The University of Texas at Arlington

Lecture 8
Hardware Connections

CSE 3442/5442

PIC18F458 Pin Diagram

- DIP (dual in line package) shown
  - QFP (quad flat package) and LLC
    (leadless chip carrier) is also available
- Vdd typical 5V but can be set; 2 pins
- Vss (GND) 2 pins (ground bounce noise reduction)
- OSC1 and OSC2: for external quartz oscillators 0-40MHz, configurable
- MCLR – master clear reset – creates power on reset
PIC18F458 Power-On Reset Circuit

At MCLR:
- PC, WREG, SP reset to 0
- TRISA-TRISE reset to FF

Minimum Connections on a Pic18F (With a Crystal)
**Configuration Registers**

- In ROM beyond addressable
- CONFIG directive in the source code is used to write to it, e.g.,
  - `CONFIG OSC=HS`
  - `CONFIG OSCS = OFF` OR
  - `CONFIG OSC=HS , OSCS = OFF`
- MPLAB can directly manipulate them inside a project
- Can be accessed through table reads and writes from the microcontroller
- CONFIG5 and up are in general for code protection settings

---

### CONFIG1H - Oscillator

| Bit 7 | Unimplemented, Read as 0 |
| Bit 6 | OSCS | Oscillator System Clock Switch Enable bit |
| Bit 5 | OSCS | 1 = Oscillator System clock switch option is disabled (main oscillator is source) 0 = Oscillator System clock switch option is enabled (oscillator switching is enabled) |
| Bit 4 | Prescaler Value, Read as 0 |
| Bit 3-0 | FOSC[2:0] | Oscillator Function bits |

- **FOSC default is RC (111)** – on chip oscillator with RC circuit attached to OSC1
- **EC = external oscillator connected to OSC1**
- **HSPLL, HS, XT, LP, are crystal options with stability and power consumption (in this descending order) differences (HSPLL quadruples the frequency)**
- **There is an internal 32kHz oscillator that can be enabled by the OSCEN bit**
Clocks

### Osc Choice vs Crystal Freq

<table>
<thead>
<tr>
<th>Osc Choice</th>
<th>Crystal Freq</th>
<th>C1 range</th>
<th>C2 range</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP</td>
<td>32 kHz</td>
<td>33 pF</td>
<td>33 pF</td>
</tr>
<tr>
<td>LP</td>
<td>200 kHz</td>
<td>15 pF</td>
<td>15 pF</td>
</tr>
<tr>
<td>XT</td>
<td>200 kHz</td>
<td>47−65 pF</td>
<td>47−65 pF</td>
</tr>
<tr>
<td>XT</td>
<td>1 MHz</td>
<td>15 pF</td>
<td>15 pF</td>
</tr>
<tr>
<td>XT</td>
<td>4 MHz</td>
<td>15 pF</td>
<td>15 pF</td>
</tr>
<tr>
<td>HS</td>
<td>4 MHz</td>
<td>15 pF</td>
<td>15 pF</td>
</tr>
<tr>
<td>HS</td>
<td>8 MHz</td>
<td>15−33 pF</td>
<td>15−33 pF</td>
</tr>
<tr>
<td>HS</td>
<td>20 MHz</td>
<td>15−33 pF</td>
<td>15−33 pF</td>
</tr>
<tr>
<td>HS</td>
<td>25 MHz</td>
<td>15−33 pF</td>
<td>15−33 pF</td>
</tr>
</tbody>
</table>

---

### CONFIG2L – Initial Transients

- Controls stable voltage and frequency during reset
- Power up timer (PWRT) keeps the processor start up so that supply voltage can be stabilized
- Oscillator start up timer does the same for the oscillator
- Brown out suspends the processor if supply voltage falls below a set level
- Recall: the higher the frequency the higher of a supply voltage we need
CONFIG2H – Rottweilers

- Errrr, watchdog timers
- Can verify if the system still is executing meaningful instructions or is stuck (ROM corruption, noises, PC failures) and issue a reset
- WDTPS determines the time-out

CONFIG4L – Debuggers, etc.

- There are special purpose in-circuit debuggers that can be used with PICs (RB6 and RB7 unusable)
- SVREN: Stack has only 31 locations if it over (or under) flows we can ask for a reset
- LVP: low voltage in circuit programming bit (RB5 becomes unusable)
LIST Directive

- The LIST directive can be used to tell the assembler about our code reading needs
  - LIST P=18F452, F=INHX32, MM=OFF, ST=OFF, X=OFF

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B=nnn</td>
<td>Set tab space (8)</td>
</tr>
<tr>
<td>C=nnn</td>
<td>Set column width for prints (132)</td>
</tr>
<tr>
<td>F=format</td>
<td>Hex file output (INHX32 (32bit ROM address), INHX8M (16bit ROM address), INHX8S (split – two files one for high one for low bytes))</td>
</tr>
<tr>
<td>MM={ON/OFF}</td>
<td>List file to contain memory map (on)</td>
</tr>
<tr>
<td>N=nnn</td>
<td>Lines per page for prints (60)</td>
</tr>
<tr>
<td>P=type</td>
<td>Set microcontroller type</td>
</tr>
<tr>
<td>R=radix</td>
<td>Hex, dec, or oct (hex)</td>
</tr>
<tr>
<td>ST={ON/OFF}</td>
<td>List file to contain symbol list (on)</td>
</tr>
<tr>
<td>X={ON/OFF}</td>
<td>Macro expansion (on)</td>
</tr>
</tbody>
</table>

How About C?

- It is simple to do all this from C as well: insert a #pragma in front of config
  - E.g., #pragma config OSC=HS
So, What’s in a HEX File?

- **BB**: how many bytes in the line, max value 0x10H
- **AAAA**: ROM address space where data needs to go (only 16 bits!)
- **TT**: 00 means hex file is continuing 01 means it is over; in INHX32: 02 segment address, 04: HHHH is used to switch to another bank (A31;A16)
- **H…H**: Data (at most 16 in a line – see BB)
- **CC**: checksum

We have the Hex File, What Now?

- We need to burn the code onto the microcontroller
- Three basic methods:
  - Off-circuit: the microcontroller is programmed as a stand alone chip and is then inserted into the circuit
  - In-circuit: microcontroller has to set aside pins that are used to program it while inside the circuit
  - Boot loader: a special code running on the microcontroller, allowing it to accept code from any of its interfaces (boot loader needs to be burned with one of the previous two methods)
Off-circuit Programming

- Special purpose programming tools need to be used
- Usually “high” voltages are needed
- Chip manufacturer usually sell them but many third parties do it as well
- Specs on how it needs to work are provided by the chip manufacturer
- Usually can be built at home easily (may not be important)
- Usually have zero insertion force sockets (ZIF)
- Have to be careful about electrostatic discharge (ESD)

In-circuit Programming

- In circuit Serial Programming (ICPS) uses RB7 and RB6
- These pins can be reused after programming but the designer needs to make sure they do not interfere
**Boot loader**

- See our QwikFlash boards
- Takes away ROM space from the developer
- Can use any communication methods available to the microcontroller
- Code space for boot loader must be reserved and protected
- May be a good choice for development but usually not for final product

**Summary**

- Microcontrollers need external connections
- Usually there are many options for clock sources (some microcontrollers have two clock sources)
- Microcontrollers have to be configured (by burning configuration values to special registers)
- After code is compiled and linked it needs to be burned onto the microcontroller (for which there are three general methods)