Today: Introduction to the PIC Architecture II

Continue our tour of the primary components of the PIC 16F628

- Digital I/O
- Timers
- Analog input
- Analog comparators
- Serial hardware
- Watch-Dog Timer

Sensor of the Day...

System Debugging Hints

- Build and test incrementally
- When things stop working: go back to the basics
  - Are the components getting power?
  - Does the PIC boot? How to tell?
- Reflecting internal state:
  - Flashing LEDs
  - Plenty of printf()'s

A Bit More on CCSC

- getc(), putc(): get and put a character
- gets(), puts(): get and put strings
- printf(): generate formatted strings
- kbhit(): indicates if a character is arriving over the serial line

Standard Set of Mathematical Ops

- cos(), abs(), tan(), atan(), sqrt()...
- +, -, *, /, += ....
- rand() and srand()
- <<, >>, ||, && ...

Some Non-Standard Ops

- shift_left(), shift_right(), rotate_left() ...
- bit_set(), bit_clear(), bit_test()
- swap(): swap the type nybbles in a byte

Standard String Manipulation

- stoi(): ascii to integer
- tolower() and toupper(): case conversion
- strlen(): string length
- strcpy(): string copy
- strcmp(): string compare
- sprintf(): formatted output to a string

- Remember: in C, the last character of a string must be 0x0
  - so make sure you plan for the extra byte

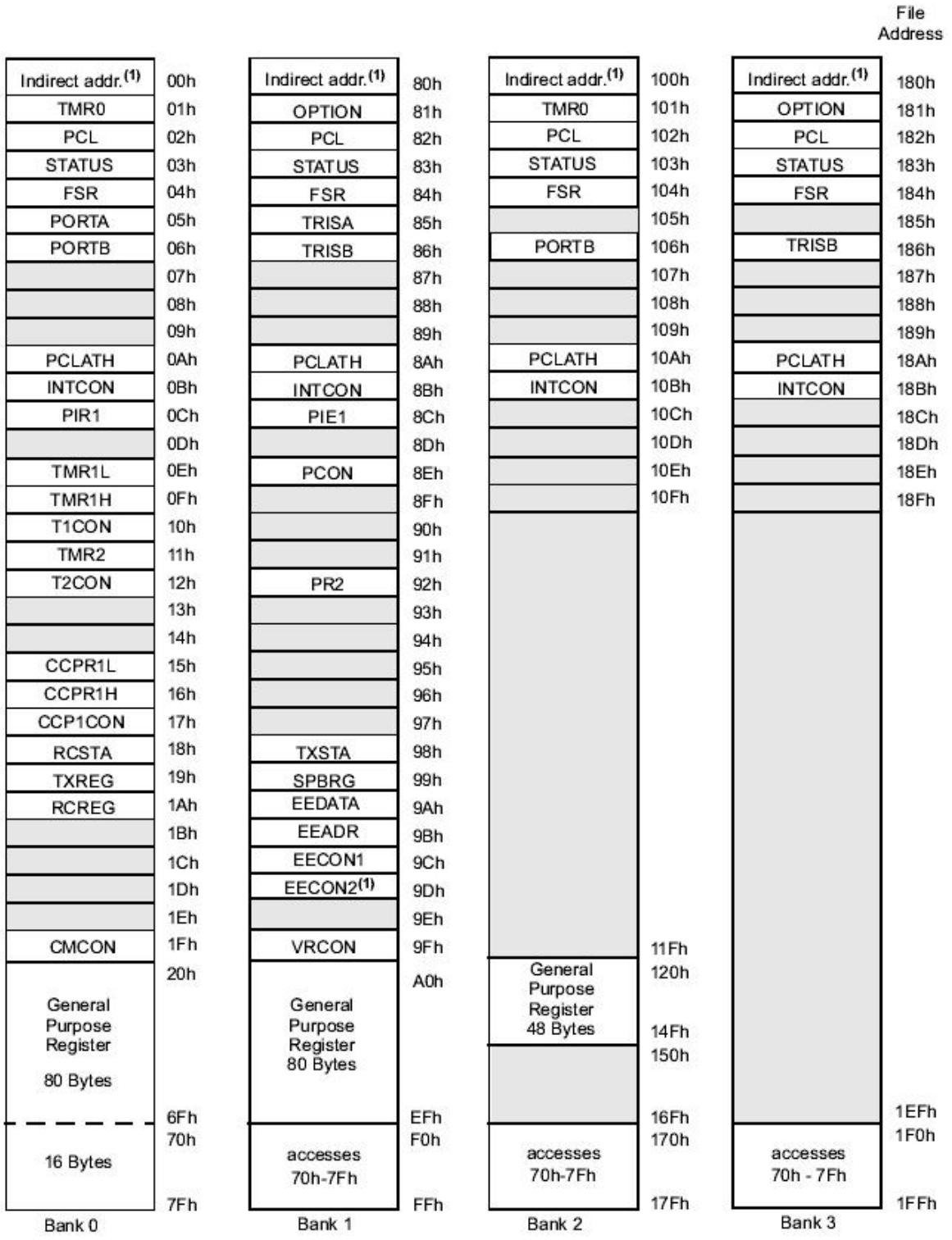
Timing

- delay_ms(): delay for the specified number of miliseconds
- delay_us(): micro seconds

