1. Identification of the proposed change

1.1. Title

FORMAT Library Function

1.2. MDC Proposer and Sponsor

This proposal originates from Richardson Computer Research and is sponsored by Chris Richardson.
Chris Richardson can be reached at:
· P. O. Box 8744, La Jolla, California 92038-8744
· home phone: 619 575 7525
· office phone: 619 535 7284
· telefax: 619 535 7341
· email: chris.richardson@forum.saic.com
· email: chris.richardson@forum.va.gov
Actual sponsorship is from the FORMATting Working Group of the String Processing Task Group of the Data Management Subcommittee.

1.3. Motion

No motion.

1.4. History of MDC actions

Date	Document	Action
Summer 1999	X11/1999-1	Final MDC Type A Write-uo
Sep 1998	X11/SC13/1998-10	Accepted as an MDC Type A Extension (12:1:3).
Jun 1998	X11/SC13/TG2/98-9	FORMAT Library Function, Revised Yet Again was discussed in Task Group 2 and the form of the Library Function was seen as inslight variant to the current library format and the globals mentioned should be in their final ssvn format. This has been applied and will be published again at this meeting as X11/SC13/TG2/98-10.
Sep 1997	X11/SC13/TG2/97-2	FORMAT Library Function, Revised Yet Again was voted as Subcommittee type A (9:4:6).
Mar 1997	X11/SC13/TG2/96-8	The status of FORMAT Library status was confused by the publishing of X11/SC13/TG2/96-2 as X11/SC13/TG2/96-8 (unchanged). There has been only a couple of criticisms of the lack of $FN document numbers which are corrected in this document. The sentiment for "Decimal Point Alternative" produced only a single vote, its author.
Oct 1996	X11/SC13/TG2/96-2	FORMAT Library Function, Revised Yet Again This document is an update of the version which superseded the $FN function extension. Voted as SC Type B.
Mar 1995	X11/SC13/TG2/95-5	FORMAT Library Function, Revised Again Comment was made to 1) modify the example with the MERGE command to eliminate a lot of code, 2) Formalize the language to the Library Format, 3) Some spelling corrections, 4) and the modification of frmask to fmask.
Oct 1995	X11/SC13/TG2/95-5	FORMAT Library Function, Revised, was voted down because the statement of this Library Function was not consistent with the current Library Function Format. There was also a criticism that the function is too complex. The problem is a complex problem which is being addressed in a standard way.
Jun 1995	X11/SC13/TG2/95-4	^$FORMAT ssvn This session demonstrated that this functionality has been presented as a function, ssvn, extrinsic function, as well as a library function. The next time it should be brought forward as a Library Function.
Jun 1995	X11/SC13/TG2/WG4/95-1	Discussion in SC12 Discussion included the need to address 1) out-of-range values and 2) separate the ssvn references into a separate document. The international pressure is encouraging a balance between complexity and the international requirements as well as additional future functionality.
June 1994	X11/SC13/94-32	Discussed in task group, no formal action taken.
April 1994	X11/SC13/TG2/WG4/93-2	Discussed in task group, superseded by new formal proposal

April 1994	X11/SC13/TG2/WG4/93-3	Discussed in task group, superseded by new formal proposal
April 1994	X11/SC13/TG2/WG4/93-4	Discussed in task group, superseded by new formal proposal
April 1994	X11/SC13/TG2/WG4/93-5	Discussed in task group, superseded by new formal proposal
Jan 1994	X11/SC13/TG2/WG4/93-7	FORMAT functionality introduced; no formal action taken
May 1993	X11/SC13/TG2/WG4/93-3	^$FORMAT ssvn - Pre-meeting mailing
May 1993	X11/SC13/TG2/WG4/93-4	FORMAT Library Routine
March 1993	X11/SC13/TG2/WG4/93-3	^$FORMAT ssvn-Pre-meeting mailing
March 1993	X11/SC13/TG2/WG4/93-2	^$FORMAT svn-Post-meeting mailing
Feb 1993	X11/SC13/TG2/WG4/93-1	Working Group formed to investigate the task of FORMATting output.
Feb 1993	X11/SC13/TG2/WG4/93-1	Working Group formed to investigate the task of FORMATting output.
Jan 1993	X11/SC12/93-3	Proposed as SC#12 Type B, remanded to task group for inclusion in a more complete proposal
Nov 1992	X11/SC12/TG2/92-13	Published Post-Meeting
Oct 1992	X11/SC12/TG2/92-12	Task Group Failed
Aug 1992	X11/SC12/TG2/92-12	Proposed as Type B/C
1984?	?	Proposal for $FNUMBER

1.5. Dependencies

None.

2. Justification of Proposed Change

2.1. Needs

In many different countries there are differing methods for inserting separators in display format numerics. Some countries recognize separation every thousand (three numeric spaces), others recognize hundreds (two numeric spaces) as separators. Some countries use commas, others use spaces, single quotes, or other delimiters. Japan uses a different delimiter for each four character separator. Some countries use a single character currency, others use multiple characters (up to 4 characters). Portugal uses its currency symbol as its decimal point. This describes some of the problems encountered when representing specialized output formatting. A very few of these features are available in some other languages (BASIC, FORTRAN, and COBOL). This will make output formatting much easier for programmers.

