Application Note

Interfacing an Incremental Rotary Encoder with Z8F64xx Series MCUs

AN034801-1212

Abstract

This application note demonstrates how to read data from the two-channel output of an incremental rotary encoder. Data from these two channels are based on the Gray Encoding System and can be used to determine the rotation of the encoder shaft, whether it is clockwise or counterclockwise.

The firmware developed for this application is used to determine the rotation of the incremental rotary encoder shaft. It is based on a method of taking the XOR result of the current Channel A bit and the previous Channel B bit.

This application uses an encoder that operates at 20 steps per revolution. For demonstration purposes, an up/downcounter for display is included in the firmware. The display counts up when the rotation of the encoder shaft is clockwise; conversely, the display counts down when the rotation is counterclockwise. This count is defined as being an increment or decrement of one step.

Note: The source code file associated with this application note, AN0348-SC01.zip, is available for download on zilog.com. This source code has been tested with version 5.0.0 of ZDS II for Z8 Encore! XP MCUs. Subsequent releases of ZDS II may require you to modify the code supplied with this application note.

Discussion

A rotary encoder is similar to a potentiometer in form factor, but without a rotational stop. Its shaft can be infinitely rotated in either a clockwise or counterclockwise direction. Typical applications of a rotary encoder can include motor control, angular or position measurement, control knobs and mechanical mice.

There are several types of rotary encoders, the most common of which are classified as *absolute* and *incremental* rotary encoders. Other classifications are based on the number of pulses, positions or steps per revolution that a rotary encoder turns, as well as its mechanical interface style and sensors used inside the encoder.

A number of rotary encoders include an integrated pushbutton switch which can be used to trigger an event when the shaft of the encoder is pressed.

In this application, a rotary encoder with a pushbutton switch is used to reset a value displayed in the HyperTerminal terminal emulation application.

General Types of Rotary Encoders

Two general types of rotary encoders exist – absolute and incremental. These two types can be defined as follows:

- An absolute rotary encoder produces a digital output value proportional to the angle of the shaft.
- An incremental rotary encoder produces two digital output signals. The phase relationship between these two signals determines whether the shaft of the encoder is rotating clockwise or counterclockwise.

Reading Direction of Incremental Rotary Encoder

Inside the incremental rotary encoder are a slotted wheel and two pairs of LEDs and sensors, as shown in Figure 1. When the shaft is rotated, the slotted wheel cuts the light from the LEDs and passes it to the sensors such that the sensors receive alternate ON and OFF light signals.

