# Start

Initializes the registers needed in the next parts of the program, begins the first uart read

# Digits

Reads in digits from the UART until there is a ‘.’   
Also makes sure that the binary value is accurate, multiplying the previous value by 10 before adding the current value

# Negative check

Runs at the start of every program to check for and handle a negative sign. Jumps back to where it was called

# Count Digit Bits

Counts how many bits the integer part takes to represent

# No Digit Dot Init

Sets up some registers assuming there was no integer part to handle

# Dot Init

Sets up to read in the fractional part

# Dot Eval

Reads in the fractional part, and keeps the digits straight. At this point, it reads it in the same as a regular integer

# Dot Eval Power Loop

Puts into a register the max value that the fractional part cannot exceed. This utilized an algorithm where you take the fractional part and multiply by 2. If the result is greater than 1, then the corresponding binary representation has a 1. If it is less than 1, put a 0 in the binary representation. Since the PLP cannot handle fractions, we get the same result by calculating as integers. For instance, 0.2345 cannot exceed 1, but it will produce the same output if you calculated 2345 not exceeding 10000.  
This loop creates the 10000.

# Dot Eval Loop Init

Sets up registers to convert the decimal fraction to a binary fraction

# Dot Eval Loop

Converts the decimal fraction to a binary fraction

# Dot Eval Loop 1

Called by Dot Eval Loop, when a 1 needs to be set in the converted value