Digital I/O, Switches, and Oscillator Options

Mark Colton

Department of Mechanical Engineering

Brigham Young University

Digital I/O

- Digital input and output (I/O) allow the microcontroller to communicate with or drive lots of different devices
- Examples:
  - Read buttons and switches
  - Drive LEDs
  - Drive LCDs
  - Interface with digital logic chips
  - Drive relays
  - Create your own serial communication systems

Digital I/O States

- Read or write digital “states”:
  - HIGH/TRUE/ON/1
  - LOW/FALSE/OFF/0

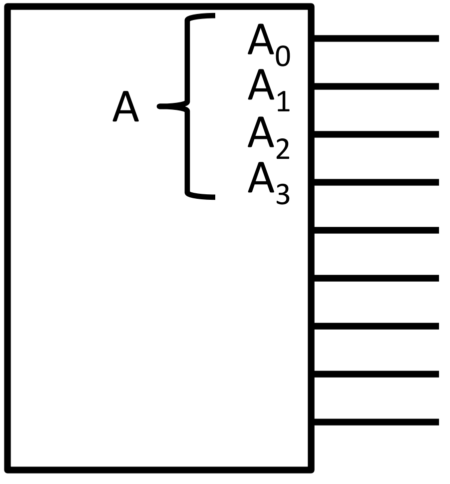
- Digital output (PIC24)
  - HIGH: V_{OH} = 3 V (V_{DD} = 3.6 V) – Depends on V_{DD}
  - LOW: V_{OL} = 0.4 V

- Digital input (PIC24)
  - HIGH: $V_{IH} = 0.8V_{DD}$
  - LOW: $V_{IL} = 0.2V_{DD}$

Source: PIC24F Electrical Characteristics, Tables 29-9 and 29-10

Digital I/O Ports

- Ports
  - Most microcontrollers group their digital I/O lines into ports
  - In most cases this is treated as a digital word
- Example: 4-line digital I/O port called A
  - We can look at this as a 4-bit digital word with bits A_0 (LSB) to A_3 (MSB)
  - If I write 1011 to A, the pins assigned to the bits will generate voltages as described on previous slide
  - I can also read from A to find out what states are applied to the pins



Number Representations

- Binary (1s and 0s) makes sense with ports because we can picture which bit (line) has which value (1 or 0)
- Hexadecimal is a nice (and universally used) shorthand
- Group each 4 bits and assign a value of 0-F (16 possible values because $2^4 = 16$)
- Examples:

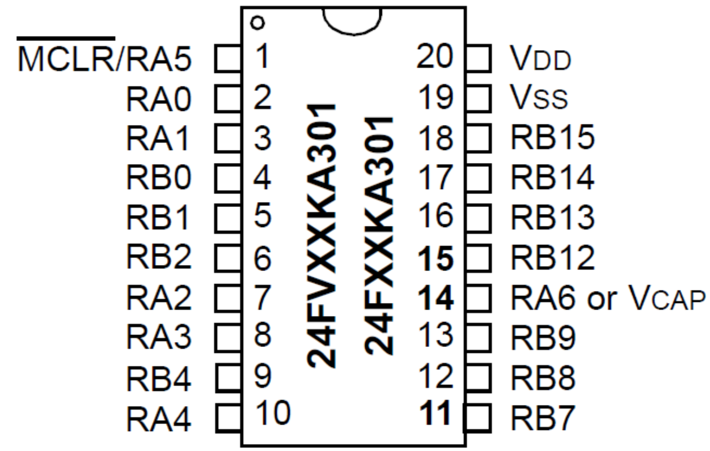
  10110010

  4C3

- In C:
  - Binary: `0b010011000011`
  - Hex: `0x4C3`
  - Decimal: `1219`
- Can “write” any of these numbers to a port and get the same result

PIC24F16KA301 – Ports

- Two digital I/O ports: PortA & PortB
- Data sheet:
  - Table 1-2
  - Table 1-3
- PortA: 7 bits/lines (RA0-RA6)
- PortB: 11 bits/lines, missing chunks
- Frustrating but true: Bits not located next to each other on chip!



