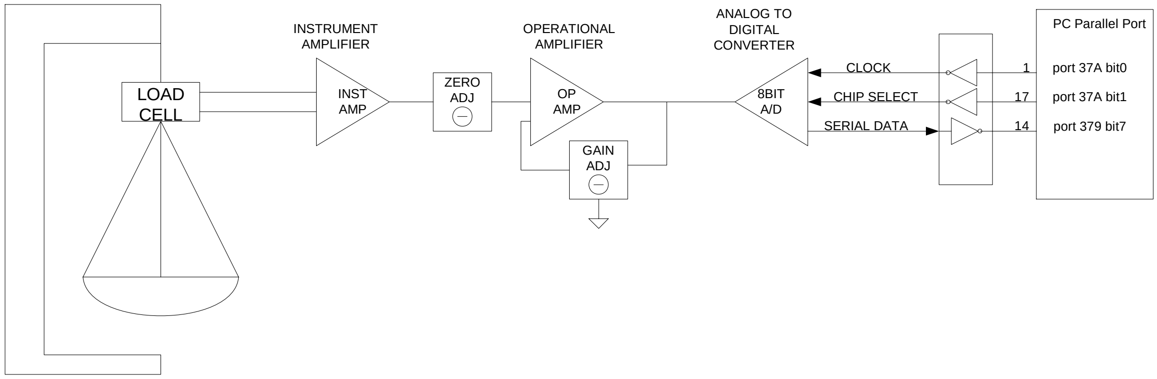
CIS 240 Scale Project (60 to 90pts)

Introduction:

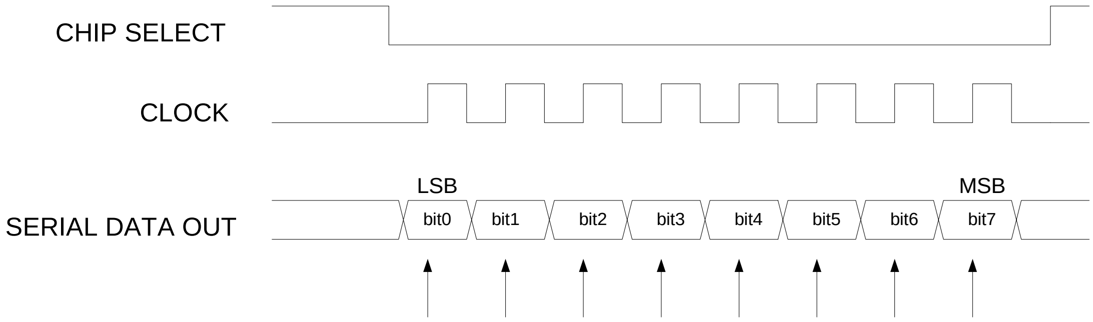
The scale project requires creating software that controls the parallel port of the PC. The scale is a hardware device as depicted in the diagram below. The scale is set up to read up to 2 lbs maximum. When 2lbs are in the scale an Analog to Digital converter output will be at FF which is the maximum hex/binary value out of the A/D converter since it only has an 8 bit output. The A/D converter feeds back the data 1 bit at a time through a serial interface. The serial interface must be controlled by your assembly language program.

The serial interface has two signals that cause a read cycle of the A/D converter. The chip select starts the conversion, then eight clock signals are supplied to read out the data one bit at a time. The data is sent out LSB to MSB and is read into the PC on port 379 bit 7. The serial data must be converted to parallel in the assembly program to generate one byte of data. The data is to be displayed as 1.99 lbs maximum therefore must be converted to fractional decimal.

The chip select and clock must be controlled by your assembly program according the timing diagram shown below the scale. Write to port 37A bits 0 and bit 1 using the OUT instruction, then read the data on port 379 bit 7 using the IN instruction.



SEND CHIP SELECT AND CLOCK OUT OF PORT 37A(bit0 & bit1) IN THE FOLLOWING SEQUENCE TO CONTROL A/D CONVERTER



READ PORT 379 bit7 SERIAL DATA AFTER RISING EDGE OF EACH CLOCK

Algorithm for converting binary to fractional decimal

The fractional result of up to 1.99 requires that there be an implied decimal place in the read 8 bit data. The decimal point is after the MSB, therefore the MSB is considered an unsigned whole number of either 1 or 0.

The fraction therefore is 7 bits of LSBs. To interpret the LSBs as decimal a binary to decimal conversion algorithm is followed.

Converting Binary to Decimal

Divide READ-DATA by 10 repeatedly saving the remainders as the decimal value LSD to MSD

```
DIV BL
Where: BL  = Ah
         AL = DATA
```

```
Example AL = 7Fh
Div bl
AH = 7  AL = C

Xor ah,ah        ;clear old remainder (should save it first)
Div bl

AH = 2  AL = 1

Xor ah,ah        ;clear old remainder (should save it first)
Div bl

AH = 1  AL = 0   ;when AL=0 the process is complete
```

The conversion is 7Fh = 127 decimal if you observe the AH remainder values you can see conversion

Converting Fractional results

The problem with the above conversion is that the result is not a fraction but a decimal whole number. We must first create a fraction out of the 7 bit LSBs in order to do this we have to divide rescale 7F to 99decimal which is the maximum fractional result.

Fraction = (7bit-LSBs* 99 + 64)/127

The 64 decimal represents a rounding value of .005 to help maintain LSB accuracy
These values as shown must be converted to hex for your program (or specified as decimal in the assembler)

The binary result of the fractional conversion, is then converted to decimal as shown in the previous example.

Displaying the output

The include file DISPLAY.INC may be used to send the converted data to the out put and align it up. The hex values must first be converted to ASCII as in previous exercises.

Adding a Filter for 30pt extra credit

Filtering A/D data can help stabilize the scale readings. The algorithm for creating an exponential averaging filter is:

History = 4 ;this is the exponential constant use between 2 and 8

New-Filtered-data = (New-data – old-Filtered-data)/History + old-Filtered-data

Old-Filtered-data = New-Filtered-data

One difficulty with this algorithm is that the data is considered unsigned therefore the subtraction of the new – old filtered data is a challenge

Deliverables

Code listing, assembled and tested code with instructor verification of operation