FLYTTAL – REAL

Flyttal används i datorsystem för s k ”flytande beräkning” vilket innebär att decimalkommat inte har någon fix (fast) position.

Flyttal består av 2 delar (mantissa och exponent).

När ett datorsystem ska hantera och utföra beräkningar av mycket stora tal som inte en dubbeloperand (32-bitars dataregister t ex DINT) kan hantera eller om det är fråga om mycket små tal är flyttal lösningen.

Små tal

Ett dataregister (16- eller 32-bitars) kan inte hantera / presentera decimaler. Eventuella decimaler ur en beräkning försummas eller placeras i följande dataregister men då dock som en rest (inte decimalen).
Minsta tal ett flyttal kan representera **$\pm 1{,}175 \times 10^{-38}$** vilket motsvarar binär exp på -126

Stora tal

Ett 16-bitars dataregister kan hantera tal mellan -32.768 till +32.767 och ett 32-bitars dataregister talen -2.147.483.647 till +2.147.483.646
Största tal ett flyttal kan representera **$\pm 3{,}403 \times 10^{38}$** vilket motsvarar binär exp på +127

Flyttal som lösning

Räcker inte dessa intervall till för beräkningar kan flyttal tas vid, de kan hantera tal med en exponent

-38 till +38 (-126 till +127 binärt)

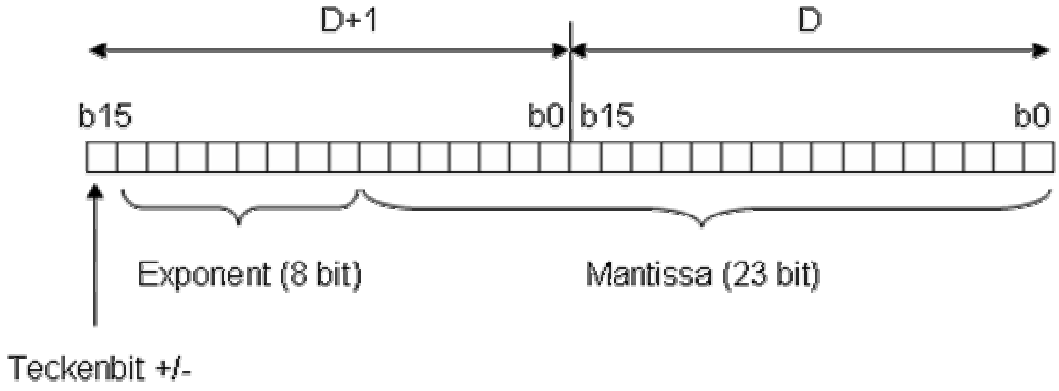
Exempel:
$1{,}8 \times 10^{19}$ 18000000000000000000

$1{,}25 \times 10^{-12}$ 0,00000000000125

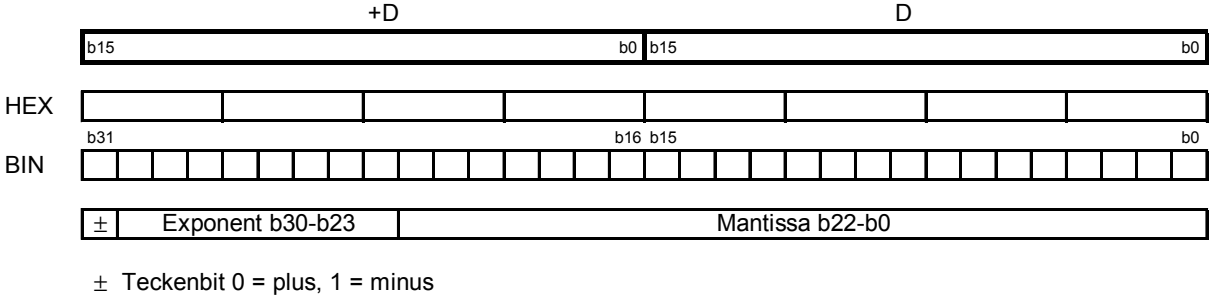
Flyttal kan också hantera ”vanliga” tal som både 16- och 32-bitars dataregister hanterar.

Struktur flyttal

Flyttal består av 2 delar (mantissa och exponent). Mantissan motsvarar decimalen och exponenten är dess multiplikator.