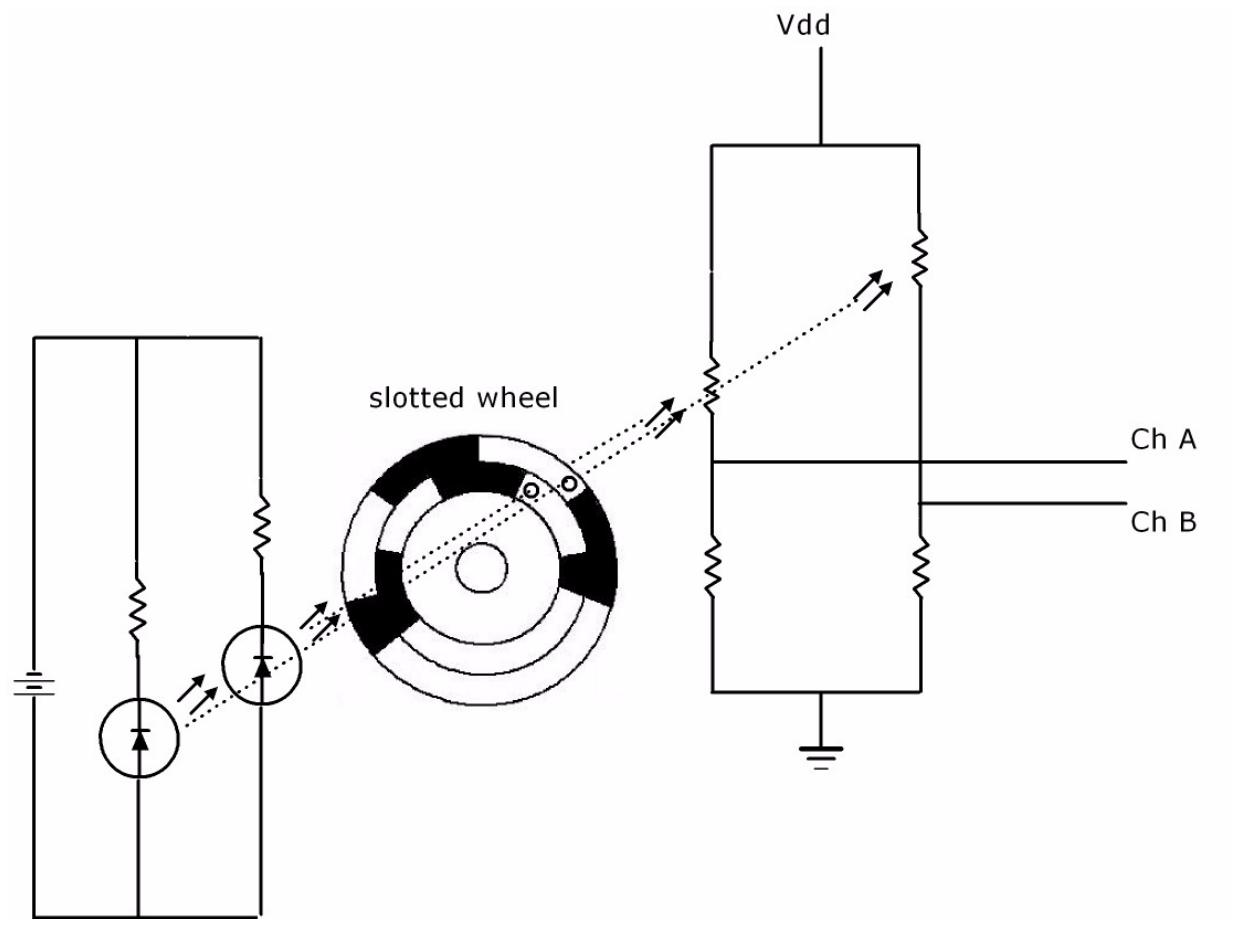


Figure 1. Basic Rotary Encoder Circuit

The sensors next produce an output signal, as shown in Figures 2 and 3.