PIC24F16KA301 – Data Direction

- PortA and PortB are bidirectional – must configure for input or output

Important note: With microcontrollers, it's all about the configuration

- Direction is set using the data direction register (DDR) associated with a port

- Section 11.0
  - Use TRISA register to set PortA direction
  - Use TRISB register to set PortB direction
  - 0 for output, 1 for input

- Example: To set bits 0-3 of PortA as outputs and bits 4-6 as inputs, write 1110000 to TRISA register:

```
TRISA = 0b1110000;
TRISA = 0x70;
```

PIC24F16KA301 – I/O

- After configuring ports or individual bits as input or output, can write to or read from ports using PORTA and PORTB registers
- Example: output 0100110 to PortA

```
TRISA = 0x00;
PORTA = 0x26;
```

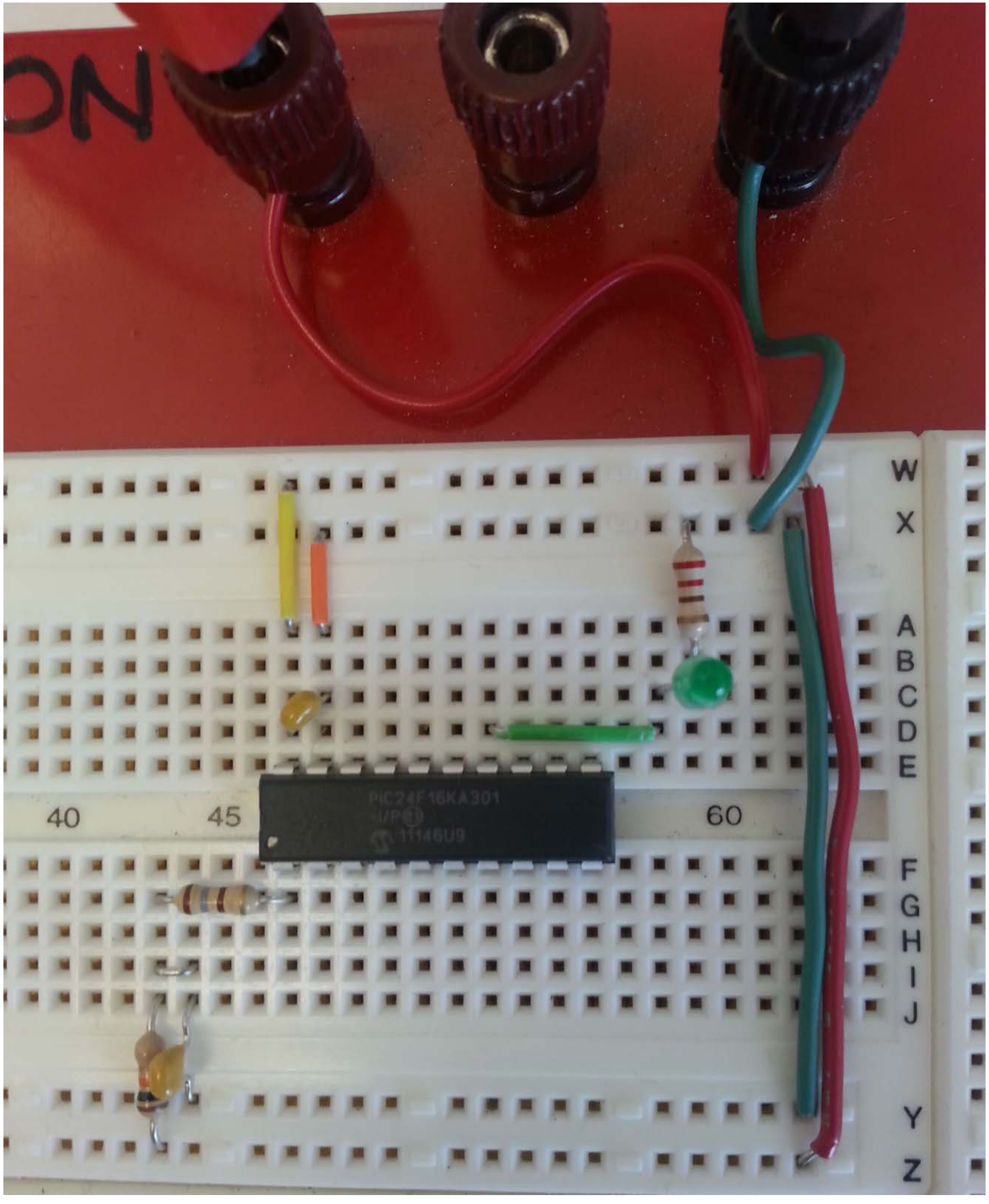
- Example: read from PORTB

```
TRISB = 0xFF;
int x;
x = PORTB;
```