FLYTTAL (REAL)



± Teckenbit 0 = plus, 1 = minus

Ett flyttal upptar 2 dataregister i följd.

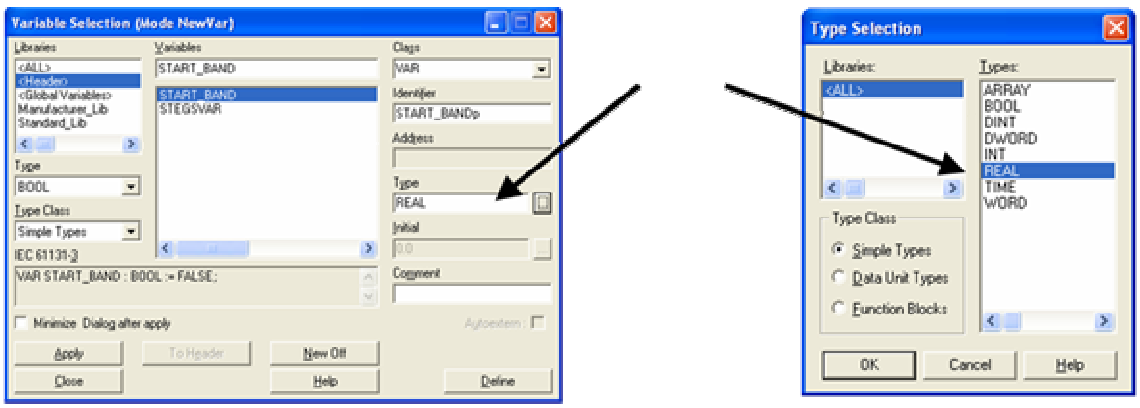
De 2 registerna samverkar så att b0-b15 i det lägre registret och b0-b6 i det högre registret upptar flyttalets mantissa där decimalen finns representerad. Det motsvarar totalt 23 bitar.

b7-b14 i det högre registret upptar flyttalets exponent.

b15 i det högre registret är tecken-bit (+ eller -).

Definiera ett flyttal

När man ska definiera en variabel som flyttal anger man ”type” REAL
De kan definieras lokalt och globalt.



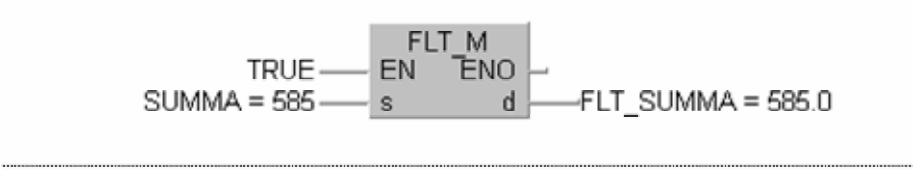
Omvandling av tal

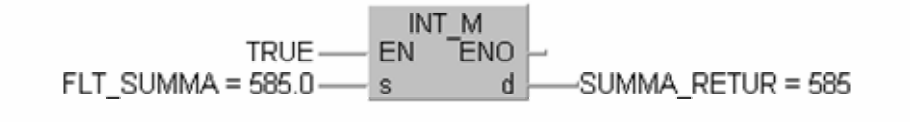
För att kunna hantera tal som flyttal måste de konverteras till flyttal. Instruktionen heter FLT / DFLT beroende på om det är fråga om ett INT eller DINT

När flyttalet ska omvandlas till INT eller DINT heter instruktionen INT eller DINT

INT → REAL
(16-bitars reg → flyttal)

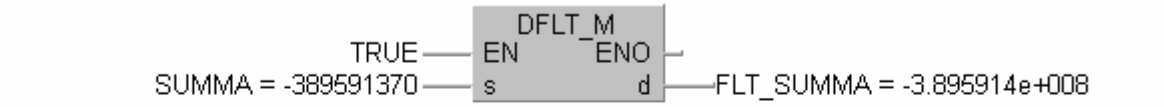
REAL → INT
(flyttal → 16-bitars reg)

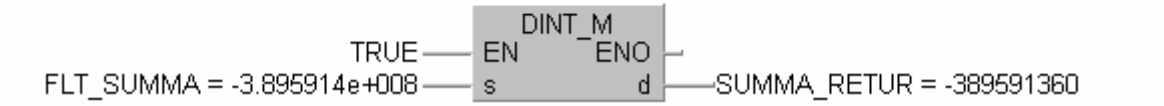




DINT → REAL
(32-bitars reg → flyttal)