- Note that the compiler simply generates the right number of loop cycles to implement these
  - Time spent in interrupts does not count!

Data Memory

- The names of the specialized registers are shown
- In many cases (but not all), a program may read or write to these registers directly



The Status Register

STATUS REGISTER (ADDRESS: 03h, 83h, 103h, 183h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

- Different bits have different functions (this is the case for most specialized registers)
- In the case of the SR, most bits are set as a result of executing an instruction

Instruction Set: Byte-Oriented

- Add the contents of register 0x45 and W and place the result in W:

```
ADD 0x45, W
```

- And the contents of register 0x88 and W and place the result in 0xf8:

```
AND 0x88, F
```

- Clear register 0x67:

```
CLRF 0x67
```

Other Byte-Oriented Instructions

- Move the contents of W to location:

```
movwf 0x43
```

- Rotate left through carry (source: 0x31, dest: W)

```
rlf 0x31, W
```

Bit-Oriented Instructions

- Clear bit:

```
bcf 0x45, 4
```

- If bit 2 is clear, then skip an instruction:

```
btfsc 0x45, 2
```

Misc and Control Instructions

- Jump to location

```
goto 0x89
```

- Sleep:

```
sleep
```

- Return from interrupt:

```
retfie
```

- Return with literal:

```
retlw 0x67
```

Interrupts

- Why are there no instructions for interrupt enable/disable?

Interrupts

- Why are there no instructions for interrupt enable/disable?
  - Interrupts are manipulated by writing directly into the correct register

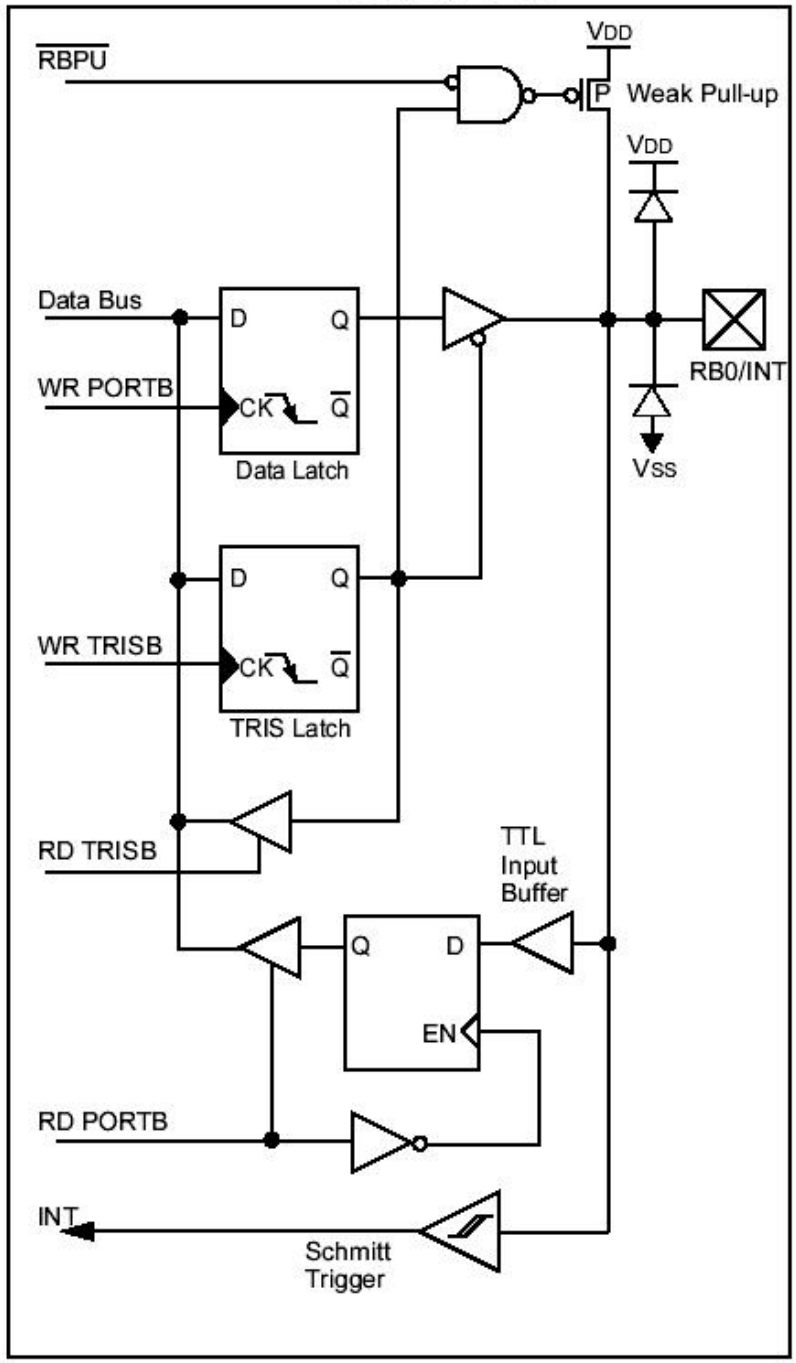
I/O Ports (Port A and Port B)

