



Interfacing!

Objective



Your task now is to DESIGN:

- to achieve objectives outlined in the Requirements Specifications
- while observing constraints – including course time!

You already have much knowledge of many circuits and principles that will help.

Research will be required to fill in any gaps!



Objective

Designing a modern electronic system requires a good deal of up-to-date *Product Knowledge*.

This set of slides is meant to help by considering “interfacing”. This will help, if applicable, to select an MCU for your design.

Recall the goal: systems-level design!

Heads-up: Course-centric Platforms

The lab is equipped to best handle particular MCUs:

Atmel AVR MCUs (e.g. Atmega32)

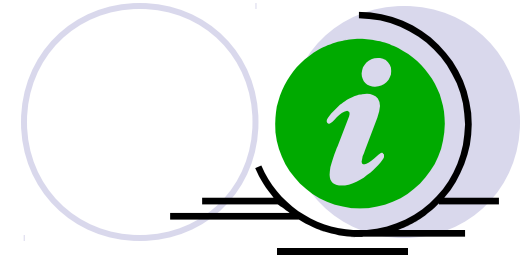
PIC 16F87X MCUs (e.g. 16F877A)

You do ***not*** need to select these devices.

You may have a trade-off to balance:

(better-suited device) vs. (effective guidance, on-hand equipment, ...)

Sources of Information



- Device datasheets and their Errata are the authoritative source of information.
- Manufacturers often have more general information about the family of microcontroller. (e.g. The PIC Midrange Microcontroller Reference)
- Textbooks.

You will get to the point that a device datasheet is often all you will need!

Interfacing 101

The title 'Interfacing 101' is positioned on the left side of the slide. The number '101' is enclosed within a light purple circle. To the right of the title, there are three more circles: a solid light purple circle, a light purple circle with a thin outline, and another solid light purple circle.

When “interfacing” two systems, there are things that need to be considered:

1. Voltage levels;
2. Current; and
3. Timing.

...and maybe other things, like the signal protocol or circuit power consumption.

All need to be appropriately addressed.



Interfacing 101: Final Exam!

Consider the situations on the handout.
(This isn't really an exam...)

Some errors are likely obvious. Others may only reveal themselves through learning how the parts work!



Peripherals

- Peripherals that form part of an MCU, FPGA design, or stand-alone IC can make a resulting design more efficient.
- Manufacturers are constantly creating peripherals that are meant to help with aspects of interfacing.
- Knowledge of what is available is key and will be part of what dictates MCU selection, if applicable!



Peripherals: Digital I/O

- Physical pins on MCUs can often be used for general-purpose digital input or output.
- (Beware that many pins are also associated with another function!)
- Different pins have different current sinking and sourcing limits, as outlined in the datasheet “DC/Electrical Characteristics” section.

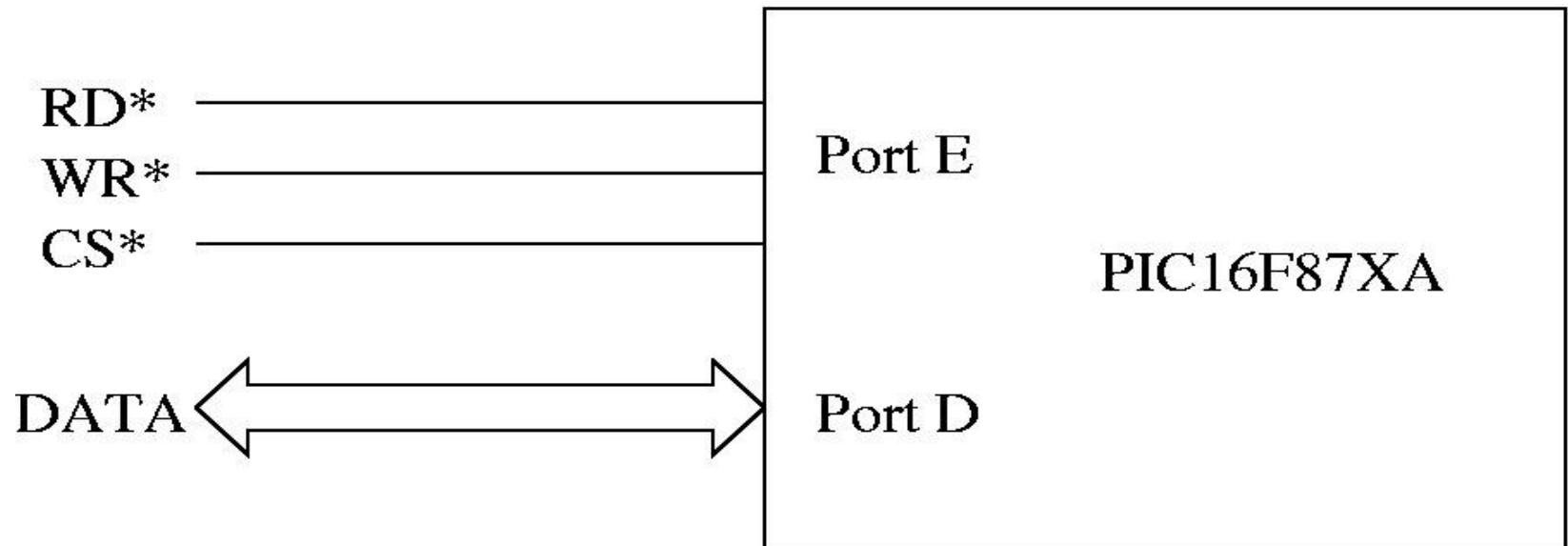
Peripherals: Digital I/O (cont'd)

