ECE 381 - Microcontrollers

Serial Peripheral Interface (SPI) & NRF24 Radio

SOUTHERN ILLINOIS UNIVERSITY EDWARDSVILLE

Lab 9 Summary

- We will develop a wireless temperature sensor
- Once a second, sample LM34CZ voltage
  - Convert to floating point with 0.1 degree accuracy
  - Send to BeagleBone Web Server at (146.163.133.81:8080) using NRF24 radio
  - Wait 1s
  - Repeat…
- Also will be given a hex file to test NRF24 operation

Let's Break It Down

- Sample ADC Voltage from LM34CZ
  - PSoC5 has built-in Delta-Sigma ADC (Also programmable amplifiers, filters, etc.)
  - Also 2 hardware SAR ADCs
- Send Using NRF24 Radio
  - NRF24 radio uses SPI to communicate
  - SPI Protocol
  - NRF24 configuration

LM34CZ

- 3 pins, (Vs = +5V, Vout → Goes to pin on Port0[1])
  - NOTE: View on datasheet is from BOTTOM!
- Output 10mV / °F → This is what you sample
- Check DelSig datasheet for API functions
  - ADC_Start()
  - ADC_StartConverstion()
  - ...

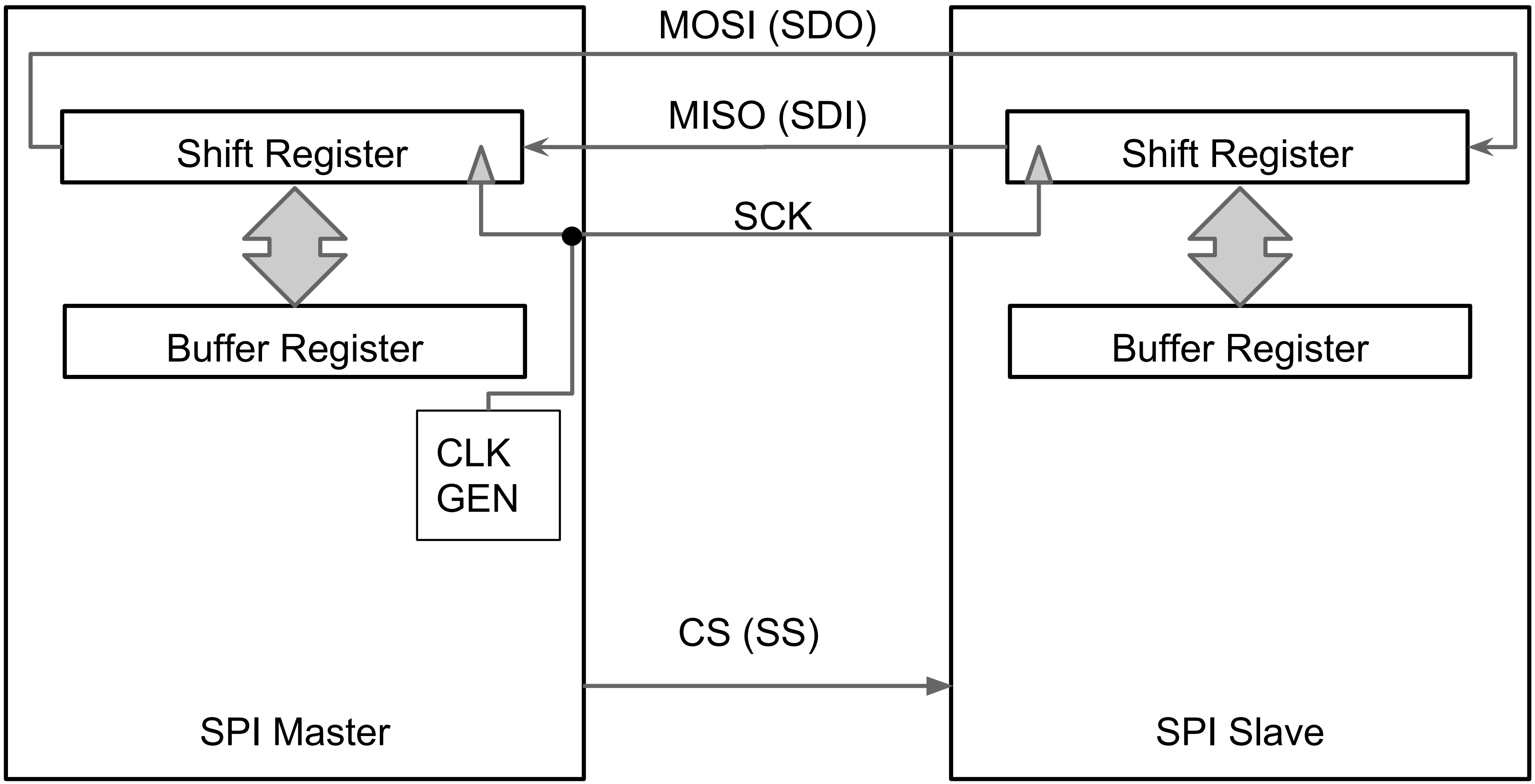
Phwew! Onto SPI

- We now know about RS-232, and I2C
- SPI is another serial protocol for communicating between chips
- 4-Wire Bus
  - MOSI (Master Out/Slave In) - Data from master to slave(s)