The working group discussed the needs expressed by the attendees. One member wanted a simpler solution than was being offered by the $%FORMAT^STRING function being proposed. His suggestion was to modify the $FNUMBER function directives to expand its capabilities to specify separators, decimal character, and grouping size in a position dependant fashion. Certain characteristics of the specification should be able to be defaulted, (like grouping size of 3 and the default decimal character of "." and a separator of ","). While the proposed solution seemed to be economic in key-strokes, it was deemed by the other attendees as too ethnocentric.

The idea of each group having ethnocentric defaults which they want to customize for their own use is most attractive. These local defaults can be over-ridden by an application which is packaged and hosted for another machine of unknown ethnic and cultural preferences is most attractive. It is the goal of this proposal to provide a method of defaulting the characteristics of currency, decimal and separator characters, as well as cluster size. These characteristics should be available to interrogate and every characteristic should be overrideable at the process level.

The FORMAT library routine will represent the proposed library function. Minor modification may be made to move these defaults into the process symbol table. Obviously, this will defeat the system-wide capabilities.

Rounding will be applied as it is with the $JUSTIFY function. The next least significant digit than specified by the format directives will result in rounding up if that digit is 5 or larger. No rounding occurs if the next least significant digit is less than 5.

2.2. Existing Practice in Area of the Proposed Change

The current practice is to do specific formatting using the $FNUMBER, $JUSTIFY, $TRANSLATE, and in some cases SET $EXTRACT function. The programmer ends up doing cut-and-try, add-a-space, delete-a-space to align the output for reports. Most languages have a method of alignment. Usually when the value will not fit in the specified field, the field is filled with an error character like asterisk, "*". The FORMAT function also provides a default error character to fill the given undersized field.

3. Description of the proposed change

3.1. General Description of the Proposed Change

```
>SET farg1="CS=""#"":SC="".""":DC="",""":EC=""*"""
>SET farg1=farg1_":FM=""nnsnnnsnnndnn+"""
>WRITE farg1
CS="#":SC=".":DC=",":EC="*":FM="nnsnnnsnnndnn+"
>WRITE $%FORMAT^STRING(-12345.678,farg1) = fstrg
    12.345,68-
```

Where farg1 is;

`CS` - Currency String where fstrg is the default or overridden currency character string.

`DC` - Decimal Character(s) where fstrg is the decimal character is the decimal character recognized by the $%FORMAT^STRING library function format descripter 'd'.

`SC` - Separator Character(s) where fstrg is the set of separator characters, usually only a single character.

The following are meta-symbols for the format mask which will be used in the example;

A - ASCII Only
B - Binary
E - Anything
H - Hex
L - Lower Case Only
O - Octal
P - Punctuation Only
U - Upper Case Only
X - Spacer
c - Currency (Left Justified)
d - Decimal Point (Singular occurrence per mask)
f - Float either type at the end of the f run
l - Left Justified Numeric
m - Money (Right Justified)
n - Numeric
s - Separator
- - Display negative only if negative
+ - Display sign always
(- Begin Negative Number
) - End Negative Number

3.2. Annotated Examples of Use

```
Ex: 1   >SET B="100000.05"   ; Load the test value
        >SET X="FM=Z:CS=""$"":DC="".""":SC="","":FC=""*"""
        >SET Z="cnsnnnsnnndnn"   ; Format specification
        >WRITE ?12,"Total:>"_$%FORMAT^STRING(B,X)   ; Application
                   Total:>$**100,000.05

Ex: 2   >SET Z="+nsnnnsnnndnn",X=X_":FC=""0"""
        >WRITE ?12,"Total:>"_$%FORMAT^STRING(B,X)   ; New Fill Character
                   Total:>+00100,000.05

Ex: 3   >SET Z="ccnsnnnsnnndnn"
        >SET X=X_"FC="" "":CS=""DM"":SC="".""",DC="","""
        >WRITE ?12,"Total:>"_$%FORMAT^STRING(B,X)   ; Multiple Char Currency
                   Total:>DM  100.000,05

Ex: 4   >SET Z="nsnnnsnnn ccc",X=X_":CS=""ATS"""
        >WRITE ?12,"Total:>"_$%FORMAT^STRING(B,X)   ; Trailing Multiple Currency
                   Total:>  100.000 ATS
        >WRITE $P(X,":",1,9)
        FM=Z:CS="$":DC=".":SC=",":FC="*":FC="0":FC=" ":CS="DM":SC="."
        >WRITE $P(X,":",10,99)
        DC=",":CS="ATS"

Ex: 5   >SET X="FM=Z:DC=""$"":SC="" "":FC="" """
        >SET Z="nsnnnsnnndnn"
        >WRITE ?12,"Total:>"_$%FORMAT^STRING(B,X)   ; Portuguese
                   Total:>  100 000$05

Ex: 6   >SET X="FM=Z:DC="".""":SC="","":FC="" """
        >SET Z="cfsfffsffndnn"
        >WRITE ?12,"Total:>"_$%FORMAT^STRING(-B,Z)   ; floating sign
                   Total:>$  -100,000.05

Ex: 7   >SET Y=X_":FL=1"
        >WRITE ?12,"Total:>"_$%FORMAT^STRING(B,Y)   ; floating currency
                   Total:>   $100,000.05

Ex: 8   >WRITE ?12,"Total:>"_$%FORMAT^STRING(B,X_":FC=""$""")   ; fill character
                   Total:>$$$$100,000.05

Ex: 9   >SET Z="(nnsnnnsnnndnn)",Y=Y_":FC="&"
        >WRITE ?12,"Total:>"_$%FORMAT^STRING(-B,Y)_"X"   ; negative braces
                   Total:>(&&&100,000.05)X

Ex:10   >SET Z="(ffsfffsffndnn)"
        >WRITE $%FORMAT^STRING(-B,X)_"X"   ; floating brace
                   Total:>   (100,000.05)X

Ex:11   >SET Z="(nnsnnnsnnndnn),Y=X_":FC=""&"""
        >WRITE ?12,"Total:>"_$%FORMAT^STRING(B,Y)_"X"   ; positive with brace
                   Total:> &&&100,000.05 X
```

```
Ex:12  >SET Z="nnsnnsnnsn",Y=X_":SC=""-"""
       >WRITE $%FORMAT^STRING(1234567,Y)   ; formatting numbers
       12-34-56-7

Ex:13  >SET Z="nns nns",Y=X_":SC="""'"
       >WRITE $%FORMAT^STRING(1210,Y)   ; multiple separators
       12' 10"

Ex:14  >SET Z="nnnnsnnnnsnnnnsnnnndnnnnsnnnnsnnnn"   ; Japanese
       >SET Y=X_":SC=""!@#"":SR=""%^"":DC="","""
       >SET YEN=12345678901234.567890123
       >WRITE ?12,"Total:>"_$%FORMAT^STRING(YEN,Y)
                   Total:>12#3456@7890!1234,5678%9012^3
```

