Project 7: Rotary Encoders

The Task

- Make a rotary encoder countdown timer with a finite state machine.
- This finite state machine can "tell" which way the encoder is spinning.
- Based on that, increment or decrement the value of a 7 segment display and output that.
- This display also counts down by 1 every second.

The Machine

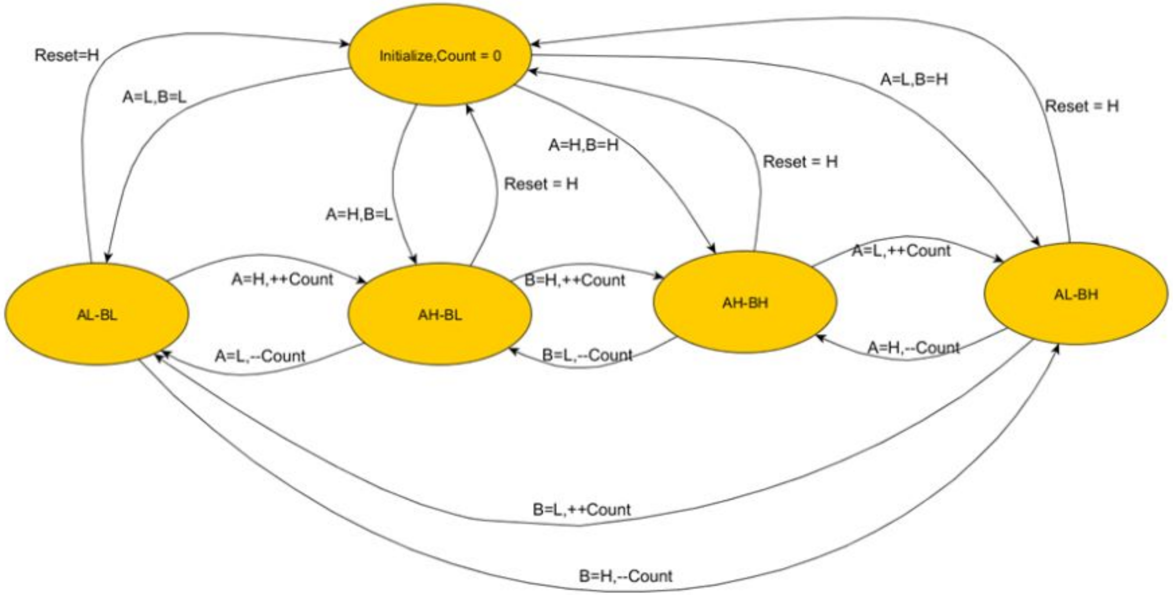


Figure 3: Rotary Encoder FSM State Transition Diagram.

The Machine

- In essence, the rotary encoder has two switches, A and B that produce square waves.
- Using the value of A and B together in time, it is possible to generate a set of combinations of states.
- If both A and B transition in time, this is a forbidden state.
- If one of A or B transitions, then switch to the new state.
- This restricts the possible transitions, which we match up to moving in a single direction.

The Code

```
case AlowBlow:
    if (myRotorEncoder->SwitchA== High && myRotorEncoder->SwitchB== Low && myRotorEncoder->Reset==Low){
        NextEncoderState=AhighBlow;
        ++myRotorEncoder->RotaryEncoderStateCount;
    }
    else if (myRotorEncoder->SwitchA== Low && myRotorEncoder->SwitchB== High && myRotorEncoder->Reset==Low){
        NextEncoderState=AlowBhigh;
        --myRotorEncoder->RotaryEncoderStateCount;
    }
    else  if(myRotorEncoder->Reset==High ){
        NextEncoderState=Initialize;
    }
    else{
        NextEncoderState=AlowBlow;
    }

    break;
```

The Code

- What was presented was one of the possible transitions on the original state diagram, which was not actually complete.
- Here we see that only one rotor can transition at a time within the code.
- Every other transition has been implemented in a similar way, with different NextState outputs.

7 Segment Display

- We set up a 7 segment display to show the value of a countdown timer.
- We hooked up the machine to this to generate the continuous countdown with increment/decrement functionality.
- Using SPI and the MOSI port of the device, we sent hex codes to the display and latched the display to change it.

