

APPLICATION NOTE

AN709

Reversing bits within a data byte on the XA

1996 Dec 09

Reversing bits within a data byte on the XA

AN709

Author: Greg Goodhue, Microcontroller Product Planning, Philips Semiconductors, Sunnyvale, CA

Implementing an algorithm to reverse the bits within a data byte is notorious for producing inefficient code on most processors. This function can serve as a case study of how to trade off code size for performance and shows some of the methods that might be employed in similar types of data conversion situations.

Here four solutions are shown to implement the byte reverse function. The first version (Listing 1) uses a very simple approach. The result is produced by shifting a bit out of the initial data value and shifting the same bit back into the result value. This is repeated in a loop for each bit. Since the two shift operations are done in opposite directions, the result value is a bit reversal of the initial value. This version is the smallest in size, using only 11 bytes. However, it takes 128 XA clock periods to complete.

Listing 2 uses the same method as the first, but “unfolds” the loop to eliminate the counting and branching overhead. What is left are the instructions from the inside of the loop repeated eight times.

Unfolding the loop gives faster execution, 64 clocks in this case. The code size grows somewhat to 16 bytes.

The third method (Listing 3) uses a partial lookup table to reverse one nibble at a time and assemble the complete byte from two lookup values. In a reversed byte, the upper nibble of the result consists of the reversed bits of the lower nibble of the initial value,

while the lower nibble of the result consists of the reversed bits of the upper nibble of the initial value. The code example uses each nibble of the initial value as an index into the lookup table, which provides a nibble of result data. The two partial results are then combined to produce the complete result. This version uses 42 bytes for both the code and the lookup table, but requires only 42 XA clock periods to complete.

The final method shown (Listing 4) uses a full lookup table to produce the entire result very quickly. The initial data byte is used as an index into the lookup table and the value from the table is the complete result byte. This method produces the result in only 12 XA clocks. However, the code plus the lookup table occupies a fairly large amount of code space: 264 bytes.

CONCLUSION

These examples show how code size may often be traded for execution speed, or execution speed for code size, depending on an application's requirements. This is summarized in Figure 1. Other solutions to this particular algorithm are certainly possible and other algorithms will likely have different types of solutions with different resulting tradeoffs.

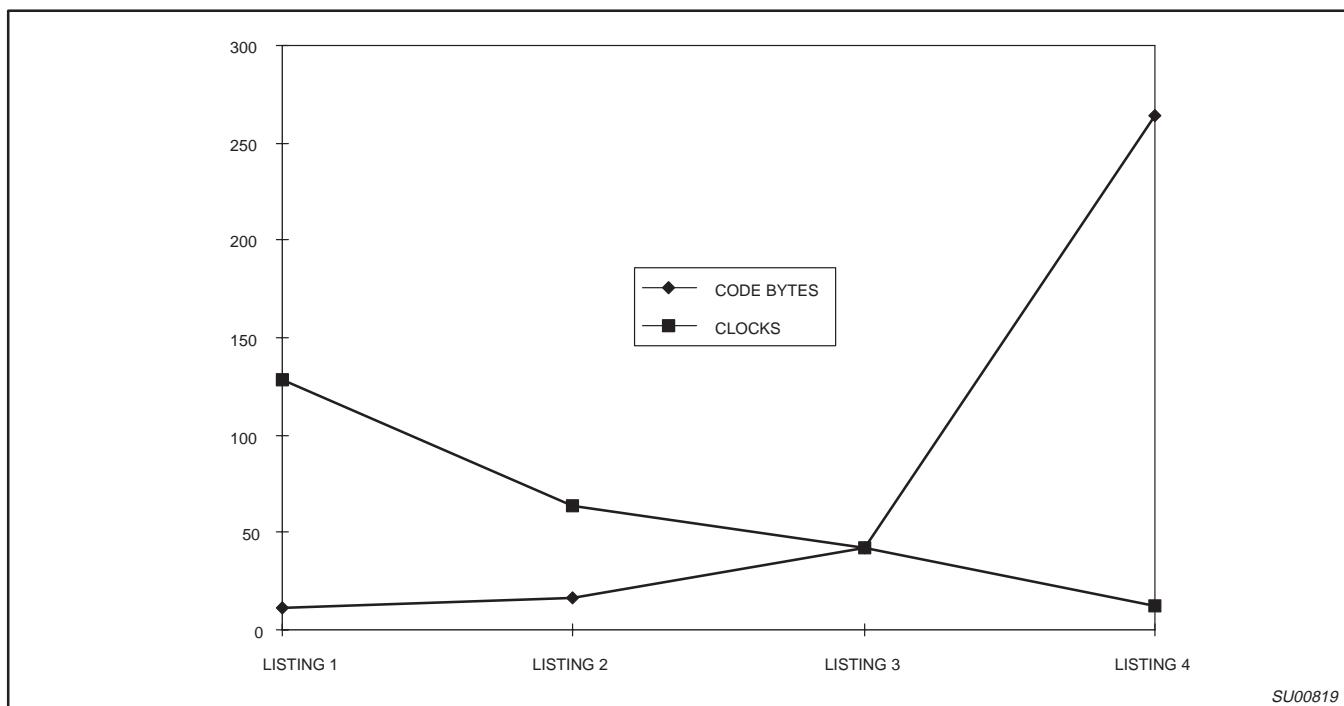


Figure 1. Tradeoff of code size to performance.