  - MISO (Master In/Slave Out) - Data from slave(s) to master
  - SCK (Shift Clock) - Clock signal generated by master
  - SS or CS (Slave Select / Chip Select) - Enable bit for slave devices

SPI Data Formatting

- The speed, data format (MSB/LSB 1st), error correction, etc. is entirely device dependent
  - There are no packets, read/write, address, etc.
  - The Master controls everything
- This allows for much faster operation than I2C!
  - (ie. NRF24 can operate at up to 10 Mbps!)
  - SD Cards can do SPI

SPI Data Flow

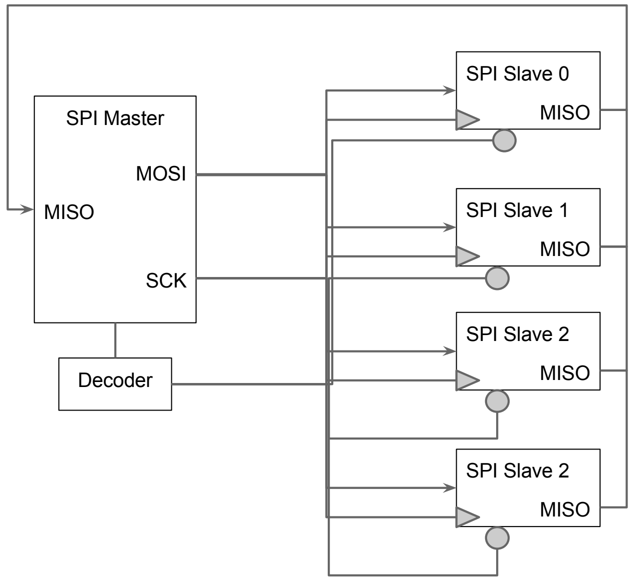


SPI Order

1. Master asserts (chip) slave select
2. Master primes data (from buffer) with first bit in shift reg.
3. Master asserts clock
    1. shifts 1 bit out of master register on one end
    2. shifts 1 bit into shift register on slave end
    3. shifts 1 bit out of slave register into master
    4. shifts 1 bit into master on opp. end of out bit
4. Master desserts clock
5. Primes data (from buffer),repeat 3
6. When done, deassert chip(slave) select

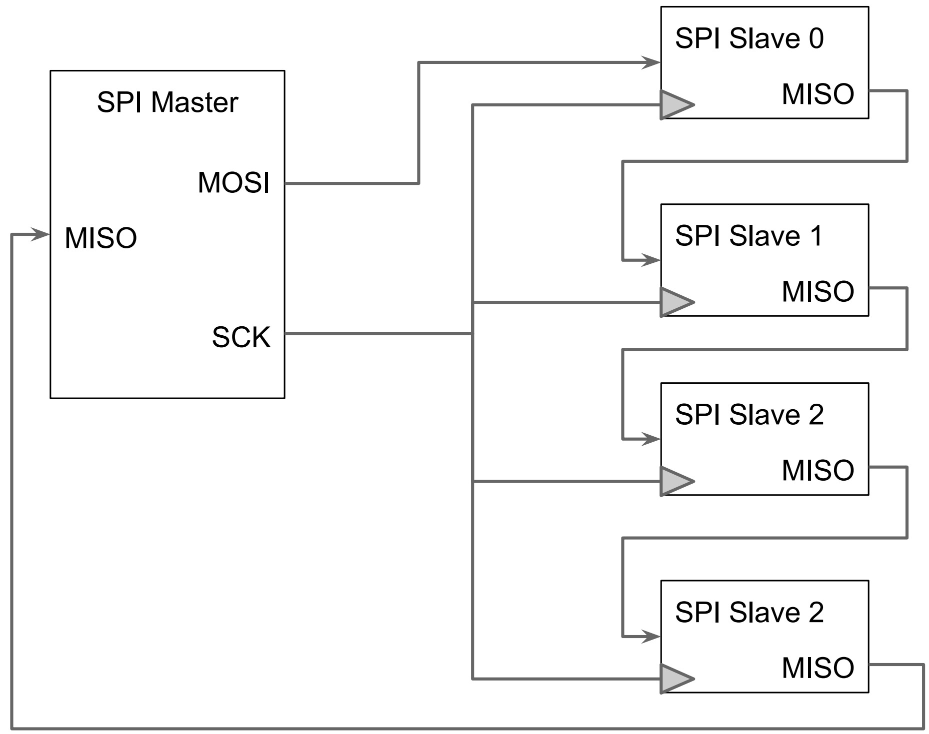
SPI Common Bus

- Decoder selects appropriate CS for device
- Pros:
  - Fast, Direct
- Cons:
  - Requires more pins/traces



