H8SX Series

Enhanced Addressing Mode (for Arrays)

Introduction

As well as having an architecture that is upward-compatible with each CPU of the H8/300, H8/300H, and H8S series, so as to inherit a full complement of peripheral functions, the H8SX microcomputer series has a maximum operating frequency of 50 MHz and uses a 32-bit H8SX core CPU as well as an on-chip multiplier/divider to improve performance.

This H8SX series Application Note provides information you may be need during software and hardware design. This is a basic edition that provides operation examples that each use a single H8SX series on-chip peripheral function.

Although the operation of each program, circuit, and other aspects covered by this application note has been checked, make sure that you conduct your own operation checks before actually using the H8SX series.

Contents

1. Overview

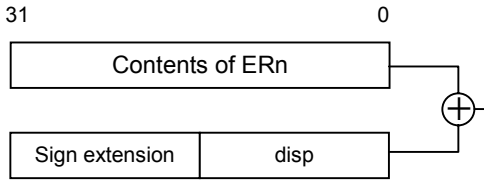
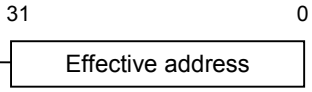
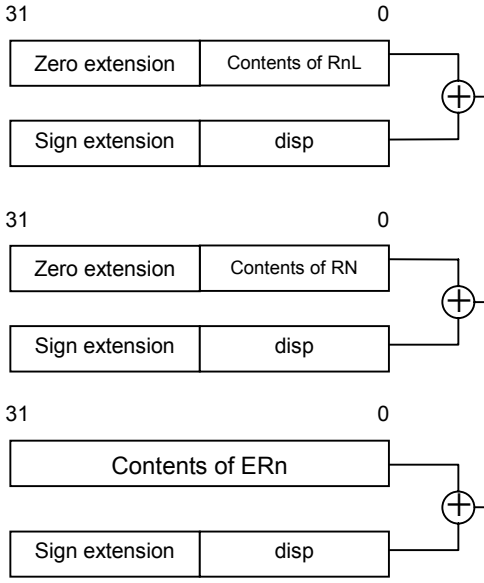
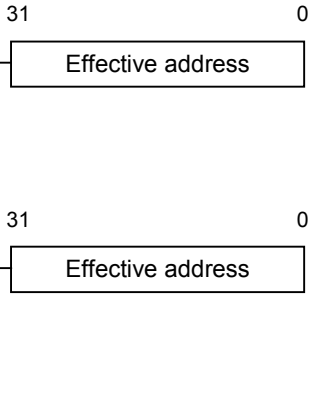
The H8SX series has an architecture that is upward-compatible with each CPU of the H8/300, H8/300H, and H8S series. Furthermore, in addition its instruction set has been enhanced to improve CPU performance. This enhancement of the instruction set has greatly improved code efficiency relative to the conventional series. This code efficiency leads to benefits such as a reduction in the ROM capacity required for storing programs and a reduction in each instruction fetch cycle. This application note describes the index register indirect with displacement, which is a new addressing mode that has been added as an enhanced instruction set item and which is particularly effective for array data processing.

2. Configuration

As an addressing mode similar to the index register indirect addressing mode with displacement, the H8SX series supports the register indirect addressing mode with displacement. The conventional H8S series also supports the register indirect addressing mode with displacement. The index register indirect addressing mode with displacement is a new addressing mode that is newly supported by the H8SX series. The following describes the difference between these two addressing modes. In the register indirect addressing mode with displacement, the value obtained by adding the displacement to the contents of the specified register (ERn only) is used as the effective address. In the index register indirect addressing mode with displacement, any of 8-, 16-, 32-bit registers (that is, RnL, Rn, and ERn) can be specified and the value obtained by adding the displacement to the contents of the specified registers that are zero-extended to 32 bits is used as the effective address. For this reason, the latter addressing mode allows more flexible addressing than the former and is better suited to a wider range of applications. Table 1 lists methods for calculating the effective address in each addressing mode. Figure 1 shows an example of accessing array data.

In the following description, a typical sort program that accesses array data is used as a sample to compare the H8SX series with the H8S series. The sample program is written in C. The results of comparing the aspects shown below are listed: Code generated by a compiler (assembler code), the instruction code length, and other items in the generated code.

