Technical Note

Floating Point Routines

TN000102-0712

General Overview

Arithmetic routines are common in a wide range of embedded applications. From home HVAC systems to industrial process parameter measurement, a certain amount of precise computation is always necessary. 8-bit controllers normally offer fixed-point arithmetic and logic units (ALUs) and typically compute using only whole numbers. However, it is more convenient to represent real numbers. The format for real numbers is described in the IEEE 754 standard and is known as *floating point representation*.

In this technical note, the basics of floating point representation are described: how to represent numbers in the real format, and how to convert whole and floating point numbers. Three companion technical notes, TN0002, TN0003, and TN0004, describe the Add/Subtract, Multiply, and Divide routines, respectively.

Discussion

The Z8 Floating Point Library is a math library that allows the user to operate with numbers presented in an exponential format corresponding to the IEEE 754 standard. This standard is represented graphically in Figure 1. The Library is a set of assembler functions which are low-time, low-space solutions for development of dedicated assembly calculation routines for Zilog's Z8 family of devices.

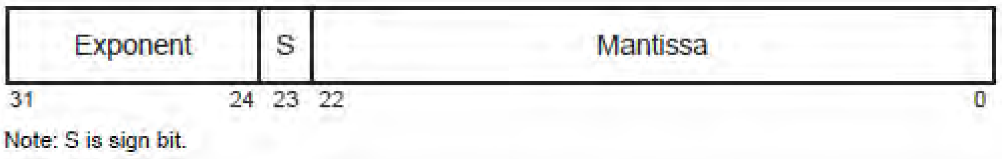


Figure 1. 32-Bit IEEE 754 Format

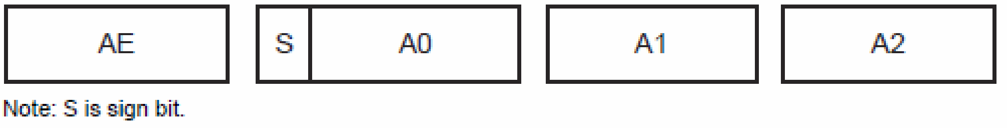
Figure 2 represents the logical allocation of registers for the purpose of floating number representation. This 32-bit format allows real or natural numbers.

For example:

0.25 = 2^{-1} = 3F000000h

0.33 = 3EA8F5C3h

3.1415 = 40490E56h



Note: S is sign bit.

Figure 2. Byte Representation of the Floating Point Number

To facilitate routine usage, the user can operate with whole numbers and use floating point routines for precise intermediate calculations. For this purpose, two functions are provided. These functions, `z8fpInt` and `z8fpFlo`, perform conversion from the integer format to the floating point format, and vice versa.

The Z8 Floating Point Library uses one register file in the Z8 architecture that can be assigned by the user in the fplib.inc control file. Up to 14 registers are in use by the floating point functions. Two registers are available to the user, as shown in Table 1.

Table 1. Available Floating Point Registers

AE	A0	A1	A2	S0	S1	S2	
BE	B0	B1	B2		TEMP	CNT	ATMP

Note: A = accumulator, B = second operand, S = intermediate sum, TEMP = temporary storage, CNT = counter, ATMP = temporary accumulator byte exchange.

The first routine, `z8fpInt`, performs conversion of a floating point number to the corresponding 16-bit integer. The number returned from the function is usually a result of a series of operations, and may be used for printing on an output device or for sending to other operation modules that do not offer floating point support. Conversion is performed by means of shifting the value of the mantissa to the left while decreasing the exponent to 0. Only numbers in the range 32768–32767 can be converted to achieve proper results. If the result is too large to be represented as a 16-bit integer , the error flag (the Carry flag in the Z8 Flags register) is set. The function provides the signed integer result. This routine uses 58 bytes of code and 4 registers.

The second routine, `z8fpFlo`, performs conversion of an integer to a floating point representation. Any 16-bit number can be converted to floating point format. A fixed-point format assumes that the decimal point is located after the least significant bit (lsb) of the operand. The floating point mantissa is at the left of its most significant bit. Conversion is performed by means of shifting the integer value to the right while increasing the exponent accordingly. The operation of setting the point at the left of the most significant bit is known as normalization . This operation uses only 17 bytes of code and 4 registers, and passes control to the normalization routine.

A normalization routine ensures that the result of any operation conforms to the IEEE 754 standard. It shifts the mantissa to the left until the value of its most significant bit is 1. Every shift is accompanied by a decrement of the exponent. The function is called automatically at the final stage of every floating point routine and normally does not require direct use.