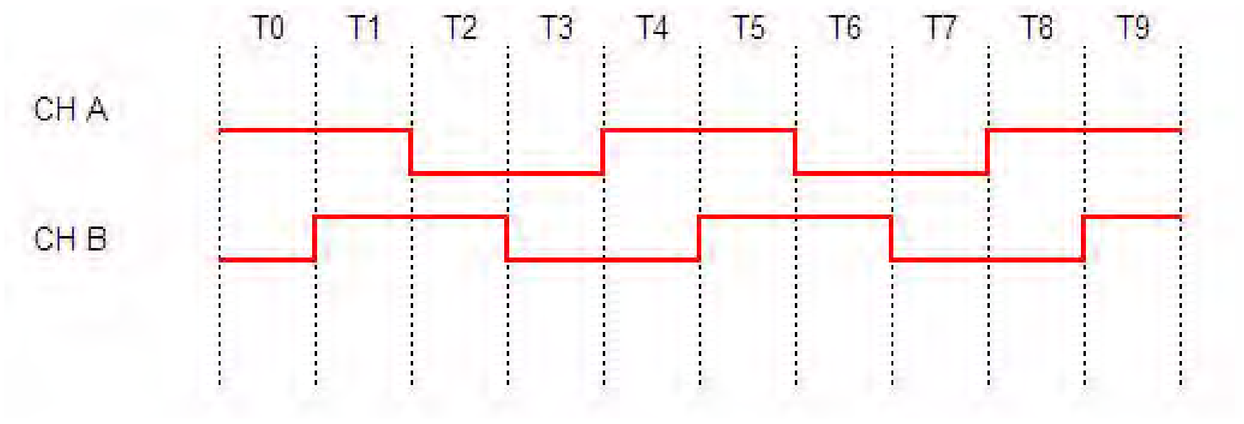


Figure 2. Clockwise Shaft Rotation Timing

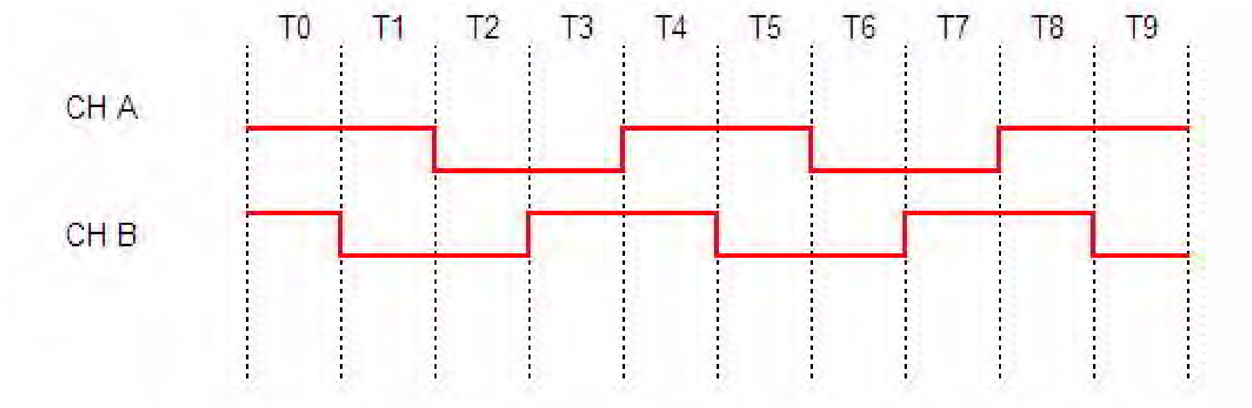


Figure 3. Counterclockwise Shaft Rotation Timing

There are three different methods for reading signals from an incremental rotary encoder to determine the direction of shaft rotation. Each of these methods is described below.

Method A: Measuring Edges and Signal Levels

Determine the relationship between the signal levels of Channel A and the edges of the signal of Channel B. If the signal level of Channel A is High and the Channel B signal is Rising Edge, the rotary encoder shaft rotation is clockwise (see Figure 2). If the signal level of Channel A is High and the Channel B signal is Falling Edge, the rotary encoder shaft rotation is counterclockwise (see Figure 3).

Method B: Gray Encoding System

Converting Figures 2 and 3 into table format results in the data presented in Tables 2 and 1. With Channel A representing the Least Significant Byte (LSB) and Channel B representing the Most Significant Byte (MSB), the binary data pattern produced is `01`, `11`, `10`, `00` (1, 3, 2, 0 as decimal-equivalent values) to represent a clockwise rotation, and `11`, `01`, `00`, `10` (3, 1, 0, 2 as decimal-equivalent values) to represent a counterclockwise rotation. These data patterns are called *gray code*; by using these patterns, the direction of shaft rotation can be determined.

