Micronix C++ Programming Guide

Introduction

This documentation describes the different implementations of the C++ programming files. Although there are different programming files, the functionality of each is identical. There are three methods of implementing the programming files:

1. micronix_header.h (.h file only)
2. MMCLibrary (.cpp and .h files) - (Recommended)
3. MicronixMMCLib_DLL.dll (.dll file)

Features

- Serial Port Communication with MMC Devices
  - Connect to / Disconnect from MMC Devices via COM Port
  - Send command to / Receive data from MMC Devices.

Supported MMC Devices

MMC-10 , MMC-100 , MMC-103 , MMC-110 , MMC-200

Current List of Functions

- openComPortMMC(string port)
- closeComPortMMC()
- writeCommandToMMC(string axis, string command)
- pollValues(string axis, string command)

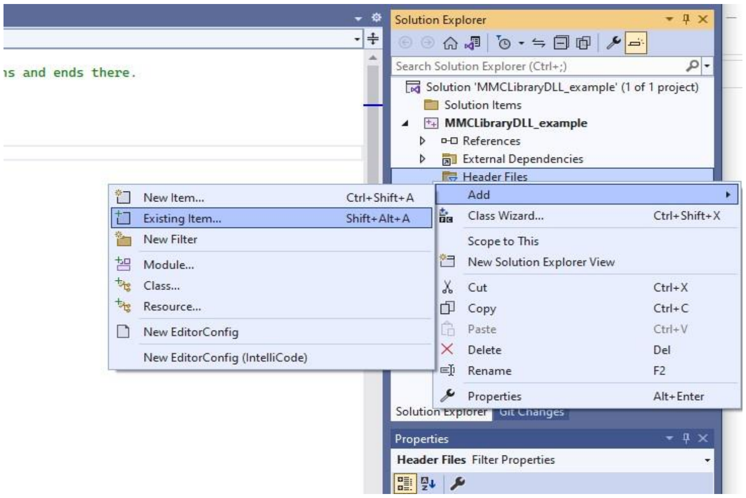
Setting Up Visual Studio for micronix_header.h

Function: Using one header file to make use of the List of Functions provided

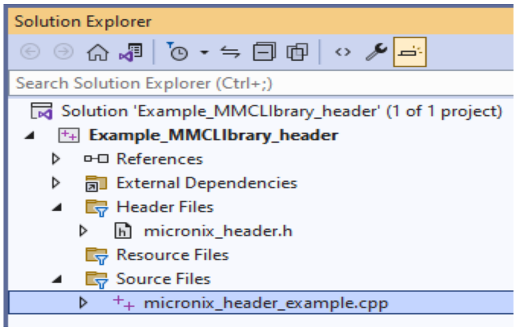
DISCLAIMER : Using multiple *#include* in the same project will give error, to use multiple inclusions of the header file use the second method of implementation - MMCLibrary.

Steps: Made in consideration with Visual Studio but similarly can be implemented in any IDE

1. Create a new Visual Studio Project in C++ using the template of Console App.
2. In the Solution Explorer, right click on '*Header File*' → '*Add*' → '*Existing Item..*' and add '*micronix_header.h*' file



3. Now, your Solution Explorer should look like this with the added file under *'Header File'*



Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Example_MMCLibrary_header' (1 of 1 project)
Example_MMCLibrary_header
References
External Dependencies
Header Files
micronix_header.h
Resource Files
Source Files
micronix_header_example.cpp

4. Now, you are all set to include the header file and use by <u>*#include "micronix_header.h"*</u>
5. You can now write your own main() function file using the header file with the list of functions.
6. Or, similarly use the '*micronix_header_example.cpp'* file by adding it into Source File and running it. (Make sure you have only one main() function running in the project)

Setting Up Visual Studio for MMCLibrary

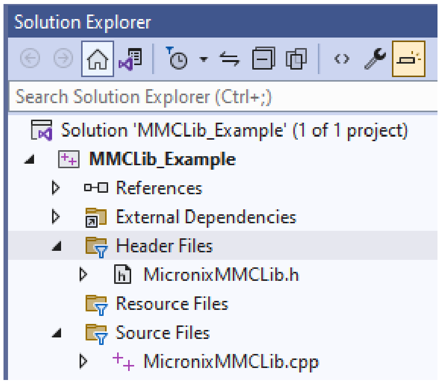
Function: To use the header file and its source file to implement functions usage

Steps : Made in consideration with Visual Studio but similarly can be implemented in any IDE

1. Create a new Visual Studio Project in C++ using the template of Console App.

2. In the Solution Explorer, right click on *'Header File'* → *'Add'* → *'Existing Item..'* and add the '*MicronixMMCLib.h'* file

3. Similarly, add '*MicronixMMCLib.cpp'* as Existing Item under the 'Source File' in the Solution Explorer



4. Now you are set to use the header file by *#include "MicronixMMCLib.h"* and write your main() function and use the list of functions given to access the MMC.

