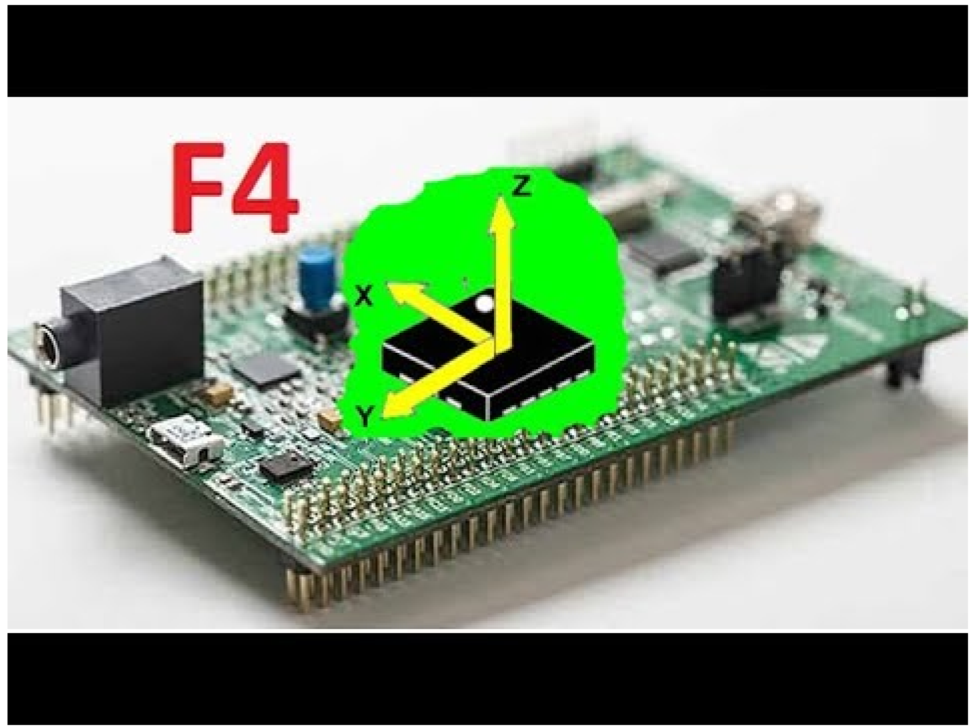
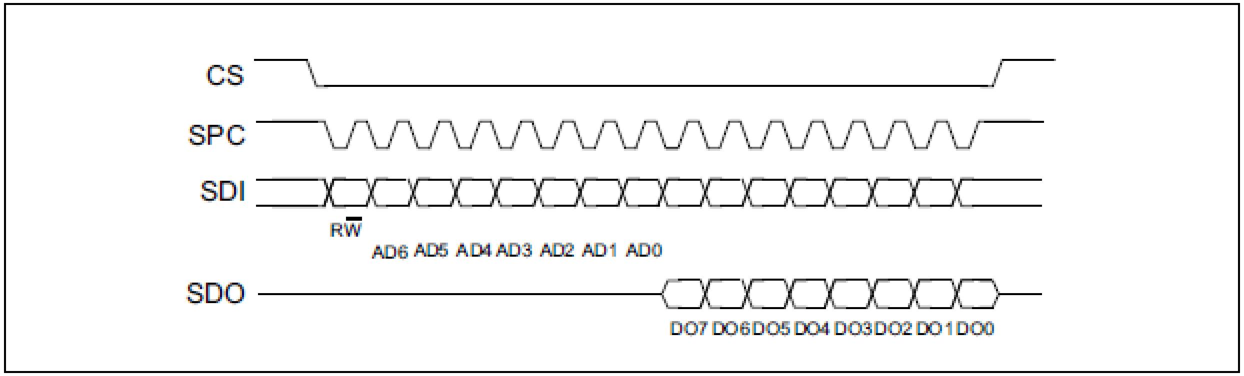
Acelerómetro en STM32F407



5.2.1 SPI read

Figure 8. SPI read protocol



The SPI Read command is performed with 16 clock pulses. A multiple byte read command is performed by adding blocks of 8 clock pulses to the previous one.

bit 0: READ bit. The value is 1.

