The "Euro" Problem, or Treating Currency as an Internal Data Type

by Winfried Gerum
Translated by Marcus Werners

We all know the look of our banknotes, "Deutsche Mark" is printed on some European ones. But there is more to that than meets the eye: Since May 2, 1998, there is still the familiar print of "Deutsche Mark" or "Lira Italiana," but their meaning is already Euro. The currency exchange rates between the countries that take part in the European monetary union have been determined at this date and will not be changed anymore. According to the eternal wisdom of our politicians, we will not know until December 31st of this year what the exchange rate of our national currency to the Euro will be.

Software and service companies have to plan for the Euro now: Starting January 2nd of 1999—less than 160 days from now—stock rates will be noted in Euro. Many companies will quote their prices in Euro and print invoices accordingly. From an account in your local currency you could choose to transfer money in Euro or the local currency.

There is a lot of work to do and waiting is surely the wrong strategy.

• You have to select a code for the Euro-sign

• You need to find a screen representation

• All your different printers need to be updated to print the Euro-sign, a common work-around being to simply print "EUR"

• All Display formats of currencies on screens or in print-outs that never needed to worry about currency symbols, because there was only one currency, need to be updated

• Commercial off-the-shelf packages ("COTS") may have different conventions, so there will be interface issues

But those "external" aspects will not be discussed here.

In addition to issues of external presentation, there is the question of internal representation. Storing the currency values will not be sufficient any more. You have to store the currency along with the value. As long as you are dealing with only one currency you will want to continue with your normal calculations. If there are different (Euro) currencies involved , you have additional conversions.

Naturally those conversions could be included in your application programs, but this will involve a multitude of changes. Some may have already started along this route.

There is, however, a more elegant method. One of the advantages of M is what is called "late binding" in computer science. What it means is to determine details late in the process of running an application and achieving a lot of flexibility this way. The basic concept of late binding could be extended.

M knows about strings, that could—as a special case—designate a numeric value. Until now a numeric value consists of a number only, without information about dimension, scale or unit.

The usage of "pure" numbers is the norm in data processing, but the circumstances under which it is appropriate to use these are rare. The real world uses pure numbers, mostly integers to describe quantities. More often numbers designate a value within a dimension like length, speed, weight, resistance or currency. Furthermore, values within a dimension could be expressed using different units of measurements. Volume can be expressed as cubic feet, liters or gallons, length in centimeters, inches, miles or light-years and currency in USD, DEM, ATS or EUR.

M could easily be extended to process numbers with scale and unit in addition to pure numbers. Wherever the M standard talks about "numexpr" the possibility exists to use a number and a scale and a unit ("valueexpr"). Such usage will have ramifications for processing "values" with operators and functions.

An overview of currently used units shows that every unit implies a dimension without ambiguity.

If you add (or subtract) values that share the same unit, the result will have the same unit. If you add values of the same dimension but different units, the unit of the second value will be automatically converted to the unit of the first value. It is erroneous to add values of different dimensions. The only exception: If one value is 0 without a unit the result of the operation will have the unit of the other value.

Examples for calculations with units:

```
> W "1kg"+"500g"
1.5kg
> W "1cm"+"1inch"
3.54cm
> S X=0
>W X+"1EUR"+"1DEM"
2.54EUR
```

The same applies to Boolean operations:

```
> W "3kg">"500g"
1
> W "1cm">"1inch"
0
W "1.20DEM"<"1EUR"
1
```

Multiplication and division will produce other units, that may be composed:

```
> W "1V"*"1A"
1W
> W "1inch"/"1cm"
2.54
```

The "Plus" operation result gives value and unit:

```
> W +"2.50cm length"
2.5cm
```

Special cases are the conversion from one unit to another . . .

```
> W "0mm"+"1inch"
25.4mm
```

. . . or a calculation to find the conversion factor:

```
> W "1mile"/"1km"
1.609344
```

Not all dimensions have units of their own. There is no unit specific for speed. We are familiar with miles (or kilometers) per hour. Within a modified M I would write "90(km/h)." Prices would not be expressed in the unit of currency, but in currency and weight, (e.g., "4.95(EUR/kg)").

