mLink Library Reference Guide for

mLink Character LCD
(HCMODU0190x)

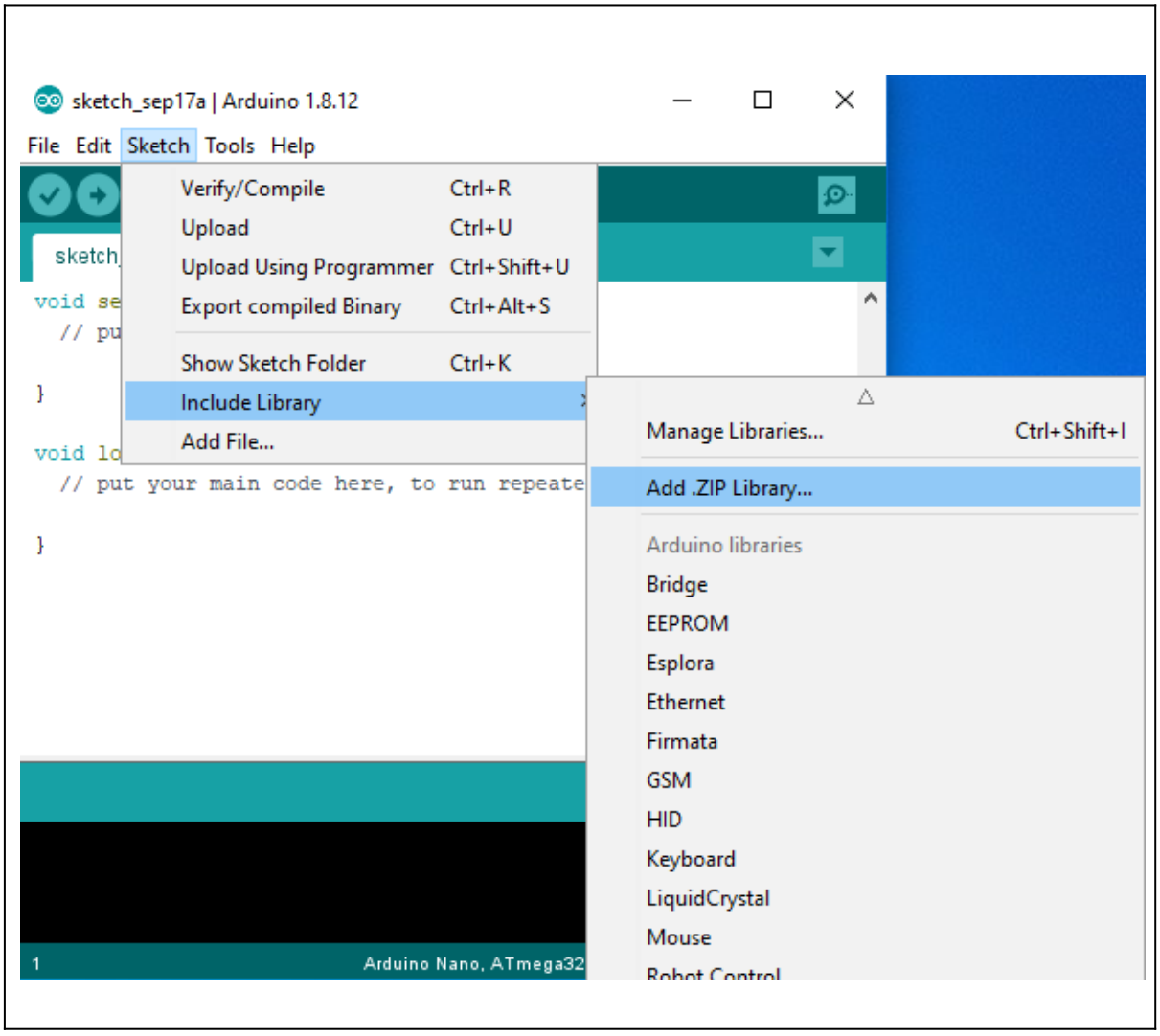
Installing the mLink library

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

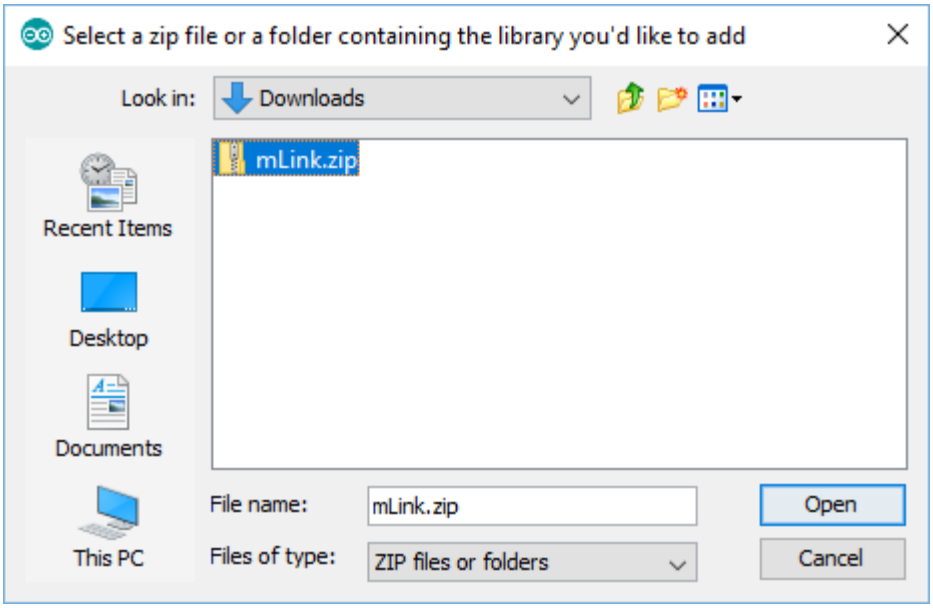
First download the mLink.zip file from the software section of our support forum here:

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
writeBit(*add*, *reg*, *bit*, *state*)	Writes to a bit in 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 write to (0 to 7) *state* = boolean value to set the bit to	n/a
write(*add*, *reg*, *data*)	Writes 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
write(*add*, *reg*, *bytes*, **data*)	Writes data to one or more consecutive registers	*add* = *byte* value containing I2C address of mLink module *reg* = *byte* value containing register index of the first register to write to *bytes* = the size of the data in bytes to write **data* = a byte pointer to the start of the data to write	n/a
sleep(*add*);	Puts the module into a low power sleep mode.	*add* = *byte* value containing I2C address of mLink module	n/a
cLCD_print(*add*, *text*)	Library macro that prints ASCII text to the display	*add* = *byte* value containing I2C address of mLink module *text* = a null terminated character array	n/a

cLCD_printFloat(*add*, *val*, *dp*)	Library macro that prints a floating point number to the display	*add* = *byte* value containing I2C *val* = the floating point number to print *dp* = the number of decimal places to display	n/a
cLCD_printCust(*add*, *i*)	Library macro that prints one of the 8 custom characters to the display	*add* = *byte* value containing I2C i = the index number (0 to 7) of the custom character to print	n/a
