Propeller Library

by George Antrobus

April 22, 2012

Table of Contents

2 Pin Functions 3

2.1 pinInput 3

2.2 pinOutput 3

2.3 pinGetDirection 3

2.4 pinSetDirection 4

2.5 pinReverseDirection 4

2.6 pinGet 5

2.7 pinSet 5

2.8 pinGetField 6

2.9 pinSetField 6

2.10 pinHigh 6

2.11 pinLow 7

2.12 pinToggle 7

2.13 pinPulseIn 8

2.14 pinPulseOut 8

3 I2C Functions 10

3.1 i2cInit 10

3.2 i2cTerm 10

3.3 i2cSendBuf 11

3.4 i2cBegin 11

3.5 i2cAddByte 12

3.6 i2cSend 12

3.7 i2cRequestBuf 13

3.8 i2cRequest 13

3.9 i2cGetByte 14

4 Serial Terminal Functions 15

5 VGA Terminal Functions 15

6 TV Terminal Functions 15

7 Terminal Functions 15

# Pin Functions

To use the pin functions, include the <propeller/pin.h” header file.

## pinInput

Prototype

void pinInput(int pin);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | pin | Pin number |

Description

Sets a pin to an input. Pins are inputs by default but if a pin has been used as an output and you then want to use it as an input you should call this function first.

Returns

Nothing

## pinOutput

Prototype

void pinOutput(int pin);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | pin | Pin number |

Description

Sets a pin to an output. Since pins are inputs by default, this function must be called before you use it as an output.

Returns

Nothing

## pinGetDirection

Prototype

int pinGetDirection(int pin);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | pin | Pin number |

Description

Gets the direction of a pin. Returns 0 for input pins and 1 for output pins.

Returns

The pin direction.

## pinSetDirection

Prototype

void pinSetDirection(int pin, int direction);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | pin | Pin number |
| [in] | direction | Pin direction |

Description

Sets the direction of a pin. Set direction to 0 for input pins and 1 for output pins. Since pins are inputs by default, this function or pinOutput must be called to set a pin to be an output before you use it as an output.

Returns

The pin direction.

## pinReverseDirection

Prototype

void pinReverseDirection(int pin);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | pin | Pin number |

Description

Reverses the direction of a pin. If the pin was an output, it becomes an input. If it was an input, it becomes an output.

Returns

Nothing

## pinGet

Prototype

int pinGet(int pin);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | pin | Pin number |

Description

Gets the value of an input pin.

Returns

The pin value.

## pinSet

Prototype

void pinSet(int pin, int value);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | pin | Pin number |
| [in] | value | The new pin value |

Description

Sets the value of an output pin. Since pins are inputs by default, the pin should be set to an output using either pinOutput or pinSetDir prior to calling this function.

Returns

Nothing

## pinGetField

Prototype

int pinGetField(int high, int low);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | high | High pin number |
| [in] | low | Low pin number |

Description

Get the value of a group of pins starting with the high pin and ending with the low pin. For convenience, the high and low pin numbers can be swapped. This function allows a group of contiguous pins to be treated as a multi-bit field.

Returns

The value of the pins in the field.

## pinSetField

Prototype

void pinSetField(int high, int low, int value);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | high | High pin number |
| [in] | low | Low pin number |
| [in] | value | Value to write to the field |

Description

Set the value of a group of pins starting with the high pin and ending with the low pin. For convenience, the high and low pin numbers can be swapped. This function allows a group of contiguous pins to be treated as a multi-bit field.

Returns

Nothing

## pinHigh

Prototype

void pinHigh(int pin);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | pin | Pin number |

Description

Sets an output pin high. Since pins are inputs by default, the pin should be set to an output using either pinOutput or pinSetDir prior to calling this function.

Returns

Nothing

## pinLow

Prototype

void pinLow(int pin);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | pin | Pin number |

Description

Sets an output pin low. Since pins are inputs by default, the pin should be set to an output using either pinOutput or pinSetDir prior to calling this function.

Returns

Nothing

## pinToggle

Prototype

void pinToggle(int pin);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | pin | Pin number |

Description

Toggles an output pin. Since pins are inputs by default, the pin should be set to an output using either pinOutput or pinSetDir prior to calling this function.

Returns

Nothing

## pinPulseIn

Prototype

void pinPulseIn(int pin, int state);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | pin | Pin number |
| [in] | state | Pin state to measure |

Description

Measures a pulse on an input pin.

If state is 0, it measures a pulse starting immediately if the pin is already low or at the next high to low transition if it is high and ending at a low to high transition.

If state is 1, it measures a pulse starting immediately if the pin is already high or at the next low to high transition if it is low and ending at a high to low transition.

Returns

The duration of the pulse in clock ticks.

## pinPulseOut

Prototype

void pinPulseOut(int pin, int duration);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | pin | Pin number |
| [in] | duration | Pulse duration in clock ticks |

Description

Generates a pulse of the specified duration on an output pin. The pulse starts by toggling the state of the specified pin and ends by toggling it again. If the pin is low when this function is called, it will be set high for the specified duration and then back low again. If it is high when this function is called, it will be set low for the specified duration and then set high again.

