

))

**CHAPTER 6****PROGRAMS FOR THE M68HC12**

))Q

**OBJECTIVES**

This chapter will show programming techniques and suggest an assembly language programming style. Examples of programs using the CASM12 assembler are given and explained. We will also show how to write structured assembly language programs that meet the goals of top down software design.<sup>1</sup>

- 6.1 Assembly Language Programming Style
  - Source Code Style
  - To Indent or not to Indent
  - Upper and Lower Case
  - Use Equates, not Magic Numbers
  - Using Include Files
  - Using Monitor Routines
  - Commenting Style
  - Subroutine or Function Headers
- 6.2 Structured Assembly Language Programming
  - Sequence
  - IF-THEN-ELSE Decision**
  - WHILE-DO Repetition
  - DO-WHILE Repetition
- 6.3 Example Programs
- 6.4 Conclusion and Chapter Summary Points

Learning to write an effective and well organized assembly language programs takes time. The essential elements are (1) it does what it is supposed to do, (2) it is written so it can be understood by another assembly language programmer, and (3) it makes use of assembler directives and the ability of the assembler to evaluate expressions.

- ! An effective assembly language programming style can consist of the following sections:
  - ! A program header with a short description of the program.
  - ! Assembler equates defining the system information, constants and the memory map.

---

<sup>1</sup> Chapter 6 in *Microcontrollers and Microcomputers: Principles of Software and Hardware Engineering* shows how to design software using the top down design method and structured pseudocode design tools. Modular software design and the problems of module coupling are discussed also.

- ! An `ORG` directive to locate the code in ROM.
- ! A section of code to initialize the stack pointer and other variable data.
- ! The main body of the program consisting mainly of subroutine calls.
- ! The program subroutines.
- ! Definitions for constants used in the program.
- ! An `ORG` directive to locate the variable data in RAM.
- ! `DS` directives to allocate memory for variables.
- ! While indentation is used for high-level languages to show structure in the design, it is not used very often in assembly language programming.
- ! Using upper and lower case letters can make your program more readable.
- ! Avoid magic numbers in your code by using the `EQU` directive to define constants.
- ! Frequently used definitions and text can be kept in include files.
- ! Include files can be imported into your source code file to reduce the amount of typing needed.
- ! An effective commenting style makes use of pseudocode design comments.
- ! Structured assembly language programming should be used to convert your pseudocode design to a program.