- Whether a pin is an input or output is established via a *data direction register*.
- The data is read or written through a *data register*.
- It is good practice to establish the output level prior to making a pin an output!
- Digital I/O comes in useful for *many* things: turning on LEDs, reading switches, etc.
- Note that Digital I/O can often be made to *emulate* another peripheral via firmware!

Peripherals: Parallel Slave Port

- This peripheral is offered by some PIC16F87XA devices.
- Used when you would like the MCU to act like a memory-mapped device: control signals (RD^* , WR^* , CS^*) and data are included.
- This could be useful when the system is connected to a PC printer port!

Peripherals: Parallel Slave Port (cont'd)





Peripherals: Timers and Counters

- Timers typically start counting up from 0 when the device comes out of reset.
- These are used to provide timing for other peripherals.
- Timers can be used to pace execution of the firmware.
- The clock signal for these timers can come from various sources (internal or external) and can often be adjusted via a “prescaler”.
- Useful for counting incoming pulses!

Peripherals: Input Capture



- This works by “capturing” an associated timer’s value when a signal edge is encountered.
- Achieved by hardware!
- The edge that capturing occurs on is often configurable.
- This facility is used to measure input pulse widths and periods (and therefore frequency).



Peripherals: Output Compare

- Opposite of input capture.
- When the associated timer's free-running value is equal to the “compare value”, a pin can:
 - Be set high;
 - Be cleared;
 - Left alone (set a flag and perhaps generate an interrupt).
- Used to generate output pulses, waveforms, and/or generate a periodic interrupt.



Peripherals: Pulse Width Modulator

- This system is very similar to the Output Compare, except there are registers for setting the duty cycle and period.
- Used for generating pulse-width modulated waveforms.
- Very useful for controlling motors, among other things, even creating a D/A converter!



Peripherals: Pulse Accumulator

- A pulse accumulator is typically a timer that uses an external clock input to count the number of incoming pulses.
- Useful for things like wheel (shaft) encoders.
- The same functionality can be achieved using an interrupt input pin and a small ISR.

