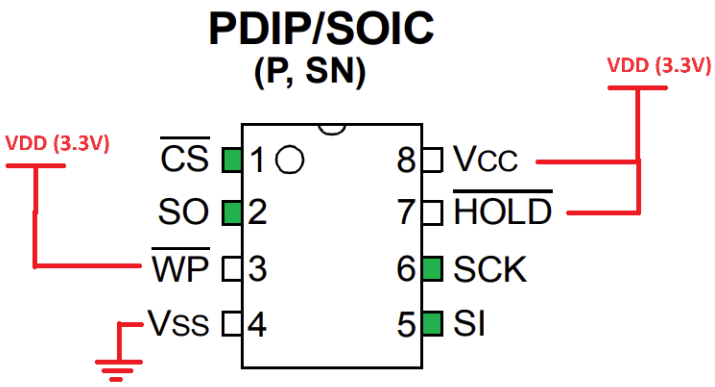
Goals:

- How to communicate with an external device using SPI
- How to write a firmware for a device



Setup:

- The device we are using is a 1 Kbit Serial Electrically Erasable Programmable Read-Only Memory (EEPROM) 25LC010A-I/P-ND
- Use SPI1 to communicate with the device.
- Connect the device **VDD to 3.3V** and **VSS to ground.**
- Chip select (CS_), Slave Data Out (SDO), Slave Data In (SDI), and SPI Clock (SCK) must be connected to the MCU using pins PE12, PE14, PE15, and PE13.
- You may manually generate the chip select signal (by software) or use the SPI dedicated SS signal.

Requirements:

- Design and create full firmware to communicate with the provided device.
- Firmware usually provides an abstraction layer of the hardware so that users have useful hardware-independent high-level functions.
- This can be done by creating a hierarchy of functions where few low-level functions are needed. For example, you will need a function to initialize the SPI device, one to check its status, one to write/read a single byte, one to write a page.
- Candidate functions (just a suggestion, you can define your own):

```
void initMem()                          // Initialize Memory
char readMem(uint16_t addr)             // Read a byte at <addr>
void writeMem(uint16_t addr, char data) // Write data to addr, returns
                                        //  number of bytes written
char void getStatus()                   // Get the device status
```

- **To test the firmware:**
  - Create a write pointer (*PTR*) that keeps track of the first empty location in the EEPROM. Assume that *PTR* starts at address 0 when a program starts.
  - Using LPUART1, the user may enter any message. Upon pressing ENTER, the message will be written to EEPROM at PTR and the number of bytes written will be printed on the serial terminal. PTR will now point at the address following the last byte that was written. The next message will be written to PTR which will be updated again.
  - Pressing the blue button will send all stored messages to serial terminal. This will reset PTR back to 0.
- Do not use any prebuilt high-level functions but you may use your functions developed in previous labs. Like what you learned in class, write your program in a register level abstraction. Add comments to each line in your code.