Fast Integer Square Root

Author: Ross M. Fosler
Microchip Technology Inc.

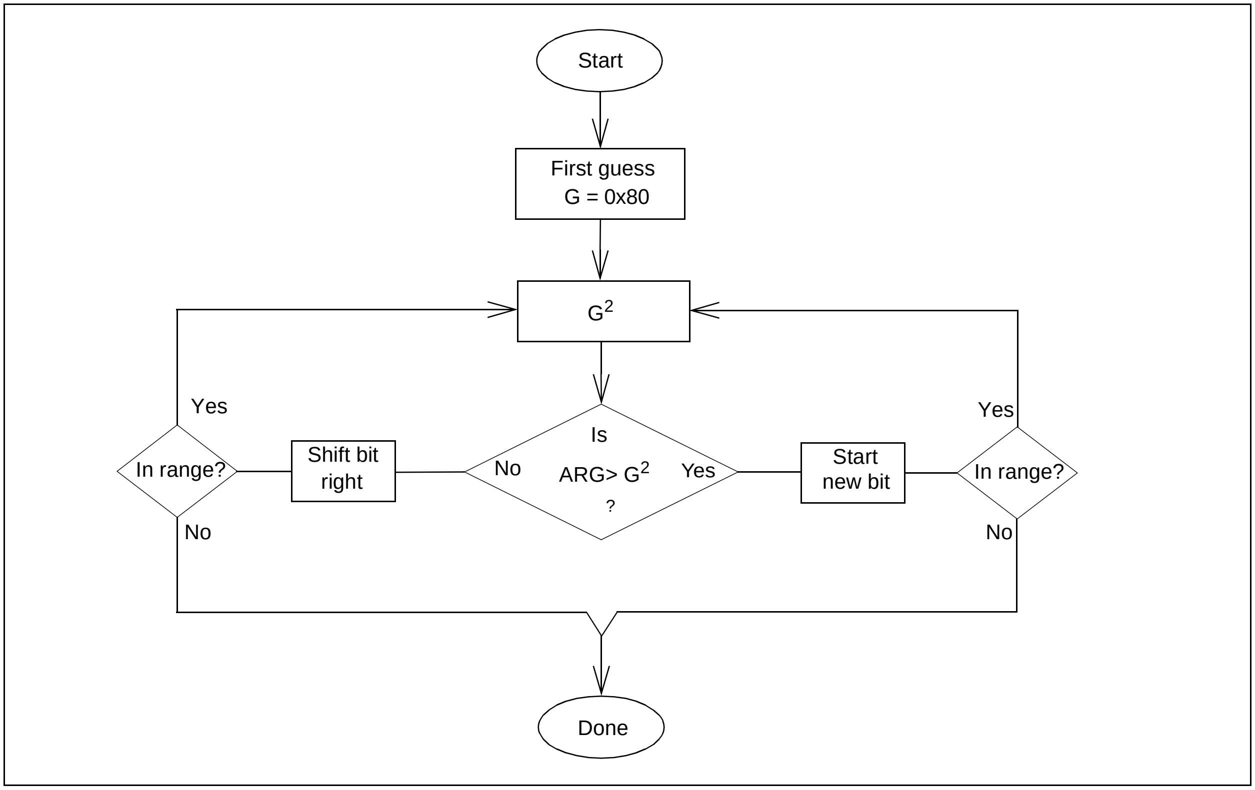
INTRODUCTION

One very common and relatively quick method for finding the square root of a number is the Newton-Raphson method. Although this method is quick in terms of mathematics, it also requires extensive use of division to produce results, usually iterating many times. In the PIC18CXX2 microcontroller family, though not difficult, division does requires several basic operations. However, with the help of the single cycle hardware multiplier, one of the many nice features in the PIC18CXX2 and the use of a technique different from the Newton-Raphson method, division is avoided. The following algorithm demonstrates how the single cycle multiplier is useful in calculating a square root and at the same time, save processor time.

THE ALGORITHM

Using the binary nature of the microcontroller, the square root of a fixed precision number can be found quickly. Each digit in a binary number represents a power of two. By successively rotating through each bit, or power of two and testing the result against the desired value, i.e. squaring the guess and checking if it is greater than the original argument, the approximate root gets closer and closer to the actual value. In conjunction with this, the value is achieved quickly. This is because each bit is tested rather than every possible 8-bit combination. The general technique is outlined in Figure 1. For a 16-bit integer, only nine program loops are required to completely test and produce a result. Example 1 is a demonstration of this procedure:

FIGURE 1: SQUARE ROOT FLOW CHART



EXAMPLE 1: 8-BIT EXAMPLE

$$A = \sqrt{0xCF48}$$

or

$$A^2 = 0xCF48$$

Step	A	Description
1	1000 0000 (0x80)	this squared is less than 0xCF48, start next cycle with a new bit
	new bit	
2	1100 0000 (0xC0)	this squared is less than 0xCF48, start next cycle with a new bit
	new bit	
3	1110 0000 (0xE0)	this squared is less than 0xCF48, start next cycle with a new bit
	new bit	
4	1111 0000 (0xF0)	this is greater than 0xCF48, shift bit right
	shifted bit	
5	1110 1000 (0xE8)	this is greater than 0xCF48, shift bit right
	shifted bit	
6	1110 0100 (0xE4)	this squared is less than 0xCF48, start next cycle with a new bit
	new bit	
7	1110 0110 (0xE6)	this squared is less than 0xCF48, start next cycle with a new bit
	new bit	
8	1110 0111 (0xE7)	this is greater than 0xCF48, shift right
	bit shifted out	
9	1110 0110 (0xE6)	right-most bit is thrown away for the integer approximation and the process is finished; otherwise, this could keep going for more accurate fractional approximation

ANALYSIS

Following the flow of this algorithm, there are only nine loops for an 8-bit number. And summing all the mathematics involved, there is only one multiplication and one conditional test for each step; a conditional test is most likely a subtraction with some bit testing. Plus, there are some logical operations to perform the bit manipulations, again one per loop. This means there are three basic operations per loop, totaling to 27 operations for the complete routine. Of course, the actual number of operations goes up some when applied to a specific microcontroller, but subjectively speaking, this is still not bad when compared to the large number of steps required to perform any number of divisions as required by the Newton-Raphson method.

The program in Appendix A is a functioning demonstration of the algorithm described above for 16-bit and 32-bit numbers. Table 1 gives the simulation results for these code examples. Also, the code is written specifically for the PIC18CXX2 series microcontrollers, but it can be modified to run on PIC17C microcontrollers that have a hardware multiplier.

TABLE 1: PERFORMANCE

	Max Cycles	Time		
		40MHz	**10MHz**	**4MHz**
16-bit Square Root	149	14.9us	59.6us	149us
32-bit Square Root	1002	100.2us	400.8us	1002us

CONCLUSION

This algorithm is just one possible way to compute the square root of a number. Its advantage is in the use of multiplication, a function easily performed on the PIC18CXX2 microcontroller, rather than division, an operation requiring a number of basic operations. In addition, the method and coding are extremely simple, requiring very little program and data memory. The end result is a fast and compact method to calculate the integer square root of a number.

MEMORY REQUIREMENTS

```
                          Section Info
  Section       Type    Address   Location Size(Bytes)
---------  ---------  ---------  ---------  ---------
   R_Vctr       code   0x000000    program   0x000004
   .cinit    romdata   0x00002a    program   0x000002
   S_ROOT       code   0x00002c    program   0x0000f4
    SRoot       code   0x000120    program   0x000022
  SimpMth      udata   0x000000       data   0x000014



             Program Memory Usage
              Start         End
           ---------   ---------
            0x000000    0x000003
            0x00002a    0x000141
284 out of 32786 program addresses used, program memory utilization is 0%
```

APPENDIX A: MAIN.ASM

```
; *******************************************************************
    Title"Square Root Calling Routine Demo"
; *******************************************************************

; *******************************************************************
; ***                                                             ***
; ***     Author: Ross Fosler                                     ***
; ***             Applications Engineer                           ***
; ***             Microchip Technology Inc.                       ***
; ***                                                             ***
; ***     Program:main.asm                                        ***
; ***             This routine calls the square root function     ***
; ***             to find the root of two arbitrary numbers.      ***
; ***                                                             ***
; ***     Last Rev:August 10, 2000                                ***
; ***     Ver 1.00                                                ***
; ***                                                             ***
; *******************************************************************

; *******************************************************************
    listp=18C252
    #include P18C252.INC
; *******************************************************************

; *******************************************************************
    EXTERN  ARGA0, ARGA1, ARGA2, ARGA3
    EXTERN  RES0, RES1
    EXTERN  Sqrt
; *******************************************************************

; *******************************************************************
W           equ         0               ; Standard constants
F           equ         1
a           equ         0
; *******************************************************************

; *******************************************************************
R_Vctr      CODE        0x0000
            goto        Main
; *******************************************************************
```

```
; *********************************************************************
; Calling Routine
SRoot       CODE

Main
            movlw       0xCF
            movwf       ARGA1, a
            movlw       0x48
            movwf       ARGA0, a

            call        Sqrt        ; Sqrt(0xCF48)
                                    ; RES0 should now contain 0xE6

            movlw       0xE0
            movwf       ARGA3, a
            movlw       0x12
            movwf       ARGA2, a
            movlw       0xA1
            movwf       ARGA1, a
            movlw       0x40
            movwf       ARGA0, a

            call        Sqrt        ; Sqrt(0xE012A140)
                                    ; RES1:RES0 should now contain 0xEF81

            bra         Main
; *********************************************************************

        END
```