REAL → DINT
(flyttal → 32-bitars reg)

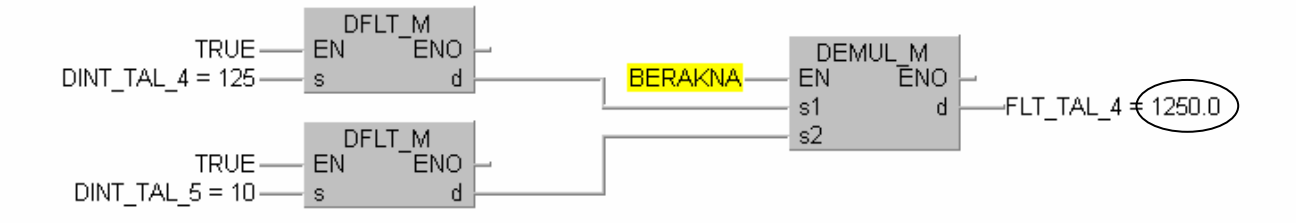




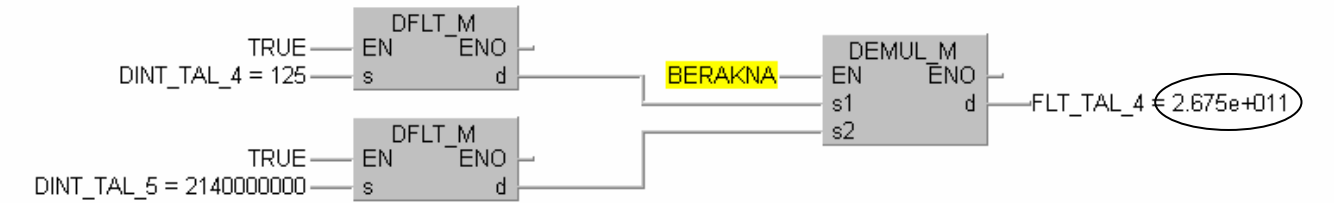
Flyttalets presentation

Beroende på om exponent fordras eller ej ser talet ut olika.

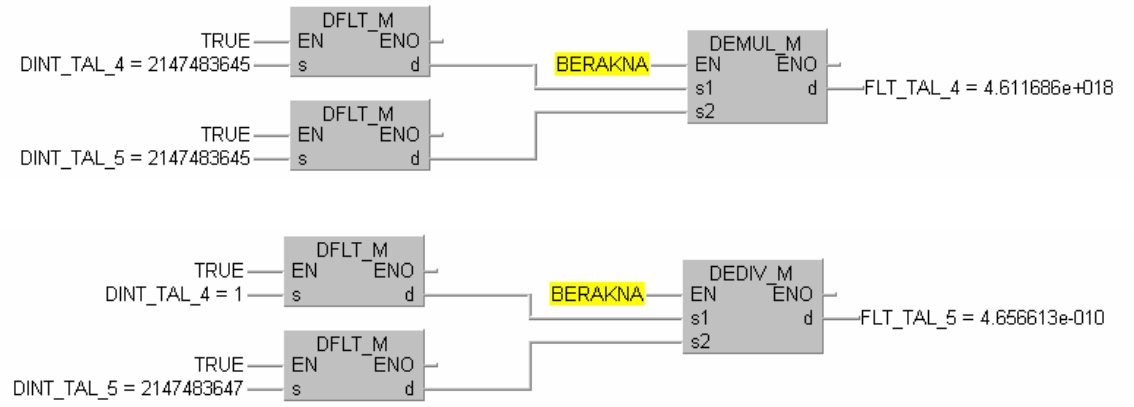
Tal som inte fordrar exponent (1250)



Tal som fordrar exponent ($2{,}675 \times 10^{11} = 267500000000$)

