Introduction:

Serial Peripheral Interface (SPI) is a serial communication bus developed by Motorola back in 20th century which was first used externally in microcontroller to communicate with the external peripherals. SPI is mostly used in communicating with the peripherals whenever speed is needed. When data streams, slow rate of communication like Inter-Integrated Circuit (I2C) will be a bad idea to use. Thus, the existence of SPI will be a great help because of the significant high speed rate that SPI has achieved. SPI-compatible interfaces often range into ten megahertz or higher.

Although SPI is good in communication, SPI needs more effort and more hardware resources than I2C when more than one slave is involved. In order to communicate with several slaves, SPI needs multiple connections. Thus, the setup for the connection between master and slave takes time and the area too. As this report goes along, the introduction to the SPI and the experiment of using SPI to communicate will be discussed and document. By the way, Figure 0(a) and Figure 0(b) are the setup in hardware for master and master with slave.

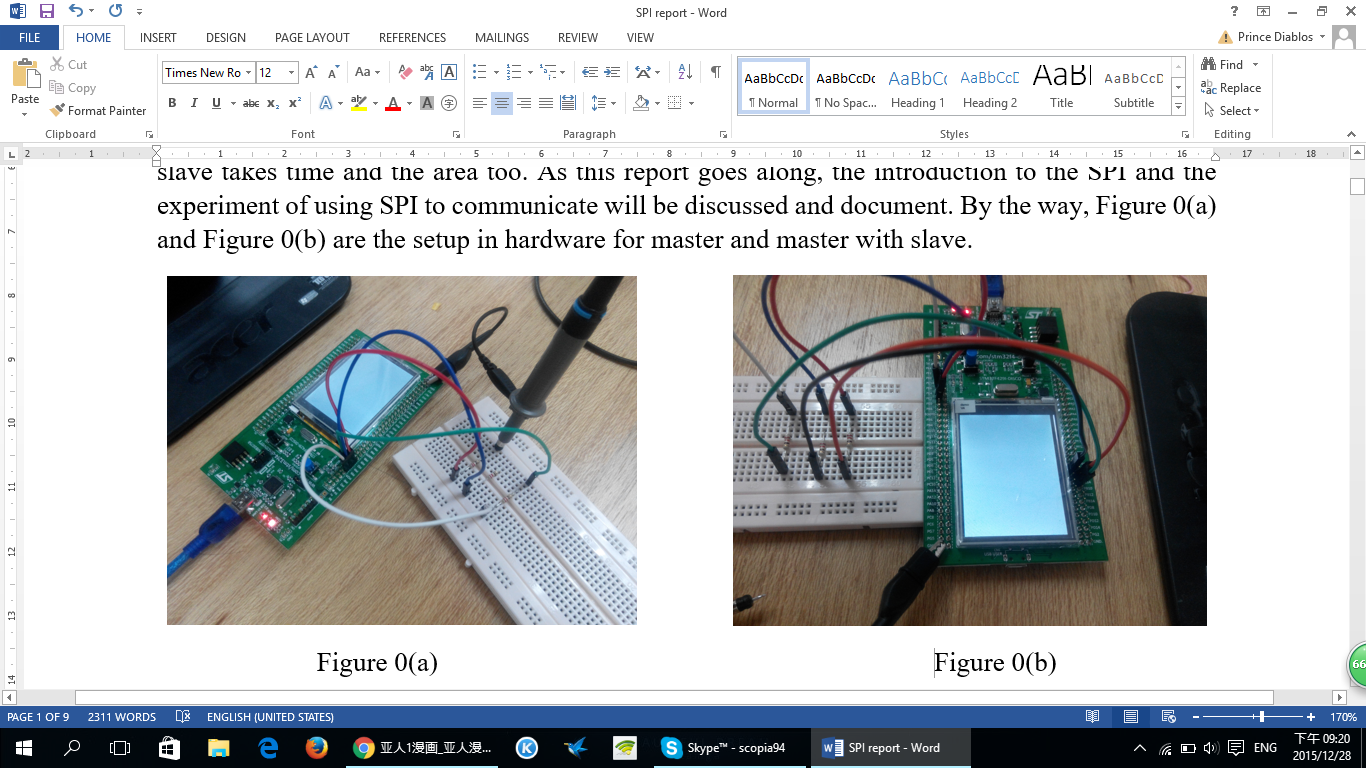


Figure 0(a) is connected in a way that only master is doing the transmission and reception of data. For Figure 0(b), master and slave is connected through connecting the SCK, NSS, MOSI and MISO pins together.

