Read This First

This preface summarizes the chapters, lists related documentation, and describes the style and symbol conventions used in this manual.

How This Manual Is Organized

This document contains the following chapters:

Chapter 1 **Introduction and Installation**
Provides an overview of the assembly language development tools, a walkthrough, and installation information.

Chapter 2 **Introduction to Common Object File Format**
Discusses the basic COFF concept of **sections** and how they can help you use the assembler and linker more efficiently. Common object file format, or COFF, is the object file format used by the MSP430 family tools. *Read Chapter before using the assembler and linker.*

Chapter 3 **Assembler Description**
Tells you how to invoke the assembler and discusses source statement format, valid constants and expressions, and assembler output.

Chapter 4 **Assembler Directives**
Divided into two parts: the first part describes the directives according to function, and the second part presents the directives in alphabetical order.

Chapter 5 **Instruction Set Summary**
Summarizes the MSP430 instruction set alphabetically.

Chapter 6 **Macro Language**
Describes macro directives, substitution symbols used as macro parameters, and how to create macros.

Chapter 7 **Archiver Description**
Contains instructions for invoking the archiver, creating new archive libraries, and modifying existing libraries.

Chapter 8 **Linker Description**
Tells you how to invoke the linker, provides details about linker operation, discusses linker directives, and presents a detailed linking example.

Chapter 9 **Absolute Lister Description**
Tells you how to invoke the absolute lister so that you can obtain a listing of the absolute addresses of an object file.

Chapter 10 **Object Format Converter Description**
Tells you how to invoke the object format converter so that you can convert a COFF object file into an Intel, Tektronix, or TI-tagged object format.

Appendix A **Common Object File Format**
Contains supplemental technical data about the internal format and structure of COFF object files.

Appendix B **Symbolic Debugging Directives**
Lists symbolic debugging directives that a high level language can use.

Appendix C **Assembler Error Messages**
Lists the assembler error messages.

Appendix D **Linker Error Messages**
Lists the linker error messages.

Appendix E **ASCII Character Set**
Provides a table of the ASCII character set.

Appendix F **Glossary**
Contains a glossary of terms and acronyms used in this book.

Appendix G **Floating Point Formats**
Contains informations about the internal format of floating point constants.

Related Documentation

The following MSP430 documents are also available.

The MSP430 Family Data Manual (literature number SPNSxxx) discusses hardware aspects of the MSP430, such as pin functions, architecture, stack operation, and interfaces, and contains the MSP430 instruction set.

The MSP430 data sheets contain the recommended operating conditions, electrical specifications, and timing characteristics of the MSP430 family devices.

- ***MSP430C201 16–Bit Microcontroller Data Sheet*** (literature number SPNSxxx)

Style and Symbol Conventions

This document uses the following conventions:

- Program listings, program examples, and interactive displays are shown in a special font. Examples use a bold version of the special font for emphasis. Here is a sample program listing:

```
1 0000  2δ      ξ    .βψτε  45
2 0001  2φ      ψ    .βψτε  47
3 0002  32      ζ    .βψτε  50
4 0003               .τεξτ
```

- In syntax descriptions, the instruction, command, or directive is in a **bold face font** and parameters are in *italics*. Portions of a syntax that are in **bold face** should be entered as shown; portions of a syntax that are in *italics* describe the type of information that should be entered. Here is an example of a directive syntax:

  .space *size*

  .space is the directive. This directive has one parameter, indicated by *size*. When you use .space, the first and only parameter must be the size.

- Square brackets (**[** and **]**) identify an optional parameter. If you use an optional parameter, you specify the information within the brackets; you don't enter the brackets themselves. This is an example of an instruction that has an optional parameter:

  .text *[address]*

- Braces ({ and }) indicate a list. The symbol **|** (read as *or*) separates items within the list. Here's an example of a list:

  { a | b | c }

  This provides three choices: a, b, or c.

- Some directives can have a varying number of parameters. For example, the .byte directive can have up to 100 parameters. The syntax for this directive is:

  .byte $value_1$ [*, ... ,* $value_n$]

  This syntax shows that .byte must have at least one value parameter, but you have the option of supplying additional value parameters, separated by commas.

Following are other symbols and abbreviations used throughout this document.

Symbol	Definition	Symbol	Definition
R0-R3	Registers with special functions	R4-R15	Working registers, general purpose
PC	Program counter register	SP	Stack pointer register
SR	Status register	CG1,CG2	Constant generator registers
LSB	Least significant bit	MSB	Most significant bit
H,h	Suffix – hexadecimal number	B,b	Suffix – binary integer
Q,q	Suffix – octal integer		
{ }	List of parameters	[ ]	Optional parameter
text	Indicates a “fill in the blank" – replace the text in *italics* with an appropriate substitute. For example, substitute an actual label for *label*; substitute an actual destination expression for *expression*.		

Trademarks

IBM, IBM PC, IBM PC/XT, PC–DOS, IBM OS/2, and PS/2 are trademarks of International Business Machines Corp.

MS, MS OS/2, MS–DOS, and MS–Windows are registered trademarks of Microsoft Corp.

Sun–3, Sun–4, SunView, SunWindows, and Sun Workstation are trademarks of Sun Microsystems, Inc.

UNIX is a registered trademark of AT&T Bell Laboratories, Inc.

VAX and VMS are trademarks of Digital Equipment Corp.