Software SPI Implementation on EZ-USB

Author: **Scott Cordner**
Cypress Semiconductor
ott@cypress.com

Introduction

This document demonstrates how to implement software SPI (Serial Peripheral Interface) on EZ-USB. This demonstration uses the EZ-USB as the master, and transfers data to and from a 25LC320 EEPROM. The sample programs include:

- spiwrite.a51 - assembly routine to write one spi byte
- spiread.a51 - assembly routing to read one spi byte
- spi.c - C program that calls these routines to read and write data to a 25LC320 EEPROM

Write Routine Listing

```
 1  NAME       spiwritebyte
 2  ;
 3  ;   spiwrite.a51   4-19-00 ott
 4  ;   This routine takes a byte variable and
 5  ;   shifts it out with the clock
 6  ;   worst case bit rate of 250kHz
 7  ;   byte rate of 35 kHz
 8  ;
 9
10  ?PR?SPIWRITEBYTE?MODULE    segment code
11  ?DT?SPIWRITEBYTE?MODULE    segment data overlayable
12
13  PUBLIC     _spiwritebyte, ?_spiwritebyte?BYTE
14
15  rseg       ?DT?SPIWRITEBYTE?MODULE
16  ?_spiwritebyte?BYTE:
17  d: ds 1
18
19  rseg       ?PR?SPIWRITEBYTE?MODULE
20
21  CLKHIGH equ 00000010B      ;Bitmask to turn clk pin high
22  CLKLOW equ 11111101B       ;Bitmask to turn clk pin low
23  BITHIGH equ 00000100B      ;Bitmask to turn out pin high
24  BITLOW equ 11111011B       ;Bitmask to turn out pin low
25
26  OUTC            XDATA        7F98H
27
28  _spiwritebyte:
29
30      mov DPTR,#OUTC         ;point to outc
31      mov R6, #8             ;set up loop
32  loop:
33      mov A, R7              ;move data to send to A
34      rlc A                  ;rotate left through carry
35      mov R7, A              ;save rotated
36      movx A, @DPTR          ;setup to change bit
37      jc highbit             ;if bit is high jump
38      anl A, #BITLOW         ;else set bit low
39      sjmp skip              ;skip setting bit high
40  highbit:
41      orl A, #BITHIGH        ;set out high and clock
42  skip:
43      orl A, #CLKHIGH        ;set clock bit high
44      movx @DPTR, A          ;output data
```

```
45  ;   nop                 ;may need this to stretch clock high time
46      anl A, #CLKLOW      ;set clock low
47      movx @DPTR, A       ;output low clock
48      djnz R6, loop       ;repeat eight times
49      ret
50      end
```

The write routine takes a byte that was passed to it and shifts it out MSB first. Data is changed when the CLK is high, the device latches the data on the falling edge of CLK. The routine uses Port C pins C1, C2, but you can change these to be any pins by changing the bitmasks. This routine clocks data at a 250kHz bit rate, or about 35kHz byte rate.