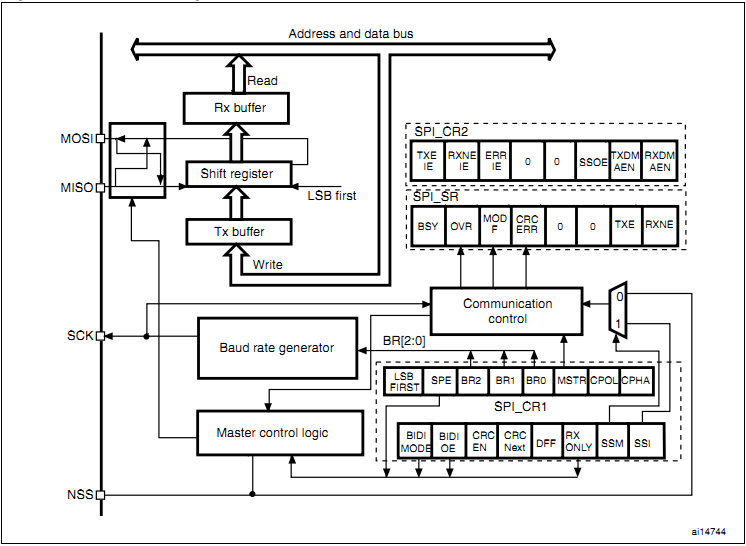


Figure 1Block Diagram of SPI [Accessed from: http://www1.coocox.org/cox/manual/STM32F1xx/group\_\_\_s\_t\_m32\_f1xx\_\_\_s\_p\_i.html]

Figure 1 shown the internal circuitry of how SPI is built. From the diagram shown, four pins are shown which is Master out Slave in (MOSI), Master in Slave out (MISO), Serial Clock (SCK) and Slave Select (NSS) or Chip Select (CS). For MOSI and MISO, both pins are not necessary be used together which have to depend on the setup of transmission and reception.

Pins of SPI:

1. MOSI: This pin is used for master to send out the data to the specific slave.
2. MISO: This pin is used for slave to send out the data to the master.
3. SCK : This pin is used to output the clock for master and input for slave. The slave will not use any clock except the clock output from master, even the clock the slave generated.
4. NSS : This is used the select slave. In other word, without this pin, master and slave cannot be communicated. Initially, this pin is high as in logic 1. To connect with the chosen slave, NSS needs to be low (logic 0) to indicate that the chosen slave is in communication with the master and other slaves cannot communicate with master.

Transmission:

For transmitting any data to the destination through using SPI, data will be first stored in the Transmission buffer (Tx buffer) whenever data is written to Data Register (DR). Once Tx buffer is full, Transmit Buffer Empty (TXE) flag will be cleared to indicate the data is ready to send. Then, the data will send to MOSI or MISO serially through the shift register. Depending on the situation, either MOSI or MISO will be chose to send out data or both will be used for the communication.

Reception:

For receiving any data sent from the destination though SPI, the data will be stored at the reception buffer (Rx buffer) whenever DR is read. Once data is stored inside the Rx buffer, Receive Buffer Not Empty (RXNE) flag is set to indicate the data has not yet read. Then, the data will serially shifted towards the targeted bus get processed. Depending on the situation, either MOSI or MISO will be chose to receive the data or both will be used for the communication.

Serial Clock:

Clock is important in any sequential circuits because flip-flop needs clock pulse to output the data whenever clock triggered from time to time. Thus, SPI has this serial clock which only generated by master. Serial clock is made up by this clock phase and clock polarity. This clock can be configured at Control Register 1 (CR1) in SPI setting with four options. (Refer to Table 1.) The purpose of having this four options of clock is because some devices need this kind of pattern of clock in order to communicate between the source and the destination and able to sample the data at the correct timing. Figure 2 will show the pattern of clock from the four options of clock.