cLCD_cursor(*add*, *col*, *row*)	Library macro sets the cursor location	*add* = *byte* value containing I2C *col* = signed byte value containing the column position (-128 to 127) *row* = signed byte value containing the row position (-128 to 127)	n/a
cLCD_clear(*add*)	Library macro that clears the display	*add* = *byte* value containing I2C	n/a
cLCD_on(*add*, *state*)	Library macro turns the display on or off	*add* = *byte* value containing I2C *state* = the required state (1 = on, 0 = off)	n/a
cLCD_cursDir(*add*, *dir*)	Library macro that sets the cursor direction	*add* = *byte* value containing I2C *dir* = is the direction of the cursor (0 = left to right, 1 = right to left)	n/a
cLCD_dispType(*add*, *type*)	Library macro that sets the display type	*add* = *byte* value containing I2C *type* =the display type (0 = 16x2, 1 = 20x4)	n/a
cLCD_backlight(*add*, *level*)	Library macro that sets the brightness level of the backlight	*add* = *byte* value containing I2C *level* = a value from 0 to 10 representing the backlight level (0 = off, 10 = maximum)	n/a
cLCD_contrast(*add*, *level*)	Library macro that sets the displays contrast level	*add* = *byte* value containing I2C *level* = bye value representing the contrast level (0 = min, 255 = max)	n/a
cLCD_setCust0(*add*, *bitmap*)	Library macro that writes a bitmap to custom character 0	*add* = *byte* value containing I2C *bitmap* = an array of 8 bytes containing the bitmap	n/a
cLCD_setCust1(*add*, *d*)	Library macro that writes a bitmap to custom character 1	*add* = *byte* value containing I2C *bitmap* = an array of 8 bytes containing the bitmap	n/a
cLCD_setCust2(*add*, *d*)	Library macro that writes a bitmap to custom character 2	*add* = *byte* value containing I2C *bitmap* = an array of 8 bytes containing the bitmap	n/a
cLCD_setCust3(*add*, *d*)	Library macro that writes a bitmap to custom character 3	*add* = *byte* value containing I2C *bitmap* = an array of 8 bytes containing the bitmap	n/a

