Encoder Usage

- Shaft encoders are often used to replace analog controls
- They are cheaper and more reliable than analog controls
- Encoders provide digital outputs that can indicate shaft movement and speed
- There are many types of encoders available for various purposes
- We will look at simple quadrature encoders that are used to sense a user selectable knob position such as a volume control.

Encoder Usage

- A pair of signals are in quadrature when they differ in phase by 90 degrees.
- Our encoders have two outputs, each 90 degrees out of phase with each other.
- As the shaft is turned, the A and B outputs pulse while maintaining a 90deg shift.

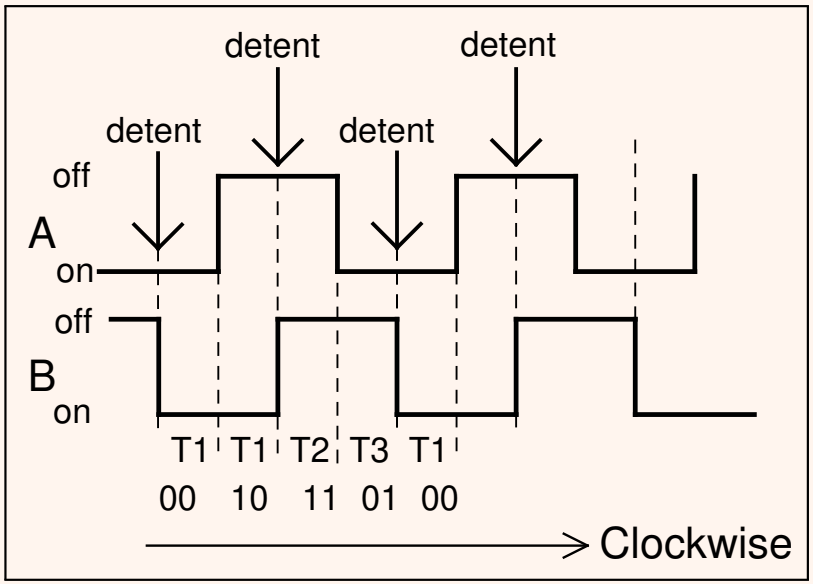

Figure 1: Quadrature Encoder Behavior

Encoder Usage

- Our encoders don't need a power supply. But, their outputs must be pulled up.
- Pulling up the encoder pins is most efficiently done with uP internal pullups.

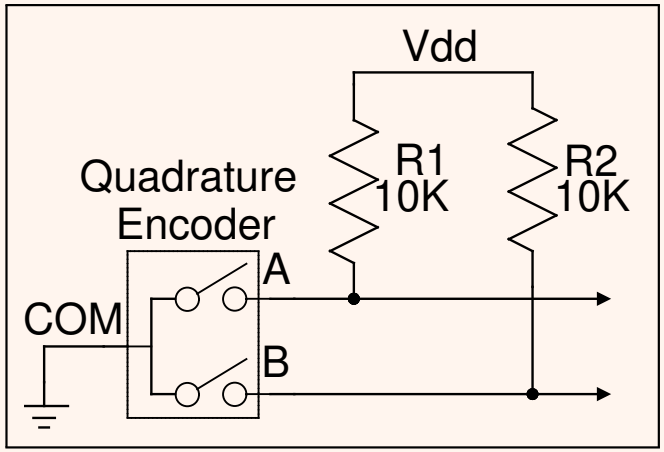

Figure 2: Typical Encoder Connection

Encoder Usage

- The encoder we will look at is similar to the Bourne 652-PEC12R-4117F-N12.
- It emits twelve pulses per 360deg rotation from its A and B outputs.
- The detent occurs between adjacent edges of A and B.
- The switch bounce is specified as 2mS(max) at a rotation speed of 15 RPM.(slow!)
- This specification also states that this measurement uses a "standard noise reduction filter", whatever that is! They don't say. Probably a capacitor-based low pass filter.
- This specification also states that "Users should verify actual device performance in their specific applications.
- A prudent engineer would test it themselves! The paranoid survive!

Encoder Usage

- How can uPs track the encoder movement?
- Some have built-in encoder circuitry.
- Microchip's dsPIC33F family has its QEI module.
- Note the three inputs.
- The index pin allows an absolute position to be determined.