5. Similarly, use the given example program *'MMCLib_Example.cpp'* and include it under the source file and run to see it working.

Setting Up Visual Studio for MicronixMMCLib_DLL

Function: To make use of the dll file to implement functions usage

Steps : Made in consideration with Visual Studio but similarly can be implemented in any IDE

1. Create a new Visual Studio Project in C++ using the template of Console App.

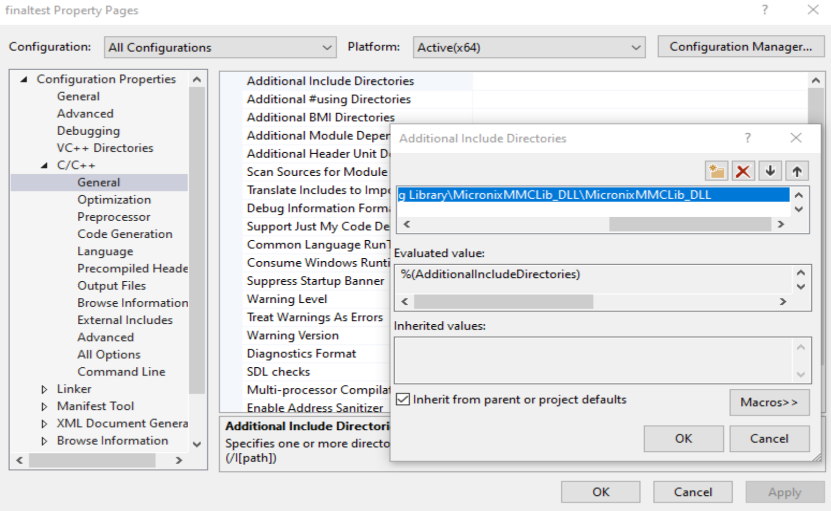
2. In Menu Bar, select *Project* → *Properties* → *C/C++* → *General*

3. Switch Configuration to “All Configuration”

4. Under ‘Additional Include Directories’ select ‘Edit’ under the dropdown menu and browse to the directory of

…\Micronix C++ Programming Library\MicronixMMCLib_DLL\MicronixMMCLib_DLL

and then ‘*Select Folder’*.



finaltest Property Pages

Configuration: All Configurations

Platform: Active(x64)

Additional Include Directories

g Library\MicronixMMCLib_DLL\MicronixMMCLib_DLL

Evaluated value:

%(AdditionalIncludeDirectories)

Inherited values:

☑ Inherit from parent or project defaults

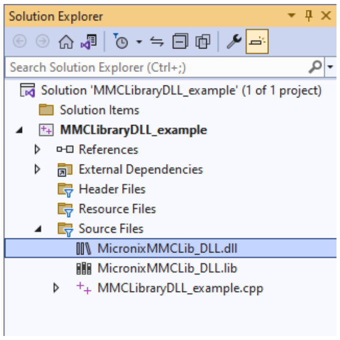
Additional Include Directories

Specifies one or more directo (/I[path])

4. After that click ‘Apply’ and ‘Ok’ and you should be able to include the header file directly by *#include “MicronixMMCLib_DLL.h”* and if it does not prompt any error it is able to find the dir.

5. In Solution Explorer, right click the Source File and select ‘*Add’ and ‘Existing Item’*.

Add the *MicronixMMCLib_DLL.dll and MicronixMMCLib_DLL.lib* file. It should look like



7. Now, you should be able to write your main() function and make use of the list of functions

8. Also, you can use the given example file *: ‘MMCLibraryDLL_example.cpp’* and add this under the source file (make sure you have only one main function under all source files) and run.

