# AN670

# Floating Point to ASCII Conversion

Authors: Rick Evans
Microchip Technology, Inc.

## INTRODUCTION

It is often necessary to output a floating point number to a display. For example, to check calculations, one might want to output floating point numbers using the PICmicro® microcontrollers serial port, or use general purpose I/O to output to a liquid crystal display (LCD). Either way, the floating point number must be converted to its ASCII equivalent. This document shows a specific example of converting a 32-bit floating point number to ASCII. Application note AN575 contains 24-bit and 32-bit floating point routines. A subroutine is provided here that does the conversion and returns the ASCII equivalent in RAM. An example "main" program is provided to show how to use the subroutine.

## FLOATING POINT FORMAT OVERVIEW

Application note AN575 describes the Microchip format of 24 and 32-bit floating point numbers. We will use the 32-bit format for this example. Table 1 reviews the 32-bit floating point format.

**TABLE 1: ARGUMENT FORMAT FOR MICROCHIP 32-BIT FLOATING POINT FORMAT**

| Name | Description |
|---|---|
| AEXP | 8-bit biased exponent from -126 to +128 |
| AARGB0 | MSB of mantissa (bit7 is the sign bit) |
| AARGB1 | middle byte of mantissa |
| AARGB2 | LSB of mantissa |

**TABLE 2: 32-BIT FLOATING POINT FORMAT**

| | AARG | | | |
|---|---|---|---|---|
| Register Name | AEXP | AARG B0 | AARG B1 | AARG B2 |
| Microchip 32-bit format | xxxx xxxx | Sxxx xxxx | xxxx xxxx | xxxx xxxx |

Table 2 depicts Microchip's 32-bit floating point register RAM usage. The bit labeled "S" is the sign bit. These registers are collectively called AARG. The floating point routines require that the arguments be put in AARG and BARG (BARG is the second argument, same format as AARG). The result of the floating point operation is stored in AARG.

### Floating Point to ASCII base 10 Conversion

Floating point numbers generated by the AN575 subroutines sometimes need to be displayed. According to AN575, the number range for the floating point numbers is: $\pm 1.17549435 \times 10^{-38}$ to $\pm 6.80564693 \times 10^{+38}$. This application note will only show how to convert numbers between 0.000 to 9.999. With modification, this method can be extended to convert other ranges of numbers as well.

The calling program should ensure that the AARG registers are loaded with the correct 32-bit floating point number: either as a result of a previous floating point operation or by manually loading the AARG. The "main" routine that calls `float_ascii` is shown in Appendix A. For demonstration purposes, lets take an approximation of $\pi$ and load it into the AARG register. We'll use the number 3.1415927. (A shortcut to determine the Microchip floating point numbers is to use `fprep.exe`. The program `fprep.exe` is provided with AN575 to convert a decimal number to Microchip floating point.) Then, the `float_ascii` subroutine is called. Upon return from the subroutine, the ASCII base 10 representation of the floating point number is stored in RAM registers: `ones, tenths, hundredths, and thousandths`. Each register (ones, tenths, etc.) has an ASCII character which represents a digit. The decimal point is not included in the register RAM. Since it is given that the number is between 0.000 and 9.999, the display routine should manually output a decimal point after it outputs the first digit. Table 3 shows the ASCII values of each digit. The numbers are 3.141.

**TABLE 3: THE ASCII VALUES FOR 3.141 DECIMAL RETURNED FROM ROUTINE `float_ascii`**

| Register Name | ASCII |
|---|---|
| ones | 33h |
| tenths | 31h |
| hundredths | 34h |
| thousandths | 31h |

### Customizing the Routine

There are several changes you can make to the `float_ascii` routine to customize it. First, the number of significant figures in the number is specified by the constant `SIG_FIG`. Suppose we wanted to display one more digit of accuracy, four digits to the right of the decimal point. It is easy to alter the `floasc.inc` assembly file to account for this change. The following steps illustrate how to change the source code to return a total of five digits.

1. Ensure that there is enough RAM registers allocated to hold each digit. In this case, we would change the cblock definition as in Figure 1.

