

# C-CODE EXAMPLES FOR SCA3000 ACCELEROMETER

## 1 INTRODUCTION

This technical note presents simple C-code examples for SPI communication between the SCA3000 sensor and a MCU. In these code examples

- the X-axis acceleration is read from the SCA3000 sensor
- the MODE register content is read
- the free fall detection is enabled

Please refer to the document "SCA3000 Product Family Specification 8257300A" for further information on SCA3000 register addressing and SPI communication. The examples are tested with SCA3000 DEMO KIT hardware. Please refer to the document "SCA3000 DEMO KIT User Manual 8259300" for further information about the SCA3000 DEMO KIT. This document applies to SCA3000-D01 and SCA3000-E01.

## 2 C-CODE EXAMPLE

This document covers the following C-code examples listed below:

- 2.1 Read X-axis acceleration
- 2.2 Read MODE register content
- 2.3 Enable free fall detection

## 2.1 Read X-axis acceleration

```

/* This example reads SCA3000 X acceleration registers MSB and LSB.
 * The register address of X channel is 0x05 (MSB).
 * NOTE: The command byte of SPI bus is :
 *       (MSB) A A A A A A A RW 0 (LSB)
 *
 *       |           | | *- always zero
 *       |           | *--- Read / Write Bit
 *       \-----*----- Register address, 0x05 at this example
 *
 * Because of the real byte sent to SPI bus must be calculated:
 *   BYTE = (Address * 4) + RW*2;
 * For Read operation RW = 0 and for Write operation RW = 1
 * 'Address *4' shifts the bit pattern to left by 2 and 'RW*2' by 1.
 * BYTE = 0x05*4 + 0;
 * BYTE = 0x14;
 * This calculation is done inside the function.
 *
 * Usage of the following function:
 *
 * int Xacc;
 *
 * Xacc = SCA3000_read_16bit(0x05);
 *
 * This example has been written for an Atmel ATmega168 processor in a GCC environment.
 * Other processors or environments might need different names ports or SPI device names.
 * SPDR - is the read/ write data to/ from SPI bus register
 * (see ATmega168 document (doc2545.pdf) pages 159 - 167 for more information) */

#define SPI_CSB 0x04 // This is PB2 at the port
#define SPI_PORT PORTB

// Function takes in the 8-bit register address and returns 16-bit register value.
int SCA3000_Read_16bit(unsigned int Address)
{
    int result;
    result = 0;
    Address = Address << 2; // RW bit is set to zero by shifting the bit
                          // pattern to left by 2

    SPI_PORT = SPI_PORT & (~SPI_CSB); // Set CSB to zero
    SPDR = Address; // Write command to SPI bus

    while(!(SPSR & (1 << SPIF))) // Wait until data has been sent
        ;

    SPDR = 0x00; // Write dummy byte to line
                // in order to generate SPI clocks for data read
    while(!(SPSR & (1 << SPIF))) // Wait until data has been sent
        ;

    result = SPDR; // Get MSB of the result
    result = result << 8; // Shift the MSB of the result to left by 8

    SPDR = 0x00; // Write dummy byte to line
                // in order to generate SPI clocks for data read
    while(!(SPSR & (1 << SPIF))) // Wait until data has been sent
        ;

    result |= SPDR; // Get LSB of the result

    SPI_PORT = SPI_PORT | SPI_CSB; // Set CSB to one
    return result;
}