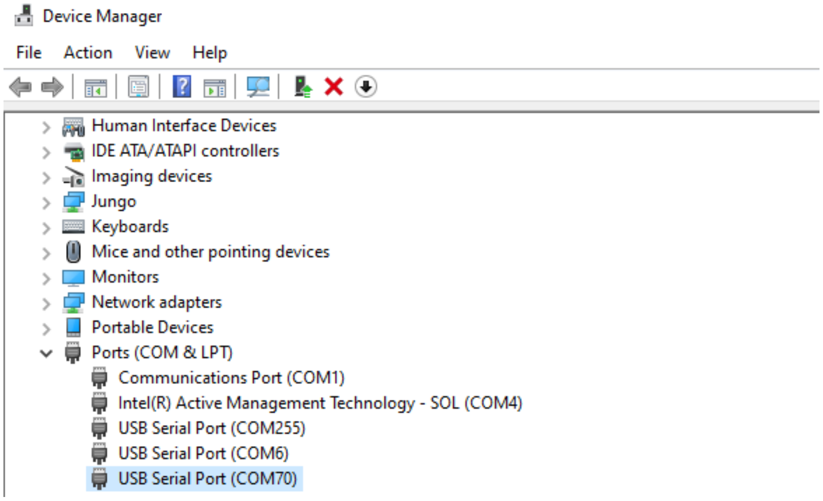
bool openComPortMMC(string *port*)

Function:

Connects to the MMC Device Virtual COM Port through the USB connection.

This function requires user to know the name of the COM port of the MMC Device.

COM port name can be found in Device Manager as "USB Serial Port (COMx)"



Return Value:

bool output (True or False)

- True is returned when COM port is successfully connected.
- False is returned when COM port fails to connect.
  - Reason for not connecting properly is usually because COM is already accessed.

C++ Example

```
string port = “COM3”;
bool isOpen = MicronixMMCLibrary.openComPortMMC(port);
if ( isOpen )
{
        std::cout << “MMC Device on ” <<  port << “ has been successfully connected.”;
}
else if ( !isOpen )
{
        std::cout << “COM port is not available.”;
}
```

bool closeComPortMMC( )

Function:

Disconnects the MMC Device opened from the openComPortMMC() function.

Return Value:

bool output (True or False)

- True is returned when COM port is successfully disconnected.
- False is returned when COM port fails to disconnect.
  - Reason for not connecting properly is usually because COM is already disconnected ie. unplugging the USB cable from the MMC Device.

C++ Example

```
string port = “COM3”;
bool isOpen =openComPortMMC(port);
if ( isOpen)
{
        std::cout << “MMC Device on ” <<  port << “ has been successfully connected.”;


        bool isClose = closeComPortMMC();
        if (isClose)
        {
                std::cout << “MMC Device successfully disconnected”
        }
}
else if ( !isOpen )
{
        std::cout << “COM port is not available.”;
}
```

bool writeCommandToMMC(string *axis* , *string command*)

Function:

Writes/Send commands to the MMC Device.

First parameter, string *axis,* is the Axis number of the MMC Device

Second parameter, string *command*, is based on Micronix Motion Control Language.

*Full list of available commands is found in the MMC Device’s reference manuals.

Return Value:

bool output (True or False)

- True is returned when MMC device’s COM port is successfully connected to be able to send command to the device.
- False is returned when COM port is not connected.

C++ Example

```
string axis = “1”;
string abs_move = “MVA”;                //Send Move to Absolute Position 10mm command
string distance = “10";

string command = abs_move + distance;

writeCommandToMMC(axis, command);
```

string pollValues(string *axis* , string *command*)

Function:

Query data based on the MMC Device’s Axis and its respective parameter to query.

For example, if the user would like to query the current speed of the MMC Device, then the user would query the velocity command (VEL).

Return Value:

string output of the *axis+command* sent

- Consult the MMC Device’s reference manual for full list of commands that can be polled and the expected return value

C++ Example

```
string axis = “1”;
string command = “VER”        //poll 1VER? which outputs the MMC device’s firmware name
string output = pollValues(axis, command);

std::cout << output ;

//Output for 1VER? on an MMC-100 would be “MMC-100v2.00.00\n\r”
```

```
string position = pollValues(“1” , “POS”);
std::cout << position ;


//Output for 1POS? would be “[Target Position],[Encoder Position]”
```

Example C++ code for using the library

Function: To showcase connection, writing command , querying, and closing the port

Example Code : Using the Implementation Method – MMCLibrary

```
#include <iostream>
#include "MicronixMMCLib.h"
int main()
{
        // Please refer to the manual for to find the MMC Device COM Port
        auto s = openComPortMMC("COM6");

        // WriteCommand return only bool values even if a query was sent
        // To get the response use the pollValues function as below
        auto s2 = writeCommandToMMC("1", "sta?");
        auto s23 = pollValues("1", "VER");
        auto s223 = pollValues("1", "pos");

        std::cout << std::endl << s23;
        std::cout << s223;

        closeComPortMMC();
}
```

Output :

//Output for 1VER? on an MMC-100 would be “MMC-100v2.00.00”

//Output for 1POS? would be “[Target Position],[Encoder Position]”

```
MMC – 200 V1.4.23
0.000000,0.000000
```