- Both:
  - 8 pins
  - Can be configured as generic digital I/O pins
  - Some pins can be used for external interrupts
- Port A:
  - Analog inputs (2)
  - Analog comparator function possible
- Port B:
  - Serial I/O
  - Pulse-Width Modulated (PWM) generation, reading

Port B, Pin 0 Circuit Diagram

- Functions provided by this pin:
  - Digital output
  - Digital input
    - with or without a weak pullup
  - External interrupt
    - Schmitt Triggered
- Tristate latch:
  - Selects input or output

FIGURE 5-8: BLOCK DIAGRAM OF RB0/INT PIN



Setting the Tristate Latch

- In C:

```
set_tris_b(0xF0);
set_tris_a(0b00001000);
```

- 0 = corresponding pin is an output
- 1 = input
- When not in use, it is best to configure as an input
- You only need to use these functions if you are using “fast_io” mode (otherwise, the tristate buffer will be configured automatically)

Digital I/O

- Input from a pin:

```
value = input(PIN_A3);
```

- Input from an entire port:

```
value = input_a();
```

- Output to a pin:

```
output_low(PIN_A5);
output_high(PIN_B1);
output_bit(PIN_B2, 0);
```

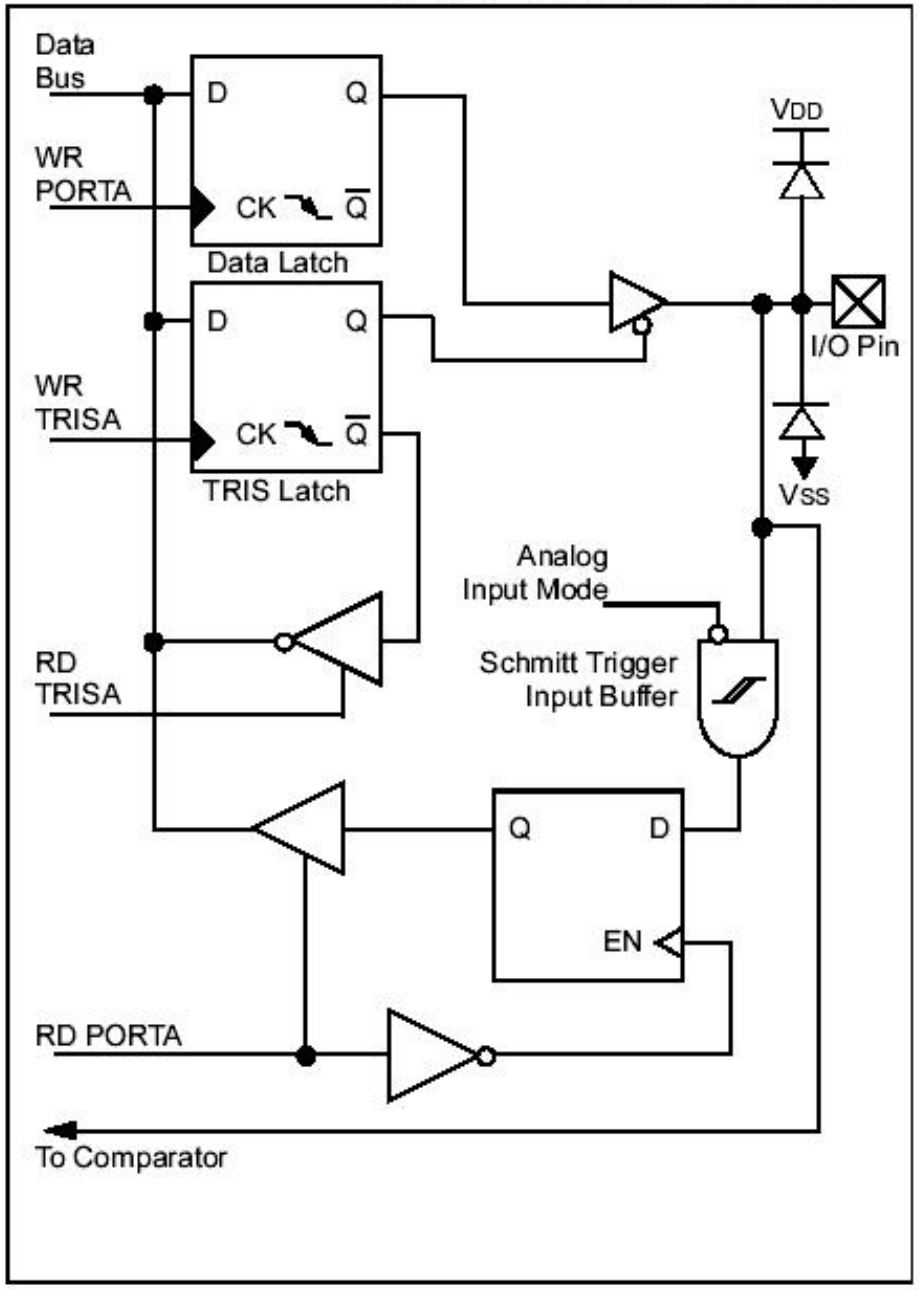
- Output to a port:

```
output_b(value);
```