APPENDIX B: SQRT.ASM

```
; *******************************************************************
    Title"16/32 bit Integer Square Root"
; *******************************************************************

; *******************************************************************
; ***                                                             ***
; ***   Author: Ross Fosler                                       ***
; ***           Applications Engineer                             ***
; ***           Microchip Technology Inc.                         ***
; ***                                                             ***
; ***  Program:sqrt.asm                                           ***
; ***           This module contains code to perform fast integer ***
; ***           square root functions on either 16 or 32 bit      ***
; ***           values.                                           ***
; ***                                                             ***
; *** Last Rev:August 10, 2000                                    ***
; ***           Ver 1.00                                          ***
; ***                                                             ***
; *******************************************************************

; *******************************************************************
    #include P18C252.INC
; *******************************************************************

; *******************************************************************
MSB         equ         7       ; general literal constants
LSB         equ         0
W           equ         0
F           equ         1
a           equ         0
; *******************************************************************

; *******************************************************************
SimpMth     UDATA_ACS

ARGA0       res         1       ; various argument registers
ARGA1       res         1
ARGA2       res         1
ARGA3       res         1

    GLOBAL  ARGA0,  ARGA1,  ARGA2,  ARGA3

ARG1H       res         1
ARG1L       res         1
ARG2H       res         1
ARG2L       res         1

    GLOBAL  ARG1H,  ARG1L,  ARG2H,  ARG2L

SARG1       res         1       ; signed arguments
SARG2       res         1

    GLOBAL  SARG1,  SARG2

RES1        res         1       ; result registers
RES0        res         1

    GLOBAL  RES0,   RES1

SQRES0      res         1
SQRES1      res         1
SQRES2      res         1
```

```
SQRES3     res        1

    GLOBAL  SQRES0, SQRES1, SQRES2, SQRES3

BITLOC0    res        1      ; temporary registers
BITLOC1    res        1
TEMP0      res        1
TEMP1      res        1
; *******************************************************************

; *******************************************************************
; The function of this square root routine is to determine the root
; to the nearest integer.  At the same time the root is found at the
; best possible speed; therefore, the root is found a little differently
; for the two basic sizes of numbers, 16-bit and 32-bit.  The following
; differentiates the two and jumps to the appropriate function.

; Sqrt(ARGA3:ARGA2:ARGA1:ARGA0) = RES1:RES0

S_ROOT     CODE

Sqrt       tstfsz     ARGA3, a       ; determine if the number is 16-bit
           bra        Sqrt32         ;  or 32-bit and call the best function
           tstfsz     ARGA2, a
           bra        Sqrt32
           clrf       RES1, a
           bra        Sqrt16

           GLOBAL     Sqrt
; *******************************************************************

; ******************** Square Root **********************************
;  Sqrt16(ARGA1:ARGA0) = RES0

Sqrt16     clrf       TEMP0, a       ; clear the temp solution
           movlw      0x80           ; setup the first bit
           movwf      BITLOC0, a
           movwf      RES0, a

Square8    movf       RES0, W, a     ; square the guess
           mulwf      RES0, a

           movf       PRODL, W, a    ; ARGA - PROD  test
           subwf      ARGA0, W, a
           movf       PRODH, W, a
           subwfb     ARGA1, W, a

           btfsc      STATUS, C, a
           bra        NextBit        ; if positive then next bit
                                     ; if negative then rotate right

           movff      TEMP0, RES0    ; move last good value back into RES0
           rrncf      BITLOC0, F, a  ; then rotote the bit and put it
           movf       BITLOC0, W, a      ; back into RES0
           iorwf      RES0, F, a

           btfsc      BITLOC0, 7, a      ; if last value was tested then get
           bra        Done               ; out

           bra        Square8            ; elso go back for another test

NextBit    movff      RES0, TEMP0        ; copy the last good approximation
           rrncf      BITLOC0, F, a      ; rotate the bit location register
           movf       BITLOC0, W, a
           iorwf      RES0, F, a
```

```
           btfsc      BITLOC0, 7, a      ; if last value was tested then get
           bra        Done               ; out

           bra        Square8

Done       movff      TEMP0,RES0         ; put the final result in RES0
           return

           GLOBAL     Sqrt16
; *******************************************************************

; ******************** Square Root **********************************
