Tiny Lisp Interpreter
Embedded Computing Annex

Brent Seidel
Phoenix, AZ

April 26, 2021

This document is ©2021 Brent Seidel. All rights reserved.

Note that this is a draft version and not the final version for publication.

Contents

1 Introduction **1**

2 Common **2**
2.1 Template . . . 2
2.1.1 Inputs . . . 2
2.1.2 Output . . . 2
2.1.3 Example . . . 2

3 Arduino Due **3**
3.1 analog-read . . . 3
3.1.1 Inputs . . . 3
3.1.2 Output . . . 3
3.1.3 Example . . . 3
3.2 bme280-read . . . 3
3.2.1 Inputs . . . 4
3.2.2 Output . . . 4
3.2.3 Example . . . 4
3.3 bme280-read-raw . . . 4
3.3.1 Inputs . . . 4
3.3.2 Output . . . 4
3.3.3 Example . . . 4
3.4 bmp180-read . . . 4
3.4.1 Inputs . . . 4
3.4.2 Output . . . 4
3.4.3 Example . . . 5
3.5 l3gd20-read . . . 5
3.5.1 Inputs . . . 5
3.5.2 Output . . . 5
3.5.3 Example . . . 5
3.6 MCP23017 General Information . . . 5
3.7 mcp23017-dir . . . 5
3.7.1 Inputs . . . 5
3.7.2 Output . . . 5
3.7.3 Example . . . 5
3.8 mcp23017-polarity . . . 5

3.8.1 Inputs . . . 6
3.8.2 Output . . . 6
3.8.3 Example . . . 6
3.9 mcp23017-pullup . . . 6
3.9.1 Inputs . . . 6
3.9.2 Output . . . 6
3.9.3 Example . . . 6
3.10 mcp23017-read . . . 6
3.10.1 Inputs . . . 6
3.10.2 Output . . . 6
3.10.3 Example . . . 7
3.11 mcp23017-set . . . 7
3.11.1 Inputs . . . 7
3.11.2 Output . . . 7
3.11.3 Example . . . 7
3.12 GPIO/Pin General Information . . . 7
3.13 pin-mode . . . 7
3.13.1 Inputs . . . 7
3.13.2 Output . . . 7
3.13.3 Example . . . 8
3.14 pin-pullup . . . 8
3.14.1 Inputs . . . 8
3.14.2 Output . . . 8
3.14.3 Example . . . 8
3.15 pin-read . . . 8
3.15.1 Inputs . . . 8
3.15.2 Output . . . 8
3.15.3 Example . . . 8
3.16 pin-set . . . 8
3.16.1 Inputs . . . 9
3.16.2 Output . . . 9
3.16.3 Example . . . 9
3.17 pca9685-set . . . 9
3.17.1 Inputs . . . 9
3.17.2 Output . . . 9
3.17.3 Example . . . 9
3.18 Stepper Control General Information . . . 9
3.19 stepper-delay . . . 9
3.19.1 Inputs . . . 9
3.19.2 Output . . . 9
3.19.3 Example . . . 10
3.20 stepper-init . . . 10
3.20.1 Inputs . . . 10
3.20.2 Output . . . 10
3.20.3 Example . . . 10
3.21 stepper-off . . . 10

3.21.1 Inputs 10
3.21.2 Output 10
3.21.3 Example 10
3.22 stepper-step 10
3.22.1 Inputs 10
3.22.2 Output 11
3.22.3 Example 11

4 Raspberry PI **12**

Chapter 1

Introduction

This document describes the extensions to the Tiny-Lisp language for embedded applications. For documentation on the core language, see `https://github.com/BrentSeidel/Ada-Lisp/tree/master/documentation`. Due to the diversity of embedded platforms, operations that work on one may not work on another. Each of the following chapters describes a specific platform, with a few operations that are common to all, or most all platforms.

None of these items are expected to be found in Common Lisp, so any program using them will not be compatible.

Chapter 2

Common

This chapter contains any operations that are common to more than one platform. Currently, this section is empty.