Example

- Part 1: Turn on an LED connected to pin 14

- Part 2: Turn off an LED connected to pin 14



ON
W
X
A
B
C
D
E
PIC24F16KA301
-I/P
1146U9
40
45
60
F
G
H
I
J
Y
Z

PIC24F16KA301 – Individual Bits

- Can also read/write individual bits:

```
PORTAbits.RA6 = 1;
_RA6 = 1;
```
(**Look in header file**)

```
x = PORTBbits.RA2;
x = _RA2;
```

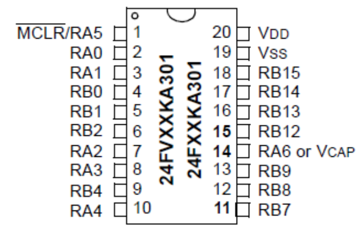
- Or set/clear individual bits in the TRISx registers:

```
TRISAbits.TRISA3 = 1;
_TRISA3 = 1;
```

PIC24F16KA301 – Shared Pins

- Every pin on the PIC24FKA301 is shared across multiple functions
- This is true for all microcontrollers
- Typically need to configure which peripheral is to be used

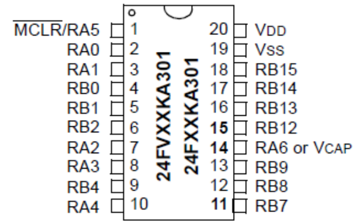
20-Pin SPDIP/SSOP/SOIC[1]



