Quadrature Encoder Position Counter With MSP430™ MCUs

TEXAS INSTRUMENTS

Introduction

Quadrature encoders are used to keep track of the angular position of knobs and motors in many applications including volume control, robotics and factory automation systems. As a quadrature encoder device rotates, it outputs square waves on two wires (Line A and Line B), which are tracked by an interpreter device to determine the device's position. By looking at which square wave leads the other, it is possible to know which direction the device is rotating. If Line B's square wave leads Line A's square wave, the device is rotating clockwise, and if Line A's square wave leads Line B's square wave, the device is rotating counter-clockwise, as shown in Figure 1 and Figure 2. The actual name of the square wave signals may be different based on the selected quadrature encoder hardware and manufacturer notation standards.

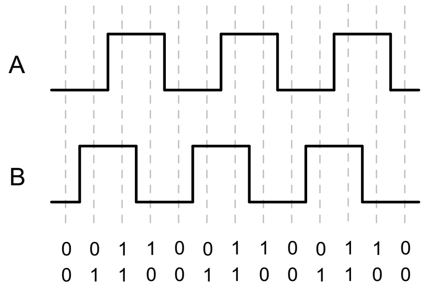

Figure 1. Quadrature Encoder Clockwise Rotation Waveforms

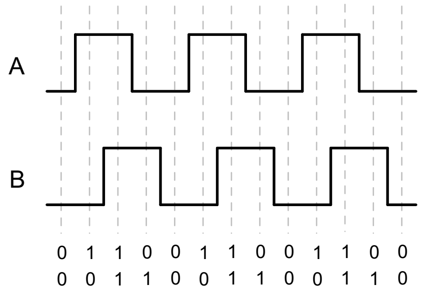

Figure 2. Quadrature Encoder Counter-Clockwise Rotation Waveforms

The values of A and B can be tracked in a state machine to determine whether the device angle is increasing or decreasing as shown in Figure 3. Each time a rising or falling edge occurs on Line A or B, the device changes states and the position counter is changed accordingly.

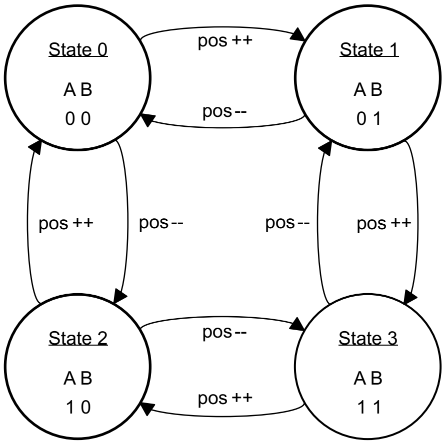

Figure 3. Quadrature Encoder State Machine

The MSP430FR2000 microcontroller (MCU) can hold a program implementing this state machine to take in the quadrature encoder input from a device and output the change in position through UART. Additionally, the internal real-time clock (RTC) can be used to provide rotational velocity information by outputting timestamps each time the position changes. This project has been optimized for smallest code size and robust handling of human interface device (HID) inputs. To get started, download project files and a code example demonstrating this functionality.

Implementation

This application uses the MSP430FR2000 MCU along with the MSP-TS430PW20 target development board. As shown in Figure 4, the MCU pins P1.1 and P1.0 are connected to channels A and B of a quadrature encoder device to track its position. UART communication occurs on P1.7 (the MSP-FET or eZ-FET backchannel UART can be used to connect to a host processor at 9600 baud to transmit position and time data). Note that the MSP-TS430PW20 target board already includes the correct connections for the UART TXD and RXD on the MSP-FET connector as long as JP14 and JP15 are populated (leave JP13 unconnected).

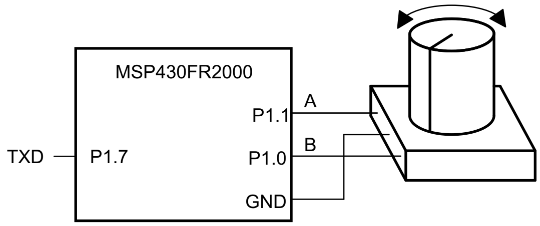

Figure 4. Hardware Connection Diagram

The firmware implements the state machine shown in Figure 3 and tracks the change in position of the connected quadrature encoder device, from the time that the MCU is powered-on, starting at a value of 0. The MCU transmits an unsigned 8-bit integer representing the device's position, followed immediately by a 16-bit integer representing the timestamp through UART each time an edge occurs on Line A or B as the quadrature encoder hardware is rotated. When the position counter is at 0 and is decremented, it wraps around to a value of 255 and vice versa. This number can be interpreted as a two's complement number if desired. The 16-bit timestamp is transmitted as two 8-bit integers, with the high byte transmitted first, and the low byte transmitted second. Figure 5 shows the data packet structure.