2.1 Template

2.1.1 Inputs

2.1.2 Output

2.1.3 Example

()

Chapter 3

Arduino Due

This chapter contains operations for the Arduino Due. To use these, include the `ada_lisp_embedded_due.gpr` project in your project. After calling `BBS.lisp.init()`, make the following calls:

```
BBS.lisp.embed.init;
BBS.lisp.embed.init_discretes;
```

If you are going to use any of the operations relating to I2C devices, the related `*_found` and `*_info` variables need to be set appropriately.

3.1 analog-read

Reads and returns a value from an analog input pin.

3.1.1 Inputs

One integer representing the analog input number.

3.1.2 Output

The value of the specified analog input.

3.1.3 Example

```
(analog-read 2)
```

returns the value of analog input number 2.

3.2 bme280-read

Reads the sensors and returns a list of three items containing the temperature (°C), pressure (Pa), and humidity (%), in that order.

3.2.1 Inputs

None.

3.2.2 Output

A list of three integers containing the temperature (°C), pressure (Pa), and humidity (%), in that order

3.2.3 Example

```
(bme280-read)
```

3.3 bme280-read-raw

Reads the sensors and returns a list of three items containing the raw values for temperature, pressure, and humidity, in that order.

3.3.1 Inputs

None.

3.3.2 Output

A list of three integers containing the raw values for temperature, pressure, and humidity, in that order.

3.3.3 Example

```
(bme280-read-raw)
```

3.4 bmp180-read

Reads the sensors and returns a list of two items containing the temperature in (°C) and atmospheric pressure (Pa) from the BMP180 sensor.

3.4.1 Inputs

None.

3.4.2 Output

A list of two integers containing the temperature in (°C) and atmospheric pressure (Pa), in that order.

3.4.3 Example

```
(bmp180-read)
```

3.5 l3gd20-read

Reads the gyroscopic sensor on a L3GD20 device and returns the values.

3.5.1 Inputs

None.

3.5.2 Output

A list of three integers containing the rotations around the x, y, and z axises in values of degrees per second (°/S).

3.5.3 Example

```
(l3gd20-read)
```

3.6 MCP23017 General Information

Note that for the MCP23017 devices, currently only addresses 0, 2, and 6 are valid. At some point, this will be extended to allow the full range of 0-7.

3.7 mcp23017-dir

Set direction of bits in the MCP23017 port.

3.7.1 Inputs

Two integers. The first is the address of the MCP23017 device, the second is a 16 bit number where each bit represents the direction of that bit in the port (0-read, 1-write).

3.7.2 Output

Returns the direction value.

3.7.3 Example

```
(mcp23017-dir 2 #x00FF)
```

3.8 mcp23017-polarity

Set polarity of bits in the MCP23017 port.

3.8.1 Inputs

Two integers. The first is the address of the MCP23017 device, the second is a 16 bit number where each bit represents the polarity of that bit in the port (0-normal, 1-inverted).

3.8.2 Output

Returns the polarity value.

3.8.3 Example

```
(mcp23017-pullup 2 #x00FF)
```

3.9 mcp23017-pullup

Enable/disable pull-up resistors for bits in the MCP23017 port.

3.9.1 Inputs

Two integers. The first is the address of the MCP23017 device, the second is a 16 bit number where each bit represents the state of the pull-up resistor of that bit in the port (0-disable, 1-enable).

3.9.2 Output

Returns the pull-up value.

3.9.3 Example

```
(mcp23017-pullup 2 #x00FF)
```

3.10 mcp23017-read

Read data from a MCP23017 port

3.10.1 Inputs

One integer representing the address (of the MCP23017 device.

3.10.2 Output

A 16 bit integer representing the data read from the MCP23017 device.

3.10.3 Example

```
(mcp23017-read 2)
```

3.11 mcp23017-set

Set output data of bits in the MCP23017 port.