Pin	Pin Features	
	PIC24FVXXKA301	PIC24FXXKA301
1	$\overline{\text{MCLR}}$/VPP/RA5	$\overline{\text{MCLR}}$/VPP/RA5
2	PGEC2/VREF+/CVREF+/AN0/C3INC/SCK2/CN2/RA0	PGEC2/VREF+/CVREF+/AN0/C3INC/SCK2/CN2/RA0
3	PGED2/CVREF-/VREF-/AN1/SDO2/CN3/RA1	PGED2/CVREF-/VREF-/AN1/SDO2/CN3/RA1
4	PGED1/AN2/ULPWU/CTCMP/C1IND/C2INB/C3IND/U2TX/SDI2/OC2/CN4/RB0	PGED1/AN2/ULPWU/CTCMP/C1IND/C2INB/C3IND/U2TX/SDI2/OC2/CN4/RB0
5	PGEC1/AN3/C1INC/C2INA/U2RX/OC3/CTED12/CN5/RB1	PGEC1/AN3/C1INC/C2INA/U2RX/OC3/CTED12/CN5/RB1
6	AN4/SDA2/T5CK/T4CK/U1RX/CTED13/CN6/RB2	AN4/SDA2/T5CK/T4CK/U1RX/CTED13/CN6/RB2
7	OSCI/AN13/C1INB/C2IND/CLKI/CN30/RA2	OSCI/AN13/C1INB/C2IND/CLKI/CN30/RA2
8	OSCO/AN14/C1INA/C2INC/CLKO/CN29/RA3	OSCO/AN14/C1INA/C2INC/CLKO/CN29/RA3
9	PGED3/SOSCI/AN15/$\overline{\text{U2RTS}}$/CN1/RB4	PGED3/SOSCI/AN15/$\overline{\text{U2RTS}}$/CN1/RB4
10	PGEC3/SOSCO/SCLKI/$\overline{\text{U2CTS}}$/CN0/RA4	PGEC3/SOSCO/SCLKI/$\overline{\text{U2CTS}}$/CN0/RA4
11	**U1TX/C2OUT/OC1/IC1/CTED1/INT0/CN23/RB7**	**U1TX/INT0/CN23/RB7**
12	SCL1/$\overline{\text{U1CTS}}$/C3OUT/CTED10/CN22/RB8	SCL1/$\overline{\text{U1CTS}}$/C3OUT/CTED10/CN22/RB8
13	SDA1/T1CK/$\overline{\text{U1RTS}}$/IC2/CTED4/CN21/RB9	SDA1/T1CK/$\overline{\text{U1RTS}}$/IC2/CTED4/CN21/RB9
14	VCAP	**C2OUT/OC1/IC1/CTED1/INT2/CN8/RA6**
15	**AN12/HLVDIN/SCK1/$\overline{\text{SS2}}$/IC3/CTED2/INT2/CN14/RB12**	**AN12/HLVDIN/SCK1/$\overline{\text{SS2}}$/IC3/CTED2/CN14/RB12**
16	AN11/SDO1/OCFB/CTPLS/CN13/RB13	AN11/SDO1/OCFB/CTPLS/CN13/RB13
17	CVREF/AN10/C3INB/RTCC/SDI1/C1OUT/OCFA/CTED5/**INT1**/CN12/RB14	CVREF/AN10/C3INB/RTCC/SDI1/C1OUT/OCFA/CTED5/**INT1**/CN12/RB14
18	AN9/C3INA/SCL2/T3CK/T2CK/REFO/$\overline{\text{SS1}}$/CTED6/CN11/RB15	AN9/C3INA/SCL2/T3CK/T2CK/REFO/$\overline{\text{SS1}}$/CTED6/CN11/RB15
19	VSS/AVSS	VSS/AVSS
20	VDD/AVDD	VDD/AVDD

PIC24F16KA301 – Shared Pins

20-Pin SPDIP/SSOP/SOIC[1]



