

# Motorola Semiconductor Engineering Bulletin

---

## EB252

### MOVB, MOVW, PSHM, and PULM Syntax Differences on MC68HC16 Assemblers

By **Brian Scott Crow**  
Austin, Texas

#### Introduction

---

The architecture of the Motorola M68HC16 Family of microcontrollers contains several new instructions compared to that of the Motorola M68HC11 Family of microcontrollers. These new instructions add more powerful addressing modes which can speed execution of certain repetitive operations. Some assemblers differ on the syntax of the various instructions.

In particular, this engineering bulletin discusses two assemblers:

- MASM16
- IASM16

MASM16 is shipped with the M68HC16Z1EVB board, while IASM16 is shipped with the M68ICD16 debugger package and also all MEVB16 boards.



## General Information

---

These instructions from the M68HC16 instruction set may require different syntax when assembled with different assemblers:

- MOVB
- MOVW
- PSHM
- PULM

MOVB and MOVW are data movement instructions which more closely resemble the MOVE instructions of CPU32-based microcontrollers more so than the load-store architecture of the 8- and 16-bit machines.

The PSHM and PULM instructions move data to and from the registers and the built-in hardware stack.

## MOVB and MOVW

---

MOVB has three distinct addressing modes, as does MOVW. Just like the machines with a CPU32, a source and destination address must be specified. Either the source address, destination address, or both can be specified using extended addressing mode. One of the source or destination addresses may be specified using post-modified index addressing.

The three allowed combinations for MOVW and MOVW are shown in [Table 1](#) and [Table 2](#).

In the tables:

aa = 8-bit signed offset used for post-modified index

bb cc = 16-bit address concatenated with the EK field to form the effective 20-bit address

**Table 1. Addressing Modes for MOVB**

Addressing Mode	Opcode	Offset Address	Operand(s)
IXP to EXT	30	aa	bb cc
EXT to IXP	32	aa	bb cc
EXT to EXT	37FE	—	bb cc bb cc

**Table 2. Addressing Modes for MOVW**

Addressing Mode	Opcode	Offset Address	Operand(s)
IXP to EXT	31	aa	bb cc
EXT to IXP	33	aa	bb cc
EXT to EXT	37FF	—	bb cc bb cc

These examples should help users of the MASM16 assembler, which is currently shipped with the M68HC16Z1 EVB.

### Case I

```
MOVW 2, X, $FFFE
```

In this case, the word pointed to by the address in XK:IX (4-bit X extension concatenated with index X) will be moved to the word pointed to by EK:\$FFFE (4-bit extended extension concatenated with the extended address).

The source data pointed to by X will not be modified, and after the operation is complete, the value 2 will be added to the Index X register. The machine language will correspond to 31 02 FF FE which describes row 1 in [Table 1](#) and [Table 2](#).

## Case II

```
MOVB $FFFE, $FE, X
```

In this case, the byte pointed to by EK:\$FFFE will be moved to the byte pointed to by EK:XK. The source data will not be modified, and after the operation, \$FE will be added to the X register.

**NOTE:** *The offset is an 8-bit signed two's complement number, so an offset of \$FE corresponds to subtracting 2 from the index X register after the move operation is complete. If addition, if the offset causes index X to overflow, then the XK field will be incremented or decremented as necessary.*

## Case III

```
MOVW $0000, $FFFE
```

In this final case, both address operands are specified using extended addressing mode. The word pointed to by EK:\$0000 will be moved to the word pointed to by EK:\$FFFE. The source data will not be modified, and index X will also not be modified.

**NOTE:** *Notice that the syntax schemes for MOVW and MOVW are the same. the only difference is the width of the data that is moved.*

IASM16, which is another assembler on the market for the MC68HC16 Family, uses this syntax to achieve the same results as each of the cases here:

```
MOVW          X( 2 ), $FFFE
MOVB          $FFFE, X( $FE )
MOVW          $0000, $FFFE
```

This syntax difference is confusing; however, careful attention to the opcodes generated by the assembler will show how each possibility of syntax translates to object code. Once the object code generated for a particular syntax can be predicted, proceed to use these instructions as originally intended for the application.

## PSHM and PULM

These two instructions are used to transfer data from hardware registers to the built-in system stack and vice versa. They use an 8-bit mask operand to specify any combination of the D, E, X, Y, Z, K (extension register) and CCR (condition code register) registers. The mask is formed by putting a logic 1 in the bit which corresponds to the register being addressed.

### *For PSHM*

#### Mask bits

- 0 = accumulator D
- 1 = accumulator E
- 2 = index register X
- 3 = index register Y
- 4 = index register Z
- 5 = extension register
- 6 = condition code register
- 7 = Motorola reserves this location for future use.

### *For PULM*

#### Mask bits

- 0 = condition code register
- 1 = extension register
- 2 = index register Z
- 3 = index register Y
- 4 = index register X
- 5 = accumulator E
- 6 = accumulator D
- 7 = Motorola reserves this location for future use.

The MASM16 assembler has eliminated the need to memorize this list by using a simpler, more intuitive syntax for PSHM and PULM. The next two examples show how to use these instructions in MASM16.

Case I                      PSHM                      D , X , K , CCR

This corresponds to the opcode 34 65. The pushes occur in the order that bits are set in the mask starting from bit 0 through bit 7, so that D is pushed, X is pushed, K is pushed, and finally the CCR is pushed onto the stack.

Case II                      PULM                      CCR , Y , X

This corresponds to the opcode 35 19. The pulls also occur in the order that bits are set in the mask starting from bit 0 through bit 7. This is why the mask bits are reversed in order.

**NOTE:** *If a PSHM and then a PULM are called with the same register list, the registers will be pushed, then pulled back into their same register.*

IASM16 uses this syntax to achieve the same results as cases I and II:

```
PSHM $65  
PULM $19
```

The programmer must refer to the mask bits list and insert the 8-bit mask which corresponds to a logic 1 in every bit that requires pushing or pulling.

This syntax requires some labor from the programmer; however, it can be achieved clearly by using the previous mask bit lists.

**NOTE:** *If a PSHM and a PULM are called with the same 8-bit mask, the registers will not be pulled into the same register they were pushed from.*

For example, PSHM \$01 followed by PULM \$01 will push accumulator D onto the stack and will be pulled back into the condition code register. This is not usually desirable; however, it can be used as an inefficient method to transfer information from one register to another.

## Conclusion

---

MASM16 and IASM16 use different syntax for certain instructions. This is certainly evident in the study of MOVB, MOVW, PSHM, and PULM.


Care must be taken to use the correct form of the instruction, depending on which assembler will be used.

The most confusing part of this difference can be found with users of the M68HC16Z1 EVB. MASM16 is shipped with the board as the assembler of choice; however, the EVB16 software, which is used to communicate with the board, was written by the authors of IASM16.

The ramifications of this are:

1. When writing code to be assembled with MASM16, one syntax must be used, but
2. The one-line assembler and disassembler contained in the EVB16 software follows the IASM16 syntax.

When using any of these instruction, take care to ensure that what you want is what you get. This can be quickly accomplished by comparing the opcodes generated in the listing file with the expected opcodes as shown in the *CPU16 Reference Manual*. This manual can be obtained from Motorola's Literature Distribution Center by referencing document number CPU16RM/AD.

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

#### 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:** Motorola Japan 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., Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, New Territories, 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/>

Mfax is a trademark of Motorola, Inc.



**MOTOROLA**

© Motorola, Inc., 1999

EB252/D