Returns

Nothing

# I2C Functions

To use the I2C functions, include the <propeller/i2c.h” header file.

## i2cInit

Prototype

int i2cInit(

I2C\_STATE \*dev,

int scl,

int sda,

int frequency);

Parameters

|  |  |  |
| --- | --- | --- |
| [out] | dev | I2C state structure |
| [in] | scl | SCL pin number |
| [in] | sda | SDA pin number |
| [in] | frequency | I2C bus frequency |

Description

Initializes an I2C device on the specified pins. The bus frequency can be set as high as 1mhz. This function must be called to initialize the I2C\_STATE structure before any of the other I2C functions can be called. This function starts an I2C driver on another COG.

Returns

|  |  |
| --- | --- |
| 0 | Success |
| -1 | Failure |

## i2cTerm

Prototype

int i2cTerm(I2C\_STATE \*dev);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | dev | I2C state structure |

Description

Terminates an I2C device and releases the COG that was running the I2C driver.

Returns

|  |  |
| --- | --- |
| 0 | Success |
| -1 | Failure |

## i2cSendBuf

Prototype

int i2cSendBuf(

I2C\_STATE \*dev,

int address,

uint8\_t \*buffer,

int count);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | dev | I2C state structure |
| [in] | address | I2C address |
| [in] | buffer | Buffer containing bytes to send |
| [in] | count | Count of bytes in the buffer |

Description

Sends a buffer of data to the specified I2C address. The I2C\_STATE structure must have been initialized by a call to i2cInit before calling this function.

Returns

|  |  |
| --- | --- |
| 0 | Success |
| -1 | Failure |

## i2cBegin

Prototype

int i2cBegin(I2C\_STATE \*dev, int address);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | dev | I2C state structure |
| [in] | address | I2C address |

Description

Begins building a command to send to the I2C device at the specified address. Bytes can be added to the command using the i2cSend function. Once all of the necessary bytes are added to the command, the i2cEnd function is used to send the command to the I2C device. The I2C\_STATE structure must have been initialized by a call to i2cInit before calling this function.

Returns

|  |  |
| --- | --- |
| 0 | Success |
| -1 | Failure |

## i2cAddByte

Prototype

int i2cAddByte(I2C\_STATE \*dev, int byte);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | dev | I2C state structure |
| [in] | byte | Byte to send |

Description

Adds a byte to a command started by a call to i2cBegin. Up to 32 bytes can be added to a command. Once all of the necessary bytes are added to the command, the i2cEnd function is used to send the command to the I2C device.

Returns

|  |  |
| --- | --- |
| 0 | Success |
| -1 | Failure |

## i2cSend

Prototype

int i2cSend(I2C\_STATE \*dev);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | dev | I2C state structure |
| [in] | byte | Byte to send |

Description

Sends the I2C command that was begun with a call to i2cBegin followed by calls to i2cAddByte.

Returns

|  |  |
| --- | --- |
| 0 | Success |
| -1 | Failure |

## i2cRequestBuf

Prototype

int i2cRequestBuf(

I2C\_STATE \*dev,

int address,

uint8\_t \*buffer,

int count);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | dev | I2C state structure |
| [in] | address | I2C address |
| [out] | buffer | Buffer for the bytes received |
| [in] | count | Count of bytes to receive |

Description

Receives a buffer of data from the specified I2C address. The I2C\_STATE structure must have been initialized by a call to i2cInit before calling this function.

Returns

|  |  |
| --- | --- |
| 0 | Success |
| -1 | Failure |

## i2cRequest

Prototype

int i2cRequest(

I2C\_STATE \*dev,

int address,

int count);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | dev | I2C state structure |
| [in] | address | I2C address |
| [in] | count | Count of bytes to receive |

Description

Receives and buffers data from the specified I2C address. After this function has been called, the i2cGetByte function can be called to get each byte from the received message. The I2C\_STATE structure must have been initialized by a call to i2cInit before calling this function.

Returns

|  |  |
| --- | --- |
| 0 | Success |
| -1 | Failure |

## i2cGetByte

Prototype

int i2cGetByte(I2C\_STATE \*dev);

Parameters

|  |  |  |
| --- | --- | --- |
| [in] | dev | I2C state structure |

Description

Gets a byte from the data that was retrieved with a call to i2cRequest. The i2cRequest function must have been called prior to calling this function and only the number of bytes requested can be fetched using i2cGetByte.

