Document Type:	Standard Problem
Document Id:	Q6UJ9A003PQX
Posted Date:	Jun 09, 2005
Area:	Development Tools

Convert Float to ASCII in C18

Issue

How do I convert a floating point number in C18 to ASCII so I can display it, such as with printf?

Solution

Due to the extreme size of the printf library, floating point (%f %g and %h type) is not supported to prevent the library from getting larger. The cost in program memory space for adding floating point support is quite extreme (it would require the library to be aproximately double its current size), so it was chosen to remove the floating point printf support to keep the code size as small as possible and a reasonable size for most applications. This can cause problems for users who would like to display floating point numbers. However, its not difficult to convert a number to a fixed point for displaying. See the example code below:

```
// Code to demonstrate displaying floats on the USART using printf
// Note, the float is converted to two variables, a part to the
// left of the decimal, and a part to the right of the decimal.
// Unused digits are simply dropped, add the kind of rounding you like.

float fInput=344.9876; // Pick a number
long lWhole=0; // Stores digits left of decimal
unsigned long ulPart=0; // Stores digits right of decimal

#define MULTIPLIER 100 //(use 1 for no decimals, 10 for 1 place,
// 100 for 2 places, etc)

#include<stdio.h>
#include<p18f242.h> // choose your device as needed

void main(void)
{
// Set up the USART for output (not required, but since printf sends to the USART, it makes the
output readable in the output window)
SPBRG=15;
```

```
TXSTA=0x24;
RCSTA=0x90;


//Convert number from float to fixed point for display.
//The number is converted to two parts.
lWhole=(long)((float)fInput);
ulPart=(long)((float)fInput*MULTIPLIER)-lWhole*MULTIPLIER;

printf((far char *)"The floating point number is: %li.%li\n",lWhole,ulPart);
while(1); // wait forever
}
```

As you can see, the code first cuts off the decimal portion by converting the number to a long and stores the result. Then, another variable simply multiplies the original floating point number by a constant (100 is shown), then subtracts that from the whole number multiplied by the same constant (100). The result is the number that remains is the decimal portion, accurate to the number of decimal places you like. Since 100 was the multiplying factor, you will get your answer accurate to 100ths, or 2 places. Use 1 for 0 places, 10 for 1 place, 100 for 2 places, etc. This multiplying factor is simply called "MULTIPLIER" in the C code example and has been #defined.

The final trick is to combine these numbers into a readable format. The function printf handles long integers without a problem. It is used in this example to combine the two numbers into one number with a decimal point shown between the two original numbers.

Note that this routine does not handle rounding. It basically always rounds down. If you want rounding, add it to the routine according to your needs or handle your rounding before converting for display.

To handle auto-ranging, check the value of the number you wish to display and compare it to a value as needed. Depending on the result of the compare, you could change the number of decimal places (by changing the multiplying factor) as needed.

One final note: You can use the software simulator of MPLAB IDE 7.10 or newer to see the output of printf, such as to test this code. Go to "Debugger", choose "Select Tool" and then "MPLAB SIM". After that, go to "Debugger", then "Settings" then click "Uart1 IO" tab and check "Enable Uart1 I/O", and choose the "Output" radio button option "Window" in that dialog. Click OK. The simulator will now display the output of printf in the Output window under the tab "SIM Uart1". You should see the following when the program is run:

The floating point number is: 344.98