Topics

1	**Introduction and Installation**	**1-3**
1.1	Development Tools Overview	1-4
1.2	Software Installation	1-6
1.2.1	Installing the Tools on IBM PC/ATs or 100% Compatible Machines With PC–DOS, MS–DOS, or OS/2	1-6
1.3	Getting Started	1-7

Figures

Fig.	Title	Page
1.1	MSP430 Assembly Language Development Flow	1-4

1 Introduction and Installation

The MSP430 devices are well supported by a full set of hardware and software development tools. This document discusses the software development tools included with the MSP430 assembly language package:

- Assembler
- Archiver
- Linker
- Absolute Lister
- ROM Utility

These tools can be installed on the following systems:

- PC/AT with PC–DOS, MS–DOS, OS/2 or MS-Windows

The MSP430 assembly language tools create and use object files that are in common object file format (COFF) to facilitate modular programming. Object files contain separate blocks (called sections) of code and data that you can load into different MSP430 memory spaces. You will be able to program the MSP430 more efficiently if you have a basic understanding of COFF.

1.1 Development Tools Overview

The figure shows the assembly language development flow. The shaded portion highlights the most common development path; the other portions are optional.

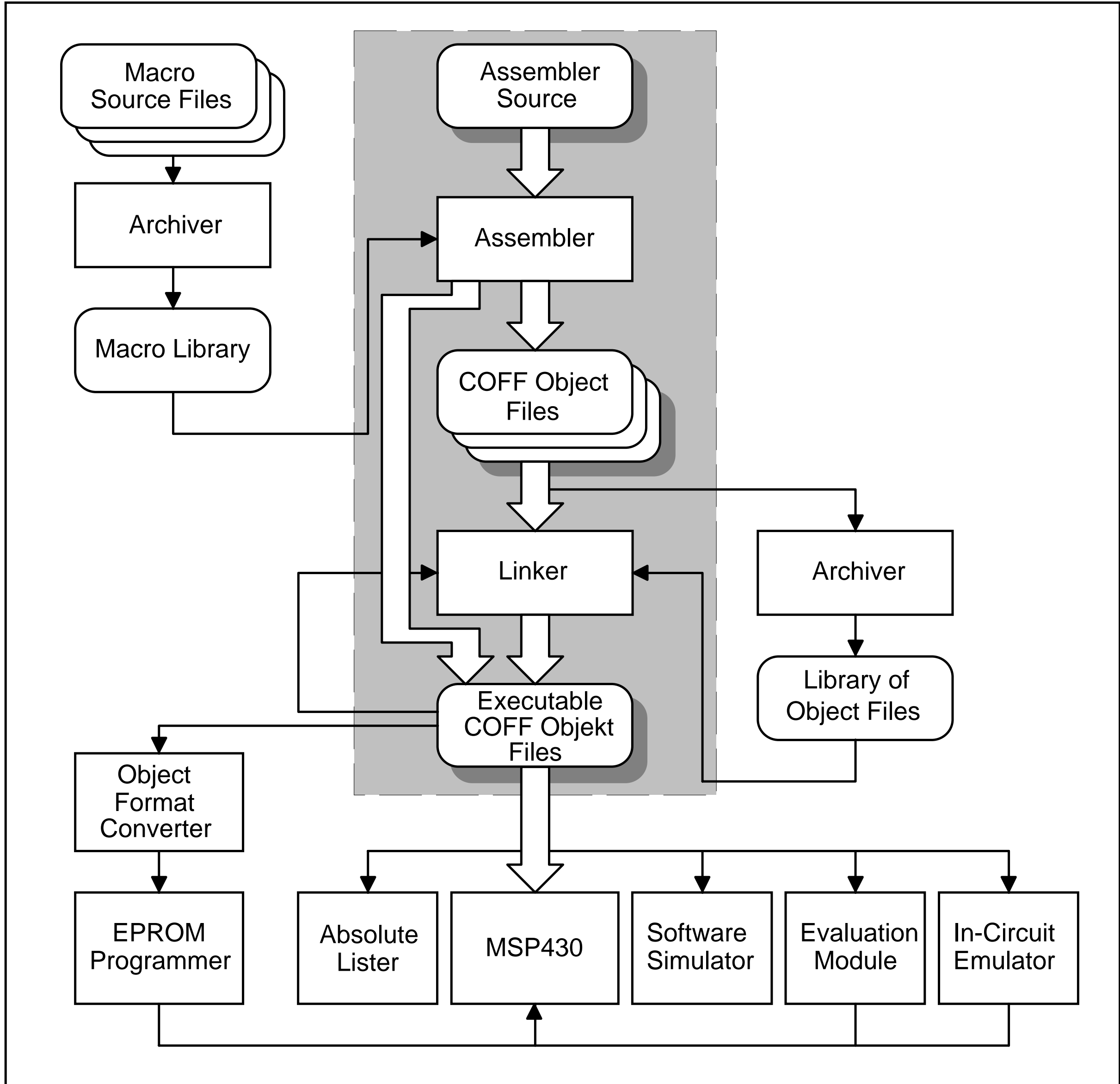

Figure 1.1: MSP430 Assembly Language Development Flow

- The **assembler** translates assembly language source files into machine language object files. Source files can contain instructions, assembler directives, and macro directives. You can use assembler directives to control various aspects of the assembly process, such as the source listing format, symbol definition, and section content.

- The **archiver** allows you to collect a group of files into a single archive file. For example, you can collect several macros together into a macro library. The assembler will search through the library and use only the members that are called as macros by the source file. You can also use the archiver to collect a group of object files into an object library. The linker will include in the library the members that resolve external references during the link.

- The **linker** combines object files into a single executable object module. As it creates the executable module, it performs relocation and resolves external references. The linker accepts relocatable COFF object files (created by the assembler) as input. It also accepts archiver library members and output modules created by a previous linker run. Linker directives allow you to combine object file sections, bind sections or symbols to addresses or within memory ranges, and define or redefine global symbols.

- The **absolute lister** provides a file that can be reassembled to produce a listing of the absolute addresses of an object file.