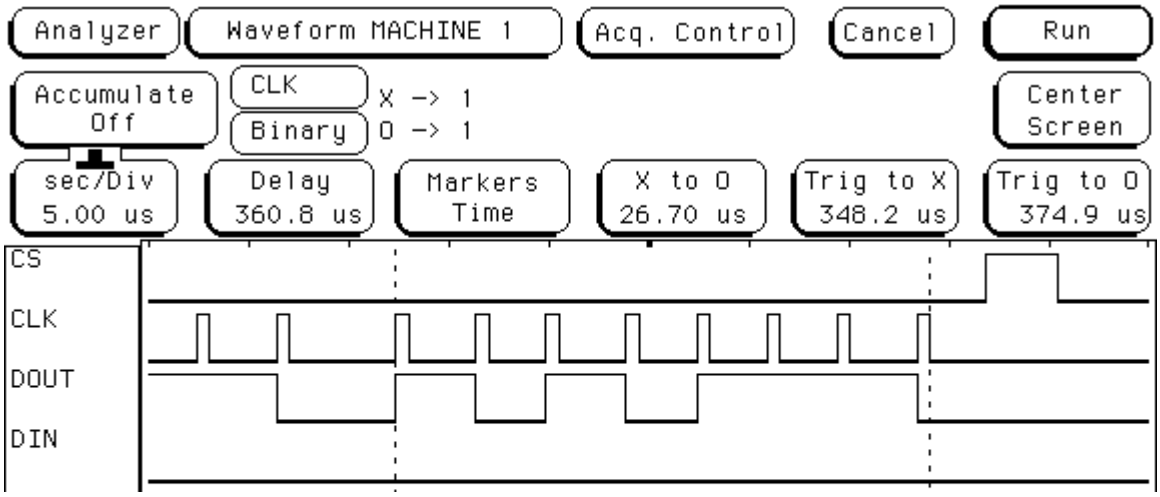

Figure 1. Typical Byte Write Sequence

Read Routine Listing

```
 1  NAME        spireadbyte
 2  ;
 3  ;   spiread.a51    4-19-00 ott
 4  ;   This routine shifts in a byte
 5  ;   with the clock and returns it
 6  ;   worst case bit rate of 222kHz
 7  ;   byte rate of 29 kHz
 8  ;
 9
10  ?PR?SPIREADBYTE?MODULE     segment code
11
12  PUBLIC      spireadbyte
13
14  CLKHIGH equ 00000010B      ;Bitmask to turn clk pin high
15  CLKLOW equ 11111101B       ;Bitmask to turn clk pin low
16  BITMASK equ 00001000B      ;Bitmask to read input pin
17
18  OUTC        XDATA          7F98H
19  PINSC       XDATA          7F9BH
20  DPS         DATA           86H
21
22  rseg        ?PR?SPIREADBYTE?MODULE
23  spireadbyte:
24
25      mov DPTR,#OUTC         ;point to outc
26      movx A,@DPTR           ;read outc
27      mov R4,A               ;save outc
28      orl A,#CLKHIGH         ;set clock high byte
29      mov R5,A               ;save contents of clkhigh
30      mov A,R4               ;read outc
31      anl A,#CLKLOW          ;set clock low byte
```

```
32      mov R4,A              ;save contents of clklow
33      inc DPS               ;switch pointer 1
34      mov DPTR,#PINSC       ;point second pointer to pinsc
35      inc DPS               ;switch pointer 0
36      mov R6,#8             ;set up loop
37  loop:
38      mov A, R5             ;get contents of clkhigh
39      movx @DPTR, A         ;output data
40      inc DPS               ;switch pointer 1
41      movx A,@DPTR          ;read in port c
42      anl A,#BITMASK        ;mask off input pin
43      jnz highbit           ;if result is not zero
44      clr c                 ;clear carry
45      sjmp skip             ;skip carry set
46  highbit:
47      setb c                ;set carry
48  skip:
49      inc DPS               ;switch pointer 0
50      mov A,R4              ;get contents of clklow
51      movx @DPTR,A          ;output low clock
52      mov A,R7              ;move byte into A
53      rlc A                 ;rotate bit in
54      mov R7,A              ;save new byte
55      djnz R6,loop          ;repeat eight times
56      ret
57      end
```

The read routine reads a byte MSB first and returns it. Data is read when the CLK is high, the device changes the data on the falling edge of CLK. The routine uses Port C pins C1, C3, but you can change these to be any pins by changing the bitmasks. This routine clocks data at a 220kHz bit rate, or about 30kHz byte rate.