The Code

```
inline void send(unsigned char value) {
    unsigned char i;
    for (i = 8; i > 0; i--) {
        if (((1 << (i - 1)) & value) > 0)  {
            P1OUT |= BIT7;
        } else {
            P1OUT &= ~BIT7;
        }
        P1OUT |= BIT5;
        P1OUT &= ~BIT5;
    }
}
```

- This is the same as the SendByte() method from the previous project.

UNIVERSITY of VIRGINIA ENGINEERING

Timers

- Lastly, we added a software timer to this that would periodically make the rotary encoder ‘tick down’.
- This entailed maintaining a counter variable that was continuously incremented until it hit some threshold.
- At this point, the counter was then reset and the output was performed, which was the reduction of the counter stored within the encoder.
- This encoder count is what was translated to a timer output.

The Code

```
void ManageSoftwareTimers(void)
{
    if (g1msTimeout) {
        g1msTimeout--;
        g1msTimer++;
    }
}
```

- This is the counter increment function for the software timer.
- The variable g1msTimer represents this time variable.

```
if (g1msTimer >= 1000) {
    g1msTimer -= 1000;
    myRotaryEncoder.RotaryEncoderStateCount -= 4;
}
```

- This performs the countdown.

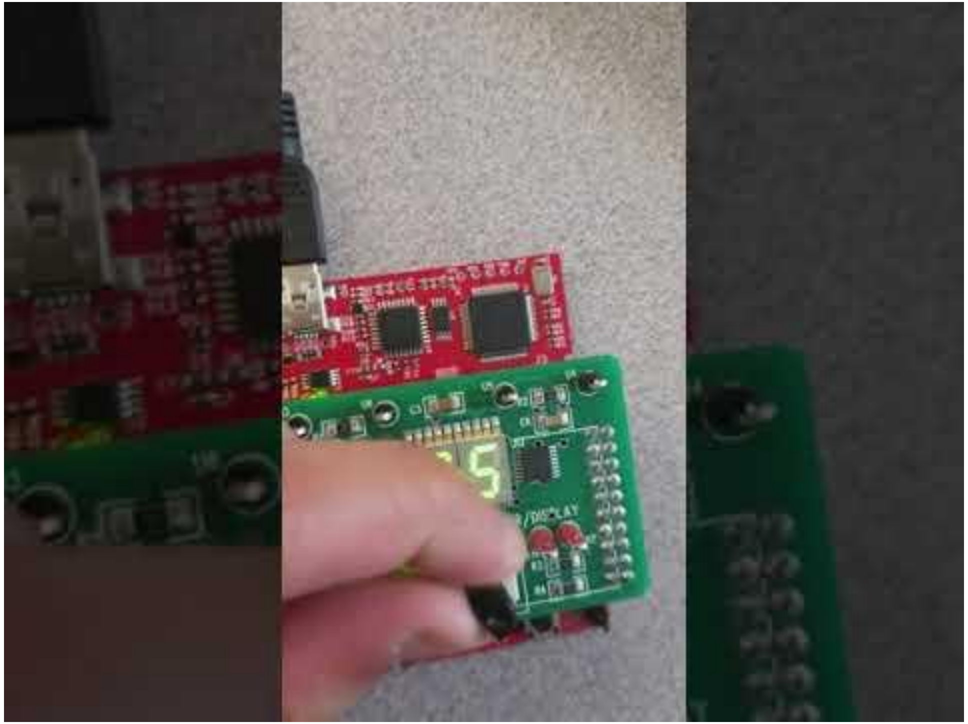
The Code

- **The last part is integrating the timer and transmitting the digits from the state count in the encoder object.**

  - SetDigit is a helper method that simply indexes the digit that I choose. It calls the SendByte() function.
  - This function then calculates the tens place and ones place without modular arithmetic.

```
void sendValue(int count, unsigned char values[]) {
    count >>= 2;

    if (count > 98) {
        setDigit(9, values);
        setDigit(9, values);
        latch();
        return;
    }

    if (count < 0) {
        setDigit(0, values);
        setDigit(0, values);
        latch();
        return;
    }

    int tens = 0;

    while (count >= 10) {
        count -= 10;
        tens++;
    }

    setDigit(count, values);
    setDigit(tens, values);

    latch();
}
```

Demonstration



Fin