SPI Daisy Chain

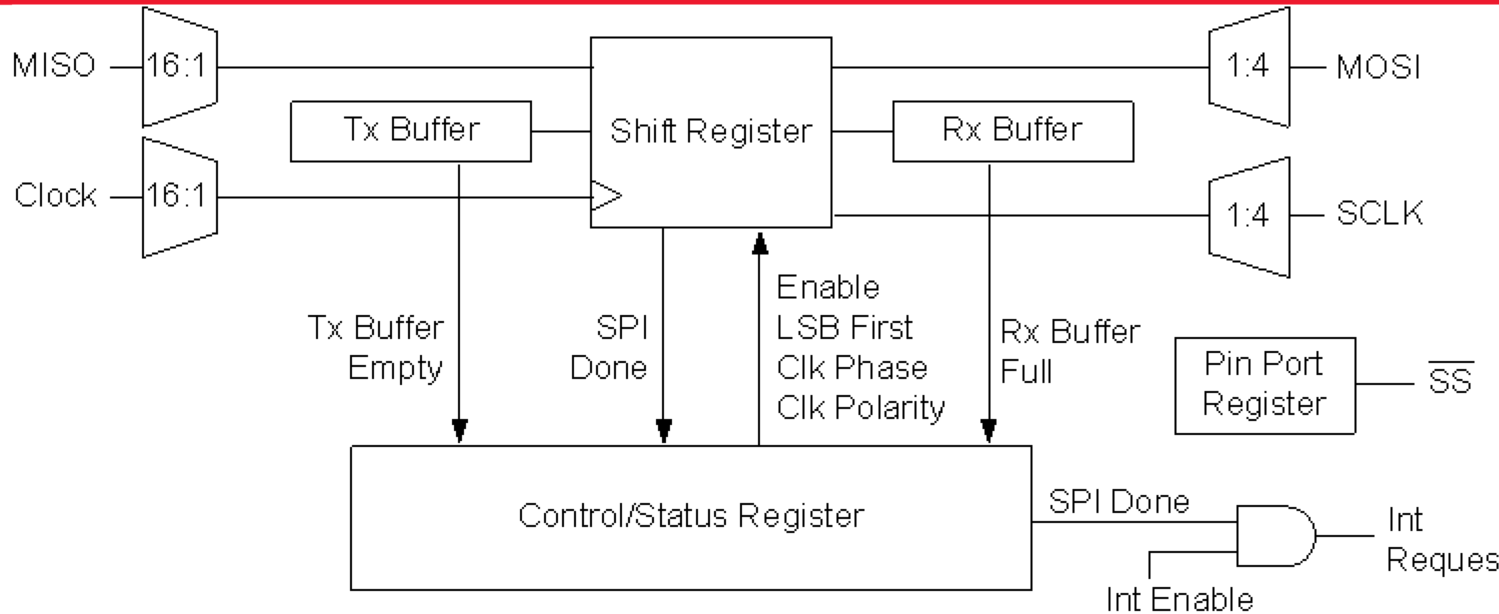
- Common CS line
- Pros:
  - Fewer wires
- Cons:
  - Slower!



Things to Keep In Mind

- For every bit shifted out, there is one shifted in and visa versa (As old TA says, “It’s like a chainsaw.”)
- To read data, dummy shifts may sometimes be necessary
- Since there is no set protocol, read datasheets!
  - Specify minimum timing (SPI may be faster than I2C)
  - Specify data format (Not just 1-byte packets always)

PSoC SPI Implementation



- Everything comes in, goes out through shift register
- Tx Buffer is used to load the shift register
- Rx Buffer is used to pull data from shift register
- Control/Status Register keeps track of buffer(s) status, SPI mode, data format

SPI Sequence on PSoC

1) Assert CS (bring low)
2) Make sure Tx Buffer is empty (TxBufferEmpty
3) Load Tx Buffer with byte to transmit (clears TxBufferEmpty bit)
4) Shift Register is loaded from Tx Buffer (sets TxBufferEmpty bit)
    1) Can now load Tx Buffer with another byte to transfer (will happen immediately after current → Potentially faster)
5) For each bit, SCK is generated to shift it out on MOSI and shift in bit on MISO
6) Ater all bits transmitted/received, shift register is transferred to Rx Buffer, Tx Buffer goes to shift reg.
    1) Sets RxBufferFull & SPIDone bits
    2) Can trigger interrupt or poll flags (Polling is slower ~ 1MHz max)
7) Bring CS high

Pay attention in the NRF24 Template Project!