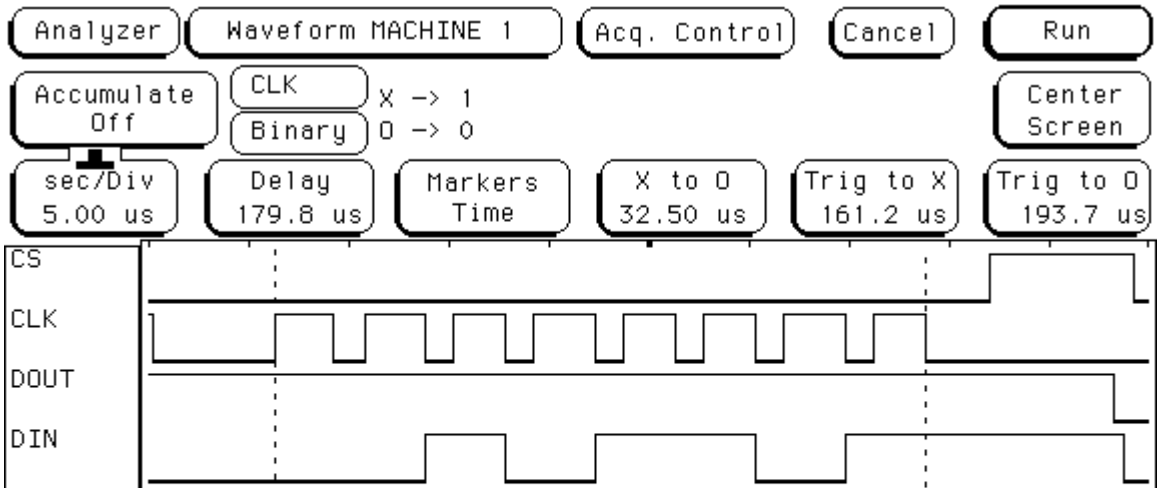

Figure 2. Typical Byte Read Sequence

C Program Listing

```
 1  /*****************************************************************************
 2
 3      spi.c   4-19-00 ott
 4      Used to test software spi functionality on EZ-USB
 5      This program will do a constant write and readback loop to an
 6      25C320 spi EEPROM with EZ-USB as the master
 7      Pins used:C0 - Chip select
 8                            C1 - Clk
 9                            C2 - Data OUT from EZ
10                            C3 - Data In to EZ
```

```
11      This pins can be changed if you change the bitmasks defines here and in
12      spiwrite.a51 and spiread.a51
13
14  ********************************************************************************/
15
16  #include <ezusb.h>
17  #include <ezregs.h>
18
19  #define CSHIGH 1
20  #define CSLOW 0xFE
21  #define CLKLOW 0xFD
22  #define INLOW 0xFB
23  #define READ_CMD 3
24  #define WRITE_CMD 2
25  #define WRITE_ENABLE 6
26  #define WRITE_DISABLE 4
27  #define READ_STATUS 5
28  #define WRITE_STATUS 1
29
30  ////////////////////////////////////// Prototypes
31  void write_25LC320 (int a, BYTE d);
32  BYTE read_25LC320 (int a);
33  void enable_25LC320 (void);
34  BYTE status_25LC320 (void);
35  void spiwritebyte (BYTE d);//Assembly routine
36  BYTE spireadbyte (void);//Assembly routine
37
38  main()
39  {
40      BYTE d;
41      int a;
42      BYTE t,x;
43
44      PORTCCFG &= 0xF0;//Turn off special functions
45      OEC |= 0x07;//C0 = CS*, C1 = CLK, C2 = OUT from EZ, C3 = IN to EZ
46      OEC &= 0xF7;
47      OUTC |= CSHIGH;//Turn CS* high
48      CKCON &= 0xF8;//Set stretch 0
49      while(TRUE)
50      {
51              enable_25LC320();//Enable write
52              write_25LC320 (a,d);//Write byte
53              while (status_25LC320() & 1);//Wait until done
54              t = read_25LC320 (a);//Try to read back
55              if (t != d)
56                      x=0;    //Test for read back, set breakpoint here
57              a++;
58              d++;
59      }
60  }
61  void write_25LC320 (int a, BYTE d)
62  {
63      OUTC &= INLOW & CLKLOW;//Make sure signals are low
64      OUTC &= CSLOW;//Turn cs low
65      spiwritebyte (WRITE_CMD);
66      spiwritebyte (a >> 8);
67      spiwritebyte (a);
68      spiwritebyte (d);
69      OUTC |= CSHIGH;//Turn cs high
70  }
71
72  BYTE read_25LC320 (int a)
73  {
74      BYTE d;
```

```
75
76      OUTC &= INLOW & CLKLOW;//Make sure signals are low
77      OUTC &= CSLOW;//Turn cs low
78      spiwritebyte (READ_CMD);
79      spiwritebyte (a >> 8);
80      spiwritebyte (a);
81      d = spireadbyte();
82      OUTC |= CSHIGH;//Turn cs high
83      return (d);
84  }
85
86  void enable_25LC320 (void)
87  {
88      OUTC &= INLOW & CLKLOW;//Make sure signals are low
89      OUTC &= CSLOW;//Turn cs low
90      spiwritebyte (WRITE_ENABLE);
91      OUTC |= CSHIGH;//Turn cs high
92  }
93
94  BYTE status_25LC320 (void)
95  {
96      BYTE d;
97
98      OUTC &= INLOW & CLKLOW;//Make sure signals are low
99      OUTC &= CSLOW;//Turn cs low
100     spiwritebyte (READ_STATUS);
101     d = spireadbyte();
102     OUTC |= CSHIGH;//Turn cs high
103     return (d);
104 }
```