Reversing bits within a data byte on the XA

AN709

LISTING 1

```
; Listing 1) Smallest solution in terms of code space:  
;  
; Enter with value to be reversed in R0L, result in R0H.  
;  
; This works by shifting the register out in one direction and back in  
;  
; the other.  
  
    mov      count,#8          ; 3 clks  
loop:   rrc      r0l,#1        ; 4 clks  
        rlc      r0h,#1        ; 4 clks  
        djnz    count,loop     ; 8/5 clks  
  
; total time = 8*(4+4) + 7*8 + 3+5 = 128 clocks
```

LISTING 2

```
; Listing 2) Solution 1 with the loop "unfolded".  
;  
; Enter with value to be reversed in R0L, result in R0H.  
;  
; This works by shifting the register out in one direction and back in  
;  
; the other.  
  
    rrc      r0l,#1        ; 4 clks  
    rlc      r0h,#1        ; 4 clks  
    rrc      r0l,#1        ; 4 clks  
    rlc      r0h,#1        ; 4 clks  
    rrc      r0l,#1        ; 4 clks  
    rlc      r0h,#1        ; 4 clks  
    rrc      r0l,#1        ; 4 clks  
    rlc      r0h,#1        ; 4 clks  
    rrc      r0l,#1        ; 4 clks  
    rlc      r0h,#1        ; 4 clks  
    rrc      r0l,#1        ; 4 clks  
    rlc      r0h,#1        ; 4 clks  
    rrc      r0l,#1        ; 4 clks  
    rlc      r0h,#1        ; 4 clks  
    rrc      r0l,#1        ; 4 clks  
    rlc      r0h,#1        ; 4 clks  
    rrc      r0l,#1        ; 4 clks  
    rlc      r0h,#1        ; 4 clks  
    rrc      r0l,#1        ; 4 clks  
    rlc      r0h,#1        ; 4 clks  
  
; total time = 8*(4+4) = 64 clocks
```

Reversing bits within a data byte on the XA

AN709

LISTING 3

```
; Listing 3) Fastest solution (without using a 256 byte lookup table):
;   Enter with value to reverse in R4L, result returned in R0L.
;   This works by reversing each nibble using a look up table and reversing
;   the two nibbles separately as part of the procedure.

    mov      r6,#LUT1           ; 3 clks
    push     r41                ; 3 clks
    and     r41,#$0f            ; 3 clks
    movc    a,[a+dptr]          ; 6 clks
    mov     r0l,r41             ; 3 clks
    rr     r0l,#4               ; 4 clks
    pop     r41                ; 4 clks
    and     r41,#$f0            ; 3 clks
    rr     r41,#4               ; 4 clks
    movc    a,[a+dptr]          ; 6 clks
    or     r0l,r41              ; 3 clks

    .
    .
    .
    .

; this is a nibble reverse lookup table:
LUT1:   db      $00           ; 0000 => 0000
        db      $08           ; 0001 => 1000
        db      $04           ; 0010 => 0100
        db      $0C           ; 0011 => 1100
        db      $02           ; 0100 => 0010
        db      $0A           ; 0101 => 1010
        db      $06           ; 0110 => 0110
        db      $0E           ; 0111 => 1110
        db      $01           ; 1000 => 0001
        db      $09           ; 1001 => 1001
        db      $05           ; 1010 => 0101
        db      $0D           ; 1011 => 1101
        db      $03           ; 1100 => 0011
        db      $0B           ; 1101 => 1011
        db      $07           ; 1110 => 0111
        db      $0F           ; 1111 => 1111

; total time = 42 clocks
```

LISTING 4

```
; Listing 4) Fastest solution (using a 256 byte lookup table):
;   Enter with value to reverse in R4L, result returned in R0L.

    mov      r6,#LUT2           ; 3 clks
    movc    a,[a+dptr]          ; 6 clks
    mov     r0l,r41             ; 3 clks

    .
    .
    .

; this is a byte reverse lookup table:
LUT2:   db      $00           ; 00000000 => 00000000
        db      $80           ; 00000001 => 10000000
        db      $40           ; 00000010 => 01000000
        db      $C0           ; 00000011 => 11000000

    .
    .
    .

; total = 12 clocks
```

Reversing bits within a data byte on the XA

AN709

NOTES

Reversing bits within a data byte on the XA

AN709

Philips Semiconductors and Philips Electronics North America Corporation reserve the right to make changes, without notice, in the products, including circuits, standard cells, and/or software, described or contained herein in order to improve design and/or performance. Philips Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified. Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

LIFE SUPPORT APPLICATIONS

Philips Semiconductors and Philips Electronics North America Corporation Products are not designed for use in life support appliances, devices, or systems where malfunction of a Philips Semiconductors and Philips Electronics North America Corporation Product can reasonably be expected to result in a personal injury. Philips Semiconductors and Philips Electronics North America Corporation customers using or selling Philips Semiconductors and Philips Electronics North America Corporation Products for use in such applications do so at their own risk and agree to fully indemnify Philips Semiconductors and Philips Electronics North America Corporation for any damages resulting from such improper use or sale.

Philips Semiconductors
811 East Arques Avenue
P.O. Box 3409
Sunnyvale, California 94088-3409
Telephone 800-234-7381

Philips Semiconductors and Philips Electronics North America Corporation register eligible circuits under the Semiconductor Chip Protection Act.
© Copyright Philips Electronics North America Corporation 1996
All rights reserved. Printed in U.S.A.

Let's make things better.

Philips
Semiconductors



PHILIPS