**FIGURE 1: CHANGING CBLOCK TO HOLD FIVE DIGITS**

```
cblock           ;reserve five bytes of
    ones         ;data RAM for each digit
    tenths       ;
    hundredths   ;
    thousandths  ;
    digit5       ;**** Add one more
                 ;RAM register
endc
```

2. The `last_digit` constant must be changed. This constant contains the address of the last variable in the cblock. In this case, the variable `digit5` is the last location.

    ```
    last_digit set digit5
    ```

3. Now the constant, `SIG_FIG` should be equated to the number of digits desired. For example, if we desire four digits to the right of the decimal point, there are a total of five digits that must be obtained.

    ```
    SIG_FIG equ   5
    ```

4. Load BARG with ten thousand. Use `fprep.exe` to find the floating point hexadecimal equivalent of 10,000.

**FIGURE 2: LOADING BARG WITH 10,000**

```
movlw  0x8C    ;BARG = 10,000 decimal
movwf  BEXP    ;(floating point) fprep.exe
movlw  0x1C    ;was used to get this
movwf  BARGB0  ;floating point
movlw  0x40    ;representation
movwf  BARGB1
movlw  0x00
movwf  BARGB2
```

## SUMMARY

This document demonstrated converting a limited range of the floating point numbers to ASCII. This is useful in order to display the results of some floating point operation. An example application of this code could be with the PIC14C000 microcontroller. Using the analog-to-digital converter module, one could read the voltage on a pin from 0.000 to 3.500 volts and output the decimal number to an LCD.

## APPENDIX A:

```
list p=16c74a,st=off,mm=off

#define P16_MAP1 0
#define P16_MAP2 1

        include “p16c74a.inc”
        nolist
        include “math16.inc”                    ;constants and varible definitions
        list
        cblock  0x70                            ;set cblock start address for
        endc                                    ;float_ascii routine


        org     0x000
        goto    start

        org     0x005
start

        movlw   0x80                            ;     1 2345678
        movwf   AEXP                            ;PI = 3.1415927
        movlw   0x49
        movwf   AARGB0
        movlw   0x0F
        movwf   AARGB1
        movlw   0xDB
        movwf   AARGB2

        call    float_ascii                     ;convert a 32-bit float to ASCII
                                                ;in this case
                                                ;ones = 3
                                                ;tenths = 1
                                                ;hundredths = 4
                                                ;thousandths = 1

done    goto    done


        include “floasc.inc”


        end
```

```
;****************************************************************************
;*Floating Point to ASCII
;*
;* INPUT: 32 bit floating point number in AEXP, AARGB0, AARGB1, AARGB2
;*       For this example, the number must be between 0.000 and 9.999.
;*       (You can easily change the number range to suit your needs.)
;*
;* OUTPUT: ones, tenths, hundredths, thousandths (ASCII digit in each)
;*
;*
;* USED: INDF,FSR,AARG,BARG,REMB,digit_count
;*
;* PROCEDURE: The floating point number in AARG is multiplied by 1000.
;*           Then the product is divided by 10 three times.  After each
;*           divide, the remainder is kept.
;*
;*           After each digit is obtained, 30H is added to it to make it
;*           an ASCII representation of that number.  Then, the ASCII
;*           value is stored in register RAM at the locations specified
;*           in the “cblock.”
;*
;*Note: The ASCII decimal point is not generated by this routine.
;*           You must output the decimal point in the correct decimal
;*           position for your application.  For this example, the
;*           decimal point is after the first digit: ones.
;*
;* The following files are needed for this routine to function.
;*  p16c74a.inc-- or any other midrange processor include file
;*                         include the processor file in your “main” file
;*
;*  math16.inc          -- constant and variable definitions for
;*                         AN575 floating point routines and
;*                         AN617 fixed point routines, both are used
;*                         in this float to ASCII routine
;*                         “include” this file in your “main” program
;*
;*  fxd26.a16 -- fixed point 32/16 divide, included at the end
;*               of this routine.
;*
;*  fp32.a16-- 32 bit float to 32 bit integer conversion
;*             included at the end of this program.
;*
;****************************************************************************


;RAM Register Definitions       Your “main” program must have a “cblock”
                                ;directive with a RAM address so the
                                ;following “cblock” will be located in RAM

      cblock                    ;reserve four bytes of data RAM for
        ones                    ;each digit
        tenths
        hundredths
        thousandths

      endc

last_digit set thousandths

      cblock
        digit_count             ;counter used to cycle through each digit
      endc


SIG_FIG equ     4               ;set SIG_FIG equal to the number of
```