This program makes calls to spiwrite and spiread to transfer data to or from a 25LC320 EEPROM. The main loop writes an incrementing byte to an incrementing address and then tries to read it back. A breakpoint can be set to confirm functionality. Any port pins can be used by properly setting up the port and changing the bitmasks.

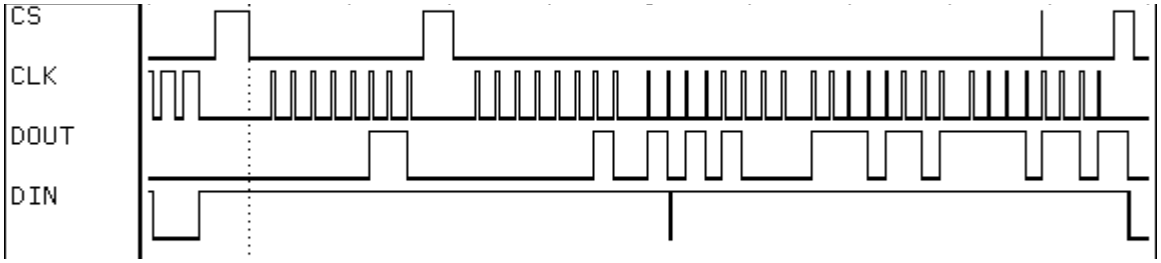

Figure 3. Entire Write Sequence

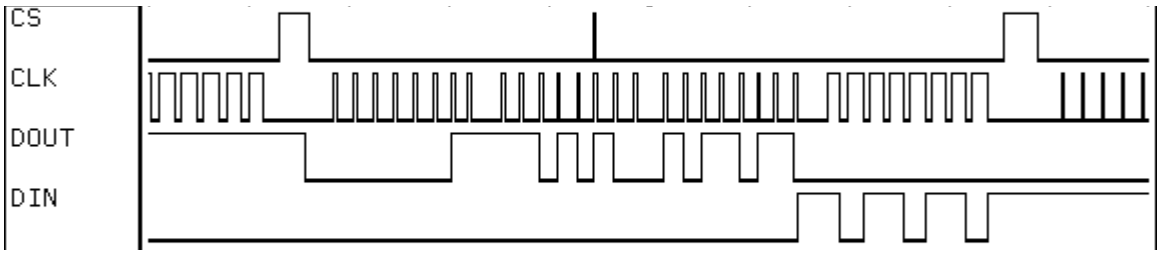

Figure 4. Entire Read Sequence

Conclusion

These programs demonstrate a simple way to implement SPI in software on the EZ-USB. The programs demonstrate communication with a 25LC320 EEPROM, but can be modified to communicate with any SPI peripheral. No attempt has been made to optimize the code for performance.