Table 1. Clockwise Shaft Rotation Signal Data

Time Slot (TN)	Channel B	Channel A
T0	0	1
T1	1	1
T2	1	0
T3	0	0
T4	0	1
T5	1	1
T6	1	0
T7	0	0
T8	0	1
T9	1	1

Table 2. Counterclockwise Shaft Rotation Signal Data

Time Slot (TN)	Channel B	Channel A
T0	1	1
T1	0	1
T2	0	0
T3	1	0
T4	1	1
T5	0	1
T6	0	0
T7	1	0
T8	1	1
T9	0	1

Method C: XOR

A third method for determining the direction of shaft rotation is to take the XOR of the current Channel A value and the previous Channel B value (Channel A TN XOR Channel

B TN–1). If the result is a logic 1, the direction is clockwise; if the result is a logic 0, the direction is counterclockwise.

As an example of this method, consider the data for T0 and T1 in Table 2.

T1 (TN) of Channel A = 1

T0 (TN–1) of Channel B = 0

Computing an XOR for these two values yields a logic 1. This computation is also true for T1 and T2; the result is a logic 1 when computing their XOR.

T2 (TN) of Channel A = 0

T1 (TN–1) of Channel B = 1

Because these two XOR results are logic 1, the shaft rotation is clockwise.

When using the XOR method upon the data in Table 1, the results are logic 0, indicating that the rotation is counterclockwise.

Software Implementation

The firmware developed for this application is based on Method C, the XOR method for reading the incremental rotary encoder.

The first part of the code serves to initialize Ports D0 and D1 as inputs, Port C0 as an interrupt source for the pushbutton switch, as indicated in Figure 4.

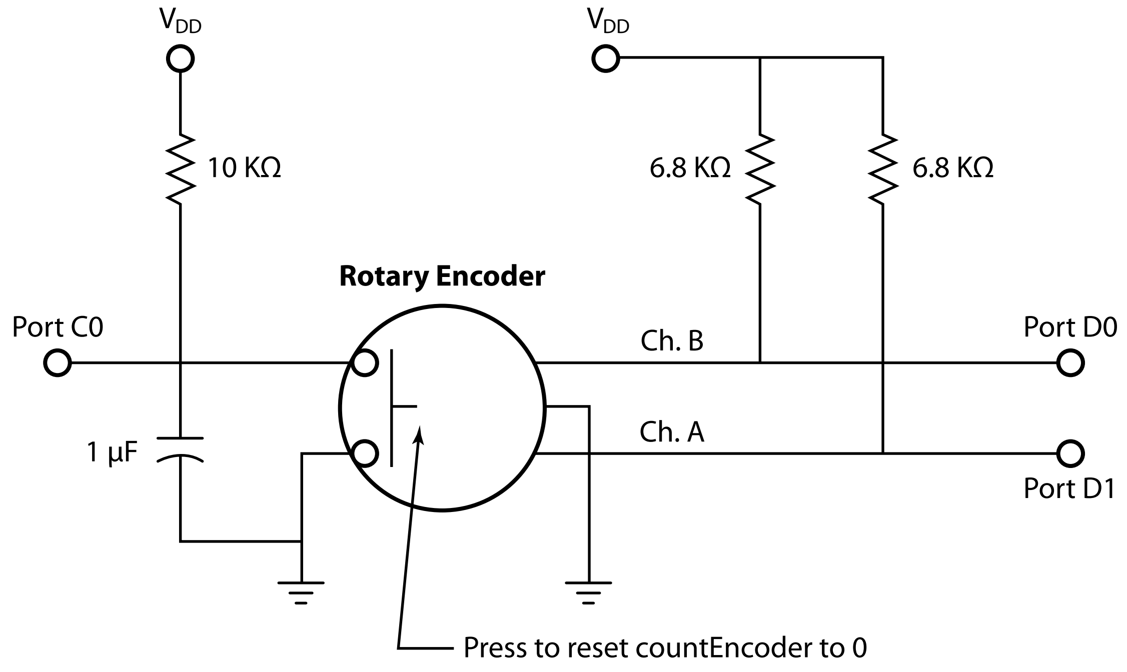


Figure 4. Rotary Encoder with Pushbutton Switch

The second part of the code reads and processes the data from Port D. The initial value read from Port D is masked for bits 1 and 0 so that the remaining Port D pins are available for other input or output devices; the data read from bits [7:2] will be a logic 0 and will not affect the bitwise XOR operation. This first data read is saved to a variable named `encoderOld`.