cLCD_setCust4(*add*, *d*)	Library macro that writes a bitmap to custom character 4	*add* = *byte* value containing I2C *bitmap* = an array of 8 bytes containing the bitmap	n/a
cLCD_setCust5(*add*, *d*)	Library macro that writes a bitmap to custom character 5	*add* = *byte* value containing I2C *bitmap* = an array of 8 bytes containing the bitmap	n/a
cLCD_setCust6(*add*, *d*)	Library macro that writes a bitmap to custom character 6	*add* = *byte* value containing I2C *bitmap* = an array of 8 bytes containing the bitmap	n/a
cLCD_setCust7(*add*, *d*)	Library macro that writes a bitmap to custom character 7	*add* = *byte* value containing I2C *bitmap* = an array of 8 bytes containing the bitmap	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 (0x56) you can use the predefined value:

CLCD_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
CLCD_CR1
CLCD_CR2

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

Reads the state of bit 1 (display type) from control register 2

```
boolean result = mLink.readBit(CLCD_I2C_ADD, MLINK_CLCD_CR2, 1);
```

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 (0x56) you can use the predefined value:

CLCD_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
CLCD_COL
CLCD_ROW
CLCD_CR1
CLCD_CR2
CLCD_BACKLIGHT
CLCD_CONTRAST

Returns

A byte value representing the state of the register.

Example Code

Reads the col register to get the cursor column position (register 12)

```
int8_t col = mLink.read(CLCD_I2C_ADD, CLCD_COL);
```

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 (0x56) you can use the predefined value:

CLCD_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
CLCD_COL
CLCD_ROW
CLCD_PRINT
CLCD_CR1
CLCD_CR2
CLCD_BACKLIGHT
CLCD_CONTRAST
CLCD_PRINT_CUST

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

Returns

None

Example Code

Prints the ASCII character ‘A’ to the display at the current cursor location.

```
mLink.write(CLCD_I2C_ADD, CLCD_PRINT, ‘A’);
```

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

Description

Writes data to one or more consecutive registers.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x56) you can use the predefined value:

CLCD_I2C_ADD

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

Register address	Label	Alternate Label
0x0B	MLINK_CLCD_PRINT_CHAR_REG	CLCD_PRINT
0x13	MLINK_CLCD_CUST0_REG	CLCD_CUST0
0x14	MLINK_CLCD_CUST1_REG	CLCD_CUST1
0x15	MLINK_CLCD_CUST2_REG	CLCD_CUST2
0x16	MLINK_CLCD_CUST3_REG	CLCD_CUST3
0x17	MLINK_CLCD_CUST4_REG	CLCD_CUST4
0x18	MLINK_CLCD_CUST5_REG	CLCD_CUST5
0x19	MLINK_CLCD_CUST6_REG	CLCD_CUST6
0x1A	MLINK_CLCD_CUST7_REG	CLCD_CUST7

bytes: byte value containing the size of the data in bytes to write.

**data:* a byte pointer to the start of the data to write.

Returns

None

Example Code

Writes a custom character bitmap to custom character 0.

```
byte bitmap = {0x0E, 0x1F, 0x11, 0x11, 0x11, 0x11, 0x11, 0x1F};

mLink.write(CLCD_I2C_ADD, CLCD_CUST0, bitmap, 8);
```

mLink.sleep(*add*);

Description

Puts the module into a low power sleep mode.

Sleep mode is automatically exited on the next register read or write.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x56) you can use the predefined value:

CLCD_I2C_ADD

Returns

None

Example Code

Puts the module into low power sleep mode.

```
mLink.sleep(CLCD_I2C_ADD);
```

mLink.cLCD_print(*add, text*);

Description

Library macro that prints one or more ASCII characters to the display at the current cursor location.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x56) you can use the predefined value:

CLCD_I2C_ADD

text: a null terminated character array containing the text to print.

Returns

None

Example Code

Prints ‘Hello World!’ to the display at the current cursor location.

```
char text[] = "Hello World!";

mLink.cLCD_print(CLCD_I2C_ADD, text);
```

mLink.cLCD_printFloat(*add, val, dp*);

Description

Library macro that prints a floating point number to the display at the current cursor location.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x56) you can use the predefined value:

CLCD_I2C_ADD

val: the value to print

dp: the number of decimal places to print the number to.