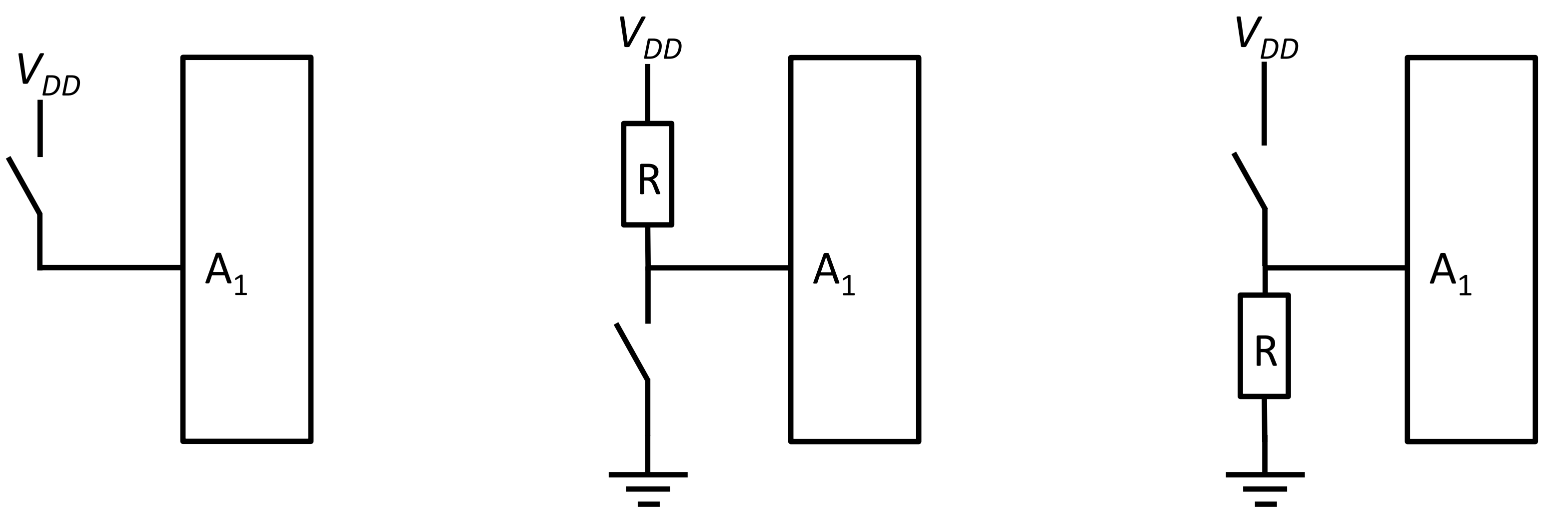
Pin	Pin Features	
	PIC24FVXXKA301	PIC24FXXKA301
1	$\overline{MCLR}$/VPP/RA5	$\overline{MCLR}$/VPP/RA5
2	PGEC2/VREF+/CVREF+/AN0/C3INC/SCK2/CN2/RA0	PGEC2/VREF+/CVREF+/AN0/C3INC/SCK2/CN2/RA0
3	PGED2/CVREF-/VREF-/AN1/SDO2/CN3/RA1	PGED2/CVREF-/VREF-/AN1/SDO2/CN3/RA1
4	PGED1/AN2/ULPWU/CTCMP/C1IND/C2INB/C3IND/U2TX/SDI2/	PGED1/AN2/ULPWU/CTCMP/C1IND/C2INB/C3IND/U2TX/SDI2/

- Example: use pin 2 (RA0) as a digital input
- Note that it is shared with AN0 (analog input)
- When analog input is enabled on that pin, we can’t use RA0 as a digital input
- Can explicitly set that pin to be used as digital I/O (Section 11.2):

```
ANSA = 0x00;          // Digital input buffer active
ANSAbits.ANSA0 = 0;
_ANSA0 = 0;
```

