

# TECHNICAL NOTE

## **ABSTRACT**

This technical note gives an example of how to use the SPI interface (master mode and interrupt driven) of the Philips Semiconductors LPC2000 microcontroller family.

## **Disclaimer**

Described applications 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.

## **MACC-06001**

### LPC2xxx SPI master code example

Author: Paul Seerden

2006 January 20

# LPC2xxx SPI master code example

MACC-06001

## SPI MASTER DEMO

The software example below configures the on-chip LPC2138 SPI block (SPI0) to interface as a master to a DS1722 digital thermometer. A simplified block diagram of the used hardware is shown in figure 1.

The code repeatedly reads the actual temperature in 8-bit mode. This digital 8-bit temperature value is simply displayed on eight LEDs connected to port P1.16-23 of the LPC2138.

The peripheral (VPB) clock is set equal to the system clock (12 MHz) and SPI bit rate is programmed to 1 Mb/s (SPCCR=12). The driver software is interrupt driven (VIC channel 0 irq).

Figure 2 and table 1 show the SPI timing parameters measured in this example and are only indicative.

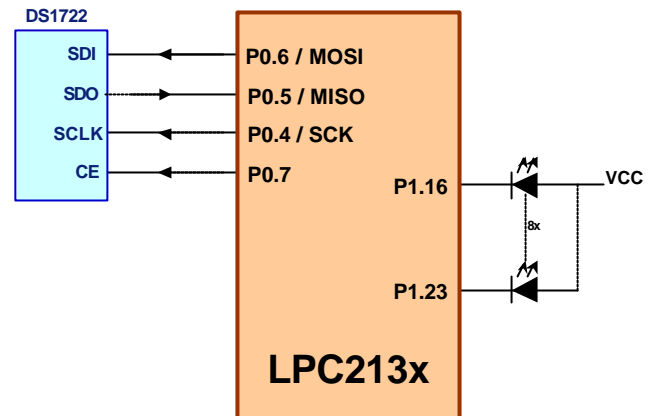


Fig 1. Simplified block diagram

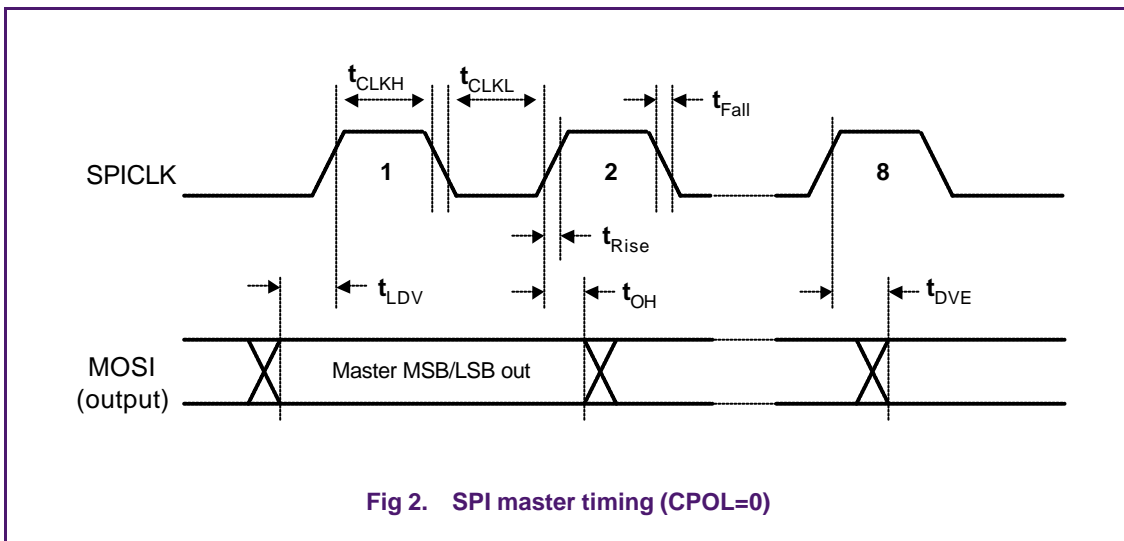


Fig 2. SPI master timing (CPOL=0)

