Creating a Link Library

Updated 09/25/2002

This article explains how to create your own custom link library for assembly language programs. If you haven't already done so, read Chapter 8 first. This article is designed as a hands-on tutorial, so we suggest that you read it sitting at a computer that has MASM installed.

There are advantages to creating your own link library and using it alongside the library in the book. A library is a good place to keep subroutines (procedures) that are general enough to be used more than once. That is, of course, why the Irvine16 and Irvine32 libraries were created. We wanted to make input-output programming as simple as possible by providing procedures that students typically need.

You certainly should not add any code to the Irvine32 library. If you did, the next time you downloaded an updated copy of this library from the book's Web site, your own code would be erased. Instead, it's best if you create your own library and use it alongside ours. We've set aside a special assembly language source code file that you can use as your library, named *MyLib32.asm*. You can modify and improve that file all you want.

Before you go any further in this tutorial, get an updated copy of the book example programs. It's a self-extracting file. It contains all the files mentioned in this article.

The Library Source Code (MyLib32.asm)

A library module is an assembly language source file containing subroutines that has been assembled in an object file. The object file can be linked to any program that calls one or more subroutines from the library. Following is *MyLib32.asm*, which you can use to hold your own procedures. For demonstration purposes, it contains the **ArraySum** procedure from page 294:

```
TITLE My Own Link Library              (MyLib32.asm)

; Personalized collection of library procedures

; Author: <you>
; Last update: 00/00/0000

INCLUDE Irvine32.inc
INCLUDE MyLib32.inc

.data
; (variables here)
.code
;-----------------------------------------------------
```

```
ArraySum PROC,
   ptrArray:PTR DWORD,     ; pointer to array
   arraySize:DWORD         ; size of array
;
; Calculates the sum of an array of 32-bit integers.
; Returns:  EAX = sum
;-----------------------------------------------------
   push ecx                ; don't push EAX
   .
   .
   ret
ArraySum ENDP
END
```

It is also necessary to create an include file containing procedure prototypes. Following is the *MyLib32.inc* file that you will use:

```
; MyLib32.inc - Include file for MyLib32.lib

; This file contains a PROTO directive for
; each procedure in MyLib32.asm.

; Last update: 08/29/2002

ArraySum PROTO, ptrArray:PTR DWORD, arraySize:DWORD
```

Producing the Object File (MyLib32.obj)

Next, you will assemble the *MyLib32.asm* file. This can be done using a batch file named *asmOnly.bat*, which contains a single command:

```
c:\Masm615\ML -c -coff %1.asm
```

Run the file from the command prompt in the \Examples\Lib32 directory, producing *MyLib32.obj*:

```
asmOnly MyLib32
```

MyLib32.obj is, for all practical purposes, a link library. (Link libraries can have extensions of DLL, OBJ, EXE, or LIB, by the way.)

Included in the WinZip file along with this areticle is *make32.bat*, which contains a special version of the LINK32 command:

```
LINK32 %1.obj MyLib32.obj irvine32.lib kernel32.lib . . .
```

Testing the Library

Before you put any library into production mode, it’s a good idea to test it with a stub program. With testing in mind, there's a file named *LibTest.asm* in the \Examples\Lib32 directory. Let’s

open it and note the call to **ArraySum**. Notice that an extra INCLUDE directive has been placed in this file for MyLib32:

```
TITLE Test the Link Library        (LibTest.asm)

; Use this program to test the link library.

INCLUDE Irvine32.inc
INCLUDE MyLib32.inc
.data
MyArray DWORD 100h,200h,300h

.code
main PROC
   INVOKE ArraySum, ADDR MyArray, LENGTHOF MyArray
   call DumpRegs
   exit
main ENDP
END main
```

When building this program, use the *make32.bat* file in the Examples\Lib32 directory. Open a command prompt, navigate to the Examples\Lib32 directory, and type the following:

```
make32 LibTest
```

If the program builds with no errors, run the executable:

```
LibTest
```

Putting MyLib32 Into Production

Once you're satisfied that *MyLib32.obj* is correct, copy it to \Masm615\LIB. Also, copy *MyLib32.inc* to \Masm615\INCLUDE.

There's one more task to complete. You need to add the name of the library to make32.bat in the \Masm615 directory. (This batch file is executed whenever **Build 32-bit MASM** is selected from the Tools menu of TextPad, JCreator, or Visual Studio.) Open the file in NotePad or any other text editor, and find the line that begins with the following:

```
LINK32 %1.obj irvine32.lib kernel32.lib . . .
```

Insert the name of our new library as follows:

```
LINK32 %1.obj MyLib32.obj irvine32.lib kernel32.lib . . .
```

After saving this file, you should be able to call MyLib32 procedures from any of your programs.

What about LIB Files?

You've probably noticed that the Irvine32 library has an extension of LIB. Creating such a file

involves running Microsoft's LIB.EXE utility.

Link libraries are designed so that only the code you're actually using is pulled from the library and attached to your programs. When building industrial-sized link libraries, source code is divided into small files and each is assembled separately into an object module. Each of the resulting object files are inserted in the library. When a subroutine is called, only the code belonging to its object module is attached to the calling program.

If you're working with a small library (such as MyLib32), there's really no need to go to the extra trouble of running the LIB utility. But if you're determined to do it, the following section shows how.

Using the Microsoft LIB Utility

There are two versions of Microsoft LIB: The 16-bit version, named Lib.exe, is located in the \Masm615 directory. The 32-bit version, also named Lib.exe, can be found in an installed copy of Microsoft Visual Studio.

Use a text editor to open the file *makeLib.bat*. You must edit the EXEPATH variable by inserting the correct path on your system to 32-bit Lib.exe in Microsoft Visual Studio. In Visual Studio 6.0, for example, the path ends with

```
. . . \Microsoft Visual Studio\VC98\BIN\LIB
```

Run *makeLib.bat* from a command prompt, passing it the base name of the object file. By default, the LIB program creates a library file having the same base name as the object file. For example, the following reads MyLib32.obj and creates MyLib32.lib:

```
makeLib MyLib32
```

Then you can copy Irvine32.lib to the \Masm615\LIB directory. Since you are using a LIB file rather than an OBJ file, the following line in \Masm615\make32.bat needs to fixed:

```
LINK32 %1.obj MyLib32.obj irvine32.lib kernel32.lib . . .
```

becomes:

```
LINK32 %1.obj MyLib32.lib irvine32.lib kernel32.lib . . .
```