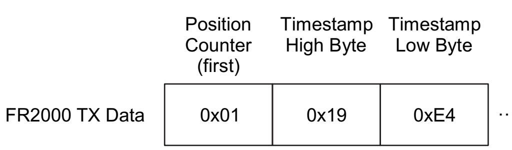

Figure 5. UART TX Data Packet Structure

If only the position data is desired to be transmitted through UART, the timestamp transmit code in the main while loop may be commented out as shown in Figure 6.

```
while(!(UCA0IFG&UCTXIFG));
UCA0TXBUF = pos;
__no_operation();

//      while(!(UCA0IFG&UCTXIFG));
//      UCA0TXBUF = time >> 8;
//      __no_operation();
//
//      while(!(UCA0IFG&UCTXIFG));
//      UCA0TXBUF = time;
//      __no_operation();
```

Figure 6. Removal of Timestamp Data

The physical angular change can be calculated from the MCU count using Equation 1. Units are shown in square brackets.

$$\Delta\theta\ [°] = \frac{(Count_{new} - Count_{old})\ [\#]}{\text{Encoder Resolution}\ [\#/\ °]} \quad (1)$$

The angular velocity of the device can be calculated using the change in the angle, the RTC frequency, and the difference of consecutive timestamps.

$$\omega\ [°/s] = \frac{\Delta\theta\ [°] * \text{RTC Freq}\ [Hz]}{Timestamp_{new} - Timestamp_{old}} \quad (2)$$

These calculations can be performed in the host processor program that receives the position and time values through UART to save code space on the MSP430FR2000 MCU.

Performance

The solution uses an MSP430FR2000 MCU and a quadrature encoder knob. Testing was performed with the TT Electronics EN11-HSM1AF15 knob, which has 20 positional latch locations, or detents, per revolution.

To run the demo, connect the hardware as previously described, load the code into the device, allow the device to run and end the debug session. Using the backchannel UART on the MSP-FET or the eZ-FET, use a host processor with UART communication set to 9600 baud none parity 1 stop bit to receive the quadrature encoder knob position and time data.

As specified by the knob data sheet, two edges occur on each of Line A and Line B every time the knob is turned to a different detent, causing the position counter to change by four. In many data sheets, the number of detents per revolution is given as *pulses per revolution* (ppr), which is used in the following calculations. The encoder resolution can then be calculated as follows:

$$\text{Encoder Resolution}\ [\#/\ °] = \frac{\Delta Count\ [\#]}{\Delta\theta\ [°]} \quad (3)$$

$$\text{Encoder Resolution}\ [\#/\ °] = \frac{\text{4 counts per pulse}}{360°/\ 20\ ppr} \quad (4)$$

$$\text{Encoder Resolution}\ [\#/\ °] = 0.2222\ \#/° \quad (5)$$

The solution configures the RTC to a frequency of 2048 Hz. The following example demonstrates how to calculate the angular velocity from real data when the knob is turned from one detent to the next.

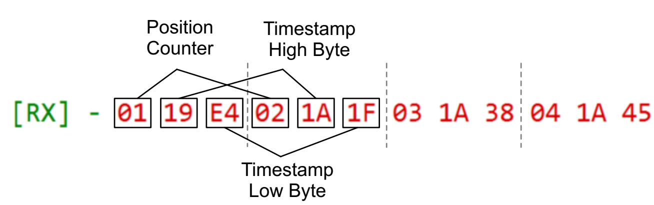

Figure 7. UART Received Position and Time Data

$$\Delta\theta\ [°] = \frac{2 - 1}{0.2222\ \#/°} = 4.5° \quad (6)$$

$$\text{Time Stamp}_{new} = 0x1A1F = 6687 \quad (7)$$

$$\text{Time Stamp}_{old} = 0x19E4 = 6628 \quad (8)$$

$$\omega\ [°/s] = \frac{4.5° \times 2048\ \text{Hz}}{6687 - 6628} = 156.2°/s \quad (9)$$

If $Timestamp_{new} – Timestamp_{old}$ is a negative number, 2^{16} (65536) must be added to the value to make it positive. For the angular velocity calculation to be accurate, the position counter must change at least once in the time it takes for the RTC counter to overflow (32 s).

The firmware can respond to changes in encoder position at a frequency of at most 200 kHz. Therefore, the maximum angular velocity that the firmware can track is given by Equation 10.

$$\omega_{max}\ [°/s] = \frac{\text{Interrupt Freq [\#/s]}}{\text{Encoder Resolution [\#/°]}} \quad (10)$$

For the knob chosen for testing, the maximum angular velocity that can be handled by the firmware (150000 RPM) is well above the maximum hardware operating speed given in the device data sheet (100 RPM).

$$\omega_{max}\ [°/s] = \frac{200000\ \#/s}{0.2222\ \#/°} = 900000°/s \quad (11)$$

$$\omega_{max} = 150000\ \text{RPM} \quad (12)$$

Device Recommendations

The device used in this example is part of the MSP430 Value Line Sensing portfolio of low-cost MCUs, designed for sensing and measurement applications. This example can be used with the devices shown in Table 1 with minimal code changes. For more information on the entire Value Line Sensing MCU portfolio, visit www.ti.com/MSP430ValueLine.

Table 1. Device Recommendations

Part Number	Key Features
MSP430FR2000	0.5KB FRAM, 0.5KB RAM, eComp
MSP430FR2100	1KB FRAM, 0.5KB RAM, 10-bit ADC, eComp
MSP430FR2110	2KB FRAM, 1KB of RAM, 10-bit ADC, eComp
MSP430FR2111	3.75KB FRAM, 1KB RAM, 10-bit ADC, eComp