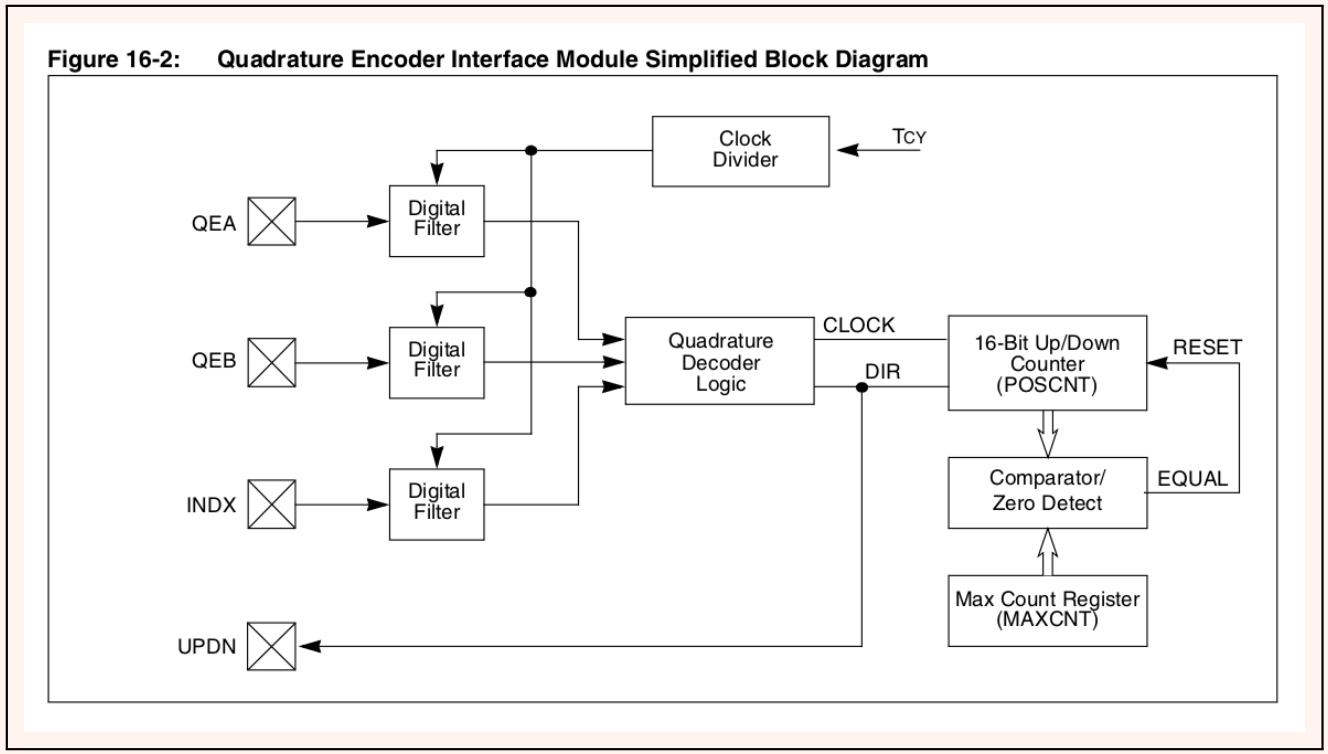

Figure 3: Microchip QEI Module

Encoder Usage

- How can we determine encoder movement with just software?
- One way is to take advantage of the fact that one output will produce an edge 90 degrees out of phase with the other output. If we sense the edge and sample the other output, movement and direction can be ascertained.