Table 1 Method of Calculating the Effective Address in Each Addressing Mode

Addressing mode	Method of calculating the effective address	Effective address
Register indirect with displacement @(disp:16,ERn)		
Index register indirect with displacement @(disp:16,RnL.B) @(disp:16,Rn.W) @(disp:16,ERn.L)		

For a displacement of 16 bits

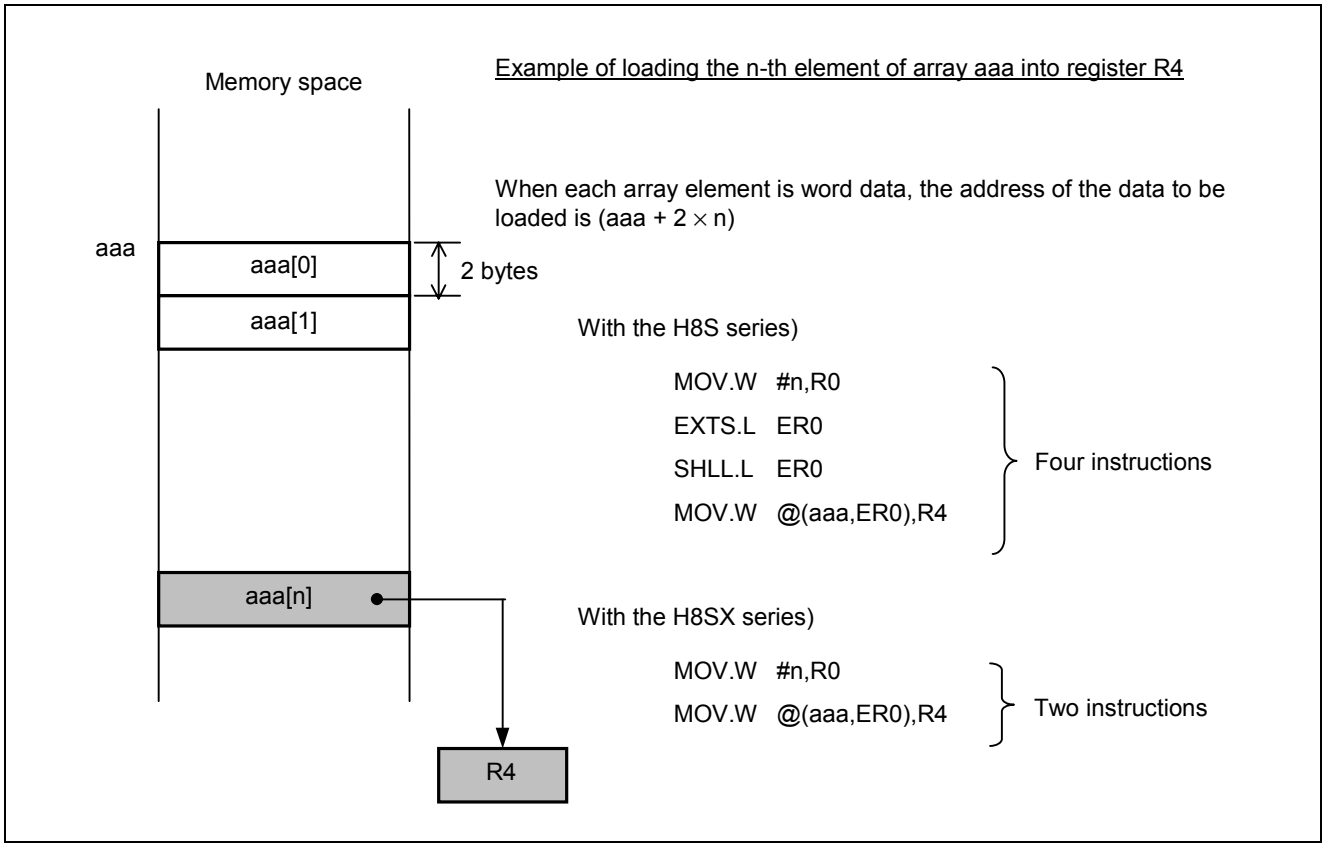
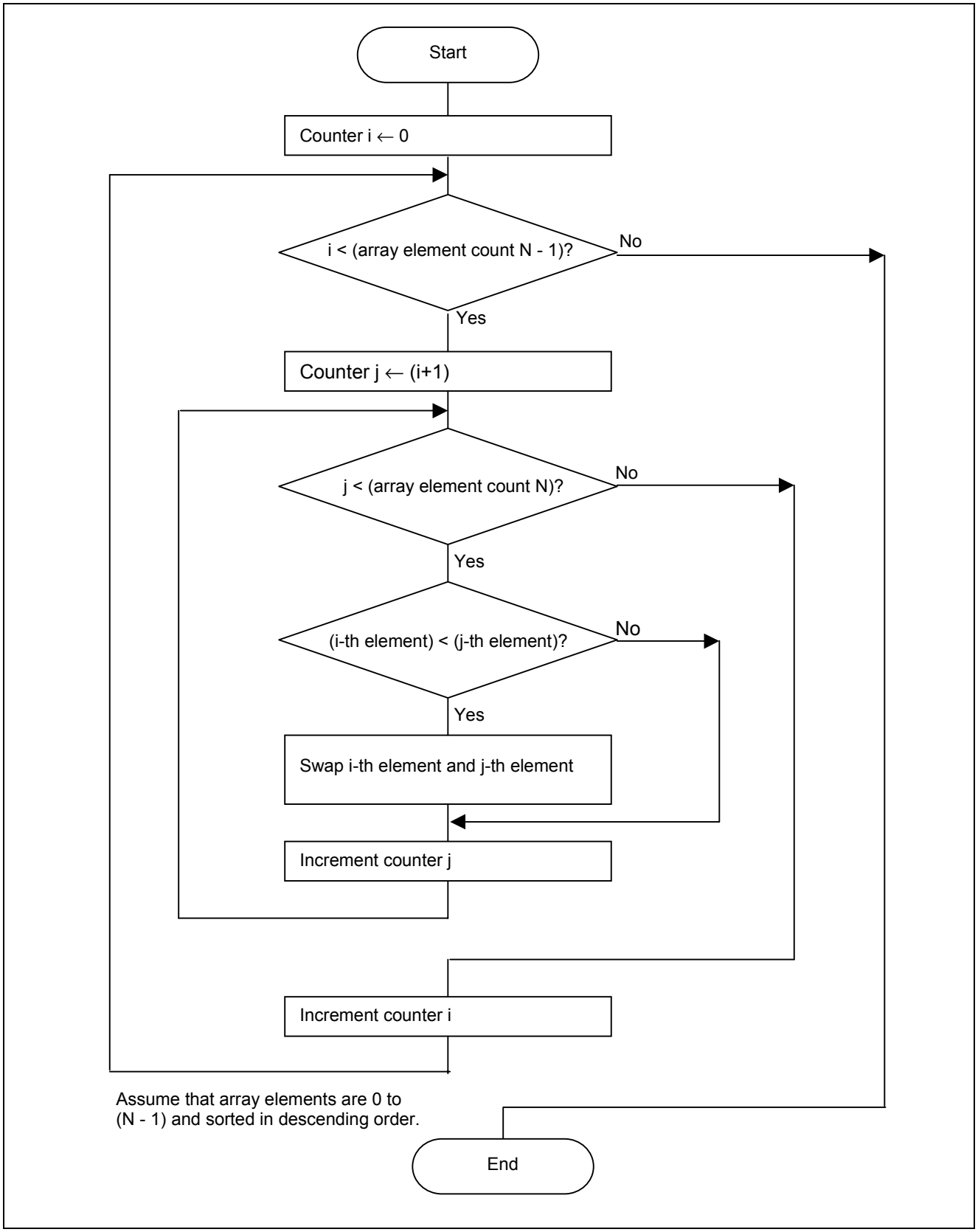


