Order this document
by EB306/D

Motorola Semiconductor Engineering Bulletin

EB306

Using Exercise 7 on the M68HC16Z1EVB and the Necessity of Word Alignment

By Brian Scott Crow
Austin, Texas

Introduction

The M68HC16Z1EVB is shipped with eight exercise programs which are examples of working programs that can be run on the evaluation board.

Exercise 7 is designed to exercise the general-purpose timer by using an output compare to drive three input captures. The output compare is set up to toggle each time the 16-bit counter reaches $1000. One input capture triggers on both edges, one on rising edges only, and the last on falling edges. When each of these interrupts occurs, a message is sent out the asynchronous serial port to a terminal. In addition to the output compares and the input captures, a PWM signal is set up to drive the pulse accumulator. When the pulse accumulator overflows 10 times, a system bell will ring on the terminal device.

Source code for this exercise is shipped on the QUICKSTART disk in a file called EXERC_7.ASM.

Why Executable Code Must Start on an Even Address

The CPU16 is the processor core on the MC68HC16 Family of microcontrollers. The architecture of this processor is designed for high performance systems. High performance in this context refers to systems which take advantage of the 16-bit data bus by interfacing

MOTOROLA

16-bit memories to the CPU16. These systems will perform at optimal levels as long as every 16-bit memory access returns 16 bits of valid data. In other words, misaligned memory accesses cause a penalty in system performance as measured in total execution time.

To help programmers and system designers keep instruction execution word-aligned, all CPU16 instructions are either two bytes, four bytes, or six bytes in length. In addition, the CPU16 only fetches instruction words and operands from word boundaries. Word boundaries in this context refer to addresses with ADR0 = 0.

Word-alignment of all instructions imposes several requirements on software writers. When defining the starting address of a code segment, the origination address must be an even address.

Example 1 shows an incorrect origination and a correct origination:

Example 1

```
ORG     $000201        ;WRONG! This is a byte-boundary since
                       ;ADR0 = 1.
ORG     $000200        ;CORRECT! This code will originate on a
                       ;word-aligned boundary
```

Some assemblers have directives which allow the programmer to define data storage locations or form constants within the code segment. These data space operands may be an odd number of bytes in length. This is allowed since data memory accesses can be mis-aligned.

However, if these constants or data storage locations are embedded in a code segment, the next instruction must begin on a word boundary. This can be accomplished in either of two ways.

- The MASM16 assembler provides an EVEN directive which forces the next instruction or data allocation to the next sequential even address, or a word aligned boundary.
- Another method is to count the number of bytes allocated for storage or for constants and ensure that the total is even by padding odd length data with a filler byte.

These two alignment methods are shown in **Example 2** and **Example 3**.

Example 2

```
mystring      DC.B    ³This is an odd-length string!²
         EVEN
label:        next_opcode   operands
```

Example 3

```
mystring      DC.B     ³This is an odd-length string!²,$00
label:   next_opcode   operands
```

Remember that assembler directives like EVEN must not begin in column 1, or they will simply be interpreted as labels. Any white space in column 1 is sufficient to ensure that the directive will be interpreted as a directive.

Fixing Exercise 7 for the M68HC16Z1

Exercise 7 is organized into these blocks:

- INCLUDE files
- Interrupt vector table
- Executable code segment
- Executable subroutines
- Data allocation for strings
- Interrupt service routines

The data allocation for strings resulted in an odd-length of total string declarations. Inserting this line on line 140 of the source code will allow proper assembly of the EXERC_7.ASM file:

```
EVEN
```

This forces the interrupt service routines to begin assembly on a word-aligned boundary.

Conclusion

The architects of the CPU16 optimized the design for high performance systems. These high performance systems will take advantage of the 16-bit word size by interfacing 16-bit memories to the processor. The advantages of a 16-bit architecture would be useless if accesses were allowed to be mis-aligned. To decrease the percentage of misaligned accesses, all instructions are forced to word boundaries. To add to the indivisibility of required word alignment, CPU16 architects designed only even numbered byte length instruction formats. Since all instructions are either two, four, or six bytes in length and must start on a word boundary, the number of mis-aligned accesses will be sharply reduced. This optimization certainly will keep instruction accesses from degrading system performance.

How to reach us:
USA/EUROPE/Locations Not Listed: Motorola Literature Distribution, P.O. Box 5405, Denver, Colorado 80217, 1-800-441-2447 or 1-303-675-2140. Customer Focus Center, 1-800-521-6274
JAPAN: Nippon Motorola Ltd.: SPD, Strategic Planning Office, 141, 4-32-1 Nishi-Gotanda, Shinagawa-ku, Tokyo, Japan. 03-5487-8488
ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd., 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298
Mfax™, Motorola Fax Back System: RMFAX0@email.sps.mot.com; http://sps.motorola.com/mfax/;
TOUCHTONE, 1-602-244-6609; US and Canada ONLY, 1-800-774-1848
HOME PAGE: http://motorola.com/sps/

MOTOROLA