Japanese uses its separators in groupings of 4 digits and the separator character reflects the magnitude of the grouping being separated. The separator character (SC) may be a sequence of characters which are ordered with the first character in the separator specification becomes the first separator to the left of the decimal position. In a similar manor, the separator right (SR), the first character in the specification is the first separator to the right of the decimal position.

```
>WRITE $%FORMAT^STRING(4567,"nnsnnsnnsn")   ; right justification
    4-56-7
>WRITE $%FORMAT^STRING(4567,"llsllsllsl")_"X"   ; left justication
45-67     X
```

3.3. Formalization

(Section numbers refer to X11/TG6/1998-1, X11.1 Draft Standard, Version 13.)
Add the following library function, and renumber the other library functions in clause 7.1.6.6 accordingly:

7.1.6.6.1 $%FORMAT^STRING

FORMAT^STRING : STRING (IN : STRING , FORMAT: fdirectives)

See 2.1 for the definition of graphic and 2.3 for the definition of expr. sepchar is a string which resolves to a single instance of a graphic.

```
fdirectives  ::=  expr V | fspec [ : fspec ] ... |

                | CS=curstr  |    Currency Specifier
                | DC=dechar  |    Decimal Character
                | EC=erchar  |    Error Character
                | FL=filstr  |    Fill String
fspec  ::=      | FM=fmask   |    Format Mask
                | FO=fillon  |    Fill On
                | SC=sepchar |    Separator Character(s) Left
                | SR=sepchar |    Separator Character(s) Right

string  ::=  expr V graphic ...

curstr  ::=  expr V string

dechar  ::=  expr V graphic
```

```
erchar  ::=  expr V graphic

fillon  ::=  expr V tvalue

filstr  ::=  expr V string

                       | c |       Currency (Left Justified)
                       | d |       Decimal (Singular occurrence per mask)
                       | f |       Float either type at the end of the f run
                       | l |       Left Justified Numeric
                       | m |       Money (Right Justified)
fmask  ::=  expr V     | n | ...   Numeric
                       | s |       Separator
                       | x |       Spacer
                       | - |       Display negative only if negative
                       | + |       Display sign always
                       |" "|       Insert space for the spaces in the mask

sepchar  ::=  expr V string
```

returns a formatted string. It will return expr unchanged. fmask is the description of the field to be output. It provides a rich set of alternative with enough undefined richness to provide future extension. Should a sign not be specified (+, (), or -), the absolute value of IN is returned.

The curstr is a string of characters which represents the single local currency designator or the multiple character international reference. This string only occurs once in a fmask specification stream.

The dechar is the definition of the decimal character. It provides a set of rich possibilities. There is the case where the currency in Portugal is used as the decimal indicator. By defining the dechar as the currency, the Portuguese' can be accommodated.

The erchar is the definition of the error condition character. It provides a method of indicating that the specification does not match the format of IN. This will generate an output string of the proscribed length which contains as many copies or parts of copies of the string in erchar as will fit in that length. This is very similar to the function of mismatched data in FORTRAN.

The fillon is a truth value which when true, fill characters are applied to the left of the implied decimal position. The default is considered 0 or False unless overridden.

The filstr is a string of repeating pattern to be used instead of spaces to fill unused numeric columns.