Table 1 Four options of clock pattern of the serial clock.

|  |  |  |
| --- | --- | --- |
| Options / Pattern | Clock phase (Bit 0 in CR1) | Clock polarity (Bit 1 in CR1) |
| 00 | First edge of SCK | Idle 0 |
| 01 | First edge of SCK | Idle 1 |
| 10 | Second edge of SCK | Idle 0 |
| 11 | Second edge of SCK | Idle 1 |

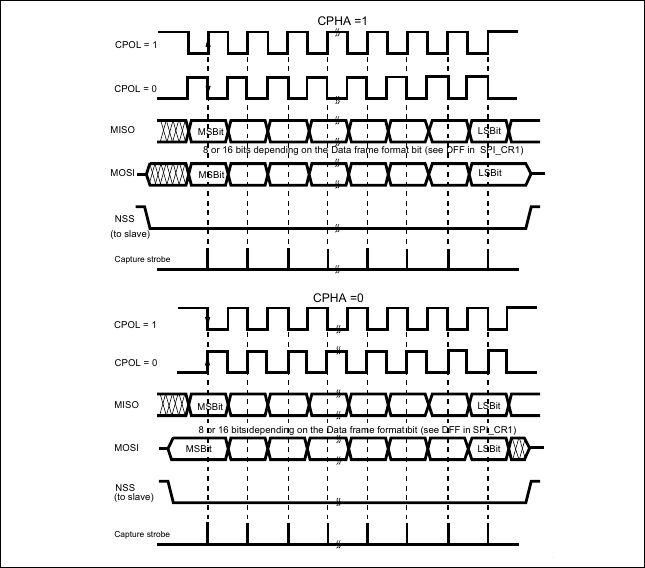


Figure 2 Data Clock Timing Diagram [Accessed from: http://wiki.csie.ncku.edu.tw/embedded/SPI]

Methodology

In order to use SPI in communication, SPI will need to configure in an appropriate setting. Thus, the control register 1 needs to be configured properly. However, there is a bit in control register 2 needs to be configured. That bit is called Frame Format (bit 4). This bit is to decide what kind of behavior that SPI have to behave ― TI (Texas Instrument) mode and Motorola mode. By selecting TI mode, some of the configuration like LSBFIRST bit in CR1 will be set as according to TI mode. As for Motorola mode, there will not be any restriction will configuring CR1 which is a full custom configuration. However, there will not be any different between these two modes under the free conditions. Thus, this experiment is conducted in TI mode because there is not any condition like clock restriction in the communication. So, by choosing TI mode, some of the configuration will be configured by this TI mode and errors will be less. Of all the bits, bit MSTR is the most important because this bit is going to decide the SPI to become a Master or Slave. The following is the procedure to configure the SPI in either master mode or slave mode.

1. Slave mode:

In the slave configuration, the serial clock is not important because the clock that the slave used is transmitted from master. This is due to the priority as a master. Thus, no matter what value that is set for the clock in the slave, data transfer rate will not be affected by the slave unless the clock from master has some alter. The following procedure is the way to setup SPI as a slave.

* 1. The data frame format bit (DFF) in CR1 is set to define the data frame as 8-bit or 16-bit data frame format. By setting this bit, the output data will be sent in 8-bit data frame or 16-bit data frame.
  2. The CPOL and CPHA bits have to be set to define one of the four relationships between the data transfer and the serial clock. In order to transfer the data correctly, the configuration of CPOL and CPHA bits in CR1 for both master and slave need to be same. Otherwise, the data will be sampled at wrong timing and output the wrong data. However, if SPI is configured as TI mode, these two bits can be ignored and these two bits will be set by default.
  3. The frame format (LSBFIRST) needs to be set to define the data to output the Least Significant Bit (LSB) first or Most Significant Bit (MSB) first. However, this bit needs to be same for both master and slave. This is because the data will be different when received if both master and slave are not the same for this bit. Example, master send out data (0x69), data received at slave will become 0x96.
  4. In hardware mode, the NSS pin needs to be connected to a low level signal during whenever communication is needed for both master and slave. In software mode, Slave Select Management (SSM) bit has to be set and Internal Slave Select (SSI) needs to be cleared in CR1. However, these two bits can be ignored is TI mode is selected.
  5. The Frame Format (FRF) bit in CR2 needs to be set to define as TI mode for serial communications. By selecting TI mode, bit SSM, SSI, LSBFIRST, CPOL and CPHA will be ignored and these bits will be set as default. However, FRF can be cleared as Motorola mode if a full control of configuration of SPI is needed.
  6. Master (MSTR) bit is cleared to become a slave and set SPI Enable (SPE) bit at the end of configuration if all the setting is done. By enabling this SPE bit, the SPI is well to communicate with the master. Any changes made after the enable of SPI will not be accepted. This is to ensure to prevent any mismatch of errors occur after sent or received.

Lastly, in this slave configuration, MOSI will become as an input pin and MISO will become as an output pin.

1. Master mode:

In the master configuration, the serial clock is generated on the SCK pin. This clock will be used by master to transfer the data and also for the slave. This is to let the master and slave be synchronous when sending data and receiving data. As for this master configuration, there is not much different as compared to slave configuration. The following procedure is the configuration for master.