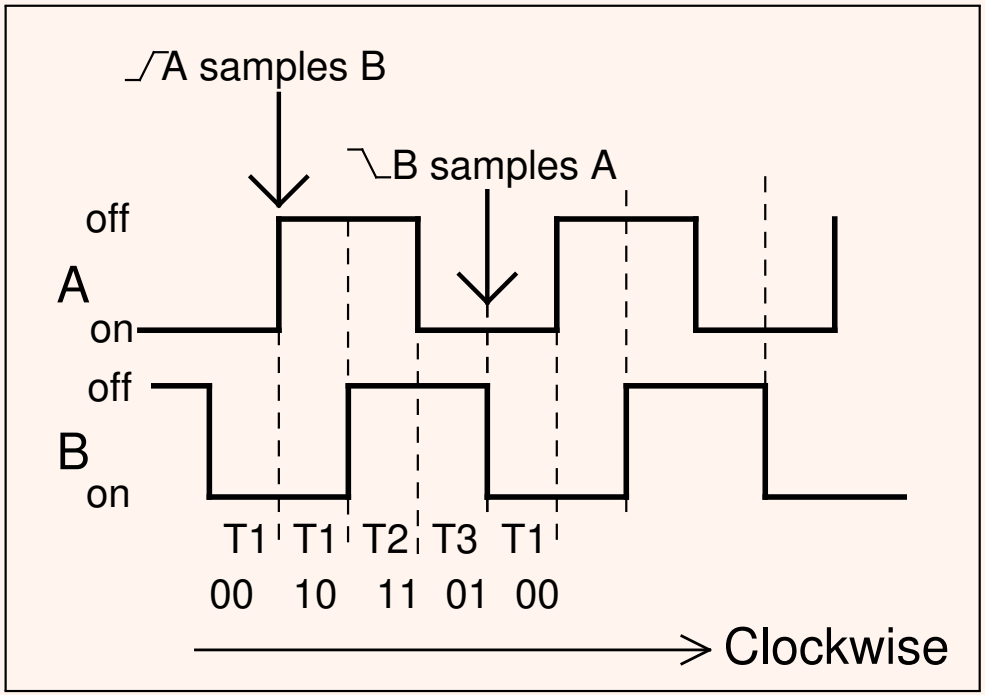

Figure 4: Quadrature Encoder Behavior

Encoder Usage

- This behavior is easily captured with the following code:

```
int8_t encoder_chk(uint8_t encoder_var) {
  //A and B are in bits 0 and 1
  static uint16_t state = {0}; //holds bits from encoder
  uint8_t a_pin, b_pin;        //encoder pin states

  //a_pin and b_pin are asserted TRUE when low
  a_pin = ((encoder_var & 0x01) == 0) ? 0 : 1;
  b_pin = ((encoder_var & 0x02) == 0) ? 0 : 1;

  //update shift using only the A pin
  state = (state << 1) | a_pin | 0xE0;

  //check for falling edge on A pin
  //if it did, then B pin state indicates direction
  //of rotation. Return 1 for CW, 0 for CCW
  if (state == 0xF0){return (b_pin) ? 1 : 0; }
  else              {return -1;} //no movement detected
} //encoder_chk
```

Encoder Usage

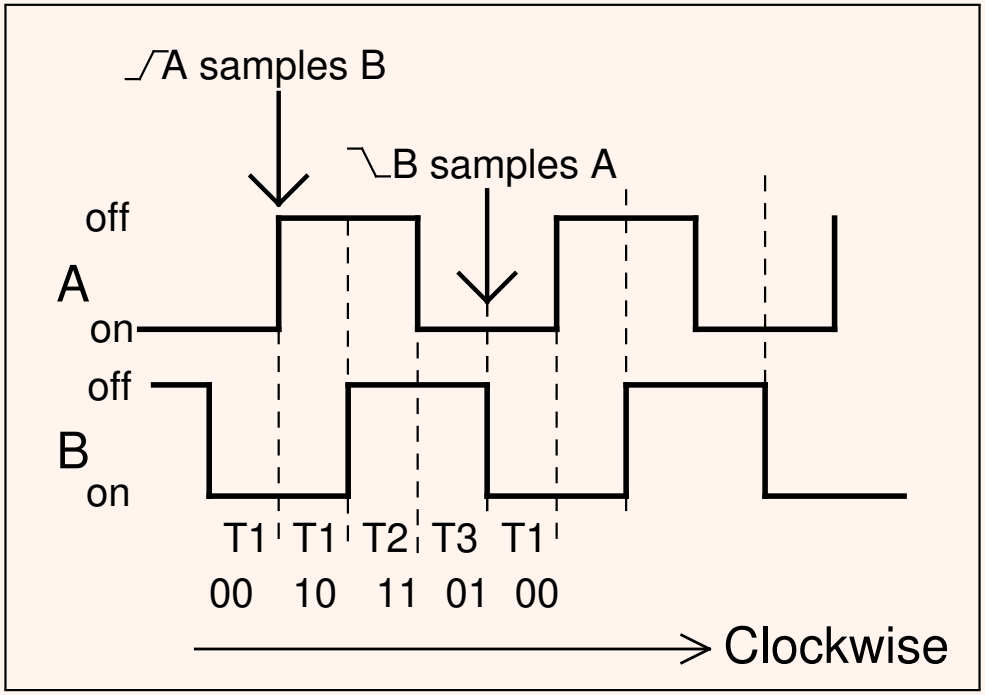

Figure 5: Tricking the Encoder Software

- This code is easy to trick however. Imagine you're in T3 and "rock" the encoder shaft between T3 and T1.
- This would cause repeated "B samples A" events.
- This would manifest itself in the encoder code indicating successive clockwise rotations.

Encoder Usage

- Here is a elegant state-machine based solution.