3.11.1 Inputs

Two integers. The first is the address of the MCP23017 device, the second is a 16 bit number where each bit represents the value of that bit in the port (0-low, 1-high).

3.11.2 Output

Returns the port data value

3.11.3 Example

```
(mcp23017-set 2 #x00FF)
```

3.12 GPIO/Pin General Information

The valid GPIO pins are numbered from 0-53. Due to the way the pins are wired on the Arduino Due, pin number 4 is unusable.

3.13 pin-mode

Sets a pin's mode to be either input or output. The first integer is the pin number and the second integer is the mode to set for the pin. The pin number is range checked as above. Mode 0 sets the pin to input mode while any other value sets the pin to output mode. Returns NIL. This should be used before trying to read or set a pin.

3.13.1 Inputs

Two integers. The first is the pin number. The second is the pin mode (0 for input, 1 for ouptut).

3.13.2 Output

NIL

3.13.3 Example

```
(pin-mode 3 1)
```

Sets pin 3 to output mode.

3.14 pin-pullup

Enable or disable the pullup resistor of a digital pin. Two parameters are read. The first parameter is the pin number. The second is the mode (NIL is disable, T is enable).

3.14.1 Inputs

Two integers. The first is the pin number. The second is a boolean (NIL for disable, T for enable).

3.14.2 Output

NIL

3.14.3 Example

```
(pin-pullup 3 1)
```

3.15 pin-read

Reads the state of a pin. It is passed one integer parameter representing the pin number and returns the state of the pin as an integer (0 for low, 1 for high).

3.15.1 Inputs

One integer representing the pin number.

3.15.2 Output

An integer with 0 being a low value on the pin and 1 being a high value.

3.15.3 Example

```
(read-read 3)
```

Returns the state of pin 3.

3.16 pin-set

Set the state of a digital pin Two parameters are read. The first parameter is the pin number. The second is the state (0 is low, 1 is high).

3.16.1 Inputs

Two integers. The first is the pin number. The second is the state (0 for low, 1 for high).

3.16.2 Output

NIL.

3.16.3 Example

```
(pin-set 3 1)
```

Sets pin 3 to a high state.

3.17 pca9685-set

Sets the PWM value for a specific channel on the PCA9685 PWM controller.

3.17.1 Inputs

Two integers. The first integer is the channel number (0-15). The second integer is the PWM value to set (0-4095).

3.17.2 Output

NIL.

3.17.3 Example

```
(pca9685-set 7 2047)
```

3.18 Stepper Control General Information

Currently four stepper motor control channels are available (0-3). Each channel requires the use of four GPIO pins.

3.19 stepper-delay

Sets the delay between steps for the specified stepper controller. The default delay is 5mS.

3.19.1 Inputs

Two integers. The first is the stepper controller number. The second is the number of mS to delay between steps.

3.19.2 Output

NIL

3.19.3 Example

```
(stepper-delay 1 10)
```

3.20 stepper-init

Initializes a stepper channel. Assigns the for GPIO pins and configures them. Sets the internal stepper phase to 1 and sets the pins appropriately.

3.20.1 Inputs

Five integers. The first is the stepper channel number. The remaining four are the four GPIO pins to use to control the stepper.

3.20.2 Output

NIL

3.20.3 Example

```
(stepper-init 1 10 11 12 13)
```

3.21 stepper-off

De-energizes the specific stepper channel. This may be done to allow the motor to be manually positioned or to reduce power consumption if holding the motor in place is no longer needed.

3.21.1 Inputs

One integer representing the stepper channel.

3.21.2 Output

NIL

3.21.3 Example

```
(stepper-off 1)
```

3.22 stepper-step

Moves the specified stepper motor the given number of steps. Positive and negative step numbers will rotate in opposite directions. The actual direction will depend on how the motor is wired.

3.22.1 Inputs

Two integers. The first represents the stepper channel. The second is the number of steps to take as a signed value.

3.22.2 Output

NIL

3.22.3 Example

```
(stepper-step 1 -100)
```

Chapter 4

Raspberry PI

No items yet.