mLink Library Reference Guide for

mLink 6 Button Pad
(HCMODU0193)

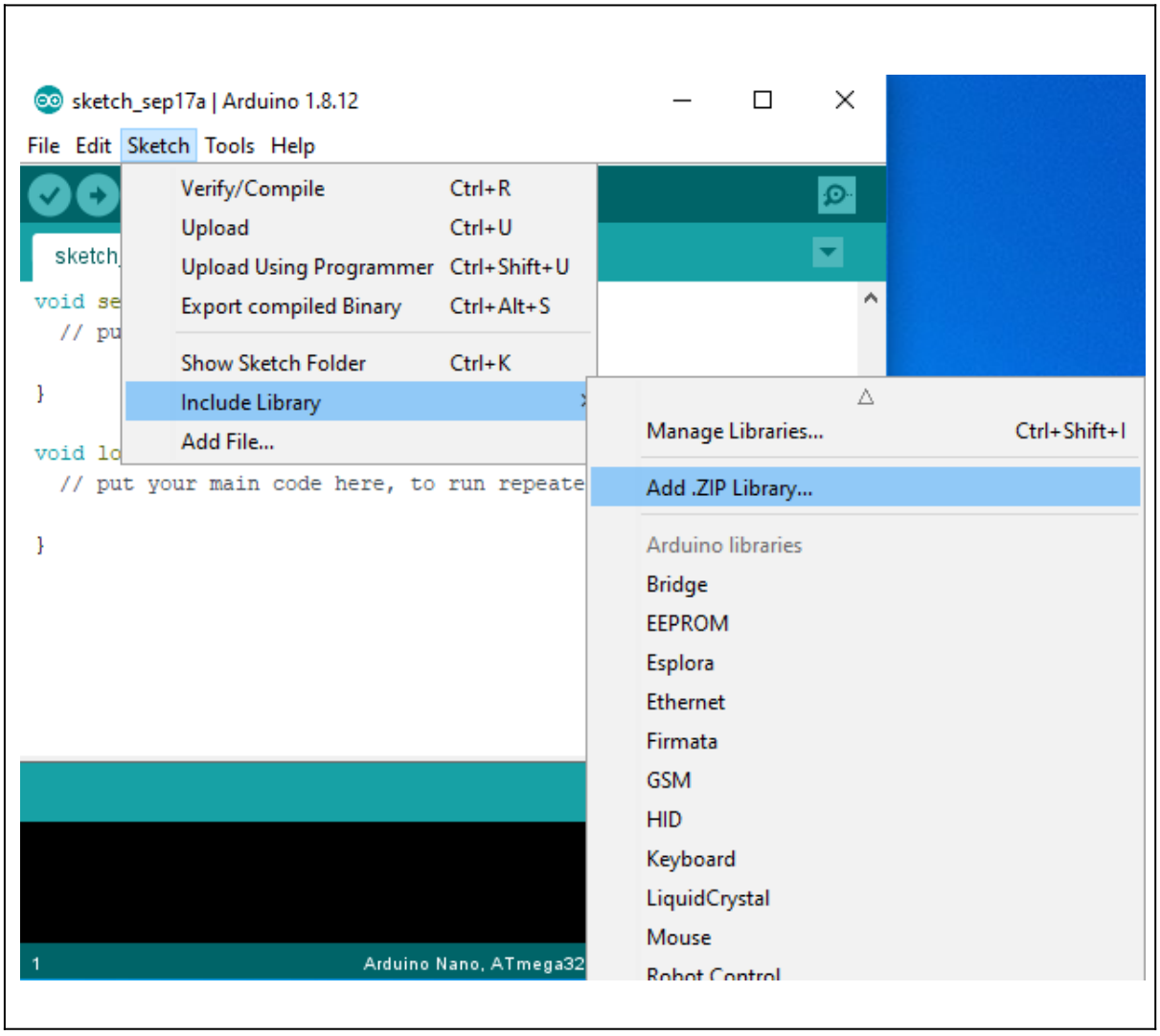
Installing the mLink library

Adding the mLink library to your Arduino IDE can be done in the same way as any other Arduino library:

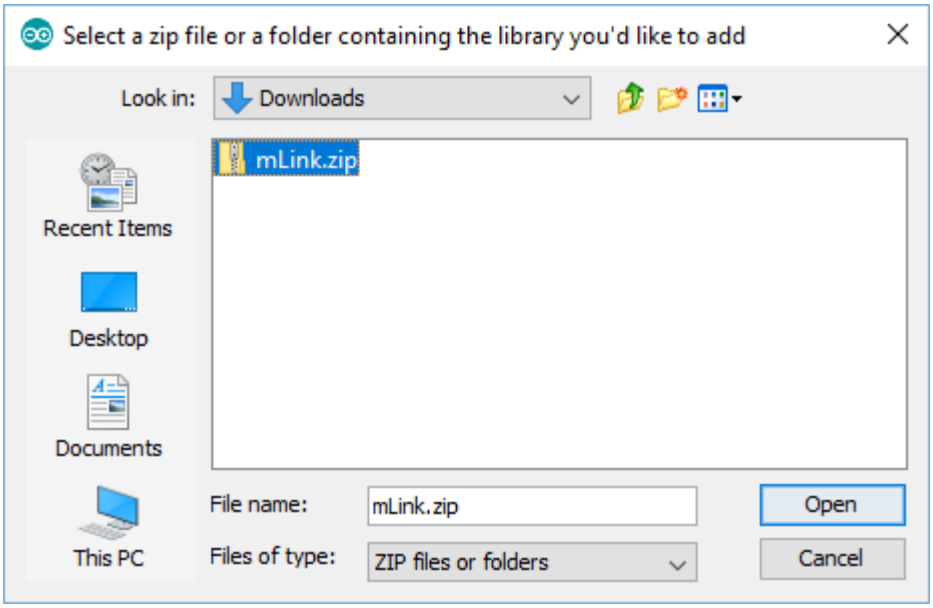
First download the mLink.zip file from the software section of our support forum here:

https://hobbycomponents.com/mLink

Once downloaded, open up your Arduino IDE and go to Sketch->Include Library->Add .ZIP Library.



In the file selection dialogue window that opens, navigate to wherever you downloaded the mLink .zip file and select it, then click the ‘Open’ button.



Including the mLink library in your sketch

Adding the mLink library to your sketch consists of 3 steps; Firstly include the mLink header file (mLink.h) at the top of your sketch, create an instance of the library, then finally initialise the library inside the startup() function:

```
// Step 1: Include the mLink library
#include "mLink.h"


//Step 2: Create an instance of the library
mLink mLink;


void setup()
{
  // Step 3: Initialise the library
  mLink.init();
}


void loop()
{
}
```

Quick library reference table

COMMAND		PARAMETERS	RETURNS
init()	Initialises the mLink library	None	n/a
readBit(*add*, *reg*, *bit*)	Reads the state of a bit from one of the mLink registers	*add* = *byte* value containing I2C address of mLink module *reg* = *byte* value containing register index *bit* = byte value containing the bit number to read (0 to 7)	*boolean* value containing the state of the bit
read(*add*, *reg*)	Reads the contents of one of the mLink registers	*add* = *byte* value containing I2C address of mLink module *reg* = *byte* value containing register index	*byte* value containing the state of the register
write(*add*, *reg*, *data*)	Writes a single byte of data to one of the mLink registers	*add* = *byte* value containing I2C address of mLink module *reg* = *byte* value containing register index *data* = *byte* value containing the data to write to the register	n/a
bPad_Empty(*add*)	Library macro that checks if the buffer is empty	*add* = *byte* value containing I2C address of mLink module	*boolean* value containing the buffer empty bit state
bpad_Read(*add*)	Library macro that reads a byte from the buffer	*add* = *byte* value containing I2C address of mLink module	*unsigned byte* containing the button index
bPad_UpState(*add*)	Library macro that reads the current state the UP button	*add* = *byte* value containing I2C address of mLink module	*boolean* value containing the current state of the UP button
bPad_LeftState(*add*)	Library macro that reads the current state the LEFT button	*add* = *byte* value containing I2C address of mLink module	*boolean* value containing the current state of the LEFT button
bPad_DownState(*add*)	Library macro that reads the current state the DOWN button	*add* = *byte* value containing I2C address of mLink module	*boolean* value containing the current state of the DOWN button
bPad_RightState(*add*)	Library macro that reads the current state the RIGHT button	*add* = *byte* value containing I2C address of mLink module	*boolean* value containing the current state of the RIGHT button
bPad_SelectState(*add*)	Library macro that reads the current state the SELECT button	*add* = *byte* value containing I2C address of mLink module	*boolean* value containing the current state of the SELECT button

bPad_BackState(*add*)	Library macro that reads the current state the BACK button	*add* = *byte* value containing I2C address of mLink module	*boolean* value containing the current state of the BACK button
bpad_Debounce(*add*, *level*)	Library macro that sets the amount of debouncing applied to the 6 keypad buttons	*add* = *byte* value containing I2C address of mLink module *level*= *byte* value containing the amount of debouncing (0 to 254)	n/a

Library Commands

mLink.init()

Description

Initialises the mLink library
Add to the setup() section of your sketch to initialise the mLink library

Syntax

mLink.init()

Parameters

None

Returns

Nothing

Example Code

```
void setup()
{
  mLink.init();
}

void loop()
{
}
```

mLink.readBit(add, reg, bit)

Description

Reads the state of a bit from one of the mLink modules 8 bit registers and returns the result as a boolean value.

