Appendix C
Building a Library

Overview

Introduction

In Lesson 16, we discussed relocatable code, and how to use a library. In this appendix, we review the mechanics of constructing a library.

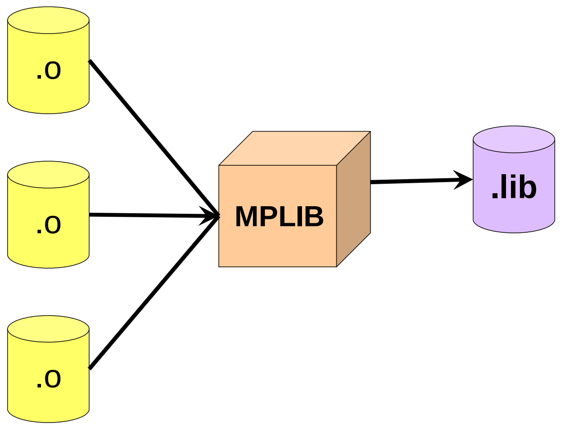
In this section

Following is a list of topics in this section:

Library Basics

Introduction

A library is basically a number of object files stuck together to make one file. The program that deals with libraries is called **MPLIB**. The basic idea is to feed a group of object files into MPLIB, which glues them together to a single file.



MPLIB Switches

MPLIB is a command line utility. MPLIB requires at least one command switch to tell the utility what to do with the library. Besides this switch, there is an optional **`/q`** (for quiet) switch to turn off displays.

The syntax is as follows:

`MPLIB` `[/q] /[`**`ctdrxh`**`]` *<library>* [*<list of objects>*]

The possible actions are: **`/c`** – create, **`/t`** – list library contents, **`/d`** – delete a member from the library, **`/r`** – add or replace a member in a library, **`/x`** – extract a member from the library, and **`/h`** - display a help message.

For All MPLAB Versions

Introduction

MPLIB has been available in MPLAB since at least version 4. Creation of a library using MPLIB as a command line utility is identical for all versions.

Using a batch file

Most of the time we will want to not only create the library, but also assemble at least some of the sources. While we could load the individual modules into the MPLAB IDE one at a time to assemble them, it is easier if we make a batch file.

Typically, the MPLAB tools won't be on our DOS PATH, so we need to spell out exactly where the executables are. In a .bat file, we could use environment variables to simplify our task.

Suppose, for example, we wanted to make a library containing only the delay routines from the LCD library:

```
: Batch file to build delay library
: MPLAB 7.01 - JJMcD 2-Mar-05
@echo off

SET TOOLS=C:\Program Files\Microchip\MPASM Suite
SET ASM="%TOOLS%\MPASMWIN.EXE"
SET LIB="%TOOLS%\MPLIB.EXE"
SET ASMFLAGS=/e+ /l+ /x- /c+ /p16F84A /o+ /q

%ASM% %ASMFLAGS% Del512ms.asm
%ASM% %ASMFLAGS% Del450ns.asm
%ASM% %ASMFLAGS% Del40us.asm
%ASM% %ASMFLAGS% Del2ms.asm
%ASM% %ASMFLAGS% Del256ms.asm
%ASM% %ASMFLAGS% Del1S.asm
%ASM% %ASMFLAGS% Del128ms.asm
%LIB% /C Delay.lib Del512ms.o Del450ns.o Del40us.o Del2ms.o Del256ms.o
Del1S.o Del128ms.o
%LIB% /T Delay.lib
```

Notice how we set a variable, `TOOLS`, containing the path for MPLAB (your path is likely to be different). Then we used that to set variables for the assembler and linker. This is not strictly necessary, it just saves typing and makes the file easier to change later. If anything in the path includes a space, for example `Program Files`, take care with the quote marks!

For version 6.x of MPLAB, the default path is:

```
SET TOOLS=C:\Program Files\MPLAB IDE\MCHIP_Tools
```

If all our files and include files fit within DOS 8.3 filenames, we could have used MPASM instead of MPASMWIN, which is quite a bit faster.

Continued on next page

For All MPLAB Versions, Continued

Using a Makefile

The DOS batch file tends to contain a lot of redundancy, which opens up opportunities for errors when we make changes. A better solution is to use **make**, if it is available. make is a sort of a scripting language that understands dependencies between files, and figures out how to get the library up to date with the minimum number of steps.

