Reading Assignment

- Textbook, Chapter 22, Serial Communication Protocols, pp. 527 – 598
  - It’s a long chapter.
  - Let’s first look at Section 22.3, SPI, pp. 568–577.
  - Next, we’ll look at Section 22.2, I2C, pp. 546–567.
  - Don’t worry so much about the USB section.
    - Read that only if you’re curious.
    - Not much we can do with that.
    - Other books are better for understanding USB.
STM32 SPI

- Two independent "channels".
  - Turn a parallel word (from 4 – 16 bits) into a serial output stream.
  - Turn a serial input stream into a 4 – 16 bit word.
  - Synchronous clock pulse for each bit.
  - Slave select to indicate that a master is writing.
• It’s a shift register.

• Assume that the microcontroller is the "master".
  − Output is MOSI: Master Out, Slave In.
  − Input is MISO: Master In, Slave Out
  − Clock is SCK
  − Slave Select is NSS
SPI is a *synchronous* protocol

- A clock pulse accompanies each data bit.
- A clock signal must be delivered to each data recipient.

- By comparison, an *asynchronous* protocol would require only a data line.
SPI is fairly fast

• STM32 can drive SPI at half the system clock speed (maximum): 24 MHz.
  - Many devices cannot handle this.

• Baud rate selection allows for up to 256 as a clock divisor. e.g. 48 MHz / 256 = ~ 187 kHz.
  - 187 kHz is the slowest the STM32 can clock SPI.
Masters / Slaves

- Devices in SPI are designated as masters or slaves.
  - A master initiates all data transfer operations.
    - It drives the clock pin and *slave select* pin(s).
  - A slave responds to transfer operations.
- Signals are named according to the devices’ standpoint:
  - e.g. MOSI: "Master Out, Slave In" is the data transmitted by a master and received by a slave.
What does serial output look like?

- Note: For this example, we’ll send the MSB first.
- Let’s look at 8-bit data size first:
- Parallel word: 1 0 1 1 0 0 0 1
- MSB-first serial representation:

In this example, data sent from master to slave on MOSI is latched in to the slave on the rising edge of the each clock.
Note about the clock

- Just because it’s called a clock doesn’t mean it must have special properties like periodicity.
- As long as the clock is not too fast, it can have enormous pauses at any point.
- This is a fringe benefit of a fully-synchronous protocol.
How do we use SPI?

- Single master to single slave configuration:

  Data can be sent and received simultaneously.
How do we use SPI?

- Single master to single slave configuration:
- (no read from slave)

Data can be only be written from master to slave.
Single master, multiple slaves

- Only one SS is active at any time.
There **can** be multiple masters

- The SS pin can be driven as well as monitored.
  - It is an error for multiple masters to assert SS at the same time.
  - Some coördination is needed to ensure that one master writes at any time.
Configuring SPI on the STM32

• Everything is more complicated than you want it to be.
  - Enable the appropriate GPIOx pins with alternate function.
  - Enable the clock to SPI1 in RCC_APB2ENR.
  - Enable the clock to SPI2 in RCC_APB1ENR.
  - Set the baud rate. (some fraction of the system clock)
  - Set the data size. (it can be from 4 – 16 bits)
  - Configure the protocol: (These things are interdependent!)
    • Configure the mode, e.g., bidirectional, but currently master.
    • Set the clock polarity and clock phase (default zero for both).
    • Enable output of NSS, and use NSSP to strobe NSS automatically.
  - Enable the SPI channel.

Not all combinations allow NSSP. See FRM 27.5.5 & 27.5.6
Configuration example

- BR[2:0] selects the SCK divisor. \( f_{SCK} = \frac{f_{SYSCLK}}{2^{(1+BR[2:0])}} \)
  - e.g. when BR is ‘111’ \( f_{SCK} = \frac{48 \text{ MHz}}{256} \)
- DS[3:0] selects the data size.
  - ‘1001’ selects a 10-bit word size.
  - **WARNING**: You must set this. Do not clear and ‘OR’ it. Why?...
CR2 DS Initialization Hazard

- The DS (data size) field in SPIx_CR2.
- Pay attention to the note very carefully.