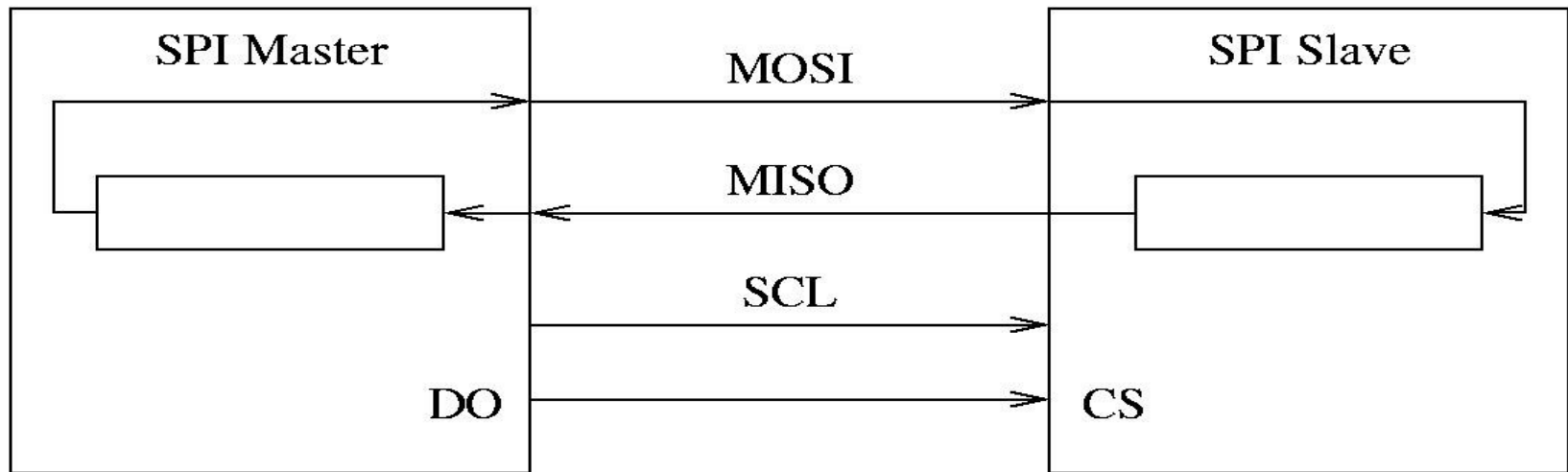
Peripherals: Synchronous Serial

- Recall that synchronous communication involves transmission of a clock signal.
- The clock is usually generated by the MCU i.e. The MCU is the master, coordinating data transfer.
- There are several synchronous serial protocols:
 - Serial Peripheral Interface (SPI)
 - Inter-IC (I²C) Interface
 - Two-Wire Interface (TWI)
- A variety of devices are available that use these protocols: A/D and D/A converters, memory, etc.
- MCUs can be networked!

Peripherals: Synchronous Serial – Serial Peripheral Interface (SPI)

- This interface uses three signals: serial data out, serial data in, and the transfer clock.
- A chip select line is also often needed.
- This is the simplest of the three synchronous serial protocols outlined here.

Peripherals: Synchronous Serial – Serial Peripheral Interface (SPI)



Peripherals: Synchronous Serial – I²C and TWI

- These interfaces use only two signals: a bidirectional serial data line and the transfer clock.
- The protocols are more complex as they include overhead to establish direction of the data line.
- Chip select lines are not required because the protocols include addressing schemes.
- A two-wire interface (TWI) is often capable of I²C.

Peripherals: Asynchronous Serial

- Asynchronous serial communication occurs via the asynchronous portion of the subsystem commonly referred to as a USART (Universal Synchronous/Asynchronous Receiver/Transmitter)
- Sometimes this is also referred to as an SCI (Serial Communications Interface)

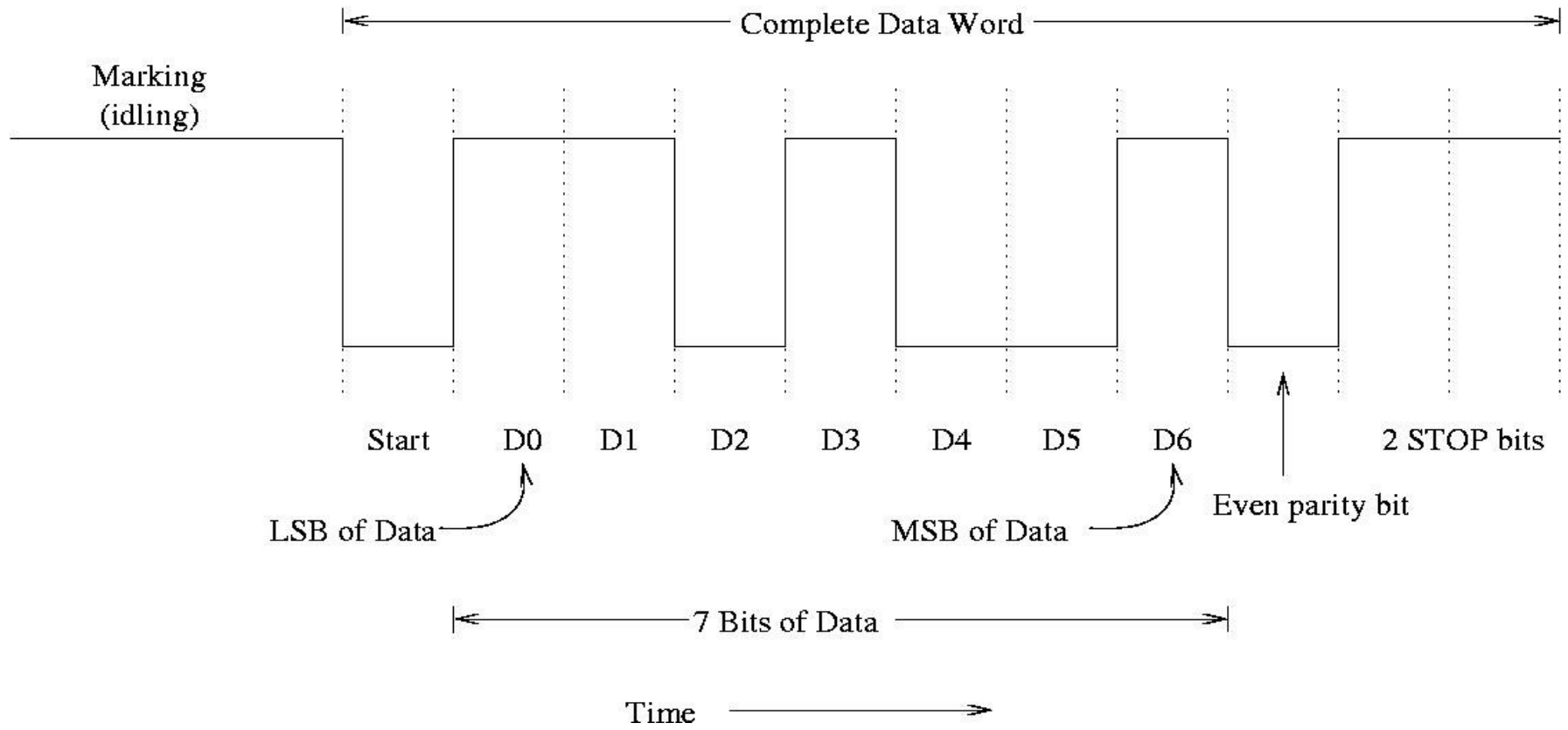
Peripherals: Asynchronous Serial (cont'd)