Interfacing Switches to Digital Inputs

- Goal: Set state of an input line HIGH or LOW using a button or switch

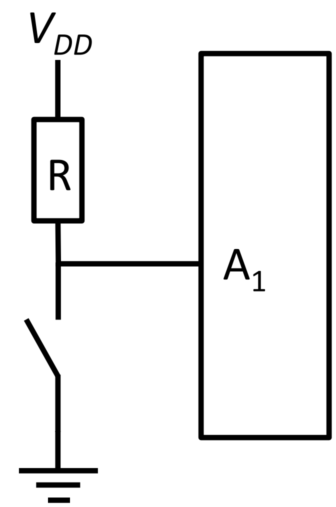


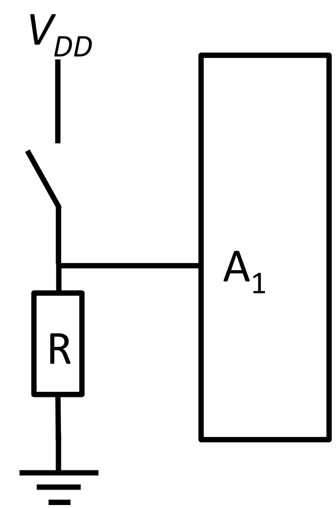
Example

- Turn on LED on pin 14 when button on pin 5 is pushed

Pull-up and Pull-down Resistor Values

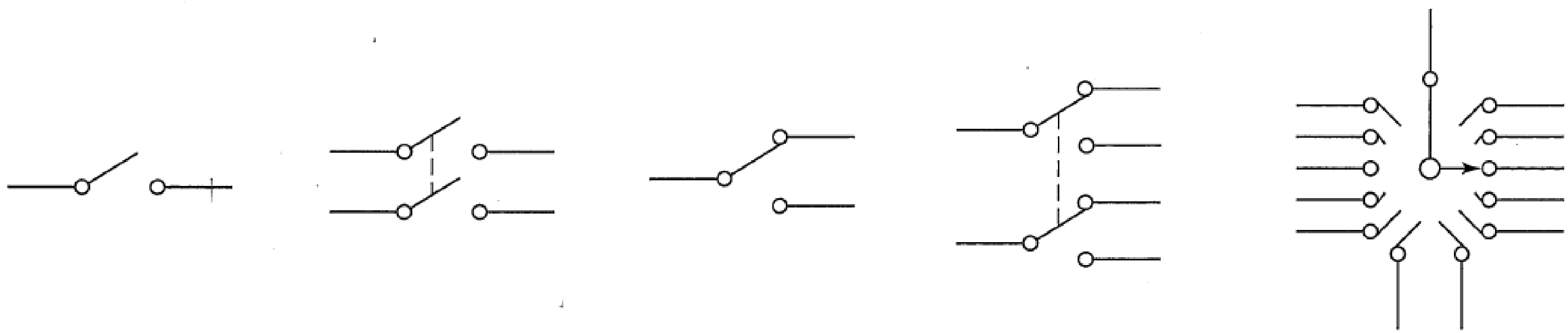
- How do we choose resistor values?
- Pull-up:
  - Open switch → want R small to make pin voltage above V_{IH}
  - Closed switch → want R large to keep current low
- Pull-down:
  - Open switch → want R small to make pin voltage below V_{IL}
  - Closed switch → want R large to keep current low
- R=10k often works in each situation, but it's easy to check



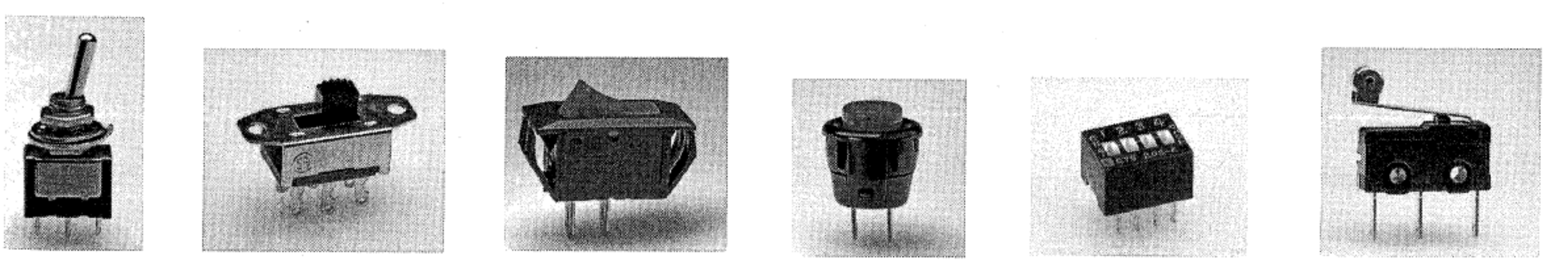


Switch Types

- **Poles:** Number of circuits it can switch
- **Throws:** Number of positions the switch can assume