Figure 1 Example of Accessing Array Data

3. Sample Program

3.1 Flowchart

The sample program shown below is a very simple sort program, that will allow you to understand the descriptions of the index register indirect addressing mode with displacement, an enhanced instruction set item. For comparison with the H8S series, the results of compilation are shown. These results are only intended to be an example, because the instruction code length generated in the compilation of an application-level program greatly depends on the source program and the compile conditions. The flowchart that is shown below is for a sample program that performs a bubble sort.


Start
Counter i ← 0
i < (array element count N - 1)?
No
Yes
Counter j ← (i+1)
j < (array element count N)?
No
Yes
(i-th element) < (j-th element)?
No
Yes
Swap i-th element and j-th element
Increment counter j
Increment counter i
Assume that array elements are 0 to
(N - 1) and sorted in descending order.
End

3.2 Program Listing

A source program that is written in C is shown below.

```
/************************************************************/
/* include file                                             */
/************************************************************/
#include <machine.h>

/************************************************************/
/* function prototype                                       */
/************************************************************/
void bubble_sort( void );

/************************************************************/
/* array variable                                           */
/************************************************************/
#define N  100
signed short wArray[N];              // Assume that data to be
                                     // sorted is set here.

/************************************************************/
/* function definition                                      */
/************************************************************/
void  bubble_sort( void )
{
   signed char  i, j;
   signed short temp;

   for(i=0; i<(N-1); i++)
   {
       for(j=(i+1); j<N; j++)
       {
           if(wArray[i]<wArray[j])
           {                         // Compares and swaps array elements.
               temp      = wArray[i];
               wArray[i] = wArray[j];
               wArray[j] = temp;
           }
       }
   }
}
```