The sepchar is a mechanism for specifying the separator character for the grouping of columns. This should be a single character for each occurrence of the separator fmask sentinel character, s.

In Annex H, add at the appropriate place in alphabetic sequence (and re-number the subsequent functiosn accordingly):

2.35 FORMAT

```
FORMAT(V,L) ; FORMAT Library Function ;06:58 PM  5 Sep 1995; RCR
V      ; Version 0.9 ; CHRIS.RICHARDSON@FORUM.SAIC.COM
       N C,CD,CM,CS,DP,E,EX,FL,FM,FO
       N GL,GV1,GV2,GVL,GVM,GX,I,J,K,ST,TV,TY,V1,V2,VP
       ;
       ; Load up Format Directives from ^$FORMAT or ^SYSTEM("FORMAT")
       D:'$D(^$FORMAT) INFORM
       S (FM,K)="",EX=0,EXS="EXS"
       ;
       ; Extract the working values from the command string
       D PRELOAD
       ;
       ; Process the directives
       D EVALU8
       ;
       ; Error Handling
       D:EX ERROR
       X:$L(EXS) "K "_EXS
       QUIT K
       ;  ======================
       ; CM  - Command Array
       ; CS  - Command String
       ; DP  - Decimal Pointer
       ; EX  - Exit Flag
       ; EXS - KILL Exit String
       ; FL  - Field Length
       ; FM  - Format String
       ; FO  - Format Option Array
       ; K   - Return Output String
       ; L   - List of Directives
       ; ST  - String Extraction String
       ; V   - Input Value
       ; =========================================================
PRELOAD              ; LOAD THE DEFAULTS PRIOR TO THE APPLICATION OF DIRECTIVES
       S K=""
       ; Load System Defaults
       F   S K=$O(^$SYSTEM("FORMAT",K))   Q:K=""   D
       .   S FO(K)=^$SYSTEM("FORMAT",K)
       .QUIT
       ; Load Process Defaults
       F   S K=$O(^$FORMAT(K))            Q:K=""   S FO(K)=^$FORMAT(K)
       S (CS,L)=$G(L)
       ; Load foargument Overrides from the List of Directives
       ;  1) Tokenize the Laterals
       D:L[""""
       . S CS=""
       . F J=2:2:$L(L,"""")  D
       . . S ST=$G(ST)+1,ST(ST)=$P(L,"""",J)
       . . S:ST(ST)="" ST(ST)=""""
       . . S CS=CS_$P(L,"""",J-1)_"%%"_ST_"%%"
       . .QUIT
       . S CS=CS_$P(L,"""",J+1)
       .QUIT
       ;  2) Evaluate the Directives
       N C,L,X
       F J=1:1:$L(CS,":")  D   Q:EX
       . S CD=$P(CS,":",J)
       . S X=$P(CD,"="),TV=$P(CD,"=",2,999)
       . I X=""     S EX=1     Q
       . ;         Uppercase Symbol Names Only
       . S TY=$TR(X,"abcdefghijklm","ABCDEFGHIJKLM")
       . S TY=$TR(TY,"nopqrstuvwxyz","NOPQRSTUVWXYZ")
       . D
```

```
      . . ; To bullet proof, ESTABLISH AN ERROR TRAP TO ERREX
      . . I ($L(TV,"%%")>1)  D LOADTV  S FO(TY)=TV  Q
      . . I TV=""                      S FO(TY)=""  Q
      . . ;   3)   Set the Directive in the FO array
      . . S FO(TY)=@TV
      . .QUIT
      .QUIT
      ;  4) Construct a KILL Exit String for directives not in default list
      N C,E
      S (C,E)=""
      F   S C=$O(FO(C))   Q:C=""      S @C=FO(C),E=E_C_","
      S EXS=E_"EXS"
      S:$D(FM) FL=$L(FM)
      QUIT
      ;  =======================
      ; DC  - Decimal Character
      ; DP  - Decimal Position
      ; EX  - Abnormal Condition Exit
      ; FL  - Format Length
      ; FM  - Format Mask
      ; GV1 - Integer Portion
      ; GV2 - Fractional Portion
      ; K   - Output Buffer
      ; NOD - No Decimal
      ; SV  - Sign Value (1 - Positive, 0 - Negative)
      ; V   - Input Value
EVALU8               ; Evaluate the input for loading into the output string
      N NOD