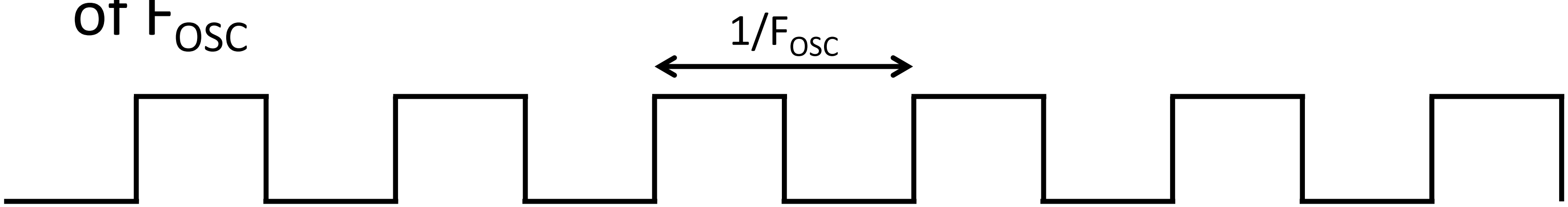
(a) SPST (b) DPST (c) SPDT (d) DPDT (e) ROTARY SP12T



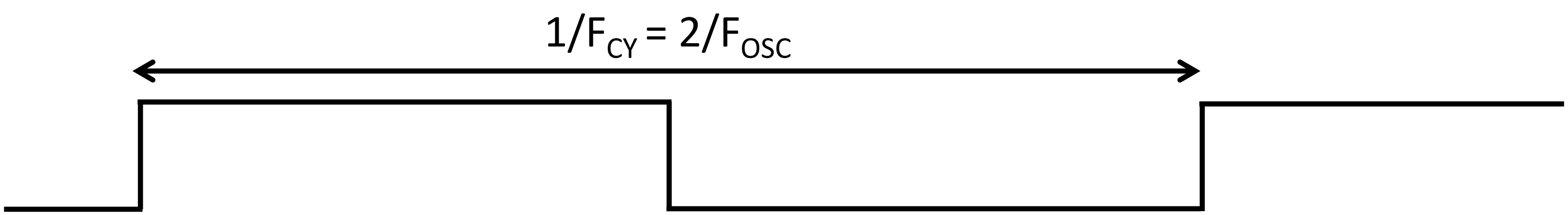
(a) Toggle (b) Slide (c) Rocker (d) Pushbutton (e) DIP (f) Micro Switch, or Snap-Action Limit Switch

Oscillators

- All microcontrollers need an oscillator to synchronize operations and data transfers
- For the PIC24F, the oscillator generates pulses at a rate of F_{OSC}



- Important: Instructions are executed every two clock cycles, so $F_{CY} = F_{OSC}/2$
- Important: Instructions = machine language or assembly instructions, *not* C instructions



Oscillator Options

- In many microcontrollers, you can select an internal (RC) or external (crystal) oscillator for the clock source
- PIC24F internal options (Section 9):
  - 8 MHz Fast RC (FRC)
  - 31 kHz Low-Power RC (LPRC)
  - 8 MHz FRC with Postscaler (FRCDIV)
  - 500 kHz FRC with Postscaler (LPFRCDIV)
- Select using _FOSCEL macro, which writes to the FNOSC bits of the FOSCEL register (Section 26 and PIC24F header file)
- Example: Select the 31 kHz LPRC

```
_FOSCSEL(FNOSC_LPRC);
```

Example

- Make an LED on pin 14 blink with a period of 2 seconds and a duty cycle of 50% using the 31 kHz LPRC oscillator

Oscillator Postscaling

- If the oscillator is too fast for what you want to do, you can apply postscaling
  - PIC24F provides 8 MHz with postscaling and 500 kHz with postscaling
  - Postscaling values of 1, 2, 4, 8, 16, 32, 64, 256
  - Divides the rate by the selected postscaling value
  - Set using the RCDIV bits of the CLKDIV register (Section 9)
- Example: How long will each instruction take if the following configuration code is executed?

```
_FOSCSEL(FNOSC_LPFRC);
_RCDIV = 0b101;
```