EXAMPLE SPI SW FOR SCA61T- AND SCA10xx SERIES

Objective

To set up communication between VTI Technologies' digital motion sensor components and a MCU of an application device. This document applies to all SCA61T, SCA100T, SCA103T, SCA1000 and SCA1020 –series products.

Description of the problem

The acceleration output data of these VTI sensors is coded to an 11 bit data word. The hardware implemented SPI routines included in some micro controller cannot be used for accelerometer output readings because the standard SPI communication is made for an 8 bit data word.

Solution

The C- language software listing example below shows an easy way to implement 11 bit data reading via the SPI bus. The example code can be implemented for the SPI read routine in any micro controller.

As VTI's digital components' temperature output is 8 bit data,most standard MCU HW SPI functions can read it.

Unlike the temperature data, the inclination output data is 11 bit. For 11 bit data reading, the hardware SPI function must be disabled and reading is done via the SW generated SPI clock and read routine. The routine included in the example code generates 167 kHz SPI clock (SPI clock is about 3µs in high state and 3µs in low state).

In the example code, 1 µs wait states are generated to adjust the SPI clock pulse shape to a 50% clock pulse ratio. When implementing the example code into the MCU, the output needs to be checked with an oscilloscope so that the SPI clock pulse ratio is about 50% and the clock speed stays below 500 kHz. Depending how fast the micro controller executes I/O commands, some wait states need to be added or removed.

A C- language software listing example:

```
/*
 * Execute SPI communication, This code extract is designed for ATMEL AVR
 */
inline uint16_t
HW_SPI(uint8_t W_len, uint16_t W_data, uint8_t R_len)
{
    uint16_t value;

    value = 0;

#ifdef SCA_SCP_DEMO
    SPI_I2C_PORT &= ~(1 << SPI_I2C_SELECT);
#endif
```

```
HW_POUT(0,0);      // pin 0 is reserved for CSB!

switch(W_len)
{
   case 0:
      break;
   case 8:          // This loop writes 8 bit data to SPI
      SPDR = (uint8_t)(W_data);
      while(!(SPSR & (1<<SPIF)))
         ;
      value = SPDR;
      break;
   case 16:           // This loop writes 16 bit data to SPI
      SPDR = (uint8_t)(W_data>>8);
      while(!(SPSR & (1<<SPIF)))
         ;
      value = SPDR << 8;
      SPDR = (uint8_t)(W_data);
      while(!(SPSR & (1<<SPIF)))
         ;
      value |= SPDR;
      break;
   default:
      SPI_DISABLE();
      W_data <<= (16 - W_len);   // This loop writes W_len bit long data to
                                 // SPI
      while(W_len > 0)
      {
         SPI_PORT &= ~(1<<SPI_CLK);

         if(W_data & 0x8000)
            SPI_PORT |= (1<<SPI_MOSI);
         else
            SPI_PORT &= ~(1<<SPI_MOSI);

         W_data <<= 1;
         wait_1us();

         SPI_PORT |= (1<<SPI_CLK);
         wait_1us();

         value <<= 1;
         if(SPI_PORT & (1<<SPI_MISO))
            value |= 0x0001;

         W_len--;
      }
      SPI_PORT &= ~(1<<SPI_CLK);
      SPI_ENABLE();
      break;
}
while(R_len != 0)          //This loop reads R_len bits from SPI
{
   if((R_len % 8) == 0) //use standard HW SPI for 8 bit and 16 bit data
   {                    //This loop is used for VTI SCA100T 8 bit temperature
                        // data
      SPDR = 0x00;
      value <<= 8;
      R_len -= 8;
      while(!(SPSR & (1<<SPIF)))
         ;
      value |= SPDR;
   }
   else
   {
      SPI_DISABLE();  //if data length is not 8 or 16 bit then disable HW SPI
      while(R_len > 0)//This loop generates SPI clock signal to the SPI port
      {               //and read SPI data bit by bit
         SPI_PORT &= ~(1<<SPI_CLK);//This loop is used for VTI SCA100T 11 bit
                                   // measurement data.
         wait_1us();  //For measurement data R_len=11
         wait_1us();
         wait_1us();  //wait_1us routines generate 1 us delay,
                      // These 4 1us waits are needed to generate 50% duty
                      // cycle to SPI clock signal
         SPI_PORT |= (1<<SPI_CLK); //with these delays the SPI clock is
```

```
                                                // high 3us and low 3us.
            wait_1us();

            value <<= 1;
            if(SPI_PORT_IN & (1<<SPI_MISO))
               value++;

            R_len--;
         }
         SPI_PORT &= ~(1<<SPI_CLK);
         SPI_ENABLE();
      }
   }

   HW_POUT(0,1);          // pin 0 is reserved for CSB!

   return value;
}
```

Document Change Control

Version	Date	Change Description
0.1	14.08.2005	Initial draft.
0.2	04.09.2006	C-code file printed on the text
0.3	25.01.2007	Text copying to clipboard enabled