SPI Communication Report
Lab 3 Group 3

In this report we give a brief description on our understanding of SPI communication during the lab 3. SPI communication protocol was used to download and upload information between Adafruit Feather HUZZAH ESP8266 and ADXL345 digital accelerometer.

1. Connection

ADXL345 support 3 or 4 wire connection. In this lab we use 4-wire configuration (Figure 4.4). These four wires are CS (Chip Select), SDI (Serial Data Iuput), SDO (Serial Data Output), SCLK (Serial Communications Clock).

ESP8266 has SCK (serial clock), MOSI (Master Out Slave In), MISO(Master In Slave Out). The connection should be:

- CS to GPIO
- SDI to MOSI
- SDO to MISO
- SCLK to SCK

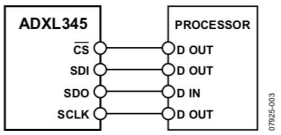

Figure 4. 4-Wire SPI Connection Diagram

2. Initialization

- SPI

The maximum SPI clock speed is 5 MHz and the timing scheme follows clock polarity (CPOL) = 1 and clock phase (CPHA) = 1. As result, the initialization code of SPI is:

```
spi = machine.SPI(1, baudrate=2000000, polarity=1, phase=1)
```

- ADXL345

Some registers need to be set up before the communication.

Register 0x2D of power control is required to be configured before we read any data from the accelerometer. Here we need to convert 6 bits register configuration binary code to hexadecimal representation. We wrote 2B into the power control register by which we let the accelerometer quit the sleep mode that can suppress DATA_READY and let the wake up frequency to be 1Hz.

Register 0x31 of data format register, we gave it a value of 0F which especially set the last four bits of the register to be 1s. This allows us to set the data format range to be +-16g. The configuration code is:

```
Cs.value(0)
spi.write(b'\x2d')
spi.write(b'\x2b')
cs.value(1)
cs.value(0)
spi.write(b'\x31')
spi.write(b'\x0f')
cs.value(1)
```

3. Communication

Noticed that we are sending 8 bits into the accelerometer instead of 6 bits, the size of a register, because the first two bits are used to configure the operation of the register while the second bit is used to determine the read length from register. As the SPI protocol, the first bit is for read or write, so in this case we should make the first bit equal to 1. Besides, the second bit should also be set to 1 which indicate the multiple bytes because we always select the second bytes and do some conversion. For example, the address of X0 is 0x32 but we have to write 0xf2 instead.

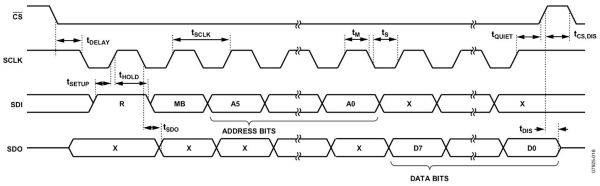

Figure 6. SPI 4-Wire Read

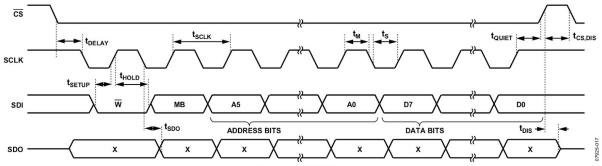

Figure 5. SPI 4-Wire Write

Register 0x31—DATA_FORMAT (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
SELF_TEST	SPI	INT_INVERT	0	FULL_RES	Justify	Range	

Register 0x2D—POWER_CTL (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
0	0	Link	AUTO_SLEEP	Measure	Sleep	Wakeup	

Last but not least, shown in figure below, for rolling texts, only x and y direction components are needed for our project. The horizontal movement of text is controlled by the x and vertical movement of text is controlled by the y. Since the magnitude of the change is sensitive the movement speed can also be set to be proportional to the reading from our accelerometer.