Port D is read again for a new value, masked, and saved to two additional variables called `encoderNew` and `encoderTemp`. The values contained in `encoderNew` and `encoderOld` are compared for changes. If these values are the same, the rotary encoder shaft is not rotating. Otherwise, the shaft is rotating but the direction of rotation must still be determined.

To determine the direction of rotation, the value contained in the `encoderNew` variable is shifted to the right by one bit position. Bit 0 of `encoderNew` is now the Channel A current value; this value will be XORed to Bit 0 of `encoderOld`, which is the previous value from Channel B. If the result is a logic 1, `countEncoder` will increment, indicating that the shaft rotation is clockwise. If the result is a logic 0, `countEncoder` will decrement, indicating that the shaft rotation is counterclockwise.

The displayed value refers to the number of steps while rotating the rotary shaft. Each step is equivalent to an increment of 1 if clockwise and a decrement of 1 if counterclockwise.

In this application, it was decided to limit the counter to display from 0 to 100 only. The `countEncoder` value is checked before displaying it to HyperTerminal through the UART0 connection to ensure that when the `countEncoder` value is equal to or greater than 100, the display will only show 100, even if the shaft is still rotating clockwise. Additionally, if the `countEncoder` value is equal to or less than 0, the display will show 0 even if the shaft is rotating counterclockwise.

After displaying the count value, the code returns to read a new Port D value.

The final part of the code is the interrupt routine for the pushbutton switch, which is integrated into the rotary encoder. This switch is connected to Port C0 of the Z8F64xx Series MCU, which is a dual-edge interrupt source. When the switch is pressed, an interrupt is generated; upon release of the switch, another interrupt is generated. In this interrupt routine, when the value of `intCntr` is equal to 2, the switch is pressed, then released. This activity resets the value of `countEncoder` to 0, and as a result, 0 is displayed in the HyperTerminal console.

Equipment Used

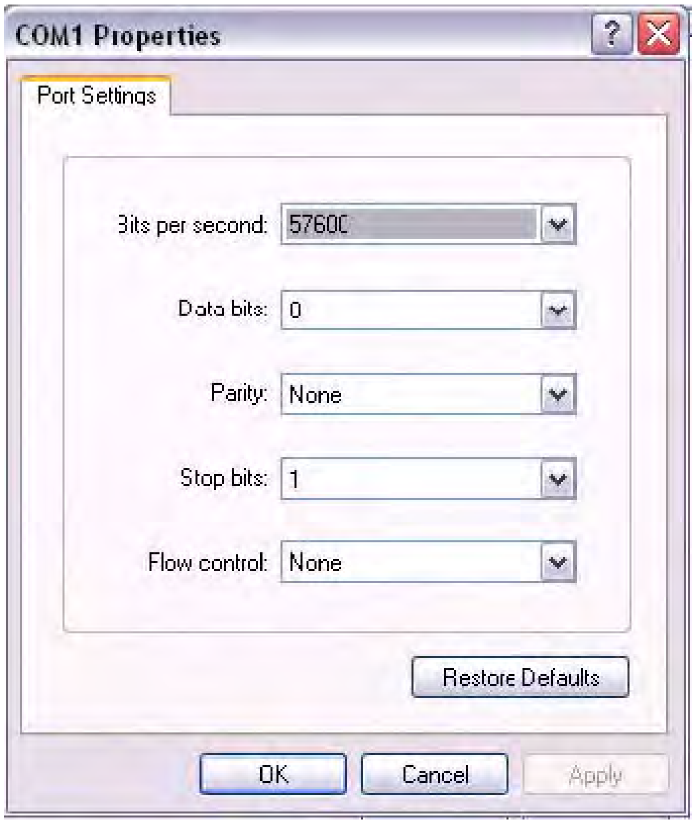
The tools used to develop this Incremental Rotary Encoder application are:

- Z8 Encore! XP Z8F64xx Series Development Kit
- Logic analyzer
- Windows-based personal computer
- DB9 serial cable
- Panasonic EVEJB encoder

Testing Procedure

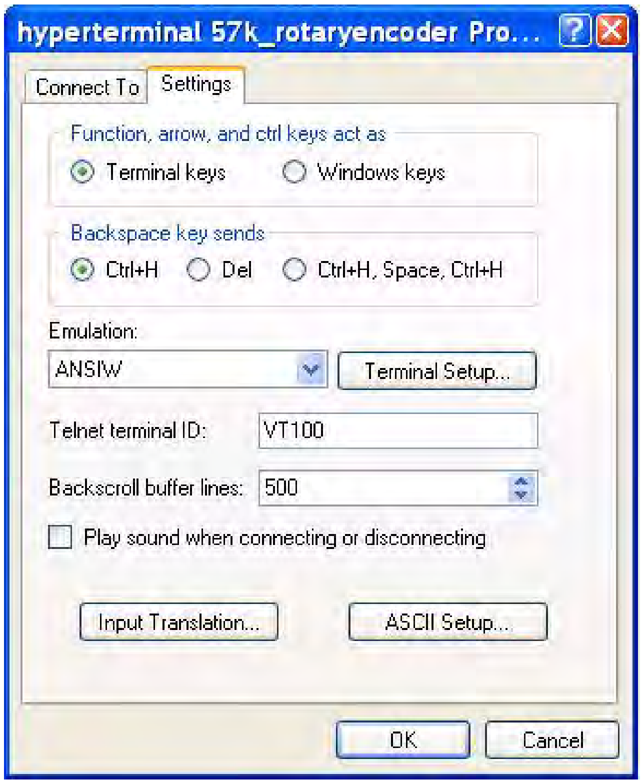
To build, configure and test the hardware and firmware for this application, observe the following procedure.

1. Download the AN0348-SC01.zip source code file, which can be obtained free from the Zilog website. Unzip this file to an appropriate location on your PC's hard drive.
2. Launch ZDS II for Z8 Encore!. Navigate via the File menu to the source code files, and open the `AN0348_RotaryEncoder.zdsproj` project file.
3. Refer to Figure 4 on page 5 to connect the rotary encoder to the Z8F64xx Series Development Board.
4. Connect one end of a serial cable to the PC, and connect the other end to the Z8F64xx Series Development Board's P1 console, which is connected to UART0.
5. Open HyperTerminal and configure it to the values indicated in Figures 5 through 7.



Bits per second: 57600
Data bits: 0
Parity: None
Stop bits: 1
Flow control: None

Figure 5. HyperTerminal Settings


hyperterminal 57k_rotaryencoder Pro...