3.3 Comparison of the H8S Series with the H8SX Series

The result of compilation (assembly code) with the H8S series is shown below.

```
SCT OFFSET LABEL     INSTRUCTION OPERAND    COMMENT

P                                           ; section
00000000 _bubble_sort:                    ; function
  00000000     STM.L       (ER4-ER6),@-SP
  00000004     MOV.L       #_wArray,ER6
  0000000A     SUB.B       R4H,R4H
  0000000C L71:
  0000000C     MOV.B       R4H,R4L
  0000000E     ADD.B       #1,R4L
  00000010     MOV.B       R4H,R1L
  00000012     EXTS.W      R1
  00000014     EXTS.L      ER1
  00000016     SHLL.L      ER1
  00000018     ADD.L       ER6,ER1
  0000001A     BRA         L73:8
  0000001C L72:
  0000001C     MOV.B       R4L,R5L
  0000001E     EXTS.W      R5
  00000020     EXTS.L      ER5
  00000022     SHLL.L      ER5
  00000024     ADD.L       ER6,ER5
  00000026     MOV.W       @ER1,R0
  00000028     MOV.W       @ER5,E0
  0000002A     CMP.W       E0,R0
  0000002C     BGE         L74:8
  0000002E     MOV.W       R0,E0
  00000030     MOV.W       @ER5,R0
  00000032     MOV.W       R0,@ER1
  00000034     MOV.W       E0,@ER5
  00000036 L74:
  00000036     INC.B       R4L
  00000038 L73:
  00000038     CMP.B       #100,R4L
  0000003A     BLT         L72:8
  0000003C     INC.B       R4H
  0000003E     CMP.B       #99,R4H
  00000040     BLT         L71:8
  00000042     LDM.L       @SP+,(ER4-ER6)
  00000046     RTS
B                                           ; section
  00000000 _wArray:                         ; static: wArray
  00000000     .RES.W      100
```

The result of compilation (assembly code) with the H8SX series is shown below.

```
SCT OFFSET LABEL     INSTRUCTION OPERAND    COMMENT

P                                           ; section
  00000000 _bubble_sort:                    ; function
  00000000     PUSH.L      ER2
```

```
   00000004     BRA/S       L8:8
   00000006     MOV.B       #0:8,R0H
   00000008 L9:
   00000008     MOV.B       R0H,R0L
   0000000A     INC.B       R0L
   0000000C     MOV.B       R0H,R1L
   0000000E     EXTS.L      #2,ER1
   00000010     BRA/S       L10:8
   00000012     MOV.L       ER1,ER2
   00000014 L11:
   00000014     MOV.W       @(_wArray:32,ER2.L),R1
   0000001C     MOV.W       @(_wArray:32,R0L.B),E0
   00000024     CMP.W       E0,R1
   00000026     BGE         L13:8
   00000028     MOV.W       E0,@(_wArray:32,ER2.L)
   00000030     MOV.W       R1,@(_wArray:32,R0L.B)
   00000038 L13:
   00000038     INC.B       R0L
   0000003A L10:
   0000003A     CMP.B       #100:8,R0L
   0000003C     BLT         L11:8
   0000003E     INC.B       R0H
   00000040 L8:
   00000040     CMP.B       #99:8,R0H
   00000042     BLT         L9:8
   00000044     RTS/L       ER2
 B                                          ; section
   00000000 _wArray:                        ; static: wArray
   00000000     .RES.W      100
```

The following source code section performs comparison and swapping of array elements.

```
        if(wArray[i]<wArray[j])
    {                       // Compares and swaps array elements.
        temp      = wArray[i];
        wArray[i] = wArray[j];
        wArray[j] = temp;
    }
```

Table 2 lists the results of comparison of the H8S series and H8SX series about the assembly code in this section.

Table 2 Comparing the Results of Compilation

CPU	Assemble code		Instruction length		Execution time	
			In bytes	Total	In states	Total
H8S series	L72:					
	MOV.B	R4L,R5L	2		1	
	EXTS.W	R5	2		1	
	EXTS.L	ER5	2		1	
	SHLL.L	ER5	2		1	
	ADD.L	ER6,ER5	2	26	1	18
	MOV.W	@ER1,R0	2		2	
	MOV.W	@ER5,E0	2		2	
	CMP.W	E0,R0	2		1	
	BGE	L74:8	2		2	
	MOV.W	R0,E0	2		1	
	MOV.W	@ER5,R0	2		1	
	MOV.W	R0,@ER1	2		2	
	MOV.W	E0,@ER5	2		2	
H8SX series	MOV.L	ER1,ER2	2		1	
	L11:					
	MOV.W	@(_wArray:32,ER2.L),R1	8		3	
	MOV.W	@(_wArray:32,R0L.B),E0	8	38	3	16
	CMP.W	E0,R1	2		1	
	BGE	L13:8	2		2	
	MOV.W	E0,@(_wArray:32,ER2.L)	8		3	
	MOV.W	R1,@(_wArray:32,R0L.B)	8		3	

Revision Record

		Description	
Rev.	Date	Page	Summary
1.00	Sept.19.03	—	First edition issued

Keep safety first in your circuit designs!

1. Renesas Technology Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation or a third party.
2. Renesas Technology Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors.
   Renesas Technology Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.