Port A, Pins 0&1

- Provides:
  - Digital input: no pullup, but Schmitt Triggered
  - Digital output
  - Analog input

FIGURE 5-1: BLOCK DIAGRAM OF RA0/AN0:RA1/AN1 PINS



Timers

- Timer 0: 8-bit counter
  - Internally or externally driven
- Timer 1: 16-bit counter
  - Timing capture (e.g., of a PWM signal)
  - Timing comparator
- Timer 2: 8-bit counter
  - Flexible timing
  - Baud rate generation
  - Pulse-width modulation (PWM) generation

More later...

Watch Dog Timer

- Special timer that operates “independently” of the CPU (has its own oscillator)
- Used to reset the PIC if the processor becomes hung (non-running state) or if the software enters a loop from which it does not exit
- Configuration:
  - `WDT` option in the `#fuses` directive
  - setup_wdt(mode) at the beginning of your code
  - Mode can be one of: WDT_18MS ... WDT_2304MS

Watch Dog Timer (cont)

- Your software must occasionally reset the timer hardware:

```
restart_wdt();
```

Analog Comparators

- Four of the Port A pins can be used for comparing two pairs of voltages
  - One of each pair may be internally generated
  - When one voltage of a pair rises above the other, an interrupt may be generated

Analog-to-Digital Conversion (ADC)

- ADC is the process of representing an analog signal with a number
- The 628 does not contain an ADC, but the 876 contains a 10-bit ADC
  - Voltages from 0 to 5V are represented with up to 10 bits (number of bits is configurable)

Universal Synchronous/Asynchronous Receiver/Transmitter (USART)

- Hardware support for serial communications
  - RS232: asynchronous, bidirectional serial communication
  - SPI: synchronous, bidirectional
  - I^2C: synchronous, unidirectional

Universal Synchronous/Asynchronous Receiver/Transmitter (USART)

- Synchronous communication: there exists a shared clock (typically generated by the “master”)
  - The transition of the clock determines when a bit is transmitted
- Asynchronous: there is no shared clock
  - The two ends of the communication must find a way to agree on when a bit is being transmitted

Event Capture Mode

- When an event is detected on a port B3, the value stored in TMR1 (the 16-bit timer state) is copied to a register CCPR1 and an interrupt is raised
- Useful for measuring the time between events
  - For example, when receiving a PWM signal, this allows us to measure the duty cycle

Event Compare and PWM Modes

- Compares Timer1 state to CCPR1; when they match, toggle the state of the output to port B3
  - Useful for generating regular changes to the output pin
- PWM Mode uses the compare mode to generate a square wave at some specified frequency
  - One can then control the duty cycle of the square wave by writing the appropriate value into TMR2

Processor Manipulation

- Power down the processor, leaving only WDT and a couple of external interrupt detectors active:

```
sleep();
```

- Cause the processor to reset state and begin execution from location 0:

```
reset_cpu();
```

- On restart, your program can query the reason for the restart:

```
value = restart_cause();
```

What does the 16F876 Add?

- Analog input pins
- Ports C and D
-

Next Time

- Lab2: Friday and Monday
- Lecture on Tuesday:
  - What widgets are you building?
  - Digital I/O
  - Reading: Design with PICs Chapter 8