      S SV=1 ; ,NOD=($P(FM,"d",2)="")
      S NOD='(FM["d")
      S:V<0 SV=0
      S V1=$P(V,"."),V2=$P(V,".",2),(GV1,GV2)=""
      D:FM'=""
      . S FL=$L(FM),DP=$F(FM,"d")-1
      . S:(DP<1) DP=$L(FM)
      .QUIT
      N C
      D GETV1,GETV2:'(NOD!EX)
      Q:EX
      ;
      S K=GV1_DC_GV2
      S:NOD K=GV1
      I $G(FC)'=""   S:K[" " K=$TR(K," ",FC)
      S:$L(K)'=FL EX=1
      QUIT
      ;  =======================
GETV1 ; Get the integer portion of the value and lay it in GV1
      N CP,SP
      I $G(SL)'=""  N SC  S SC=SL
      S GVM=$P(FM,"d"),GVL=$L(GVM),GL=0     ;  1)
      S:$E(V1)="-" V1=$E(V1,2,999)          ;  2)
      S GV1=$J("",GVL),VP=$L(V1),(CP,SP)=1  ;  3)
      ;
      ; Rounding of Integer (NO DECIMAL PORTION)
      S:$P(FM,"d",2)="" V1=$E((V+.5)\1,1,$L(V1))
      F L=GVL:-1:1  S C=$E(GVM,L) D   Q:EX  ;  4)
      . S GX=0
      . D TRANSV1
      . Q:GX!EX
      . ;
      . S:GC'=" " $E(GV1,L)=GC
      .QUIT
      S:VP EX=1
      QUIT
      ;  =======================
      ; GETV1;
      ;  1) Set the Integer Portion of the Mask (GVM) and Length (GVL)
      ;  2) Get the absolute value of V1
      ;  3) Establish Blank Mask, GV1
      ;  4) Extract value for each position in the mask and set it
      ;
GETV2 ; Get the fractional portion of the value and lay it in GV2
      N CP,SP
      I $G(SR)'=""   N SC  S SC=SR
      S GVM=$P(FM,"d",2),GVL=$L(GVM),GL=0,SP=1
      D:GVL<$L(V2)  ; Rounding of Decimal
      .  N J,N
      .  S N=$E($TR($J("",GVL)," ",0)_5_$TR($J("",$L(V2))," ",0),1,$L(V2))
      .  S V2=$E(V2+N,1,$L(V2))
      .QUIT
      S GV2=$J("",GVL),VP=1,CP=1
      F L=1:1:GVL  S C=$E(GVM,L) D   Q:EX
      . S GX=0
      . D TRANSV2
      . Q:GX!EX
      . ;
      . S:GC'=" " $E(GV2,L)=GC
      .QUIT
      QUIT
      ;  =======================
      ; C   - Current Mask Character from the FM
      ; CP  - Character Position
      ; L   - Position within
      ; VP  - Value Position
      ;    (integer - Right to Left, fraction - Left to Right)
TRANSV1          ; Translate the value into the mask
      S (GC,GL)=" "
      Q:"x "[C
      ;
      ; Value Completed, Apply Currency/Float/etc, if requested
      I 'VP       D
      . I "c"[C   D                      Q
      . . S:$G(CP)="" CP=$L(CS)
      . . S GC=$E(CS,CP),CP=CP-1
      . . S:CP<1 CP=$L(CS)
      . .QUIT
      . I GVM["f"               D       Q
      . . N F,I,LI,LX,X,Q
      . . S X=" ",LI=L,LX=0
      . . D    ; Identify the Value Represented
      . . . I GVM["+"!(GVM["-")  D       Q
      . . . . S:GVM["+" X="+"
      . . . . S:V<0 X="-"
      . . . .QUIT
      . . . I GVM["("           D:V<0   Q
      . . . . S X=" ",LX=1
      . . . .QUIT
      . . .QUIT
      . . F I=L:1:GVL  S Q=$E(GV1,I)     Q:Q?1N   Q:("("_DC)[Q   D
      . . . S F=$E(GVM,I),LI=I
      . . . S:"fs("[F $E(GV1,I)=X
      . . .QUIT
      . . S BYE=1
      . . S:LX $E(GV1,LI)="("
      . .QUIT
      .QUIT
```

```
        I C="+"  S GC="+" S:'SV GC="-"  S GL=GC   Q
        I C="-"           S:'SV GC="-"  S GL=GC   Q
        I C="("           S:'SV GC="("  S GL=GC   Q
        I C=")"           S:'SV GC=")"  S GL=GC   Q
        D:VP
        . I C="c"         D                       Q
        . . S:$G(CP)="" CP=$L(CS)
        . . S GC=$E(CS,CP),CP=CP-1
        . . S:CP<1 SP=$L(CS)
        . .QUIT
        . I "fn+-"[C      S GC=$E(V1,VP),VP=VP-1  Q
        . I C="s"         D                       Q
        . . S:$G(SP)="" SP=$L(SC)