```

## 2.2 Read MODE register content

```

/* This example reads SCA3000 MODE register content.
 * The MODE register address is 0x14.
 * NOTE: The command byte of SPI bus is :
 *      (MSB) A A A A A A A RW 0 (LSB)
 *      |         |         | *-- allways zero
 *      |         |         | *--- Read / Write Bit
 *      |-----|----- Register address, 0x14 at this example
 *
 * Because of that the real byte send to SPI bus must be calculated:
 *      BYTE = (Address * 4) + RW*2;
 * For Read operation RW = 0 and for Write operation RW = 1
 * 'Address *4' shifts the bit pattern to left by 2 and 'RW*2' by 1.
 *      BYTE = 0x14*4 + 0;
 *      BYTE = 0x50;
 * This calculation is done inside the function.
 *
 * Use of the following function:
 *
 * SCA3000_read_mode();
 *
 * This example has been written for an Atmel ATmega168 processor in a GCC environment.
 * Other processors or environments might need different names ports or SPI device names.
 * SPDR - is the read/ write data to/ from SPI bus register
 * (see ATmega168 document (doc2545.pdf) pages 159 - 167 for more information) */

#define SPI_CSB 0x04 // This is PB2 at the port
#define SPI_PORT PORTB
#define SCA3000_MODE_REG_READ (0x14*4) // calculate address to MODE register read

// Read SCA3000 MODE register and return its value.
int SCA3000_read_mode(void)
{
    int mode;

    mode = 0;
    SPI_PORT = SPI_PORT & (~SPI_CSB); // Set CSB to zero
    SPDR = SCA3000_MODE_REG_READ; // Write command to SPI bus

    while(!(SPSR & (1 << SPIF))) // Wait until data has been sent
        ;

    SPDR = 0x00; // Write dummy byte to line
                // in order to generate SPI clocks for data read

    while(!(SPSR & (1 << SPIF))) // Wait until data has been sent
        ;

    mode = SPDR;
    SPI_PORT = SPI_PORT | SPI_CSB; // Set CSB to one

    return mode;
}

```

## 2.3 Enable free fall detection

```

/* This example enables SCA3000 free fall detection.
 * The register address of free fall control is 0x14 (MODE register).
 * NOTE: The command byte of SPI bus is :
 *      (MSB) A A A A A A RW 0 (LSB)
 *      |         |         | *-- always zero
 *      |         |         | *--- Read / Write Bit
 *      |-----|----- Register address, 0x14 at this example
 *
 * Because of that the real byte send to SPI bus must be calculated:
 *      BYTE = (Address * 4) + RW*2;
 * For Read operation RW = 0 and for Write operation RW = 1
 * 'Address *4' shifts the bit pattern to left by 2 and 'RW*2' by 1.
 *      BYTE = 0x14*4 + 1*2;
 *      BYTE = 0x52;
 * This calculation is done inside the function.
 *
 * Usage of the following function:
 *
 * SCA3000_Enable_FreeFall_Detection();
 *
 * This example uses the "SCA3000_read_mode()" function above.
 *
 * This example has been written for an Atmel ATmega168 processor in a GCC environment.
 * Other processors or environments might need different names ports or SPI device names.
 * SPDR - is the read/ write data to/ from SPI bus register
 * (see ATmega168 document (doc2545.pdf) pages 159 - 167 for more info) */

#define SPI_CSB 0x04 // This is PB2 at the port
#define SPI_PORT PORTB
#define SCA3000_MODE_REG_WRITE (0x14*4+1*2) // calculate address to MODE register write
#define SCA3000_FREEFALL 0x10 // FFD_EN bit is bit nro: 4
// binary: 0001 0000 == 0x10

// Set SCA3000 Free Fall detection ON
void SCA3000_Enable_FreeFall_Detection(void)
{
    int mode = 0;

    mode = SCA3000_read_mode(); // Read MODE register content, this
                                // function is defined above
    mode = mode | SCA3000_FREEFALL; // Set FFD_EN bit to one,
                                    // enable free fall detection

    SPI_PORT = SPI_PORT & (~SPI_CSB); // Set CSB to zero
    SPDR = SCA3000_MODE_REG_WRITE; // Write command to SPI bus
    while(!(SPSR & (1 << SPIF))) // Wait until data has been sent
        ;
    SPDR = mode; // Write new MODE register content to line
    while(!(SPSR & (1 << SPIF))) // Wait until data has been sent
        ;

    SPI_PORT = SPI_PORT | SPI_CSB; // Set CSB to one
}

```