 

```
                                ;significant figures in your decimal number
                                ;for example: ones, tenths,hundredths,
                                ;thousandths, requires 4 sig figs

float_ascii

        movlw   0x88            ;BARG= 1000 decimal (floating point)
        movwf   BEXP            ;fprep.exe was used to get this
        movlw   0x7A            ;floating point representation of 1000
        movwf   BARGB0
        movlw   0x00
        movwf   BARGB1
        movlw   0x00
        movwf   BARGB2


        call    FPM32           ;AARG = AARG * 1000

        call    INT3232         ;AARG  <--  INT( AARG )

        movlw   last_digit
        movwf   FSR             ;pointer = address of smallest digit
        movlw   SIG_FIG         ;load counter with the number of
        movwf   digit_count     ;significant figures the decimal number

flo_asclp
        clrf    BARGB0          ;Make the divisor 10.
        movlw   d’10’
        movwf   BARGB1

        call    FXD3216U        ;divide (32-bit fixed) / 10 (to get remainder)

        movf    REMB1,w         ;put remainder in w register
        movwf   INDF            ;put number into appropriate digit position

        movlw   0x30
        addwf   INDF,f          ;add 30h to decimal number to convert to ASCII

        decf    FSR,f           ;move pointer to next digit

        decfsz  digit_count,f
        goto    flo_asclp

        return

        nolist

        include “fxd26.a16”     ;fixed point 32/16 divide from AN617

        include “fp32.a16”      ;32 bit float routines
                                ;we are using FPM32 for 32-bit multiply
                                ;and INT3232 for 32-bit float to 32-bit int
                                ;conversion.  Routines are in AN575
```

**NOTES:**

**Note the following details of the code protection feature on PICmicro® MCUs.**

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.



Printed on recycled paper.

*Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company’s quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001 certified.*

**Note the following details of the code protection feature on PICmicro® MCUs.**

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.



**Trademarks**



Printed on recycled paper.

*Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company’s quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001 certified.*

# WORLDWIDE SALES AND SERVICE


## AMERICAS

**Corporate Office**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

**Rocky Mountain**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

**Atlanta**
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

**Boston**
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

**Chicago**
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

**Dallas**
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

**Detroit**
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

**Kokomo**
2767 S. Albright Road
Kokomo, Indiana 46902
Tel: 765-864-8360 Fax: 765-864-8387

**Los Angeles**
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

**New York**
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

**San Jose**
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

**Toronto**
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

## ASIA/PACIFIC

**Australia**
Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

**China - Beijing**
Microchip Technology Consulting (Shanghai) Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

**China - Chengdu**
Microchip Technology Consulting (Shanghai) Co., Ltd., Chengdu Liaison Office
Rm. 2401, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-6766200 Fax: 86-28-6766599

**China - Fuzhou**
Microchip Technology Consulting (Shanghai) Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

**China - Shanghai**
Microchip Technology Consulting (Shanghai) Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

**China - Shenzhen**
Microchip Technology Consulting (Shanghai) Co., Ltd., Shenzhen Liaison Office
Rm. 1315, 13/F, Shenzhen Kerry Centre,
Renminnan Lu
Shenzhen 518001, China
Tel: 86-755-2350361 Fax: 86-755-2366086

**Hong Kong**
Microchip Technology Hongkong Ltd.
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

**India**
Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

**Japan**
Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

**Korea**
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

**Singapore**
Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-334-8870 Fax: 65-334-8850

**Taiwan**
Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

## EUROPE

**Denmark**
Microchip Technology Nordic ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

**France**
Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

**Germany**
Microchip Technology GmbH
Gustav-Heinemann Ring 125
D-81739 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

**Italy**
Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

**United Kingdom**
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820


01/18/02