Many compilers include make, and there are a number of implementations of make available in various freeware libraries. Microsoft, in it's earlier compilers, included a version of make which had an unusual syntax. To prevent confusion, Microsoft's version of make that uses "normal" makefiles is called nmake.

A makefile to accomplish the same problem the batch file solved would look like:

```
# nMake script for building Delay library - MPLAB 7.01
#
TOOLROOT=C:\Program Files\Microchip\MPasm Suite
ASM="$(TOOLROOT)\MPASMWIN.EXE"
LIB="$(TOOLROOT)\MPLIB.EXE"
ASMFLAGS=/e+ /l+ /x- /c+ /p16F84A /o+ /q
OBJS=Del512ms.o Del450ns.o Del40us.o Del2ms.o Del256ms.o\
     Del1s.o Del128ms.o

.SUFFIXES: .asm .o .lib

.asm.o:
        $(ASM) $(ASMFLAGS) $<

ALL: Delay.lib

Delay.lib : $(OBJS)
        $(LIB) /C $@ $(OBJS)
        $(LIB) /T $@
```

The first few lines are very much like in the batch file; they simply set up some variables. The `OBJS` variable is a little different in that it consists of the names of the object files that will be entered in the library. The `.SUFFIXES` line tells make the order of precedence for the types of files we are concerned with; that is, you need a .asm to get a .o, and a .o to get a .lib.

The `.asm.o:` line is especially interesting. This tells make that any time it is considering a .o, it should look to see if there is a corresponding .asm. If the .asm is newer (or if the .o doesn't exist), then run the assembler on that .asm. The `$<` is a special variable that means whatever the dependent happens to be.

The remaining lines are action lines. A line like `A: B` means A is a target, and B is the dependent (or dependents). To make an A from a B, run the lines below (which must start with a tab … 8 spaces don't count!)

If we don't tell make what to make, it will make the first thing it encounters, in this case ALL. To get ALL it needs Delay.lib, and to make ALL from Delay.lib it does nothing.

To get a Delay.lib, make needs `$(OBJS)`, so it executes statements that look a lot like the ones we had in the batch file. The $@ is another special variable that means the target, so `$(LIB) /C $@ $(OBJS)` expands to:

```
MPLIB /C Delay.lib Del512ms.o Del450ns.o Del40us.o …
```

For MPLAB 6.60 and Later

Introduction

In 6.60, Microchip delivered library build support within the MPLAB IDE. This makes library construction more similar to other tasks within the IDE.

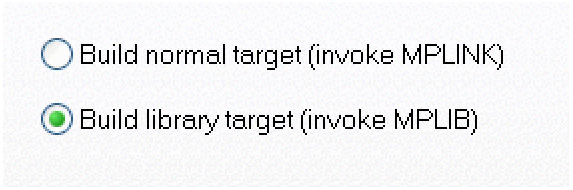
Setting up the Project

To build a library, set up a project just like any other. The project name should be the same as the desired library name, and select a directory like a normal project.

In order to build a library instead of a .hex file, go to

Projects->Build Options->Project

and select the MPASM/C17/C18 Suite tab. On that tab select "Build Library Target":

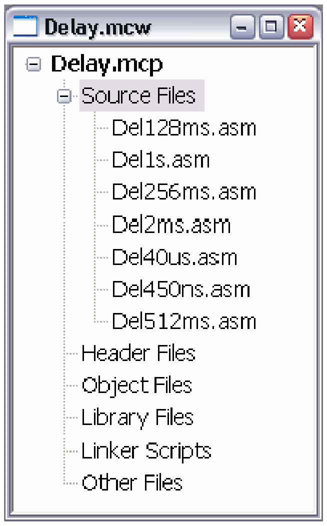


Build normal target (invoke MPLINK)
[✓ Build library target (invoke MPLIB)]

Then click OK.

Adding files to the project

Add assembler source files for the modules to be included in the project just like for a normal project. However, you do not need a linker script nor any libraries:



```
Delay.mcw
Delay.mcp
    Source Files
        Del128ms.asm
        Del1s.asm
        Del256ms.asm
        Del2ms.asm
        Del40us.asm
        Del450ns.asm
        Del512ms.asm
    Header Files
    Object Files
    Library Files
    Linker Scripts
    Other Files
```

Building the Library

Simply select

Project->Build All

Or click on the build toolbar button. Instead of a .hex file, a .lib will be created.