TN06008

LPC2000 SPI slave code example

Paul Seerden – 2006 June 16

Technical note

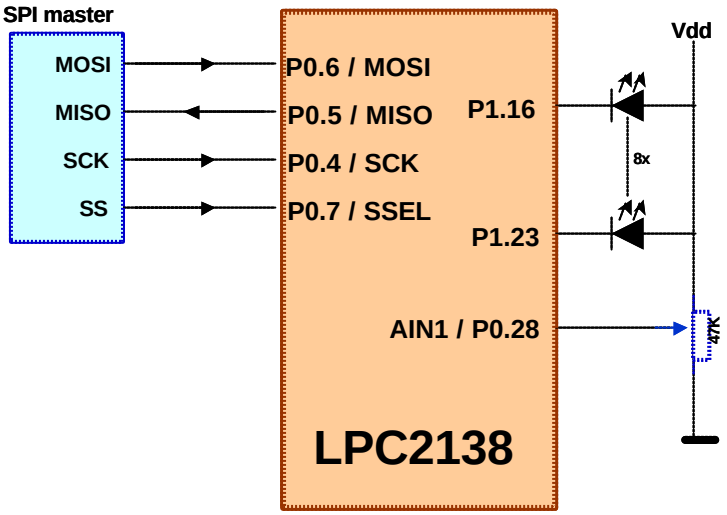
Rev: 1.0

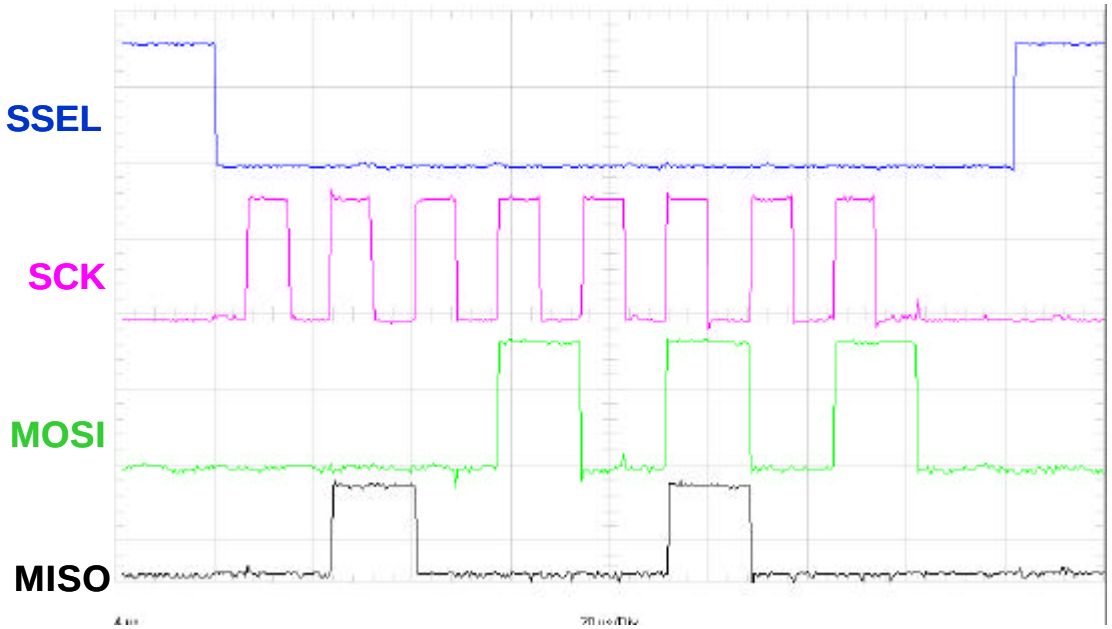
Introduction

This technical note shows an SPI slave software example for the Philips Semiconductors LPC2000 microcontroller family. The software is written for the LPC2138 and tested on an MCB2130 board. It supports interrupt driven SPI slave message transfers.

SPI slave demo

SPI slave mode functions are very specific to the system design, and therefore, very difficult to make generic. In the example below 'some' SPI master generates the slave select SSEL signal, sends an 8-bit value to the LPC2138 and receives a byte in one SPI transfer. The byte transmitted by the LPC2138 SPI slave, called ***SlaveSnd***, is actually the analog value at AIN1 (P0.28). The byte received by the LPC2138, called ***SlaveRcv***, is reflected to port pins P1.16 to P1.23.





PHILIPS

```
extern void SPI0_Init(void);

extern unsigned char SlaveSnd;
extern unsigned char SlaveRcv;

static unsigned char ADC_Read(void)
{
  unsigned int i;

    AD0CR  = 0x00200302;                  // Init ADC (Pclk = 12MHz) and select channel AD0.1
    AD0CR |= 0x01000000;                  // Start A/D Conversion
    do
    {
        i = AD0DR;                        // Read A/D Data Register
    } while ((i & 0x80000000) == 0);      // Wait for end of A/D Conversion

    return (i >> 8) & 0x00FF;             // bit 8:15 is 8-bit AD value
}

void main(void)
{
    PINSEL1 |= 0x01000000;                // P0.28 = AD0.1
    IODIR1   = 0x00FF0000;                // P1.16..23 defined as Outputs
    SPI0_Init();                          // initialize SPI bus

    while (1)
    {
        IOCLR1 = 0x00FF0000;              // Turn off LEDs
        IOSET1 = SlaveRcv << 16;          // Turn on LED

        SlaveSnd = ADC_Read();            // convert and send channel AD0.1
    }
}
```

```
unsigned char SlaveRcv = 0xAA;
unsigned char SlaveSnd;

void SPI0_Isr(void) __irq
{
    if (S0SPSR) ;                 // (dummy) read status register

    SlaveRcv = S0SPDR;            // read data received
    S0SPDR   = SlaveSnd;          // next data to transmit

    S0SPINT  = 0x01;              // reset interrupt flag
    VICVectAddr = 0;              // reset VIC
}

void SPI0_Init(void)
{
    PINSEL0 |= 0x00005500;        // configure SPI0 pins
    S0SPCR   = 0x88;              /* 1000 1000     Initialize SPI hardware:
                                     |||| ||||
                                     |||| | -----> reserved
                                     ||||  ------> SPI clock phase select
                                     |||  -------> SPI clock polarity = low when idle
                                     ||  --------> SPI slave mode
                                     |  ---------> SPI data order = msb first
                                      ----------> SPI interrupt enabled        */

    VICVectAddr0  = (unsigned int) &SPI0_Isr;
    VICVectCntl0  = 0x2A;         // Channel0 on Source#10 ... enabled
    VICIntEnable |= 0x400;        // 10th bit is SPI0 interface
}
```