Reading: Your lecture notes
Spec sheet for SG-531P EPSON SG-531P 5.0 mHz TTL clock oscillator chip.
Spec sheet for LS7366 Quadrature Decoder/Counter chip.
Spec sheet for Pololu 2284 motor and quadrature encoder.
MaEvArM manual pages for the following commands:
  • SPI_MasterInit, SPI_send, SPI_read, SS_low, SS_high
    [maevarmADC.h/.c]

Apparatus: MaEvArM, JHURSAEB, AVR Studio 4 on a PC, RS232 cable,
Oscilloscope, LS7366 Quadrature Decoder/Counter Chip, EPSON SG-531P-5.0000MC 5.0 mHz TTL clock oscillator.

Review the LS7366 quadrature decoder, SG-531P clock, Pololu 2284, and the circuit diagram shown in Figure 1, the circuit schematic shown in Figure 2.

- Construct the circuit shown in Figure 1 on the JHURSAEB
- See the TA’s sample setup for reference.
- Spin your motor by hand and verify with your oscilloscope that the Pololu 2284 quadrature encoder is providing two channels of incremental quadrature signals to the A and B input pins of the LS7366 quadrature decoder.

1) Make an annotated sketch showing your entire circuit. Your drawing should show your entire circuit and all of its connections, with all signals labeled.

* This laboratory © Louis L. Whitcomb
Figure 1: Quadrature Decoder Interface Circuit Diagram Sketch
2) Write and debug an AVR program to interrogate and print the encoder position from the quadrature decoder chip as outlined in class. Here is a sample code:

```c
// Quadrature decode
// 20011-10-29 LLW
#include "maevarmGEN.h"
#include "maevarmUSB.h"
#include "maevarmSPI.h"

// function to initialize the quadrature decoder
void initialize_quad()
{
    //Initialize SPI as master
    //Leave SPI_CLK to 1/4 of CPU clock
    //Leave data order at MSB first
    SPI_MasterInit();

    // Write to MR0
    SS_low(); //bring SS line low to start
    SPI_send(0x88); //0x88 (10001000) = write to Mode Register 0
    SPI_send(0xC3); //0xC3 (11000011) = 4xQuad, freerun,
    //disable index, async index, fclk/2
    SS_high(); //pull SS line high to end

    // Write to MR1
    SS_low(); //bring SS line low to start
    SPI_send(0x90); //0x90 (10010000) = write to Mode Register 1
    SPI_send(0x02); //0x02 (00000010) = two-byte count,
    //enable, no flags
    SS_high(); //bring SS line high to end

    return;
}

// function to read from quadrature decoder
unsigned int read_quad()
{
    // declare working variables
    unsigned char MSB;
    unsigned char LSB;
    unsigned int counter_value;

    SS_low(); //bring SS line low to start
    SPI_send(0x60); //0x60 (011000000) = read from counter
    MSB = SPI_read(); //read first byte
    LSB = SPI_read(); //read second byte
    SS_high(); //pull SS line high to end

    // assemble a 16 bit unsigned int from the two bytes
    counter_value = (MSB << 8) + LSB;
    return counter_value;
}
```
3) Verify that your program produces the correct write sequence (as outlined in class and in the LS7336 spec sheet) on the signals SS (on CH1), SCK (on CH2), MISO (on CH3), MOSI (on CH 4) for writing to the MR0 register.

   *Hint: add a temporary loop statement, with 200ms delay, around the section of your program which programs the MRO register. Set your scope to trigger on the falling edge of SS.*

   Print and annotate a scope plot to identify the ship select, data bytes, data bits, clock signals, and chip-de-select.

4) Verify that your program produces the correct write sequence (as outlined in class and in the LS7336 spec sheet) on the signals SS (on CH1), SCK (on CH2), MISO (on CH3), MOSI (on CH 4) for writing to the MR1 register.

   Print and annotate a scope plot to identify the ship select, data bytes, data bits, clock signals, and chip-de-select.
5) Verify that your program produces the correct write sequence (as outlined in class and in the LS7336 spec sheet) on the signals SS (on CH1), SCK (on CH2), MOSI (on CH3), MISO (on CH 4) for reading the **CNTR** register.

Print and annotate a scope plot to identify the chip select, data bytes, data bits, clock signals, and chip-de-select.

6) When your circuit and your program are OK, you should be able to observe changes in motor position and velocity in the data reported by the terminal outputs of your program.
   a) Observe and describe the relationship between the values on the terminal outputs and the shaft position, velocity, and direction.
   b) Hand in an annotated copy of your program, and a printout of the debug screen.

7) Use **banana-to-clip leads** to connect the red and black power terminals of the Pololu motor to one of your variable output power supplies. Use the variable voltage supply to spin the motor by applying various voltage magnitudes and polarities.
   a) Observe and describe the operation of your program, and the relation between the debug statement output and the shaft position, velocity, and direction.
   b) Observe and describe what happens when this 16-bit counter wraps-around at 0xFFFF and 0x0000 counts. What happens when the motor is spinning the other direction? If there are any problems here, modify your program to fix them.

8) Modify your program to compute and display the motor speed both in raw counts and also in the more natural units of **revolutions per second**.
   - Add a new **double** variable to your program to store the velocity in revolutions per second.
   - Modify the program’s printf statements so that it prints out
     i) the count as a hexadecimal unsigned integer,
     ii) delta as a signed decimal integer
     iii) rps as signed decimal floating point number
   - in a loop – the output should look something like
     count=0AF3  delta=+00034, rps=+1.234567
   - Now, looking at your scope, adjust the length of the delay in the loop so that the program loop (as indicated by the falling edge of the SS line) executes 10 times per second.
   - Modify the program to compute, based on the delta variable, the value of the motor shaft speed in revolutions per second. Assign the result to the new variable RPS.
   - Hand in a commented copy of your program

Set up the oscilloscope as follows:
   - Connect channel 1 of your oscilloscope to the SS pin. Adjust the scope trigger so that it triggers on the falling edge signal.
   - Connect channel 2 of your oscilloscope to display CHA pulses of the quadrature decoder.

9) With the motor running at a constant speed, get a scope plot of your setup in action, a printout of the debug screen.
   a) Hand in the annotated scope plot
   b) Hand in the debug screen
10) Show in a separate calculation how you determined the formula for calculating revolutions per second from delta counts. Show how your program implements this formula. Show that the terminal output values for delta and rps variables are consistent with the scope plot.
Cover Sheet for 530.420 Lab #8:
Motion Control I/0 with the MaEvArM:
Quadrature Decoding and SPI Serial I/O

My Secret Code: ______

My Lab Station: ______

My Partner’s Secret Code: ______

Lab Station Clean!
TA’s Signature & Date:

Your TA will sign here after you have finished your lab, cleaned up your lab station to perfection, and shown your lab station to your TA.