Connect To | Settings

Function, arrow, and ctrl keys act as
[✓ Terminal keys] Windows keys

Backspace key sends
[✓ Ctrl+H] Del Ctrl+H, Space, Ctrl+H

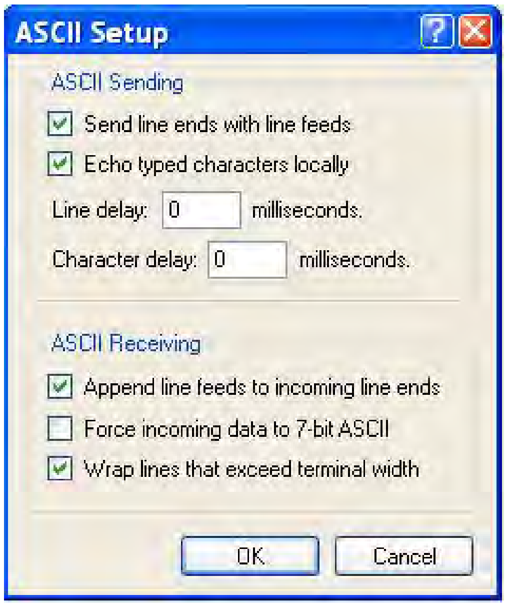
Emulation: ANSIW

Telnet terminal ID: VT100

Backscroll buffer lines: 500

☐ Play sound when connecting or disconnecting

Figure 6. The HyperTerminal Properties Dialog


ASCII Setup

ASCII Sending
☑ Send line ends with line feeds
☑ Echo typed characters locally
Line delay: 0 milliseconds.
Character delay: 0 milliseconds.

ASCII Receiving
☑ Append line feeds to incoming line ends
☐ Force incoming data to 7-bit ASCII
☑ Wrap lines that exceed terminal width

Figure 7. The HyperTerminal ASCII Setup Dialog

6. In ZDSII, click the **Compile** button to compile the code. After compilation is complete, click the **Go** button to run the code.
7. Start turning the shaft of the rotary encoder, both clockwise and counterclockwise. Also try to push down the shaft.

Results

As expected, the incremental rotary encoder produces two signals that are 90 degrees out of phase with each other.

Summary

In this application, firmware reads signals from the incremental rotary encoder and determines whether the shaft of the incremental rotary encoder is rotating clockwise or counterclockwise. The results obtained are based on a method of computing the XOR result of the current Channel A bit and the previous Channel B bit to determine the direction of shaft rotation.

References

The following supporting documents are available free for download from the Zilog website.

- Z8 Encore! XP F64xx Series Product Specification
- Z8 Encore! XP F64xx Series Development Kit User Manual (UM0151)
- See also: Rotary encoder on Wikipedia

Appendix A. Flowcharts

Figure 8 presents the flow for reading an incremental rotary encoder.