Returns

|  |  |
| --- | --- |
| 0 | Success |
| -1 | Failure |

# Serial Terminal Functions

TERM \*serialTermStart(

TERM\_SERIAL \**serialTerm*,

FILE \**fp*);

void serialTermStop(TERM \* *term*)

# VGA Terminal Functions

**TERM** \* **vgaTermStart**(**TERM\_VGA** \*vgaTerm, int basepin);

void **vgaTermStop**(**TERM** \*term);

# TV Terminal Functions

**TERM** \* **tvTermStart**(**TERM\_TV** \*tvTerm, int basepin);

void **tvTermStop**(**TERM** \*term);

# Terminal Functions

**void termBin (TERM \* *term*, int *value*, int *digits*)**

Term Bin function prints a binary number at current position

**Parameters:**

|  |  |
| --- | --- |
| *value* | is number to print |
| *digits* | is number of digits in value to print |

**void termClearScreen (TERM \* *term*)**

**void termDec (TERM \* *term*, int *value*)**

Term Dec function prints a decimal number at current position

**Parameters:**

|  |  |
| --- | --- |
| *value* | is number to print |

**int termGetColors (TERM \* *term*)**

Term GetColors function gets palette color set index

**Returns:**

number representing color set index

**int termGetColumns (TERM \* *term*)**

Term GetColumns function gets screen width.

**Returns:**

screen column count.

**int termGetRows (TERM \* *term*)**

Term GetRows function gets screen height.

**Returns:**

screen row count.

**int termGetTileColor (TERM \* *term*, int *x*, int *y*)**

Term SetTileColor sets tile data color at x,y position

**Parameters:**

|  |  |
| --- | --- |
| *x* | is current x screen position |
| *y* | is current y screen position |

**int termGetX (TERM \* *term*)**

Term GetX function gets column position

**Returns:**

column position

**int termGetY (TERM \* *term*)**

Term GetY function gets row position

**Returns:**

row position

**void termHex (TERM \* *term*, int *value*, int *digits*)**

Term Hex function prints a hexadecimal number at current position

**Parameters:**

|  |  |
| --- | --- |
| *value* | is number to print |
| *digits* | is number of digits in value to print |

**void termNewLine (TERM \* *term*)**

Term NewLine go to the start of a new line

**int termOut (TERM \* *term*, int *c*)**

Term Out function prints a character at current position or performs a screen function based on the following table:

$00 = clear screen $01 = home $08 = backspace $09 = tab (8 spaces per) $0A = set X position (X follows) $0B = set Y position (Y follows) $0C = set color (color follows) $0D = return others = printable characters

**Parameters:**

|  |  |
| --- | --- |
| *value* | is number to print |
| *digits* | is number of digits in value to print |

**void termPrint (TERM \* *term*, const char \* *s*)**

Term Print null terminated char\* to screen with normal stdio definitions

**Parameters:**

|  |  |
| --- | --- |
| *s* | is null terminated string to print using putchar |

**int termPutChar (TERM \* *term*, int *c*)**

Term PutChar print char to screen with normal stdio definitions

**Parameters:**

|  |  |
| --- | --- |
| *c* | is character to print |

**void termSetColorPalette (TERM \* *term*, const char \* *palette*)**

Term Setcolors function sets the palette to that defined by pointer.

Override default color palette palette must point to a list of up to 8 colors arranged as follows (where r, g, b are 0..3):

fore back

------------

palette byte %rgb, %rgb 'color 0 byte %rgb, %rgb 'color 1 byte %rgb, %rgb 'color 2 ...

**Parameters:**

|  |  |
| --- | --- |
| *palette* | is a char array[16]. |

**void termSetColors (TERM \* *term*, int *value*)**

Term SetColors function sets palette color set index

**Parameters:**

|  |  |
| --- | --- |
| *value* | is a color set index number 0 .. 7 |

**void termSetCoordPosition (TERM \* *term*, int *x*, int *y*)**

Term SetCoordPosition function sets position to cartesian x,y.

**Parameters:**

|  |  |
| --- | --- |
| *x* | is column counted from left. |
| *y* | is row counted from bottom. |

**void termSetCurPosition (TERM \* *term*, int *x*, int *y*)**

Term SetCurPositon function sets position to x,y.

**Parameters:**

|  |  |
| --- | --- |
| *x* | is column counted from left. |
| *y* | is row counted from top. |

**void termSetTileColor (TERM \* *term*, int *x*, int *y*, int *color*)**

Term SetTileColor sets tile data color at x,y position

**Parameters:**

|  |  |
| --- | --- |
| *x* | is current x screen position |
| *y* | is current y screen position |
| *color* | is color to set |

**void termSetX (TERM \* *term*, int *value*)**

Term SetX function sets column position value

**Parameters:**

|  |  |
| --- | --- |
| *value* | is new column position |

**void termSetXY (TERM \* *term*, int *x*, int *y*)**

Term SetXY function sets position to x,y.

**Parameters:**

|  |  |
| --- | --- |
| *x* | is column counted from left. |
| *y* | is row counted from top. |

**void termSetY (TERM \* *term*, int *value*)**

Term SetY function sets row position value

**Parameters:**

|  |  |
| --- | --- |
| *value* | is new row position |

**void termStr (TERM \* *term*, const char \* *sptr*)**

termStr function prints a string at current position

**Parameters:**