Table 1: Measured timing characteristics

Symbol	Parameter	Conditions	Typical (measured)	Unit
$t_{CLKH}$	SPI clock high time	See figure 2, PCLK = 12 MHz CCR = 12, SCK = 1MHz	500	ns
$t_{CLKL}$	SPI clock low time	See figure 2, PCLK = 12 MHz CCR = 12, SCK = 1MHz	500	ns
$t_{Rise}$	SPI outputs Rise time	See figure 2 SCK0 and MOSI0	20	ns
$t_{Fall}$	SPI outputs Fall time	See figure 2 SCK0 and MOSI0	20	ns
$t_{LDV}$	SPI Leading data valid	See figure 2	560	ns
$t_{DVE}$	SPI Data valid from enable	See figure 2	180	ns
$t_{OH}$	SPI Output data hold time	See figure 2	150	ns

## LPC2xxx SPI master code example

MACC-06001

```

#include <LPC21xx.H>                // LPC21xx definitions

#define SPI_OK          0           // transfer ended No Errors
#define SPI_BUSY       1           // transfer busy
#define SPI_ERROR      2           // SPI error

static unsigned char  state;        // State of SPI driver
static unsigned char  spiBuf[4];   // SPI data buffer
static unsigned char  *msg;        // pointer to SPI data buffer
static unsigned char  count;       // nr of bytes send/received

void SPI_Isr(void) __irq
{
    if ((S0SPSR & 0xF8) == 0x80)
    {
        *msg++ = S0SPDR;           // read byte from slave
        if (--count > 0)
            S0SPDR = *msg;        // sent next byte
        else
            state = SPI_OK;       // transfer completed
    }
    else
    {                               // SPI error
        *msg = S0SPDR;           // dummy read to clear flags
        state = SPI_ERROR;
    }
    S0SPINT = 0x01;              // reset interrupt flag
    VICVectAddr = 0;            // reset VIC
}

static void DS1722_Write(unsigned char add, unsigned char val)
{
    spiBuf[0] = add;             // DS1722 address
    spiBuf[1] = val;
    msg = spiBuf;
    count = 2;                   // nr of bytes
    state = SPI_BUSY;           // Status of driver

    IOSET0 = 0x00000080;        // SS_DS1722 = 1
    S0SPDR = *msg;              // sent first byte
    while (state == SPI_BUSY) ; // wait for end of transfer
    IOCLR0 = 0x00000080;        // SS_DS1722 = 0
}

static void SPI_Init(void)
{
    VICVectAddr0 = (unsigned int) &SPI_Isr;
    VICVectCntl0 = 0x2A;        // Channel0 on Source#10 ... enabled
    VICIntEnable |= 0x400;      // 10th bit is the SPI

    IODIR0 |= 0x00000080;       // P0.7 defined as SS_DS1722
    IOCLR0 = 0x00000080;       // SS_DS1722 = 0
    PINSEL0 |= 0x00001500;      // configure SPI0 pins (except SSEL0)

    S0SPCCR = 12;               // SCK = 1 MHz, counter > 8 and even
    S0SPCR = 0xA8;             // CPHA=1, CPOL=0, master mode, MSB first, interrupt enabled
}

int main (void)
{
    IODIR1 |= 0x00FF0000;       // P1.16-23 defined as output
    IOCLR1 = 0x00FF0000;       // All LEDs off

    SPI_Init();
    DS1722_Write(0x80,0xE0);    // initialize DS1722

    while(1)
    {
        DS1722_Write(0x02,0x02); // read temperature
        if (state == SPI_OK)     // no error ?
        {
            IOCLR1 = 0x00FF0000;
            IOSET1 = spiBuf[1] << 16;
        }
    }
}

```