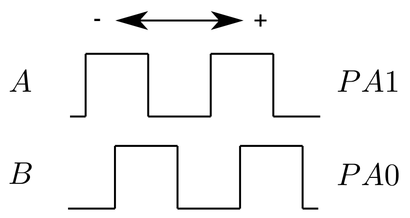
Quadrature Decoding in Software

There are several ways to perform incremental encoder "quadrature decoding" in software, but the method below is one of the simplest.

Consider the encoder quadrature waveform and associate *state transition table* shown in Figure 1 below.

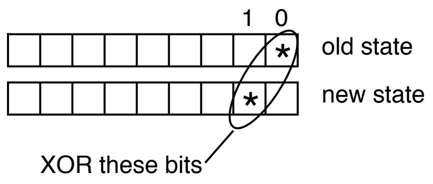


	+ direction		- direction	
state	A	B	A	B
1	0	0	0	0
2	1	0	0	1
3	1	1	1	1
4	0	1	1	0

Figure 1: Encoder waveform and state transition table.

If you look at the state transition table for each direction, you see that only 1 channel switches at each step—this is called a *gray code.* Also, it you look at phase B of the "old" state and phase A of the "new" state you will see in each direction the results of an exclusive-or (XOR, or 1-bit ADD) are a "1" (+ direction) or an "0" (- direction). This gives the way to discern direction.

Phase A of the encoder is wired to IC2, which is also bit PA1. Phase B of the encoder is wired to IC3, which is also bit PA0. Hence if you simply read PORT A (*e.g.* LDAA PORTA,X if you're using indexed addressing as you ought), you will obtain the current state of both encoder phases.



Initialization. Beyond the normal initialization of edge polarity, *etc.* you must initialize the *encoder state.* You can do this simply by reading PORTA and storing in whatever variable holds the "old state."

Interrupt Service Routines. You will have an ISR for both phase A and B; they will be *almost* identical. At every interrupt you will either increment or decrement the count—the trick is in deciding which. Here are some function/methods my ISRs used:

- Read "new state" by `ldaa PORTA,x`
- Shift "old state" left one bit by `asla` "old state"
- Perform the XOR operation by `EORA` "old state" (result of XOR in A)
- Check whether bit 1 of result is 1/0 using the "bit test" instruction `bita` with `IMM` addressing and a binary mask similar to that discussed in class
- Conditional branching to increment/decrement encoder count

Monitor Display. Remember that the screen buffering requires that you update the monitor at a relatively "slow" rate—put a software delay in your main loop and adjust the delay to get a satisfactory display.