- The MSP430 microcontroller programmer accepts COFF files as input, but most EPROM programmers do not. The **object format converter** converts a COFF object file into TI–tagged, Intel, Motorola or Tektronix object format. The converted file can be downloaded to an EPROM programmer.

- The main purpose of this development process is to produce a module that can be executed in a system that contains a MSP430 device. You can use one of several debugging tools to refine and correct your code before downloading it to a MSP430 system.

1.2 Software Installation

This section contains instructions for installing the assembly language tools.

1.2.1 Installing the Tools on IBM PC/ATs or 100% Compatible Machines With PC–DOS, MS–DOS, OS/2 or MS-Windows

The MSP430 assembly language software package is shipped on a double–sided, high–density disk. Your system must have at least 512K bytes of memory space and 1MB of harddisk space.

First make a backup of the product disk.
Insert the backup disk into the floppy disk drive of your choice.
Change to that drive and enter:

```
INSTALL
```

Follow the instructions displayed on screen.

1.3 Getting Started

The tools you will probably use most often are the assembler and the linker. This section provides a quick walkthrough so that you can get started without reading the whole user's guide. These examples show the most common methods for invoking the assembler and linker.

1) Create two short source files to use for the walkthrough; call them `file1.asm` and `file2.asm`.

file1.asm	file2.asm

<pre>   .global   addq
start   clr R10
        clr R11
 	
loop   call #addq
        jnc loop
        .end</pre>
	<pre> .global addq
	addq inc R10
	        inc R11
	skp ret
	        .end</pre>

2) Enter the following command to assemble file1.asm.

   asm430 file1

3) The **asm430** command invokes the MSP430 assembler; `file1.asm` is the input source file.

   If the input file extension is .asm, you don't have to specify the extension; the assembler uses .asm as the default. This example creates an object file called `file1.obj`. The assembler creates an object file only if there are no errors. You can specify a name for the object file, but if you don't, the assembler will append the .obj extension to the input filename.

4) Now assemble `file2.asm`; enter:

   asm430 file2 -l

5) This time, the assembler creates an object file called `file2.obj`. The -l (lowercase "L") option tells the assembler to create a listing file; the listing file for this example is called `file2.lst`.

6) Link `file1.obj` and `file2.obj`; enter:

   lnk430 file1 file2 -o prog.out

7) The **lnk430** command invokes the linker. `file1.obj` and `file2.obj` are the input object files. (If the input file extension is .obj, you don't have to specify the extension; the linker uses .obj as the default.) The linker combines `file1.obj` and `file2.obj` to create an executable object module called `prog.out`. The -o option supplies the name of the output module.