        . . S GC=$E(SC,SP),SP=SP-1
        . . S:SP<1 SP=$L(SC)
        . .QUIT
        .QUIT
        QUIT
        ;  ======================
        ; "c"    - Currency
        ; "f"    - Floating
        ; "n"    - Numeric
        ; "s"    - Separator
        ; "+-()" - Sign Representations
        ;  --------------------------
TRANSV2               ; Translate the value into the mask
        S GC=" "
        Q:"x "[C
        ;
        D:VP
        . I "f"[C  D                       Q
        . . S:$G(CP)="" CP=1
        . . S GC=$E(CS,CP),CP=CP+1
        . . S:CP>$L(CS) CP=1
        . .QUIT
        . I C="n"  D                       Q
        . . S GC=$E(V2,VP),VP=VP+1
        . . S:VP>$L(V2) VP=0
        . .QUIT
        . I C="s"  D                       Q
        . . S GC=$E(SC,SP),SP=SP+1
        . . S:SP<$L(SP) SP=1
        . .QUIT
        .QUIT
        I "c"[C    D                       Q
        . S:$G(CP)="" CP=1
        . S GC=$E(CS,CP)
        . S:CP>$L(CS) GC=" "
        . S CP=CP+1
        .QUIT
        I C="+"  S GC="+"  S:'SV GC="-"  S GL=GC  Q
        I C="-"            S:'SV GC="-"  S GL=GC  Q
        I C="("            S:'SV GC="("  S GL=GC  Q
        I C=")"            S:'SV GC=")"  S GL=GC  Q
        QUIT
        ;  ======================
ERREX   ; Error Exit point
        D ERROR
        QUIT K
        ;  ======================
        ; EC  - Error Coded String (1 character or longer)
        ; EL  - Error Code Length
        ; FL  - Field Length
```

```
      ; K   - Output String, The Error Message goes here.
ERROR ; ERROR HANDLING
      N C,E,EL,L
      S:$G(FL)<1 FL=$$FLDLNG(0)
      S E=$G(EC),K="",L=1
      S:E="" E="*"
      S EL=$L(E)
      F I=1:1:FL S C=$E(E,L),L=L+1 S:L>EL L=1 S K=K_C
      QUIT
      ;  =======================
LOADTV               ; DO THE TRANSLATION OF THE TEMP. VALUE WITH THE STRING
      N X
      S X=""
      F M=1:2:$L(TV,"%%")  D
      . S X=X_$P(TV,"%%",M)
      . S N=$P(TV,"%%",M+1)
      . S:N X=X_ST(N)
      .QUIT
      S TV=X
      QUIT
      ;  =======================
FLDLNG(F)            ; FIELD LENGTH Callable from Just About Anywhere
      S F=$G(F)
      QUIT:F F
      ;
      S F=$L($G(FO("FM")))
      I 'F                D
      .  S F=$G(FO("FL"))
      .  I 'F             D
      .  .  S F=$L($G(^$FORMAT("FM")))
      .  .  I 'F          D
      .  .  .  S F=$G(^$FORMAT("FL"))
      .  .  .  I 'F     D
      .  .  .  .  S F=$L($G(^$SYSTEM("FORMAT","FM")))
      .  .  .  .  I 'F  S F=$G(^$SYSTEM("FORMAT","FL"))
      .  .  .  .QUIT
      .  .  .QUIT
      .  .QUIT
      .QUIT
      S:'F F=10
      QUIT F
      ;  =======================
      ; vvvvvvvvvvvvvvvvvvv
      ; Format Default Load
INFORM               ; Load up the defaults
      N K,X
      S K="",X="FORMAT"
      I '$D(^$FORMAT)         D   QUIT:$D(^$SYSTEM(X))
      . I '$D(^$SYSTEM(X))    D   QUIT
      . . S ^$FORMAT("SC")=",",^$FORMAT("DC")="."
      . . S ^$FORMAT("CS")="$",^$FORMAT("EC")="*"
      . .QUIT
      . ; F   S K=$O(^$SYSTEM(X,K)) Q:K=""   S ^$FORMAT(K)=^$SYSTEM(X,K)
      . MERGE ^$FORMAT=^$SYSTEM("FORMAT")
      .QUIT
      ; IF ^SYSTEM DOES NOT EXIST, CREATE IT
      D:'$D(^$SYSTEM(X))
      . ; N K   ; If you don't have the MERGE Command
      . ; S K=""
      . ; F   S K=$O(^$FORMAT(K)) Q:K=""   S ^$SYSTEM(X,K)=^$FORMAT(K)
      . MERGE ^$SYSTEM("FORMAT")=^$FORMAT
      .QUIT
      QUIT
```

```
       ;  ======================
```