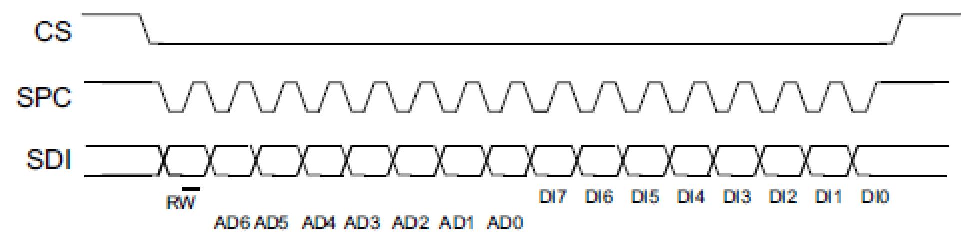
bit 1-7: address AD(6:0). This is the address field of the indexed register.

bit 8-15: data DO(7:0) (read mode). This is the data that is read from the device (MSb first).

bit 16-... : data DO(...-8). Further data in multiple byte reads.

5.2.2 SPI write

Figure 10. SPI write protocol



The SPI Write command is performed with 16 clock pulses. A multiple byte write command is performed by adding blocks of 8 clock pulses to the previous one.

bit 0: WRITE bit. The value is 0.

bit 1 -7: address AD(6:0). This is the address field of the indexed register.

bit 8-15: data DI(7:0) (write mode). This is the data that is written inside the device (MSb first).

bit 16-... : data DI(...-8). Further data in multiple byte writes.

Acelerómetro en STM32F407

6 Register mapping

Table 15 provides a list of the 8/16-bit registers embedded in the device and the corresponding addresses.

Table 15. Register address map

Name	Type	Register address		Default	Comment
		Hex	Binary		
OUT_T	r	0C	00001100	-	Temperature output
INFO1	r	0D	00001101	0010 0001	Information register 1
INFO2	r	0E	00001110	0000 0000	Information register 2
WHO_AM_I	r	0F	00001111	0011 1111	Who I am ID
OFF_X	r/w	10	00010000	0000 0000	X-axis offset correction
OFF_Y	r/w	11	00010001	0000 0000	Y-axis offset correction
OFF_Z	r/w	12	00010010	0000 0000	Z-axis offset correction
CS_X	r/w	13	00010011	0000 0000	Constant shift X
CS_Y	r/w	14	00010100	0000 0000	Constant shift Y
CS_Z	r/w	15	00010101	0000 0000	Constant shift Z
LC_L	r/w	16	00010110	0000 0001	Long counter registers
LC_H	r/w	17	00010111	0000 0000	
STAT	r	18	00011000	-	Interrupt synchronization
PEAK1	r	19	00011001	-	Peak value
PEAK2	r	1A	00011010	-	Peak value
VFC_1	r/w	1B	00011011	-	Vector filter coefficient 1
VFC_2	r/w	1C	00011100	-	Vector filter coefficient 2
VFC_3	r/w	1D	00011101	-	Vector filter coefficient 3
VFC_4	r/w	1E	00011110	-	Vector filter coefficient 4
THRS3	r/w	1F	00011111	-	Threshold value 3
CTRL_REG4	r/w	20	00100000	0000 0111	Control register
CTRL_REG1	r/w	21	00100001	0000 0000	SM1 control register

7.20 CTRL_REG4 (20h)

Control register 4.

Table 53. Control register 4

ODR3	ODR2	ODR1	ODR0	BDU	Zen	Yen	Xen

Table 54. CTRL_REG4 register description

ODR 3:0	Output data rate and power mode selection. Default value: 0000 (see *Table 55*)
BDU	Block data update. Default value: 0 (0: continuous update; 1: output registers not updated until MSB and LSB have been read)
Zen	Z-axis enable. Default value: 1 (0: Z-axis disabled; 1: Z-axis enabled)
Yen	Y-axis enable. Default value: 1 (0: Y-axis disabled; 1: Y-axis enabled)
Xen	X-axis enable. Default value: 1 (0: X-axis disabled; 1: X-axis enabled)

ODR[3:0] is used to set the power mode and ODR selection. In *Table 55* (output data rate selection) all available frequencies are shown.

Acelerómetro en STM32F407

Table 15. Register address map (continued)

Name	Type	Register address		Default	Comment
		Hex	Binary		
OUT_X_L	r	28	00101000	0000 0000	Output registers
OUT_X_H	r	29	00101001		
OUT_Y_L	r	2A	00101010		
OUT_Y_H	r	2B	00101011		
OUT_Z_L	r	2C	00101100		
OUT_Z_H	r	2D	00101101		
FIFO_CTRL	r/w	2E	00101110	0000 0000	FIFO registers
FIFO_SRC	r	2F	00101111	-	

Acelerómetro en STM32F407

7.27 OUT_X (28h - 29h)

X-axis output register.

