Propeller Library

by George Androbus

April 22, 2012

Table of Contents

2 Pin Functions 3

2.1 pinInput 3

2.2 pinOutput 3

2.3 pinGetDir 3

2.4 pinSetDir 4

2.5 pinGet 4

2.6 pinSet 5

2.7 pinHigh 5

2.8 pinLow 5

2.9 pinToggle 6

2.10 pinPulseIn 6

2.11 pinPulseOut 7

3 I2C Functions 8

3.1 i2cInit 8

3.2 i2cTerm 8

3.3 i2cSendBuf 9

3.4 i2cBegin 9

3.5 i2cAddByte 10

3.6 i2cSend 10

3.7 i2cRequestBuf 11

3.8 i2cRequest 11

3.9 i2cGetByte 12

# Pin Functions

## 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

## pinGetDir

Prototype

int pinGetDir(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.

## pinSetDir

Prototype

void pinSetDir(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.

## 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

## 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

## 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 |