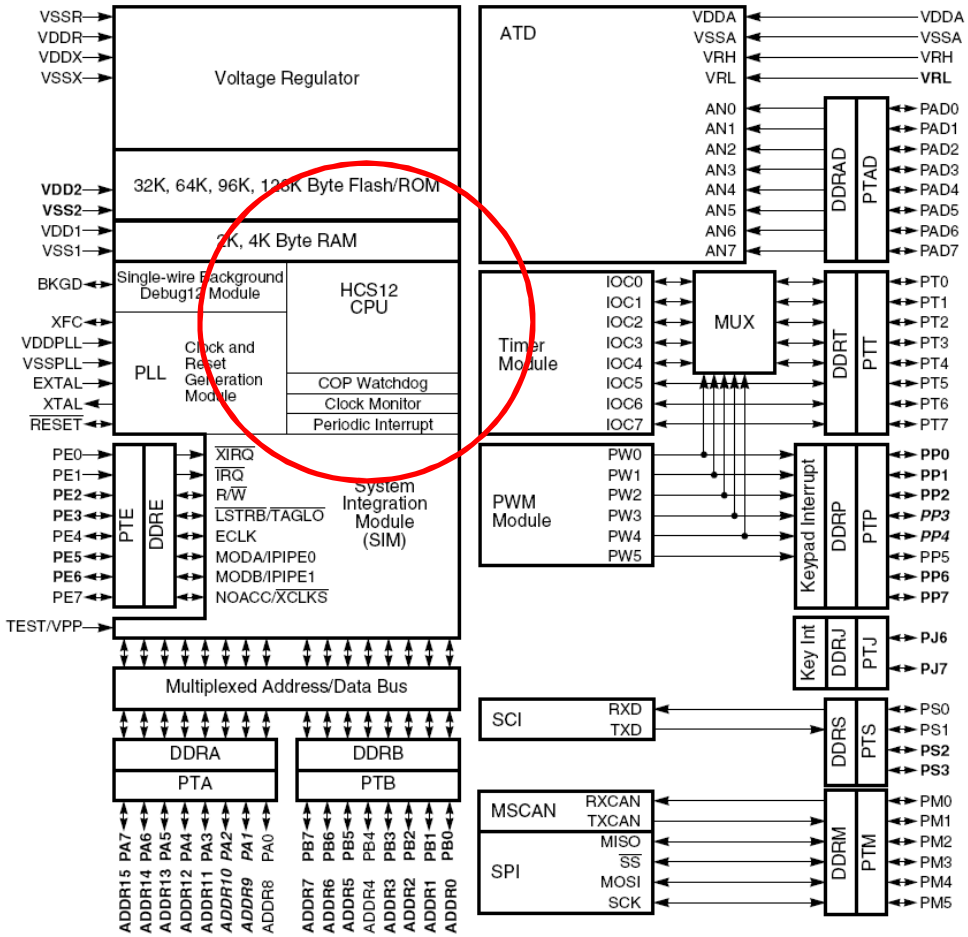
Floating Point IEEE 754 e definizione di numeri Fixed Point

Introduzione al trattamento di numeri Floating Point in sistemi a microprocessore

Esempio: MC9S12C32 Freescale Microcontroller



- CPU with Hardware multiplier
- Very good timing unit
- Very good PWM module

- Widely used in automotive applications
- NO FPU (Floating Point Unit)

Floating Point Numbers

$$value = (-1)^{sign} \cdot \left(1 + \sum_{i=1}^{23} b_i \cdot 2^{-i}\right) \cdot 2^{(e-127)}$$

The IEEE 754 is the most widely accepted standard representation for floating point numbers. The standard provides definitions for *single precision* and *double precision* representations. The single precision IEEE FPS format is composed of 32 bits, divided into a 23 bit mantissa, M, an 8 bit exponent, E, and a sign bit, S:

S (31)	**e (30-23)**								**m (22-0) "b_i"**																			
0	0	0	0	1	0	1	0	1	1	0	0	0	1	1	0	0	1	1	1	1	0	0	0	1	0	1	0	0	0	0	0
								1	Hidden bit M = m-1																			

- The normalized mantissa, m, is stored in bits 0-22 with the hidden bit, b_0, omitted. Thus M = m-1.
- The exponent, e, is represented as a bias-127 integer in bits 23-30. Thus, E = e+127.
- The sign bit, S, indicates the sign of the mantissa, with S=0 for positive values and S=1 for negative values.

IEEE754: Floating Point Precision

Normally PC and emulation environment use Double precision format, while the CodeWarrior Compiler for HCS12 microcontroller uses single precision format

Type	Sign	Exponent	Significand	Total bits	Exponent bias	Bits precision
Half (IEEE 754-2008)	1	5	10	16	15	11
Single	1	8	23	32	127	24
Double	1	11	52	64	1023	53
Quad	1	15	112	128	16383	113

Due to absence of FPU, all Floating point operations and conversions are software executed by the microcontroller.

Floating Point/Fixed Point Conversion

Single Precision Floating Point:

$$Value = (-1)^S * 1.M_2 * 2^{E-127}$$

Fixed Point (32.20):

$$Value = I.F$$

The floating/Fixed point conversions are planned and executed by software routines that are necessarily iterative due to the different possible values of floating and fixed numbers.

Barrel shifters are used to vary the base (mantissa value) and to change the exponent until the number is normalized for conversion

normalized_mantissa = 1.xxxxxx.....x

After each shift a comparison is executed by the ALU microcontroller in order to control if the mantissa (significand) is zero, for normalization.

Floating Point/Fixed Point Operations

Addition/subtraction

Barrel shifters are used to vary the base (mantissa value) and to change the exponent until the number is normalized for conversion before and after addition/subtraction.

After each shift, a comparison is executed by the ALU microcontroller, in order to control if the two exponent of the operands are equal for addition/subtraction.

Multiplication/division

Multiplication and division are similarly complex: exponent are added/subtracted, significands are multiplied/divided, but, after the operation, the resulting number must be normalized!

Example on Freescale Microcontroller:

long int f;	f = px * p2gain;	5,5 µs
float f;	f = (float)px * p2gain;	45 µs

Where P2gain is a 2 byte integer (16 bit)

Example of Circuitry/Operations for Floating Point Addition & subtraction