The routine contains an initialization segment which will be setup the first time the module is executed and ignored from that point on. The global structures, ^$SYSTEM, and ^$FORMAT may be modified to reflect the cultural bias of the host system. This module is meant to be an initial attempt at the implementation of this function.

4. Implementation impacts

4.1. Impact on Existing User Practices and Investments

Adds newer concise functionality. Simpler syntax than the current MUMPS code required to accomplish this functionality.

4.2. Impact on Existing Vendor Practices and Investments

Minimal impact expected

4.3. Techniques and Costs for Compliance Verification

The following routine is offered as an example of the described functionality and augment the validation.

```
FORMSMPL              ; Sample $$FORMAT Routine Calls ;07:00 PM  5 Sep 1995; RCR
      K  ; Kill off any old values
      W !,"Test # 1 - US Example",!
      D T1
      W !,"Test # 2 - Fill Characters",!
      D T2
      W !,"Test # 3 - Multiple Currency Example",!
      D T3
      W !,"Test # 4 - Trailing Currency String Example",!
      D T4
      W !,"Test # 5 - Portuguese Example",!
      D T5
      W !,"Test # 6 - Floating Sign Example",!
      D T6
      W !,"Test # 7 - Floating Currency Example",!
      D T7
      W !,"Test # 8 - Fill Characters",!
      D T8
      W !,"Test # 9 - Negative Parentheses Example",!
      D T9
      W !,"Test #10 - Floating Parentheses Example",!
      D T10
      W !,"Test #11 - Positive Parentheses Become Spaces Example",!
      D T11
      W !,"Test #12 - Formatting Numbers Example",!
      D T12
      W !,"Test #13 - Multiple (left-hand) Separators Example",!
      D T13
      W !,"Test #14 - Japanesse Example",!
      D T14
      W !,"Test #15 - Post-Signed Example",!
      D T15
      QUIT
      ;  ======================
T1    ;  Test Run #1 - US Standard
      S:'$D(B) B=100000.05
      S X="FM=Z:CS=""$"":DS=""."":SL="","":FL=""*"""
      S Z="cnsnnnsnnndnn"
      W "Total: "_$%FORMAT^STRING(B,X),!?7,Z
```

```
         QUIT
         ;  =======================
T2       ;  Test Run #2 - Leading Sign, Zero Fill Character
         S:'$D(B) B=100000.05
         S X="FM=Z:CS=""$"":DS="".""":SL="","":FL=""0"""
         S Z="-zszzzszzndnn"
         W "Total: "_$%FORMAT^STRING(B,X),!?7,Z
         W "Total: "_$%FORMAT^STRING(-B,X)_"<Negative",!?7,Z
         QUIT
         ;  =======================
T3       ;  Test Run #3 - Float as space
         S:'$D(B) B=100000.05
         S X="FM=Z:CS=""$"":DC="","":SL=""."":FL="" """
         S Z="-nsnnnsnnndnn"
         W "Total: "_$%FORMAT^STRING(B,X),!?7,Z,!
         W "Total: "_$%FORMAT^STRING(-B,X)_"<Negative",!?7,Z
         QUIT
         ;  =======================
T4       ;  Test Run #4 - Trailing currency symbol, German Deutsch Marks
         S:'$D(B) B=100000.05
         S (DM,X)="FM=Z:CS=""DM"":DC="","":SL=""."":FL="" """
         S Z="-nsnnnsnnndnn ccc"
         W "Total: "_$%FORMAT^STRING(B,DM),!?7,Z,!
         W "Total: "_$%FORMAT^STRING(-B,DM)_"<Negative",!?7,Z,!
         S (AT,X)="FM=Z:CS=""ATS"":DC="","":SL=""."":FL="" """
         W "Total: "_$%FORMAT^STRING(B,AT),!?7,Z,!
         W "Total: "_$%FORMAT^STRING(-B,AT)_"<Negative",!?7,Z
         QUIT
         ;  =======================
T5       ;  Test Run #5 - Currency Symbol as Decimal - Portuguese
         S:'$D(B) B=100000.05
         S PR="FM=Z:DC=""$"":SL="" "":FL="" """
         S Z="-nsnnsnnsnndnn"
         W "Total: "_$%FORMAT^STRING(B,PR),!?7,Z,!
         W "Total: "_$%FORMAT^STRING(-B,PR)_"<Negative",!?7,Z
         QUIT
         ;  =======================
T6       ;  Test Run #6 - Floating Sign
         S:'$D(B) B=100000.05
         S PR="FM=Z:CS=""$"":DC="".""":SL="","":FL="" """
         S Z="+fsfffsffndnn"
         W "Total: "_$%FORMAT^STRING(B,PR),!?7,Z,!
         W "Total: "_$%FORMAT^STRING(B,PR)_"<Negative",!?7,Z
         QUIT
         ;  =======================
T7       ;  Test Run #7 - Floating Currency
         S:'$D(B) B=100000.05
         S X="FM=Z:CS=""$"":DC="".""":SL="","":FL="" """
         S Z="cfsfffsffndnn+"
         W "Total: "_$%FORMAT^STRING(B,X),!?7,Z,!
         W "Total: "_$%FORMAT^STRING(B,X)_"<Negative",!?7,Z
         QUIT
         ;  =======================
T8       ;  Test Run #8 - Fill Character='$'
         S:'$D(B) B=100000.05
         S X="FM=Z:DC="".""":SL="","":FL=""$"""
         S Z="cfsfffsffndnn"
         W "Total: "_$%FORMAT^STRING(B,X),!?7,Z,!
         W "Total: "_$%FORMAT^STRING(B,X)_"<Negative",!?7,Z
         QUIT
         ;  =======================
T9       ;  Test Run #9 - Negative Parenthesis & Fill Character='&'
         S:'$D(B) B=100000.05
```

```
        S X="FM=Z:DC="".""":SL="","":FL=""&"""
        S Z="c(nsnnnsnnndnn)"
        W "Total: "_$%FORMAT^STRING(B,X),!?7,Z,!
        W "Total: "_$%FORMAT^STRING(-B,X)_"<Negative",!?7,Z
        QUIT
        ;  =======================
T10     ;  Test Run #10 -
        S:'$D(B) B=100000.05
        S X="FM=Z:DC="".""":SL="","":FL=""&"""
        S Z="c(fsfffsffndnn)"
        W "Total: "_$%FORMAT^STRING(B,X),!?7,Z,!
        W "Total: "_$%FORMAT^STRING(-B,X)_"<Negative",!?7,Z
        QUIT
        ;  =======================
T11     ;  Test Run #11 -
        S:'$D(B) B=100000.05
        S X="FM=Z:DC="".""":SL="","":FL=""&"""
        S Z="c(fsfffsffndnn)"
        W "Total: "_$%FORMAT^STRING(B,X),!?7,Z,!
        W "Total: "_$%FORMAT^STRING(-B,X)_"<Negative",!?7,Z
        QUIT
        ;  =======================
T12     ;  Test Run #12
        S B=123456789
        S X="FM=Z:DC="".""":SC=""-"""
        S Z="nnnsnnsnnnn"
        W "SS: "_$%FORMAT^STRING(B,X),!?4,Z,!
        S Z="nnsnnsnnsnnsn"
        W "SS: "_$%FORMAT^STRING(B,X_":SC=""\"""),!?4,Z
        K B
        QUIT
        ;  =======================
T13     ;  Test Run #13 - Multiple left hand separators
        S B=1234
        S X="FM=Z:DC="".""":SL=""""""'"""
        S Z="nnns nns"
        W "Coordinates: "_$%FORMAT^STRING(B,X),!?13,Z,!
        S B=12345
        W "Coordinates: "_$%FORMAT^STRING(B,X),!?13,Z
        K B
        QUIT
        ;  =======================
T14     ;  Test Run #14
        S:'$D(B) B=12345000.03456789
        S X="FM=Z:CS=""YEN"":DC="".""":SL=""!@#"":SR=""%^"":FL="" """
        S Z="+fsfffsfffsffndnnnsnnnsn ccc"
        W "Total: "_$%FORMAT^STRING(B,X),!?7,Z,!
        W "Total: "_$%FORMAT^STRING(-B,X)_"<Negative",!?7,Z
        K B
        QUIT
        ;  =======================
T15     ;  Test Run #15 -  Post-signed Numeric
        S:'$D(B) B=100000.05
        S X="FM=Z:DC="".""":SL="","":FL=""&"""
        S Z="cfffsfffsffndnn-"
        W "Total: "_$%FORMAT^STRING(B,X),!?7,Z,!
        W "Total: "_$%FORMAT^STRING(-B,X)_"<Negative",!?7,Z
        QUIT
        ;  =======================
```