- In asynchronous communications, only data is sent (no clock).
- Timing is provided independently at both the transmitting and receiving ends by baud rate generators.
- These baud rate generators must match in order for communication to be successful.
- The MCU's timing element will allow for a certain set of baud rates to be generated.

Peripherals: Asynchronous Serial (cont'd)

- The number of data and stop bits transmitted and expected can often be set.
- *Parity* is a simple error-detection scheme that establishes the number of high bits transmitted to ODD or EVEN.
- Asynchronous serial is convenient for having your system communicate with a PC via its RS-232 serial port, although voltage level-conversion is required.

Peripherals: Asynchronous Serial (cont'd)





Peripherals: A/D Converter

- Many devices contain a built-in A/D converter, often with an analog multiplexer ahead of it to provide multiple input channels.
- The voltage reference can often be established (within tolerances).
- A/D converters are used whenever you need to gain a digital measure of an analog signal!

Peripherals: Analog Comparator



- Analog comparators on MCUs are just like their counterparts in the op-amp world.
- Some peripheral versions allow the selection of particular reference voltages, generation of interrupts, etc.

Peripherals: USB



- Several MCUs now offer USB functionality augmented by the use of appropriate firmware.
- In lower-end microcontrollers, these are almost always *only for implementing endpoints*. i.e. They are **not** host controllers and are not intended to connect to things like flash drives, cameras, etc.



Peripherals: Ethernet

- Several MCUs now offer Ethernet connectivity.
- Again, this is augmented by the use of appropriate firmware to perhaps provide TCP/IP functionality, and even some protocols that run on that:
 - http (web-page)
 - smtp (email)
 - ...

Peripherals: Miscellaneous

- The peripheral subsystems we have covered so far are only a sampling of those most commonly encountered in MCUs.
- Note that we did not touch on the interrupt capabilities of these peripherals!
- Other facilities which many microcontrollers offer are:
 - In-Circuit Debugging
 - External Interrupt Pins (Keypad connection?)
 - Brown-out detection
 - Watchdog timer
 - Low power modes
 -
- All microcontrollers also offer *memory*: the discussion of this is reserved for later.

Recommendation: Inward vs. Outward Design

- Now you are better-prepared by knowing what typical modern MCUs and FPGAs offer.
- It would be normal to work (inward) from the boundaries of your design to determine what peripherals are required. In turn, this will help with the selection of an MCU or FPGA.
- If you *knew* which device you were using, however, you may include the use of subsystems that take advantage of particular peripherals (outward).

Recommendation: Inward vs. Outward Design

- You are free to follow the inner design approach, bearing in mind the project constraints, including budget.
- If you are *truly lost or overwhelmed*, then you will be nudged down the Atmel AVR path:
 - Atmega32 (tightest)
 - 8-Bit AVRs supported by JTAG Mkl programmers
 - AVRs supported by JTAG MkII programmers (loosest)
- These will not be appropriate for all projects.

Recommendation: Through-Hole/Reading

- In this course it is best if you use technology that is *through-hole* rather than surface mount!
- Start reading about peripherals that you suspect will be used in your design, even prior to selecting the main computational element. (An Atmel AVR PWM peripheral, say, operates almost identically to a Microchip PIC PWM peripheral!)
- More on MCU selection later!
- Start learning how to use Eagle Schematic Capture!