* 1. The Baud Rate Control (BR) bits in CR1 needs to be configure to define the serial clock baud rate.
  2. Configuration for CPOL and CPHA have to be configured and same with the slave. These bits will be ignored if TI mode is selected.
  3. DFF bit has to be configured to define as 8- or 16-bit data frame format. This is also needs to be same as the slave.
  4. LSBFIRST bit needs to be configured to define the frame format as LSB first or MSB first. This bit has to be same as slave and will be ignored if TI mode is selected.
  5. If the NSS pin is required in input mode, in hardware mode, the NSS pin has be connected to a high-level signal during the complete byte transmit sequence. In NSS software mode, SSM and SSI bit needs to be set in CR1. However, if NSS is needed in output mode, Slave Select Output Enable (SSOE) needs to be set. This step is not required if TI mode is selected.
  6. FRF is set as TI mode for serial communication. Otherwise, Motorola mode is another option and also a full configuration for CR1.
  7. Lastly, MSTR bit needs to be set to define the SPI as master and set SPE if all the configuration has done. Highly restriction and not recommended for any changes made after SPE is set.

In this configuration, MOSI will become as an output pin and MISO will become as an input pin.

Direction of communication for SPI

SPI can be configured in either half duplex (1-line bidirectional data wire) or full duplex (2-line unidirectional data wire) as shown in Figure 3. By configuring SPI in full-duplex mode, the data transfer rate is actually 2x times faster than half duplex. This is because data can be exchanged between master and slave at the same time in full-duplex due to two wire connection them but not for half-duplex. Transmit and receive in half-duplex can only be done either one is finished due to one wire connection.

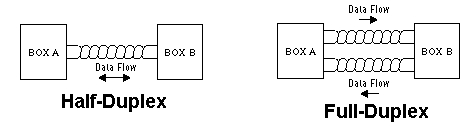


Figure 3 Half-Duplex versus Full-Duplex [Accessed from: http://www.embarcados.com.br/comunicacao-spi-parte-2/]

In order to configure SPI in half-duplex or full-duplex mode, the following procedure is the configuration for either half-duplex or full-duplex.

1. Half-duplex:
   1. Bidirectional Data Mode Enable (BIDIMODE) bit in CR1 needs to be set.
   2. Output Enable in Bidirectional mode (BIDIOE) bit in CR1 is cleared or set. For clearing BIDIOE will define SPI as receive-only mode due to output is disabled. In opposite, setting BIDIOE will define SPI as transmit-only mode.
2. Full-duplex:
   1. BIDIMODE will be cleared in order to communication in full-duplex mode.
   2. RXONLY bit in CR1 will need to be cleared to enable the transmission and reception. If RXONLY bit is set, master or slave can only receive data.

Both half-duplex and full-duplex must be the same in configuration for both master and slave. Otherwise, communication between the master and slave might not work.

Cyclic Redundancy Check (CRC):

CRC is a check that help to reduce any errors during transmission and reception. CRC is implemented for communication reliability because CRC is calculated using programmable polynomial serially on each bit. This CRC has two modes ― CRC8 and CRC16. The selection of these two modes depend on the data frame format (DFF). If DFF is set to define for 8-bit communication, CRC will become CRC8. This is the same for CRC16.

CRC will be functioned whenever CRC is enabled by setting CRCEN bit in CR1. However, by setting CRCEN bit is not enough. Value has to be assigned in CRC polynomial register (CRCPR). This register will help to calculate CRC and attached to the data when transmit. But, CRCPR has an original value inside. Thus, assigned or not will not be a matter. Then, CRC Transfer Next (CRCNEXT) has to be set as soon as the last data is transmitting. This is to tell CRC to be ready and send out once the last data is finished in transmitting. If CRCNEXT bit is cleared, CRC will not be sent out whenever the last data is sent out.

In every transmission of a pack of data, CRC will be sent out once the last data is sent out. If a new set of data is going to send out with CRC enabled. CRC needs to be reset. This is because CRC will mess up with the next data if CRC has not reset yet but the new transmission of data is ongoing. Thus, the following procedure is the step to clear CRC before sending any new data.

Clear CRC:

1. Disable SPI by clearing SPE bit (SPE = 0).
2. Clear CRCEN bit.
3. Set CRCEN bit.
4. Enable SPI by setting SPE bit (SPE = 1).

Result and Discussion:

Throughout the whole experiment, TI mode is used due to the ease of work in configuration as the experiment is communicating in serial communication. However, in either TI mode or Motorola Mode is used, the result is same.