If you want to calculate the price for one pound (indeed, a unit for weight was "officially" done away with in Germany about a hundred years ago) you would write:

```
>W "4.98(EUR/kg)"*"500g"
2.49EUR
```

Composed units must be expressed within parentheses, if we want to be able to omit the quotes in "canonic" values:

```
> W 4.98(EUR/kg)*500g
2.49EUR
```

Without parentheses the result would be:

```
> W 498EUR/kg*500g
<Undefined variable>kg
```

If you write:

```
> W +2EUR
```

you do not have to worry about backwards compatibility since this would not be possible using traditional M. A different case would be:

```
> W +"2EUR"
```

Here the semantic would not be backwards compatible.

To my knowledge, the number of resulting problems from this issue would be small in real-world applications. On the other hand, errors would be found, that stayed undiscovered up to now.

If your calculation is `"12Volt"*"3Ampere"+"3Ohm,"` right now you will get "39" instead of an error message. Valid expressions would be: `"12V"*"3A"+"3W"` or `"12V"/"3A"+"3Ohm"`.

Our aim is a system that implements this concept across all units.

Right now we implemented an M system that will accept currency as a dimension with the units EUR, DEM, ATS, BFR, ESP, FMK, FRF, IEP, ITL, NLG, and PTE.

Software updates for the upcoming Euro are relatively easy using this implementation: All currency data stored until now get the unit "DEM." New data will have the unit DEM, EUR or a currency designation from one of the other Euro-countries. If the values within an operation share a common unit, the operation will process nearly as before. If there is more than one (Euro) currency involved, the necessary conversions will take place automatically. The result will have the unit of the first value involved in the calculation. If you are calculating in DEM throughout the year 1999 and only in Euro starting in the year 2000, the programs need not be changed. If you compare turnover figures between both years, your results will be correct.

The effort to implement this functionality has been limited. The necessary changes in application programs have been limited to the external issues mentioned above and a one-time change of all existing currency values to the unit DEM. Without further changes we were able to use all existing algorithms.

Statistics will not tell you about a fall in turnover if your figures change from "1200000DEM" to "800000EUR." Using traditional M this would mean a fall of one third. With the concept explained above it would be recognized as the rise it really is.

If I were to try to incorporate this functionality within the M standard using the usual approach, we would have to wait until 2003. Given the backwards incompatibility issue it may take even longer. This would be too late at least for the Euro change. Even the government—not exactly known for speed in embracing change—will start calculating in Euro in January 2002.

If you want this concept, you will have to "convince" your M supplier. You will have to use every tactic in the book in order to have this functionality offered to you for an affordable price. ***M***

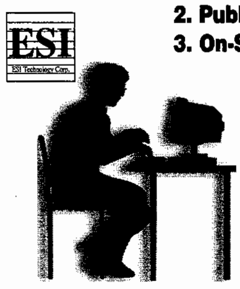
References

Novak, Gordon. "Conversions of Units of Measurement." *IEEE Transactions on Software Engineering*, 21, 8 (1995).

Winfried Gerum is employed with Winner Software GmbH, Röttenbach

ESI
The Leader in M Training

ESI offers 3 options for training:
1. **Computer Based Instruction**
2. **Publicly Scheduled Courses**
3. **On-Site Training**



- M Programming
- File Manager
- Alpha Series
- SQL Series
- Object Technology
- MSM System Management
- Unix and more…

(508) 651-1400 • Fax (508) 651-0708
Internet: sales@esitechnology.com
www.esitechnology.com
ESI Technology Corp.
5 Commonwealth Rd. Natick, MA 01760

CPU

The successful placement of **M/MUMPS** programmers at all levels has been our exclusive business for over 6 years.
We place in both Contract and Permanent positions, Nationwide.

Our service is:

Consistent, Professional, Up–to–date

Please contact **Cécile Marlowe**

(800) 600-3506
Fax: (619) 755-3066
E-Mail: jobs@cpumumps.com

Career Professionals Unlimited
424 Stratford Court, Ste. A32
Del Mar, CA 92014

Browse our Website at:
http://www.cpumumps.com