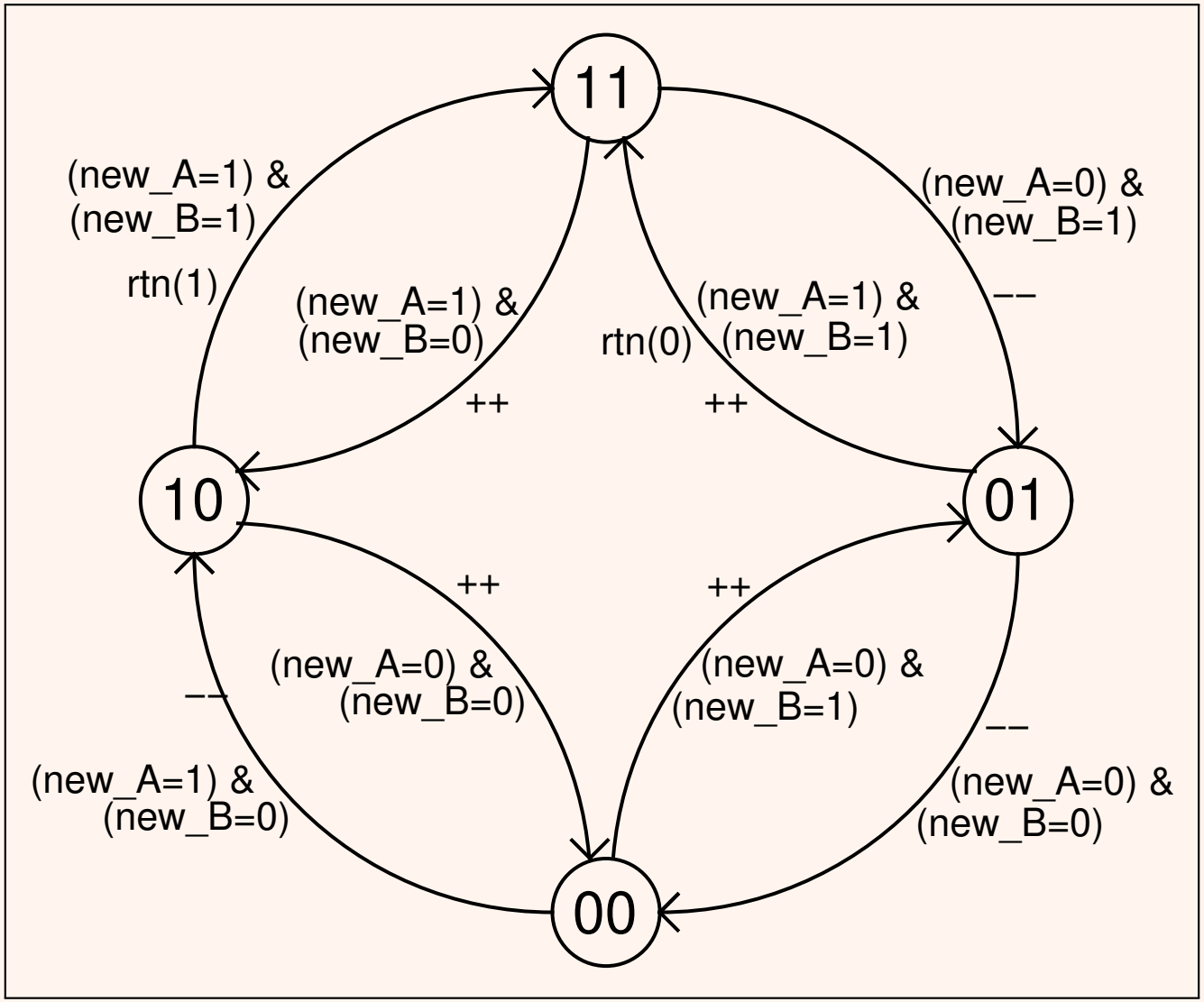

Figure 6: Encoder-Tracking State Machine

Encoder Usage

- Here are the "bones" of the software state machine.

```
return_val = -1;  //default return value, no change
if ((new_A != old_A) || (new_B != old_B)){ //if change occurred
  if ((new_A == 0) && (new_B == 0))       {
    if (old_A == 1){count++;}
    else           {count--;}
  }
  else if ((new_A == 0) && (new_B == 1)) {
    if (old_A == 0){count++;}
    else           {count--;}
  }
  else if ((new_A == 1) && (new_B == 1)) {//detent position
    if (old_A == 0){if(count== 3){return_val=0;}}  //one direction
    else           {if(count==-3){return_val=1;}}  //or the other
    count = 0;     //count is always reset in detent position
  }
  else if ((new_A == 1) && (new_B == 0)) {
    if (old_A == 1)          {count++;}
    else                    {count--;}
  }
  old_A = new_A;  //save what are now old values
  old_B = new_B;
} //if change occurred
return (return_val); //return encoder state
```