```

### Square Root

```
make var root = sqrt(value)
```

### Complete Math Example

```
def calculate_distance(x, y) {
    # Calculate Euclidean distance
    make var x_squared = x * x
    make var y_squared = y * y
    make var sum = x_squared + y_squared
    make var distance = sqrt(sum)
    return distance
}

def calculate_angle(opposite, adjacent) {
    make var ratio = opposite / adjacent
    make var angle = atan(ratio)
    return angle
}
```

---

## Comments

Single-line comments begin with **#**:

```
# This is a comment
turn on 13    # LED control

# Function to blink LED
def blink {
    turn on 13
    wait 500
}
```

---

## Complete Examples

### Example 1: Blinking LED

```
program blink_led {
    forever {
        turn on 13
        wait 1000
        turn off 13
        wait 1000
    }
}
```

### Example 2: Button-Controlled LED

```
program button_led {
    forever {
        if pin 2 is high {
            turn on 13
        } else {
            turn off 13
        }
        wait 50
    }
}
```

### Example 3: Light-Sensitive Night Light

```
program night_light {
    forever {
        make var light = read analog 0

        if light < 300 {
            turn on 13
            print "Dark - LED ON"
        } else {
            turn off 13
            print "Bright - LED OFF"
        }

        wait 500
    }
}
```

### Example 4: Obstacle Avoidance Robot

```
def measure_distance {
    make var duration = read pulse pin 7
```

```
    make var distance = duration / 58
    return distance
}

def forward {
    set pin 3 to 200
    set pin 5 to 200
}

def backward {
    set pin 3 to 150
    set pin 5 to 150
    wait 500
    set pin 3 to 0
    set pin 5 to 0
}

def turn_right {
    set pin 3 to 200
    set pin 5 to 0
    wait 400
}

def stop_motors {
    set pin 3 to 0
    set pin 5 to 0
}

program obstacle_avoidance {
    forever {
        make var dist = measure_distance
        print "Distance:"
        print dist

        if dist < 20 {
            stop_motors
            backward
            turn_right
        } else {
            forward
        }

        wait 100
    }
}
```

**Example 5: Temperature Monitor with Alert**

```
def play_alert {
    tone pin 8 freq 1000
    wait 200
    notone pin 8
    wait 100
    tone pin 8 freq 1000
    wait 200
    notone pin 8
}

program temp_monitor {
    make var threshold = 500

    forever {
        make var temp = read analog 0
        print "Temperature:"
        print temp

        if temp > threshold {
            print "WARNING: High temperature!"
            play_alert
            turn on 13
        } else {
            turn off 13
        }

        wait 1000
    }
}
```

**Example 6: Line Following Robot**

```
program line_follower {
    forever {
        make var left_sensor = read analog 0
        make var right_sensor = read analog 1

        # Both sensors on white (low values) - go straight
        if left_sensor < 500 and right_sensor < 500 {
            set pin 3 to 200
            set pin 5 to 200
        }

        # Left sensor on black - turn left
```

```
        if left_sensor > 500 {
            set pin 3 to 100
            set pin 5 to 200
        }

        # Right sensor on black - turn right
        if right_sensor > 500 {
            set pin 3 to 200
            set pin 5 to 100
        }

        wait 50
    }
}
```

**Example 7: Servo Sweep**

```
program servo_sweep {
    forever {
        # Sweep from 0 to 180
        make var angle = 0
        while angle < 180 {
            servo pin 9 set angle
            change angle by 5
            wait 50
        }

        # Sweep from 180 to 0
        while angle > 0 {
            servo pin 9 set angle
            change angle by -5
            wait 50
        }
    }
}
```

**Example 8: Drone IMU Reader (I2C)**

```
def init_mpu6050 {
    i2c begin
    i2c start 0x68
    i2c send 0x6B
    i2c send 0
    i2c stop
}
```

```
def read_gyro_data {
    i2c start 0x68
    i2c send 0x43       # GYRO_XOUT_H
    i2c stop

    make var gyro_x = read i2c 0x68
    return gyro_x
}

program drone_stabilization {
    init_mpu6050
    print "MPU6050 Ready"

    forever {
        make var rotation = read_gyro_data
        print "Gyro X:"
        print rotation

        # Adjust motor speeds based on gyro
        if rotation > 100 {
            set pin 3 to 180    # Reduce left
            set pin 5 to 220    # Increase right
        } else {
            if rotation < -100 {
                set pin 3 to 220
                set pin 5 to 180
            } else {
                set pin 3 to 200
                set pin 5 to 200
            }
        }

        wait 20
    }
}
```

---

## Language Reserved Keywords

```
program, def, make, var, set, change, by, to
turn, on, off, wait, forever, if, else
repeat, while, break, return
read, analog, serial, pulse
servo, tone, notone, freq, print
pin, is, high, low
```

```
array, size, index, of
i2c, begin, start, send, stop
and, or
sin, cos, tan, sqrt, asin, acos, atan, atan2
```

---

## Pin Assignments (Raspberry Pi/Arduino)

### Digital Pins: 2-13

- Pins 2-13 can be used for digital I/O
- Pin 13 typically has an onboard LED

### Analog Pins: 0-5

- Read analog values (0-1023)
- Used for sensors (temperature, light, etc.)

### PWM Capable Pins: 3, 5, 6, 9, 10, 11

- Support analog output (0-255)
- Used for motor speed control, LED brightness

### Common Pin Usage

- **Pin 13**: Built-in LED
- **Pins 3, 5**: Motor control
- **Pin 7**: Ultrasonic sensor trigger/echo
- **Pin 8**: Buzzer/speaker
- **Pin 9**: Servo motor
- **Analog 0-1**: Sensor inputs

---

## Compilation and Deployment

1. Write your Kinetrix program (`.kx` file)
2. Compile: `kinetrix robot.kx`
3. Generated output: `Kinetrix_Arduino.ino`
4. Upload to Arduino using Arduino IDE or CLI

---

## Best Practices

1. **Use descriptive variable names**

```
make var motor_speed = 200      # Good
make var s = 200                # Bad
```

2. **Comment your code**

```
# Initialize sensors before main loop
make var sensor_threshold = 500
```

3. **Create reusable functions**

```
def blink_times(count) {
    repeat count {
        turn on 13
        wait 200
        turn off 13
        wait 200
    }
}
```

4. **Use appropriate delays**

```
wait 50     # Responsive
wait 5000   # May cause lag
```

5. **Check sensor ranges**

```
make var reading = read analog 0
if reading > 0 and reading < 1024 {
    # Valid range
}
```

---

## Troubleshooting

### Common Issues

**Program doesn't start:** - Ensure `program` block exists - Check for syntax errors (missing braces)

**Unexpected behavior:** - Verify pin numbers match hardware - Check sensor wiring - Add debug `print` statements

**Compilation errors:** - Ensure keywords are lowercase - Check for unclosed braces `{}` - Verify function names match calls

---

## Additional Resources

- **Compiler Source**: `compiler.c`
- **Example Libraries**: `drive.kx`, `sonar.kx`, `music.kx`, `math.kx`
- **Pin Reference**: Arduino/Raspberry Pi documentation

---

**Kinetrix V1.0** - A compiled robotics language for makers and educators.