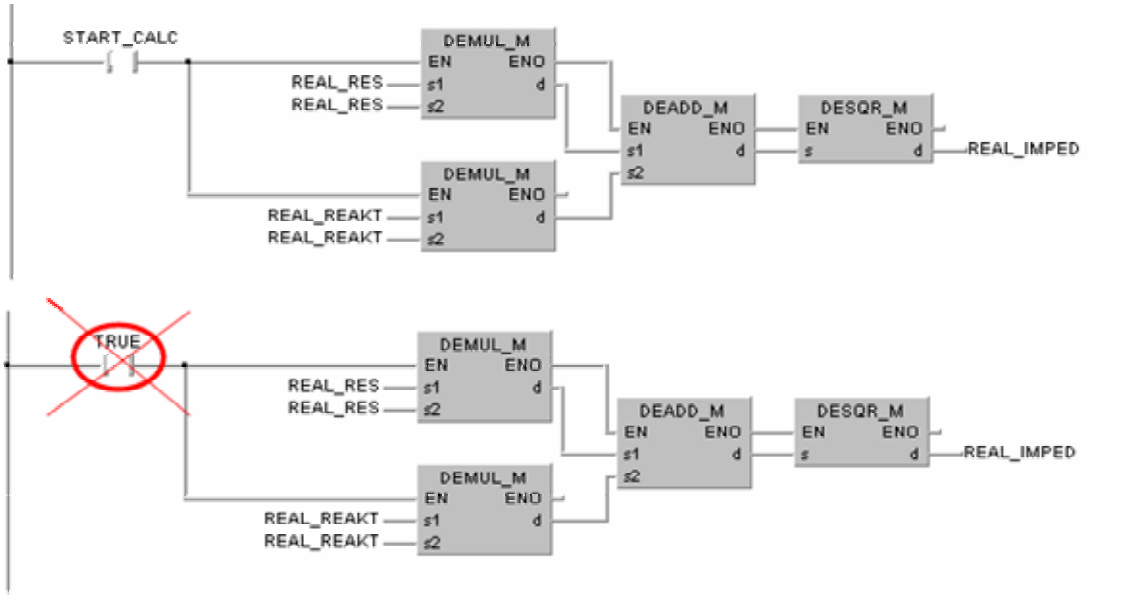


Exempel på mycket stort tal och mycket litet tal



Flyttalsberäkning i GX IEC Developer

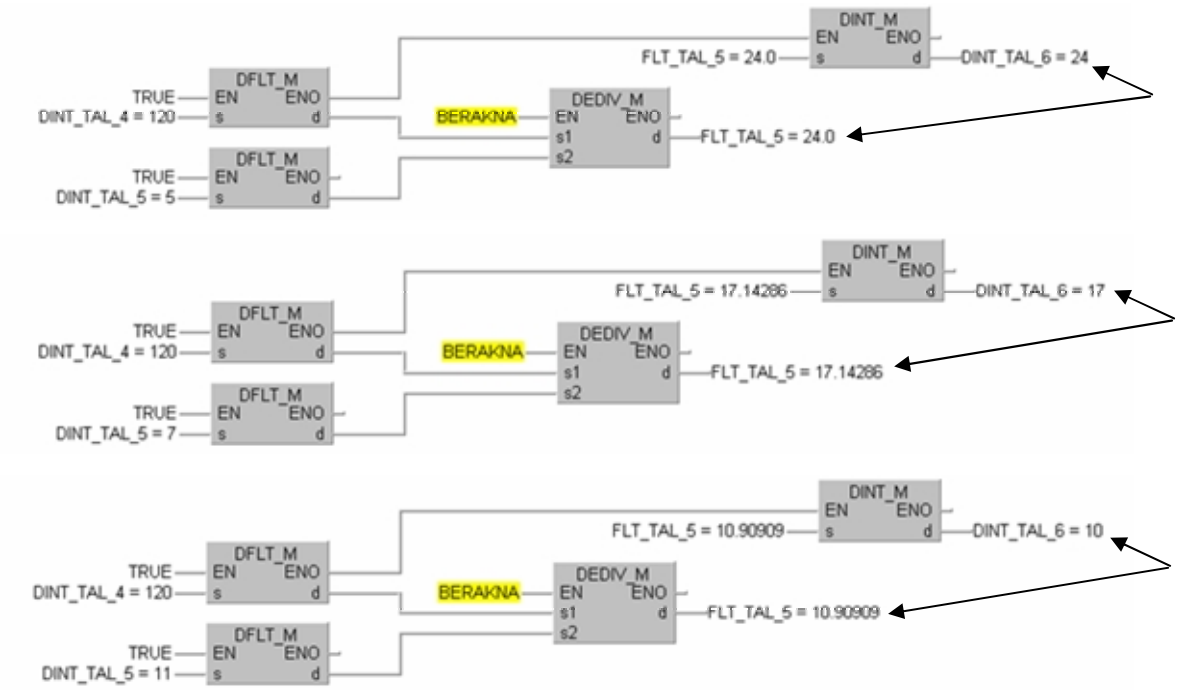
Matematiska beräkningar av flyttal kräver ett ingångsvillkor som kan gå ON/OFF.
En TRUE-funktion accepteras normalt inte av programmet. Däremot kan TRUE användas för omvandling från INT/DINT till FLT och tvärtom.