;  Sqrt32(ARGA3:ARGA2:ARGA1:ARGA0) = RES1:RES0

Sqrt32     clrf       TEMP0, a           ; clear the temp solution
           clrf       TEMP1, a
           clrf       BITLOC0, a         ; setup the first bit
           clrf       RES0, a
           movlw      0x80
           movwf      BITLOC1, a         ; BitLoc = 0x8000
           movwf      RES1, a            ; RES = 0x8000

Squar16    movff      RES0, ARG1L        ; square the guess
           movff      RES1, ARG1H
           call       Sq16

           movf       SQRES0, W, a       ; ARGA - PROD  test
           subwf      ARGA0, W, a
           movf       SQRES1, W, a
           subwfb     ARGA1, W, a
           movf       SQRES2, W, a
           subwfb     ARGA2, W, a
           movf       SQRES3, W, a
           subwfb     ARGA3, W, a

           btfsc      STATUS, C, a
           bra        NxtBt16            ; if positive then next bit
                                         ; if negative then rotate right

           addlw      0x00               ; clear carry
           movff      TEMP0, RES0        ; move last good value back into RES0
           movff      TEMP1, RES1

           rrcf       BITLOC1, F, a      ; then rotote the bit and put it
           rrcf       BITLOC0, F, a
           movf       BITLOC1, W, a      ; back into RES1:RES0
           iorwf      RES1, F, a
           movf       BITLOC0, W, a
           iorwf      RES0, F, a

           btfsc      STATUS, C, a       ; if last value was tested then get
           bra        Done32             ; out

        bra        Squar16            ; elso go back for another test

NxtBt16 addlw      0x00               ; clear carry
        movff      RES0, TEMP0        ; copy the last good approximation
        movff      RES1, TEMP1

        rrcf       BITLOC1, F, a      ; rotate the bit location register
        rrcf       BITLOC0, F, a
        movf       BITLOC1, W, a      ; and put back into RES1:RES0
        iorwf      RES1, F, a
        movf       BITLOC0, W, a
        iorwf      RES0, F, a
```

```
       btfsc      STATUS, C, a       ; if last value was tested then get
       bra        Done32             ; out

       bra        Squar16

Done32 movff      TEMP0,RES0         ; put the final result in RES1:RES0
       movff      TEMP1,RES1
       return

       GLOBAL     Sqrt32
; ******************************************************************

; *********** 16 X 16 Unsigned Square *****************************
; SQRES3:SQRES0 = ARG1H:ARG1L ^2


Sq16   movf       ARG1L, W, a
       mulwf      ARG1L              ; ARG1L * ARG2L ->
                                     ; PRODH:PRODL
       movff      PRODH, SQRES1      ;
       movff      PRODL, SQRES0      ;

       movf       ARG1H, W, a
       mulwf      ARG1H              ; ARG1H * ARG2H ->
                                     ; PRODH:PRODL
       movff      PRODH, SQRES3      ;
       movff      PRODL, SQRES2      ;

       movf       ARG1L, W, a
       mulwf      ARG1H              ; ARG1L * ARG2H ->
                                     ; PRODH:PRODL
       movf       PRODL, W, a        ;
       addwf      SQRES1, F, a       ; Add cross
       movf       PRODH, W, a        ; products
       addwfc     SQRES2, F, a       ;
       clrf       WREG, a            ;
       addwfc     SQRES3, F, a       ;

       movf       ARG1H, W, a        ;
       mulwf      ARG1L              ; ARG1H * ARG2L ->
                                     ; PRODH:PRODL
       movf       PRODL, W, a        ;
       addwf      SQRES1, F, a       ; Add cross
       movf       PRODH, W, a        ; products
       addwfc     SQRES2, F, a       ;
       clrf       WREG, W            ;
       addwfc     SQRES3, F, a       ;

       return

       GLOBAL     Sq16
; ******************************************************************
    end
```

MICROCHIP

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-786-7200 Fax: 480-786-7277
Technical Support: 480-786-7627
Web Address: http://www.microchip.com

Rocky Mountain
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-786-7966 Fax: 480-786-7456

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: 978-692-3838 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

Dayton
Two Prestige Place, Suite 130
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

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

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

China - Beijing
Microchip Technology Beijing Office
Unit 915
New China Hong Kong Manhattan Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Shanghai
Microchip Technology Shanghai Office
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

Hong Kong
Microchip Asia Pacific
RM 2101, 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, 560 027, India
Tel: 91-80-207-2165 Fax: 91-80-207-2171

Japan
Microchip Technology Intl. Inc.
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
Tel: 82-2-554-7200 Fax: 82-2-558-5934

ASIA/PACIFIC (continued)

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

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

Italy
Arizona 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

9/01/00

DNV Certification, Inc.
USA
ANSI★RAB
QMS
ACCREDITED

DNV

DNV MSC
The Netherlands
Accredited by the RvA

ISO 9001 / QS-9000
REGISTERED FIRM

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.