Parameters

add: byte value containing I2C address of mLink module. Alternatively, if the mLink module is set to its default I2C address (0x59) you can use the predefined value:

BPAD_I2C_ADD

reg: byte value containing the register number to read. You can either specify the register number (see register table) or you can use one of the following predefined values:

MLINK_STATUS_REG
BPAD_BUFFER_STATUS
BPAD_KEYSTATE

bit: byte value containing the bit number within the specified register to read. Valid values are 0 to 7.

Returns

A boolean value representing the state of the bit.

Example Code

Checks to see if the buffer is empty by reading the EMPTY bit (bit 0) in the buffer status register.

```
boolean empty = mLink.readBit(BPAD_I2C_ADD, BPAD_BUFFER_STATUS, 0);
if(empty)
  Serial.println("Buffer is empty");
else
  Serial.println("Buffer contains at least one key");
```

mLink.read(*add*, *reg*)

Description

Reads the state of one of the mLink modules 8 bit registers and returns the result as a byte.

Parameters

add: byte value containing I2C address of mLink module. Alternatively, if the mLink module is set to its default I2C address (0x59) you can use the predefined value:

BPAD_I2C_ADD

reg: byte value containing the register number to read. You can either specify the register number (see register table) or you can use one of the following predefined values:

MLINK_STATUS_REG
MLINK_ADD_REG
MLINK_MOD_TYPE_REG
MLINK_MOD_SUBTYPE_REG
MLINK_SW_VER_REG
BPAD_BUFFER_STATUS
BPAD_BUFFER
BPAD_KEYSTATE
BPAD_DEBOUNCE

Returns

A byte value representing the state of the register.

Example Code

Reads the software version register (register 4)

```
byte version = mLink.read(BPAD_I2C_ADD, MLINK_SW_VER_REG);
```

mLink.write(*add*, *reg*, *data*)

Description

Writes to one of the mLink modules 8 bit registers.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x59) you can use the predefined value:

BPAD_I2C_ADD

reg: byte value containing the register number to write to. You can either specify the register number (see register table) or you can use one of the following predefined values:

MLINK_STATUS_REG
MLINK_ADD_REG

data: byte value containing the data to write to the register

Returns

None

Example Code

Clears the status register by writing any value to it.

```
mLink.write(BPAD_I2C_ADD, MLINK_STATUS_REG, 0);
```

mLink.bPad_Empty(*add*);

Description

Library macro that checks if the buffer is empty. This macro will return a ‘1’ (true) if there is nothing pending in the buffer or a ‘0’ (false) if there are one or more button codes pending in the buffer.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x59) you can use the predefined value:

BPAD_I2C_ADD

Returns

A *boolean* representing the empty bit state:

0 = buffer has one or more button codes pending
1 = The buffer is currently empty

Note: This function should always be called to check if there is valid data in the buffer before attempting to read it.

Example Code

Reads the state of the empty bit in the button status register and stores the result in the boolean variable ‘empty’.

```
boolean empty = mLink.bPad_Empty(BPAD_I2C_ADD);
```

mLink.bPad_Read(*add*);

Description

Library macro that reads a key code from the buffer.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x59) you can use the predefined value:

BPAD_I2C_ADD

Returns

A byte value containing the key code of the next button pressed (since the last time the register was read). Subsequent reads of this register will return keycodes for any additional buttons pressed in the order they were pressed until the buffer is empty. A returned keycode will be one of the following:

0 = UP KEY
1 = LEFT KEY
2 = DOWN KEY
3 = RIGHT KEY
4 = SELECT KEY
5 = BACK KEY
255 = KEY INVALID

Note1: Before reading data from the buffer you should always check if there is valid data in the buffer by reading the buffer empty bit in the buffer status register using the bBad_Read() macro function (see example below). Attempting to read data from the buffer when it is empty will result in this function returning a key code of 255 (key invalid).