Bits 11:8  DS [3:0]: Data size

These bits configure the data length for SPI transfers:
- 0000: Not used
- 0001: Not used
- 0010: Not used
- 0011: 4-bit
- 0100: 5-bit
- 0101: 6-bit
- 0110: 7-bit
- 0111: 8-bit
- 1000: 9-bit
- 1001: 10-bit
- 1010: 11-bit
- 1011: 12-bit
- 1100: 13-bit
- 1101: 14-bit
- 1110: 15-bit
- 1111: 16-bit

If software attempts to write one of the "Not used" values, they are forced to the value "0111" (8-bit).
10-bit word size?

- Yes, this is strange.
- That’s what the OLED LCD display requires.
- You may not ever see anything else that uses a 10-bit SPI interface.
Example waveforms

- This works with the OLED display.
  - Initialization sending 0x38 prefaced with two ‘0’ bits.
  - MOSI changes on falling edge of SCK.
  - MOSI latched in to slave on rising edge of clock (when it is stable).
8-bit Usage Hazard

- When sending 8-bit data, you can write a 16-bit word to SPIx_DR.
  - That writes two bytes into the transmitter buffer.
    - Which byte gets sent first?
  - What if you want to write only one byte?
    - You can do a one-byte store to SPIx_DR (e.g. use the STRB assembly language instruction).
    - You can cast the SPIx_DR to an 8-bit integer:
      e.g., *(uint8_t *)&SPI1->DR = one_byte_value;
    - See Appendix A.17.3.
Other interesting configurations

• Cyclic Redundancy Check (CRC)
  - After sending/receiving a group of words, compute a multi-byte checksum.
    • Uses mathematical field theory to do this.
  - If receiver does not compute the same value, using the same algorithm, it indicates an error in transmission.
  - Much better than parity check of RS-232.
    • Parity is a degenerate form of 1-bit CRC.
  - Does not handle error correction.
Applications of SPI

- If you wanted a simple means of loading a 16-bit shift register (e.g. two 74HC595s), you could use SPI for this. (But not for more than 16 bits if you want to use NSS.)
Longer chains of shift registers

- If you use GPIO instead of an automatic NSS pin, then you can have an SPI chain of any length. Just issue multiple words of output.
How a Shift Register works

- One data input pin to specify the next bit.
- One clock pin shifts in the new data.
- One more pin to “store” or “output” the data newly shifted in.
- (One more pin to shift data out of the top bit of the internal shift register. This allows you to chain multiple shift registers.)
Chaining 74HC595 Shift Registers
ShiftRegisterUse

- Shift a pattern into the shift register with DIN and SH_CK.
- Store the pattern to the output registers with ST_CK.
- Leave it there while you shift in a new pattern.
- This gives you $8 \times N$ outputs by using 3 GPIO pins for N shift registers.

- This is the basis for simple serial peripherals.
The Gentle Art of Multiplexing

• Consider the seven-segment display in your dev kit.
  – 8 wires connected to 8 GPIO ports through a driver.
  – Not enough pins to drive lots of 7-segment LEDs.

• If you want an LED display on your project, you will need to multiplex the LED segments.
Turn on one display at a time. Rotate through them rapidly enough that your "persistence of vision" makes it appear they are all on simultaneously and displaying different digits.

Four displays with 10 GPIO pins.
Only using 11 lines on the STM32.
Even Better Example

Only using 3 lines on the STM32.
Secure Digital (SD) Media Cards

- Consider the pins of a (full size) SD card:
  1) nCS (NSS)
  2) DI (MOSI)
  3) VSS
  4) VDD (3.3V)
  5) CLK (SCK)
  6) VSS
  7) DO (MISO)
  8) NC
  9) NC
SD card commands

• The host (master) sends 48-bit commands (3 16-bit words) to the card (slave) to:
  - Prepare to read a block.
  - Check if block is ready to read.
  - Read the block.
  - Prepare to write a block.

• These blocks are not files. They’re just linearly addressed chunks of data on the storage device.
File systems

- You might not want to just write and read blocks.
  - FATFS: a library for reading/writing Microsoft FAT/exFAT filesystem on an SD card.
  - Look for port for STM32F4.
    - Then do a lot of porting.