Table 69. OUT_X_L register

XD7	XD6	XD5	XD4	XD3	XD2	XD1	XD0

Table 70. OUT_X_L register description

XD[7:0]	X-axis output, low values. Default value: 0000 0000

Table 71. OUT_X_H register

XD15	XD14	XD13	XD12	XD11	XD10	XD9	XD8

Table 72. OUT_X_H register description

XD[15:8]	X-axis output, high values. Default value: 0000 0000

```
/*Configure GPIO pin : PE3 – Chip Select del acelerómetro */
 GPIO_InitStruct.Pin = GPIO_PIN_3;
 GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
 GPIO_InitStruct.Pull = GPIO_NOPULL;
 GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
 HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);
```

Acelerómetro en STM32F407

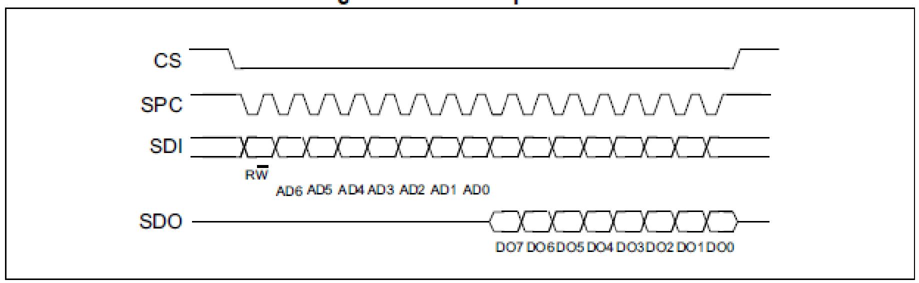
```
/* SPI1 init function */
static void MX_SPI1_Init(void)
{
  /* SPI1 parameter configuration*/
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_2LINES;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_16;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 10;
  if (HAL_SPI_Init(&hspi1) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}
```

```
void Inicializa_Acelerometro ()
{
 /*To transmit data in SPI follow the next steps: */
 // 1. Bring slave select to low
 HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_RESET);
 // 2. Transmit register + data
 spiTxBuf[0] = 0x20; //Register
 spiTxBuf[1] = 0x17; //Data
 //                                  size, timeout
 HAL_SPI_Transmit(&hspi1, spiTxBuf,     2,    50);
 // 3. Bring slave select high
 HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_SET);

 /*To receive data in SPI follow the next steps: */
 // 1.Bring slave select low
 HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_RESET);
 // 2.Transmit register + 0x80 (To set MSB high) Most Significant Bit(MSB) high = read mode
 spiTxBuf[0] = 0x20|0x80; //Register
 HAL_SPI_Transmit(&hspi1, spiTxBuf, 1, 50);
 // 3.Receive data
 HAL_SPI_Receive(&hspi1, spiRxBuf, 1, 50);
 // 4.Bring slave select high
 HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_SET);
}
```

```
uint8_t SPI_Read (uint8_t address)
{
  // 1.Bring slave select low
  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_RESET);
  // 2.Transmit register + 0x80 (To set MSB high) Most Significant Bit(MSB) high = read mode
  spiTxBuf[0] = address|0x80; //Register
  HAL_SPI_Transmit(&hspi1, spiTxBuf, 1, 50);
  // 3.Receive data
  HAL_SPI_Receive(&hspi1, spiRxBuf, 1, 50);
  // 4.Bring slave select high
  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_SET);

  return spiRxBuf[0];
}
```

5.2.1 SPI read

Figure 8. SPI read protocol



The SPI Read command is performed with 16 clock pulses. A multiple byte read command is performed by adding blocks of 8 clock pulses to the previous one.

bit 0: READ bit. The value is 1.

bit 1-7: address AD(6:0). This is the address field of the indexed register.

bit 8-15: data DO(7:0) (read mode). This is the data that is read from the device (MSb first).

bit 16-... : data DO(...-8). Further data in multiple byte reads.

```
void Tarea_Control_Inclinacion(void const * argument)
{
/* Calculo de la rotación en el eje X e Y, dentro de la tarea que
controla la inclinación de la cabeza */

        Ix1 = SPI_Read (0x28);
        Ix2 = SPI_Read (0x29);
        Ix = (Ix2 << 8) + Ix1;
        if (Ix >= 0x8000) Ix = -(65536 - Ix);
        X = Ix/16384.0;

        … ídem para eje Y
        Y = Iy/16384.0;

        … ídem para eje Z
        Z = Iz/16384.0;

        rotX = atan2(Y, sqrt(X*X+Z*Z)) * 180.0/3.1416;
        rotY = - atan2(X, sqrt(Y*Y+Z*Z)) * 180.0/3.1416;
}
```