Avrundning av tal

När ett flyttal omvandlas till INT eller DINT kommer eventuella decimaler att reduceras bort helt. Därför ska man beakta att inte omvandla alltför många gånger, då ”felet” kan växa. Sträva efter att flyttalet omvandlas först när hela den matematiska beräkningen är klar innan omvandling till INT/DINT sker, då blir felet så litet som möjligt.

Exempel

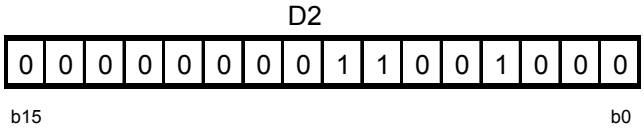


Att beräkna det decimala talet

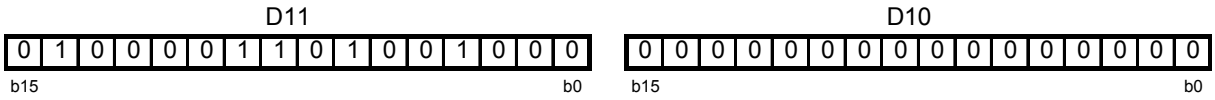
Att beräkna fram det decimala talet sker genom att beräkna exponent och mantissa på följande vis. Vi omvandlar ett 16-bitars tal (INT) VARDE_1 (D2) till flyttal (REAL) DELNING (D10+D11).

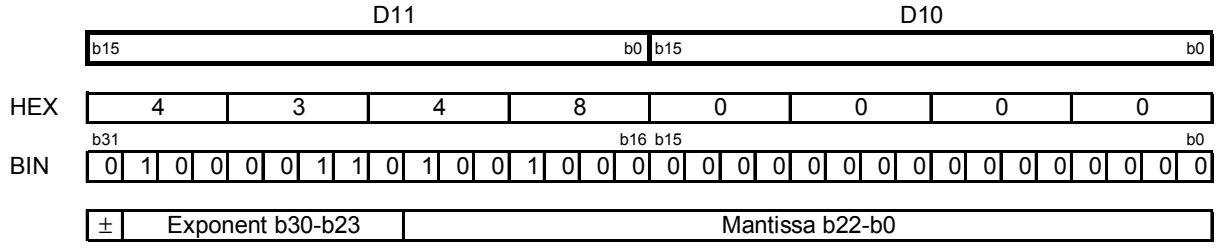
Om vi skickar in det decimala värdet 200 i variabel VARDE_1 presenteras detta i variabel DELNING med 200.0

Så här kommer den binära koden att se ut i D2 för talet 200.



Så här kommer flyttalets register att se ut för talet 200.





Bit 23 utgör vikten 0. Följaktligen får bit 0 vikten -23

Teckenbit

0 = plus (+)

Exponent

$1x2^7 + 0x2^6 + 0x2^5 + 0x2^4 + 0x2^3 + 1x2^2 + 1x2^1 + 0x2^0$

$128 + 0 + 0 + 0 + 0 + 4 + 2 + 0 = 134$

Från exponenten subtraheras **alltid** talet 127

$134 - 127 = 7$

Exponenten är alltså 7

Eftersom varje cell är binär (kan bara anta 2 tillstånd 0 eller 1) blir multiplikatorn **alltid** 2

Vi kan nu skriva 2^7

Mantissa

Mantissan ska alltid adderas med 1 enligt $1x2^0$ (även om b0 är 0). Så här kommer det att se ut.