Example

	A (decimal)	AE	A0	A1	A2
	100	00	00	03h	E8h
z8fpFlo	Intermediate*	4Ah	00	03h	E8h
z8fpNorm	100,000	42h	C8h	00	00
z8fpInt	100	00	00	03h	E8h
Note: *An automatic call to z8fpNorm follows.					

Source Code

```
include "fplib.inc"
;* ---------------------------------------------------------------------------
; Floating Point to Integer conversion routine.
; Parameter A = (AE, A0, A1, A2) - 32 bit float number to convert to integer.
; Return value is 16 bit integer in least significant bytes of operand A = (A1,
; A2)
;
; Numbers in the module range from 1 to 32768 can be converted. Numbers that
; are lower than 1 are converted to 0. In case number is out of the 16-bit
; range, overflow flag is set, no conversion takes place.
;
z8fpInt:
  clr   S1
  ld    S2, #1          ; explicit 1
  rlc   A2
  rlc   A1
  rlc   A0              ; to restore exponent
  rlc   AE              ; C = sign
  rlc   ATMP            ; ATMP.0 = sign
  cp    AE, #7Fh
  jr    C, int_zero     ; exponent is less than 0, write 0
  cp    AE, #8Eh
  jr    NC, z8_ovr      ; exponent is greater than 15
  sub   AE, #7Fh
  jr    Z, int_done
int_r:
  rlc   A2
  rlc   A1
  rlc   A0              ; moving mantissa
  rlc   S2
  rlc   S1              ; to produce integer
  djnz  AE, int_r       ; until exponent is 0
int_done:
  ld    A1, S1
  ld    A2, S2
  ret
```

```
int_zero:
  clr   A1
  clr   A2
  ret

z8_ovr:
  or    FLAGS, #OVF
  ret

;* ------------------------------------------------------------------------------
; Integer to Floating Point Representation conversion routine.
; Parameter A = (A1, A2) - 16 bit integer to convert.
; Result is 24 bit value in operand A = (AE, A0, A1, A2)
z8fpFlo:
  ld    AE, #FP_BASE + 22
  clr   A0
  rlc   A1              ; sign out
  rrc   AE              ; write sign
  jr    nc, flo_pos
  or    A0, #80h        ; store LSB of AE
flo_pos:
  rcf
  rrc   A1
  jr    z8fpNorm        ; proceed to normalization

;* ------------------------------------------------------------------------------
; Normalization routine for floating point format
; Normal floating number format:
; seee eeee e.xxx xxxx xxxx xxxx xxxx xxxx
; Parameter is a value in operand A location, the value to be normalized.
; Normalization assumes 0 at MSB. The routine is used internally by other
; floating point routines, often as a final step of every mathematical function
; in this series of Technical Notes. Operand in location A is normalized. The
; function contains check section which determines if the normalization is
; necessary and the normalization cycle that aligns the mantissa
; and adjusts the exponent in due order.
z8fpNorm:
  rcf                   ; to intermediate format first
  rlc   A0
  rlc   AE
  rlc   TEMP            ; TEMP.0 = sign
  rcf
  rrc   A0              ; for normalization assume MSB=0!

do_norm:
  rcf
; rounded normalization suppose that LSB was computed and set
; before in a calling routine, must ensure that this cycle is
; the last norm cycle!
do_norm_rounded:
  rlc   A2
  rlc   A1
  rlc   A0
  dec   AE
  jp    Z, z8fp_ovr     ; overflow
  cp    A0, #0
  jr    nz, check_norm
  cp    A1, #0
  jr    nz, check_norm
  cp    A2, #0
  jr    z, zero_result  ; three zeros - finishing normalize

check_norm:
  tm    A0, #80h        ; check if normalized
  jr    z, do_norm

end_norm:
  rlc   A0              ; restore IEEE 754 format
  rrc   TEMP
  rrc   AE
  rrc   A0
  ret
zero_result:
  clr   AE              ; zero to be returned as a result
  ret
```

Customer Support

To share comments, get your technical questions answered, or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at http://support.zilog.com.

To learn more about this product, find additional documentation, or to discover other facets about Zilog product offerings, please visit the Zilog Knowledge Base at http://zilog.com/kb or consider participating in the Zilog Forum at http://zilog.com/forum.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, please visit the Zilog website at http://www.zilog.com.

Warning: DO NOT USE THIS PRODUCT IN LIFE SUPPORT SYSTEMS.

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer

Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZILOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZILOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering.

Z8, Z8 Encore! and Z8 Encore! XP are trademarks or registered trademarks of Zilog, Inc. All other product or service names are the property of their respective owners.