Note2: The keypad buffer can store a maximum of 16 key codes.

Example Code

Checks to see if there is valid data in the buffer and if so, reads the next key code and then prints it to the serial port.

```
boolean empty = mLink.bPad_Empty(BPAD_I2C_ADD);

if(!empty)
{
  byte code = mLink.bPad_Read(BPAD_I2C_ADD);
  Serial.println(code);
}
```

mLink.bPad_UpState(*add*);

Description

Library macro that reads the current status of the UP button.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x59) you can use the predefined value:

BPAD_I2C_ADD

Returns

A *boolean* value containing the current state of the UP button where:

0 (false) = button is not currently pressed
1 (true) = button is pressed

Example Code

Reads the current state of the UP button.

```
boolean state = mLink.bPad_UpState(BPAD_I2C_ADD);

if(state)
   Serial.println("UP is pressed!");
```

mLink.bPad_LeftState(*add*);

Description

Library macro that reads the current status of the LEFT button.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x59) you can use the predefined value:

BPAD_I2C_ADD

Returns

A *boolean* value containing the current state of the LEFT button where:

0 (false) = button is not currently pressed
1 (true) = button is pressed

Example Code

Reads the current state of the LEFT button.

```
boolean state = mLink.bPad_LeftState(BPAD_I2C_ADD);

if(state)
   Serial.println("LEFT is pressed!");
```

mLink.bPad_DownState(*add*);

Description

Library macro that reads the current status of the DOWN button.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x59) you can use the predefined value:

BPAD_I2C_ADD

Returns

A *boolean* value containing the current state of the DOWN button where:

0 (false) = button is not currently pressed
1 (true) = button is pressed

Example Code

Reads the current state of the DOWN button.

```
boolean state = mLink.bPad_DownState(BPAD_I2C_ADD);

if(state)
   Serial.println("DOWN is pressed!");
```

mLink.bPad_RightState(*add*);

Description

Library macro that reads the current status of the RIGHT button.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x59) you can use the predefined value:

BPAD_I2C_ADD

Returns

A *boolean* value containing the current state of the RIGHT button where:

0 (false) = button is not currently pressed
1 (true) = button is pressed

Example Code

Reads the current state of the RIGHT button.

```
boolean state = mLink.bPad_RightState(BPAD_I2C_ADD);

if(state)
   Serial.println("RIGHT is pressed!");
```

mLink.bPad_SelectState(*add*);

Description

Library macro that reads the current status of the SELECT button.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x59) you can use the predefined value:

BPAD_I2C_ADD

Returns

A *boolean* value containing the current state of the SELECT button where:

0 (false) = button is not currently pressed
1 (true) = button is pressed

Example Code

Reads the current state of the SELECT button.

```
boolean state = mLink.bPad_SelectState(BPAD_I2C_ADD);

if(state)
   Serial.println("SELECT is pressed!");
```

mLink.bPad_BackState(*add*);

Description

Library macro that reads the current status of the BACK button.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x59) you can use the predefined value:

BPAD_I2C_ADD

Returns

A *boolean* value containing the current state of the BACK button where:

0 (false) = button is not currently pressed
1 (true) = button is pressed

Example Code

Reads the current state of the BACK button.

```
boolean state = mLink.bPad_BackState(BPAD_I2C_ADD);

if(state)
   Serial.println("BACK is pressed!");
```

mLink.bPad_Debounce(*add, level*);

Description

Library macro that sets the amount of debouncing applied to the 6 buttons (default = 200).

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x59) you can use the predefined value:

BPAD_I2C_ADD

level: byte value containing the amount of debouncing applied to the 6 buttons. Valid values are between 0 (no debouncing) and 254 (max debouncing).

Note: This value is stored in the modules non-volatile memory as will be applied even when the modules power is cycled.

Returns

None

Example Code

Sets the amount of debouncing to 100.

```
mLink.bPad_Debounce(BPAD_I2C_ADD, 100);
```

DISCLAIMER

The mLink range is a series of modules intended for the hobbyist and educational markets. Where every care has been taken to ensure the reliability and durability of this product it should not be used in safety or reliability critical applications.

This library and document is provided "as is". Hobby Components Ltd makes no warranties, whether express, implied or statutory, including, but not limited to, implied warranties of merchantability and fitness for a particular purpose, accuracy or lack of negligence. Hobby Components Ltd shall not, in any circumstances, be liable for any damages, including, but not limited to, special, incidental or consequential damages for any reason whatsoever.

COPYRIGHT NOTICE