$1x2^{0} + 1x2^{-1} + 0x2^{-2} + 0x2^{-3} + 1x2^{-4} + 0x2^{-5} + 0x2^{-6} + 0x2^{-7}.....0x2^{-23}$
$1 + 0{,}5 + 0 + 0 + 0{,}0625 + 0 + 0 + 0..... + 0 = 1{,}5625$
Mantissan får nu talet 1,5625

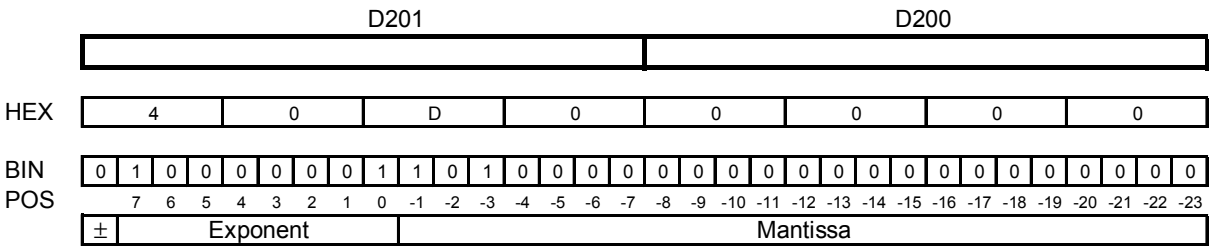
Slutlig beräkning av det decimala talet
Mantissa x Multiplikator^{exp}
$1{,}5625x2^{7} = 200$
Eftersom teckenbiten (b31) är 0 blir talet plus (+).

Ovanstående exempel visar att ett decimalt tal hanteras inte på samma vis som registerna INT och DINT. Därför kan man inte direkt utläsa talets värde i flyttalsregisterna.

Största exponenten för flyttal
Om de 8 bitarna är 1 blir den maximalt +127 (motsvarar 10^{38})

Minsta exponenten för flyttal
Om de bit0 i exponenten är 1 och de andra 0 blir den minimalt -126 (motsvarar 10^{-38})

Exempel 1



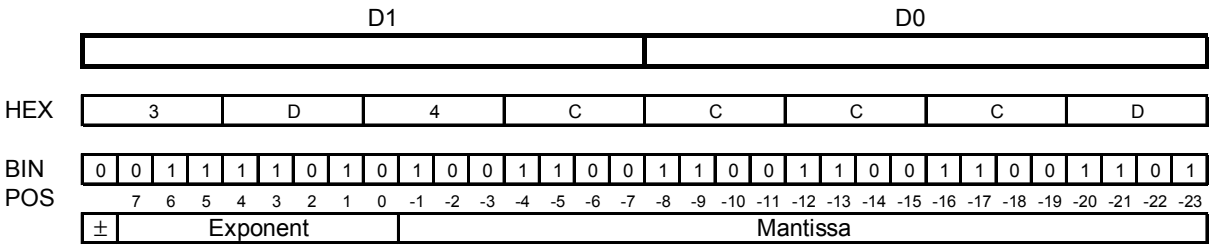
Exponent
$1x2^7+0x2^6+0x2^5+0x2^4+0x2^3+0x2^2+0x2^1+1x2^0$
128+0+0+0+0+0+0+1
129-**127**=2
± =0=plus

Mantissa
$1x2^0+1x2^{-1}+0x2^{-2}+1x2^{-3}+0x2^{-4}+0x2^{-5}+0x2^{-6}+0x2^{-7}+0x2^{-8}+0x2^{-9}+0x2^{-10}+0x2^{-11}+0x2^{-12}+0x2^{-13}+0x2^{-14}+$
$0x2^{-15}+0x2^{-16}+0x2^{-17}+0x2^{-18}+0x2^{-19}+0x2^{-20}+0x2^{-21}+0x2^{-22}+0x2^{-23}$
1+0,5+0+0,125+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
1,625

RES + mantissa x exp
+ 1,625 x 2^2 = 6,5 (decimalt) Exponentens bas är alltid 2 eftersom det är fråga om binära tal

Ovanstående flyttal motsvarar alltså det decimala talet 6,5

Exempel 2



Exponent
$0x2^7+1x2^6+1x2^5+1x2^4+1x2^3+0x2^2+1x2^1+0x2^0$
0+64+32+16+8+0+2+0
122-**127**=-5
± =0=plus