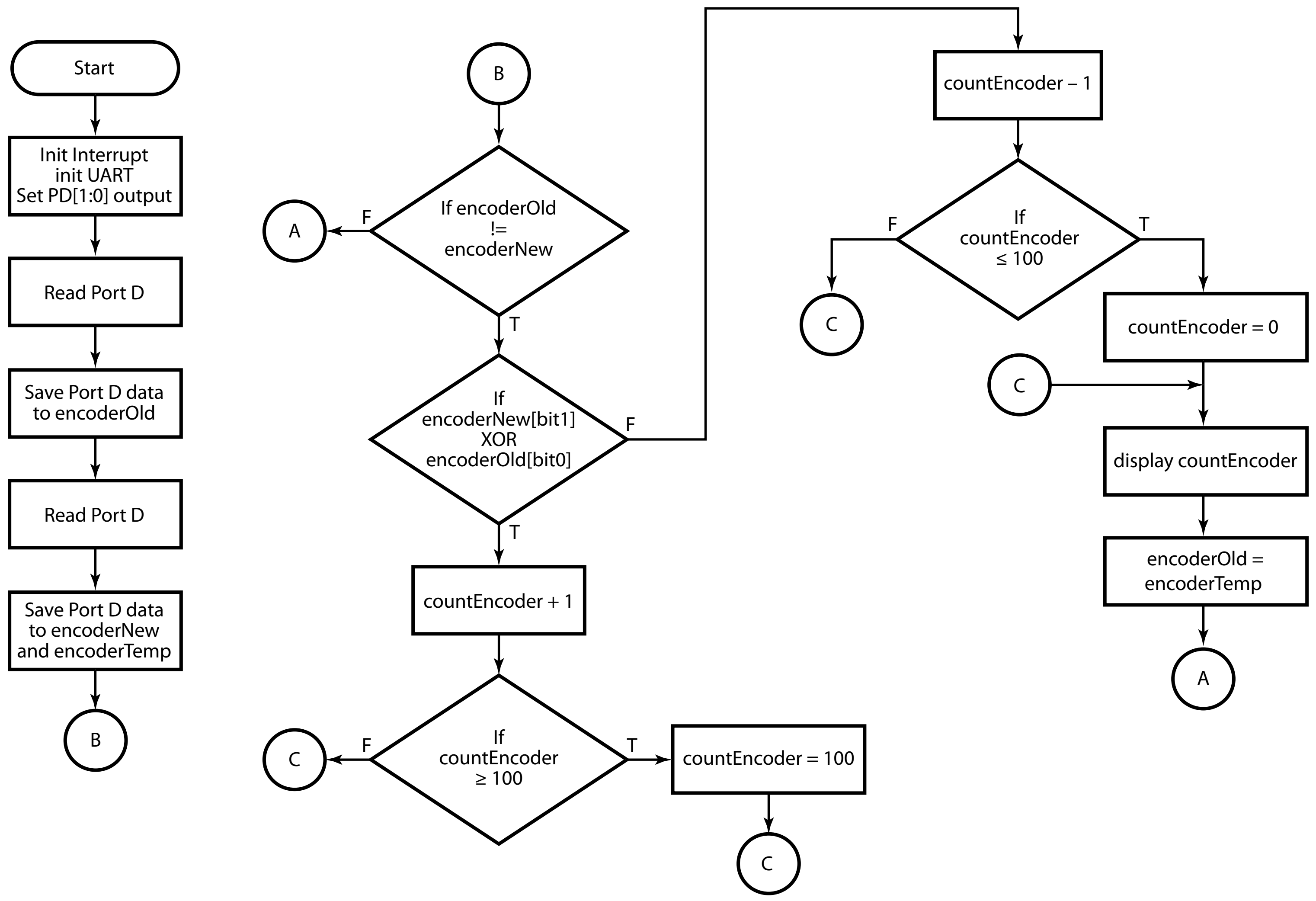


Figure 8. Flow for Reading an Incremental Rotary Encoder

Figure 9 presents the ISR flow for the pushbutton switch routine.

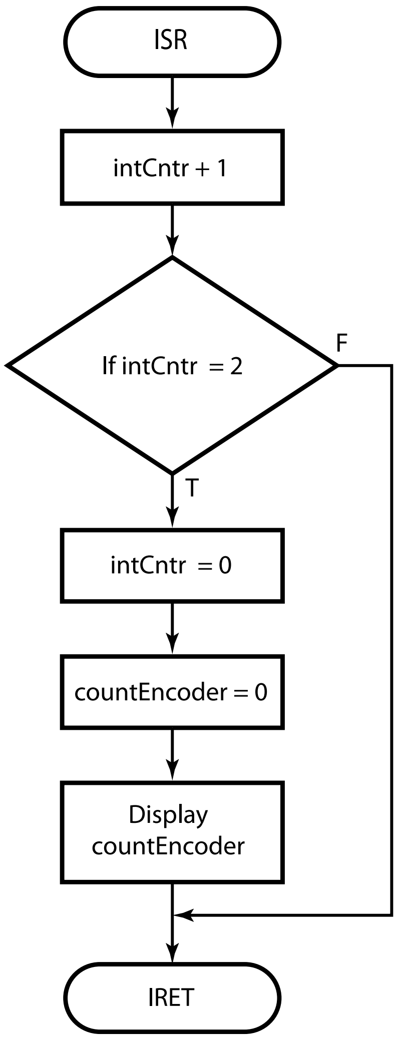


Figure 9. Interrupt Service Routine for the Pushbutton Switch

Customer Support

To share comments, get your technical questions answered, or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at http://support.zilog.com.

To learn more about this product, find additional documentation, or to discover other facets about Zilog product offerings, please visit the Zilog Knowledge Base at http://zilog.com/kb or consider participating in the Zilog Forum at http://zilog.com/forum.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, please visit the Zilog website at http://www.zilog.com.