NRF24L01

- 2.4 GHz Radio – All low level things (sync., framing, error checking, retransmission, etc.)
  - Divides ISM spectrum into 126 sub-channels
- Cheap (~$2 each)
- 3.3V Part (NOT 5V!!! – USE LD4391 regulator for +3.3V)
  - However, pins are 5V tolerant (weird, right?)
- Uses SPI for communication
- Sender & Receiver configs. must match

NRF24 Configuration

- Config. happens like thermostat chip
  - Command Byte + Argument Byte(s) (Table 20 – pg. 51 in datasheet)
  - R_REGISTER/W_REGISTER differ by 1 bit
    - R: 000A AAAA where A AAAA is address of register
    - W: 001A AAAA where A AAAA is address of register
    - Section 9.1 Shows Register Map
    - Note, some registers, like pipe address registers are multi-byte registers
  - Writing certain values to certain registers configures things like auto. acks, payoad size, crc length, rf channel, rf speed, etc.
- Data is sent/received through addressable pipes
  - TX pipe is used for sending data (Only 1)
  - RX pipe(s) used for receiving data (RX pipe address must match senders TX pipe address!)
    - 6 RX pipes → Can act as hub for up to 5 radios
  - Pipes are actually FIFO memory on chip (more in a second)

NRF24 Configuration for Lab

- To properly talk to the receiver, the nrf must be configured the same, and the pipe addresses must match for sender and receiver
- Set the number or retransmissions to 15 with a 4000 us delay between retransmissions
- Set the nrf24L01 to use channel 0x60
- Set the data rate to 250 kbps at maximum power level
- Set the transmit address to 0xc2c2c2c2c2
- Set the receive address of data pipe 0 to also be 0xc2c2c2c2c2
- Set the receive address of the transmitter (RX_ADDR_P1) to be 0xe7e7e7e7e7
- Set the payload size to be fixed at 8 bytes
- Turn on auto-acknowledge for your data pipes
- Turn on your data pipes
- Set the config register to do 2-byte CRCs, power up the radio, and put it in transmit mode

Retransmit Example

- Set the number or retransmissions to 15 with a 4000 us delay between retransmissions
  - From Register Map in datahseet: 04 is SETUP_RETR register
  - Bits 7:4 are ARD (Automatic Retransmission Delay)
    - 0000 – wait 250 us, 0001 wait 500 us, … 1111 wait 4000 us (Each increment adds 250 us to delay)
  - Bits 3:0 are ARC (Auto. Retransmit Count)
    - 0000 – Disable retransmit
    - 0001 – 1 automatic. retransmit, 0002 – 2 auto. retrans, …, 1111 – 15 automatic. retransmits
  - So, we would want to write 0xFF to register 04 for 15 auto. retransmits that are spaced 4000 us apart
  - This would be 2 SPI writes
    - 1st Byte: W_REGISTER at address 04 (001A AAAA → 0010 0100 → 0x24)
      - Note in the API this is already defined as NRF_WRITE_SETUP_RETR
    - 2nd Byte: ARD = 1111, ARC = 1111 → 1111 1111 → 0xFF

Data Flow

- To send data:
  - First send the NRF_WRITE_PAYLOAD byte (0xA0), followed by a number of bytes equal to the payload size (8 in our case)
  - After sending, poll the IRQ bit and wait for it to go to zero. This indicates either a successful transmission (due. to auto ACK) or that the max. number of retries has been reached
    - Proper error checking would read the STATUS (07) register and check the TX_DS or MAX_RT bits to see which interrupt caused the IRQ → WE DON'T HAVE TO FOR THIS LAB!!!
  - After the IRQ bit goes low, clear the appropriate flags by writing 1's to them in the nrf24L01 status register (07).

Data Format For Lab

- You should send the string “#: YYY.Y” to the receiver, where # is your lab station letter (ask if you don’t know) and YYY.Y is the string you would display (minus the degree symbol)
  - # is the lab station letter:
    - A is group on far wall closest to whiteboard
    - B is group immediately to their right
    - Wraparound to the middle table from back wall
    - Wraparound to the printer and work back towards wall
    - Group closest to cabinets would have highest letter
- Sample hex file sends “#: 381.0” → Use to test nrf24 wiring!