This software shows how to run a mechanical encoder using digital inputs and interrupts. It was written by KM6SN, and is released under GPL v3.

I have used this software in over 6 projects on different hardware platforms, with 16 ppr encoders, and 24 ppr encoders with great success.

Terms

"foreground" refers to code that is executed as a result of an interrupt service routine.

"background" refers to code that is executed in a traditional loop in the main() function; N.B. background code runs with interrupts enabled.

This code appears to be rock-solid in that:

1. it successfully debounces mechanical encoder contacts in the foreground, and

2. to the best of my testing, it does not miss an encoder transition, nor does it give spurious encoder counts, and

3. the foreground is "lean", in that the foreground processing does not lock out interrupts for a period of time which would prevent other real-time interrupt activity, such as USART, etc.

However, this code is presented with no guarantee of merchantability or fitness for any purpose.

As presented, this code uses the Make software available as open-source, and avr-gcc tools. As presented, the makefile runs on a Linux system and is setup to use /dev/ttyUSB0 at 57600 baud. One can modify the makefile to accommodate other environments such as Windows and Mac.

To compile:

in a terminal window, go to the directory containing the code and type

make[ENTER]

To upload type

make install [ENTER]

There is also a make size and make clean command.

This software written to run on an hardware environment with an ATMEGA328, and an I2C 1602 LCD, and the encoder wired to

#define PB\_E\_SIN (1<<PB2)

#define PD\_E\_COS (1<<PD4)

encoder.c shows the heart of encoder handling is a state-machine that runs in the foreground.

Be aware there are subtleties that must be observed in the background code to prevent spurious responses or crashes. Look at how encMain.c handles changing pointers, etc.

Please take a deep breath before looking in encMain.c. It was written some time ago. For reasons I cannot recall at this moment, but I am sure were valid then, I limited the size of variable that the foreground manipulates to 16 bits. Therefore, in the background I used a “base” value for frequency, and added the encoder count (positive or negative) times a hzPerClick value to the “base” value. Throughout encMain.c, there are calls to “normalize” the VFO, meaning the base=(encoder count x hxPerClick)+ base. Then the encoder count is cleared. Note the necessary control of the interrupt system during those machinations.

EncMain.c is written to accommodate some number of VFOS. It is easy to save the whole vfo data structure to EEPROM for subsequent restarts. As presented, encMain.c only uses vfo[0], shown on the lcd display as <A >=

PostScript:

included in this release is a file z.geany. Geany can use this project file to to provide a simple and powerful IDE. One does NOT need to use Geany to use this software.