Returns

None

Example Code

Prints the value 123.456 to 2 decimal places at the current cursor location.

```
float val = 123.456;

mLink.cLCD_printFloat(CLCD_I2C_ADD, val, 2);
```

mLink.cLCD_printCust(*add, i*);

Description

Library macro that prints one of the 8 custom characters to the display at the current cursor location.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x56) you can use the predefined value:

CLCD_I2C_ADD

i: the index number of the custom character to print (0 to 7)

Returns

None

Example Code

Prints custom character 0 to the display at the current cursor location.

```
mLink.cLCD_printCust(CLCD_I2C_ADD, 0);
```

mLink.cLCD_cursor(*add, col, row*);

Description

Library macro that sets the location of the cursor.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x56) you can use the predefined value:

CLCD_I2C_ADD

col: a signed byte value containing the column number to move the cursor to where 0 = the left most displayable column on the display and 15 (16x2 LCD), or 19 (20x4 LCD) is the right most displayable column. Writing a value outside this range will move the cursor to a position outside the displayable area.

row: a signed byte value containing the row number to move the cursor to where 0 = the top most displayable row on the display and 1 (16x2 LCD), or 3 (20x4 LCD) is the bottom, most displayable row. Writing a value outside this range will move the cursor to a position outside the displayable area.

Returns

None

Example Code

Prints ‘Hello World!’ to the display starting from column 3, row 1.

```
mLink.cLCD_cursor(CLCD_I2C_ADD, 3, 1);
mLink.cLCD_print(CLCD_I2C_ADD, “Hello World”);
```

mLink.cLCD_clear(*add*);

Description

Library macro that clears the display. Note that the cursor location will also be reset to col 0, row 0.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x56) you can use the predefined value:

CLCD_I2C_ADD

Returns

None

Example Code

Clears the display.

```
mLink.cLCD_clear(CLCD_I2C_ADD);
```

mLink.cLCD_on(*add, state*);

Description

Library macro that turns the display on or off.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x56) you can use the predefined value:

CLCD_I2C_ADD

state: a boolean value that specifies the required on off state (0 = display off, 1 = display on). Alternatively you can use the predefined values:

OFF
ON

Returns

None

Example Code

Turns the display off.

```
mLink.cLCD_on(CLCD_I2C_ADD, OFF);
```

mLink.cLCD_cursDir(*add, dir*);

Description

Library macro that sets the direction the cursor will move after printing an ASCII or custom character.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x56) you can use the predefined value:

CLCD_I2C_ADD

dir: a boolean value that specifies the required direction of the cursor (0 = cursor moves right, 1 = cursor moves left). Alternatively you can use the predefined values:

CURS_LTOR
CURS_RTOL

Returns

None

Example Code

Sets the cursor direction to move left after printing a character.

```
mLink.cLCD_curDir(CLCD_I2C_ADD, CURS_RTOL);
```

mLink.cLCD_dispType(*add, type*);

Description

Library macro that sets the display type (16x2 or 20x4).

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x56) you can use the predefined value:

CLCD_I2C_ADD

type: a boolean value containing the display type (0 = 16x2, 1 = 20x4). Alternatively you can use the predefined values:

CLCD_TYPE_1602
CLCD_TYPE_2004

Returns

None

Example Code

Sets the display type to a 20x4 character display.

```
mLink.cLCD_dispType(CLCD_I2C_ADD, CLCD_TYPE_2004);
```

mLink.cLCD_backlight(*add, level*);

Description

Library macro that sets the display's backlight level.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x56) you can use the predefined value:

CLCD_I2C_ADD

level: is a byte value containing the backlight brightness level in 10% increments where 0 = off and 10 = 100%. Note, setting the backlight level to a value above 10 will cause the level to be set to 100%

Returns

None

Example Code

Sets the backlight level to 50%.

```
mLink.cLCD_backlight(CLCD_I2C_ADD, 5);
```

mLink.cLCD_contrast(*add, level*);

Description

Library macro that sets the display's contrast level.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x56) you can use the predefined value:

CLCD_I2C_ADD

level: is a byte value containing the display's contrast level where 0 = min and 255 = max.

Returns

None

Example Code

Sets the contrast level to 0x55.

```
mLink.cLCD_contrast(CLCD_I2C_ADD, 0x55);
```

mLink.cLCD_setCust0…7(*add, bitmap*);

Description

Library macros (cLCD_setCust0 to cLCD_setCust7) that write a bitmap to one of the 8 custom characters.

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x56) you can use the predefined value:

CLCD_I2C_ADD

bitmap: is an 8 byte array containing the bitmap to write

Returns

None

Example Code

Writes a bitmap (battery icon) to custom character 0 then prints it to the display at the current cursor location.

```
byte bitmap = {0x0E, 0x1F, 0x11, 0x11, 0x11, 0x11, 0x11, 0x1F};

mLink.cLCD_setCust0(CLCD_I2C_ADD, bitmap);
mLink.cLCD_printCust(CLCD_I2C_ADD, 0);
```

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