4.4. Legal considerations

None expected.

5. Closely related standards activities

5.1. Other X11 Proposals (Type A or Type B) Under Consideration
The $FN Extension proposal (X11/SC13/TG2/WG4/93-6) has been superceded by this document.

5.2. Other Related Standards Efforts
ISO/IEC SC22/WG20.

5.3. Recommendations for Co-ordinating Liaison
None.

6. List of Associated Documents

None.

7. Issues, Pros and Cons, and Discussion

In the past, this proposal has been offered as an svn, an ssvn, an intrinsic function, and library function, an extrinsic function, and now, a library function. This function can probably be implemented more efficiently at some location where the default monetary system is known. The ^$FORMAT global is a mechanism for the setting of the default conversion issues. The target symbols in the array FO are the local and run-time copies of the default customized format information for a given site. These can always be overridden by the directives presented in the second argument on the call. The same directive can occur multiple times in that argument. Only the last directive of a specific type will be used.

Take a look at the examples of how these features can be applied to get a wide set of expression of number representation.

7.1. 4 June 1995, Chicago, Illinois
The proposal failed elevation to Subcommittee Type B status (8:10:5).
PRO:
1) Stops dithering
2) Allows for Localization
3) Addresses an issue reffered by ISO
4) Complete/good functionality

CON:
1) Poor Cost/Benefit ratio
2) Should be Library Function
3) Too Complex
4) Too Rarely used
5) Not extensible to date & time
6) Effect on rounding is unclear
7) Error handling for codes outside of specified ranges unclear
8) FMASK unreferenced
9) Default should be explicitly and formally specified
10) We're on record that it could be better done with code

1) In that none have tried to use this rich tool set, this is an assumption. This attempt provides for cultural default impacts, unaddressed by many other proposals. 2) It is, again. 3) All solutions run the risk of being too limited or too complex to solve a given problem. The alternative solved only a single aspect (comma

and decimal swapping). Another issue would have required additional functions. This solves all of the cultural issues I could identify. 4) Too rarely used?, maybe because it hasn't been available? It was asked for by ISO. 5) I'm not sure this is true. If an example can be provided, I believe this function can be extended to accommdate. 6) This has been addressed in the new proposal. 7) It behaves like most other FORMAT functionality found in FORTRAN and other compiled languages. 8) FMASK addressed in this document. 9) It can be within the application or overridden by the system cultural default, the best of both worlds. 10) It could be done better perhaps, but hasn't. The goal of this tool is a lowest common code attempt which will run on older MUMPS implementations.

7.2. 22 March 1996, Boston, Massachusetts

The proposal is elevated to Subcommittee Type B Status (13:0:6)
PRO:
1) Enhances I18N
2) Should be a Library Function

No cons were given at this meeting.

7.3. 26 September 1997, Chicago, Illinois

The proposal is elevated to Subcommittee Type A status (9:4:6)
Pro:
1) Needed Functionality
2) Has been everything

Con:
1) Excessive

The one Con reflects the richness of this proposal to provide functionality for a variety of different application needs. The FORMAT features in other languages is frequently even more expansive than this humble attempt (Octal and Hex representations).

8. Glossary

None.