In order to send out any data, GPIO pins for MOSI pin, MISO pin, NSS pin, and SCK pin need to be activated. Besides that, peripheral clock needs to be asserted too. Otherwise, data could not send out.

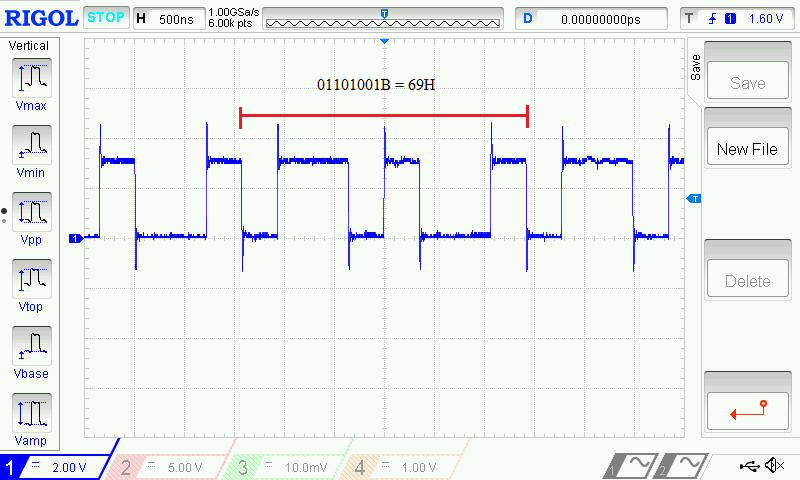


Figure 4 Transmission data of 69H

Figure 4 shows the result of transmitting 8 bits data, 69H. In order to send the data, transmit buffer (TXE) flag needs to be checked. When TXE flag is set, this indicates transmit buffer is emptied. Hence, data can be stored into transmit buffer by writing the data to data register. Once everything is ready, data will send through the output pin, either MOSI or MISO, to the destination.

In Figure 4, noise can be seen. Noise can be any unwanted signals like light. Whenever there is a hanger wire or similar that can be acted like an antenna, noise will be received that antenna-liked. To solve this problem, resistor can be a good helper. By placing a resistor at the output, noise will be trimmed.



Figure 5 NSS signal between master and slave

NSS pin between master and slave is initially high level signal (logic 1). In order for master to communicate with the slave, NSS pin needs to put out a low level signal (logic 0) between master and slave. This is to indicate that the master is communicating with the specific slave if multiple slaves are connected to the master (multislave system). During the communication, any communication from other source are not allowed except the specific slave. This is to prevent any signal collision occurs that will cause the loss of main data. Thus, this is also an advantage of using SPI as communication tool. However, if multiple slaves are connecting the master, area will be a problem.

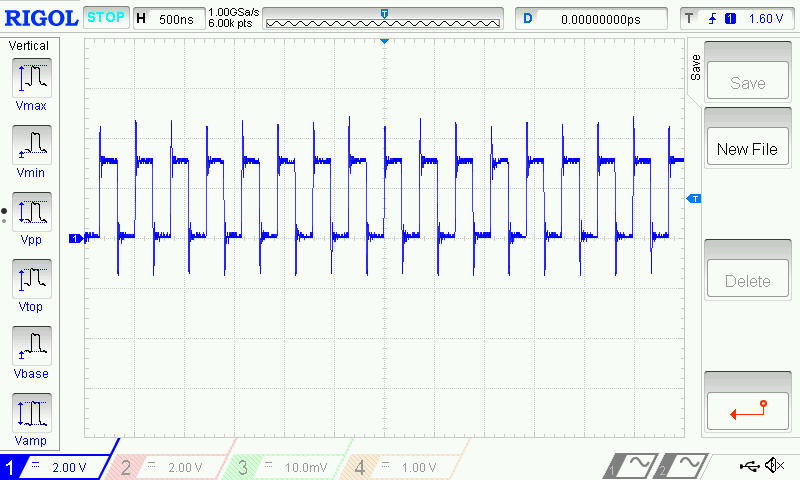


Figure 6 Serial Clock (SCK) generated by master and transmitted to slave

Figure 6 shows the serial clock generated by the master. From Figure 6, serial clock was generated continuously because the data was kept on transmitting to slave once TXE flag is set and RXNE is cleared which indicated new data can be sent and received. In order to generate this clock, peripheral clock for SPI needs to be asserted. If not, SPI will not be functioning and hence, no data can be sent and received.