Mantissa
0..-7 $0x2^0+1x2^{-1}+0x2^{-2}+0x2^{-3}+1x2^{-4}+1x2^{-5}+0x2^{-6}+0x2^{-7}+1x2^{-8}+1x2^{-9}+0x2^{-10}+0x2^{-11}+1x2^{-12}+1x2^{-13}+0x2^{-14}+0x2^{-15}+1x2^{-16}+1x2^{-17}+0x2^{-18}+0x2^{-19}+1x2^{-20}+1x2^{-21}+0x2^{-22}+1x2^{-23}$
-8..-20 $0+0,5+0+0+0,0625+0,03125+0+0+3,90625x10^{-3}+1,953125x10^{-3}+0+0+2,44140625x10^{-4}+1,220703125x10^{-4}+0+0+1,525878906x10^{-5}+7,629394531x10^{-6}+0+0+9,536743164x10^{-7}+$
-21..-23 $4,768371582x10^{-7}+0+1,192092896x10^{-7}$

RES + mantissa x exp
+ 1,6 x 2-5^{-5} = 0,05 (decimalt) Exponentens bas är alltid 2 eftersom det är fråga om binära tal

Ovanstående flyttal motsvarar alltså det decimala talet 0,05

D15+D16	FLT_TAL_3	0.05		
D15	D15	-13107	11001100 11001101	CCCD
D16	D16	15692	00111101 01001100	3D4C

Nyttan med flyttal

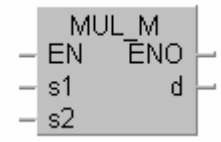
Förhoppningsvis ska man nu kunna se nyttan med att använda flyttal. De kan som sagt hantera mycket stora tal och mycket små tal med decimaler.

Om man är behöver utföra omfattande matematiska beräkningar i ett program kan övergång till flyttal göra att dels ev. decimaler inte försummas och själva valet av funktioner underlättas avsevärt.

Eftersom varje matematisk funktion endast kan ta emot och returnera variabeltyper av viss typ måste dessa variabler vara rätt.

Exempel

För en MUL_M funktion måste de båda ingångarna (s1 och s2) vara av typen 16-bitars upplösning. Utgången (d) måste vara av typen 32-bitars.



Här följer ett litet exempel

Låt oss säga att man behöver följande lilla matematiska beräkning i en programdel.
Vi förutsätter att typerna för variablerna TAL_1 …..TAL_5 överensstämmer med den typ som varje funktion kräver.

Redan efter DDIV_M funktionen blir det stopp. Resultatet från den funktionen skall vara en ARRAY om 2st 32-bitars register. Detta värde leds in i den sista funktionen (MUL_M) vilken skall ha typen 16-bitars upplösnig på sina ingångar (s1 och s2). Detta kommer alltså inte att fungera.

Alternativet blir att ”stympa” sönder ARRAY till ett 16-bitars men då kan information gå förlorad. Det bästa alternativet är att göra om talen till flyttal och tillämpa flyttalsfunktioner för den matematiska beräkningen och när den är klar omvandla flyttalet (svaret) till antingen 16- eller 32-bitars. Det man också vinner på detta är att både stora och små tal hanteras vilket medför att svaret avrundas så snävt det går i systemet.

Så skulle funktionen då se ut med flyttalsteknik. Vi måste även se till att variablerna TAL_1…..TAL_5 är deklarerade som flyttal (REAL).

OBS! Samtliga funktioner kräver variabeltypen REAL på alla in- och utgångar.

Exempel på flyttal

Editorn -Strukturerad Text [ST]

```
("Beräknar impedansen för en induktiv last")
DESQR_M (START_CALC, (REAL_RES*REAL_RES+REAL_REAKT*REAL_REAKT),REAL_IMPED);
FLT_M (TRUE, RESISTANS, REAL_RES);
DFLT_M (TRUE, REAKTANS, REAL_REAKT);
DINT_M (TRUE, REAL_IMPED, IMPEDANS);
```

```
REAL_RES = 112.0; REAL_RES = 112.0; REAL_REAKT = 150.0; REAL_REAKT = 150.0; REAL_IMPED = 187.2004
RESISTANS = 112; REAL_RES = 112.0
REAKTANS = 150; REAL_REAKT = 150.0
REAL_IMPED = 187.2004; IMPEDANS = 187
```

Monitoreringsläget

Motsvarande program i Editorn Ladder [LD]