Organisation des variables 4D dans les Blob

Par
Thierry OZIL, Marketing Technique 4D S.A.
Note technique 4D-200005-14-FR
Version 1

Date 1 Mai 2000

Résumé

Cette note technique décrit le codage des variables 4 D dans les blob, pour permettre une meilleure visibilité des structures et une manipulation des valeurs stockées.

4D Notes techniques

Introduction

Cette note technique se propose d'analyser la structure sous laquelle 4D stocke les variables dans les Blob.

Récapitulatif sur le codage des données.

Type

Le Type de la variable est codé sur un octet tel que décrit dans le tableau suivant.

Valeur	TYPE
0	Est un champ alpha
24	Est une variable chaîne
2	Est un texte
1	Est un numérique
8	Est un entier
9	Est un entier long
4	Est une date
11	Est une heure
6	Est un booléen
3	Est une image
7	Est une sous-table
30	Est un BLOB
5	Est une variable indéfinie
23	Est un pointeur
21	Est un tableau chaîne
18	Est un tableau texte
14	Est un tableau numérique
15	Est un tableau entier
16	Est un tableau entier long
17	Est un tableau date
22	Est un tableau booléen
19	Est un tableau image
20	Est un tableau pointeur
13	Est un tableau 2D

Ecriture des nombres

Un nombre dans 4D peut être écrit en décimal ou en Hexadécimal, dans ce cas, il commence par 0x suivi du nombre Hexadécimal.
Ainsi l'instruction : $MaVar := 100 équivaut à l'instruction : $Mavar := 0x0064

Ecriture des heures

Une heure est stockée sous la forme d'un entier long indiquant le nombre de secondes écoulées depuis †00 ;00 ;00†
Pour faire la conversion d'une heure en entier long, il suffit d'appliquer une opération algébrique.
Exemple :
C_Heure($vhHeure)
$vhHeure :=†01 :02 :03†
`Si l'on veut connaître la valeur décimale
$vlNbSecondes := $vhHeure +0 ` vlNbSecondes = 3723

Byte Ordering

L'ordre des octets n'est pas le même sous Windows et sous MacOS.
L'octet de poids fort est celui de gauche sur un Mac et celui de droite sur un PC

Si on prend l'exemple d'un entier (codé sur 16 bits - 2 octets) dont la valeur =1
Il s'écrira, sans surprise, sur un Mac : 0000 0000 0000 0001 et sur PC : 0001 0000 0000 0000.

Prenons un autre exemple avec un entier long (codé sur 32bits - 4octets), dont la valeur décimale sera 1112299090 (cet exemple pour arbitraire qu'il paraisse, n'est pas dépourvu d'intérêt comme on le verra un peu plus loin)

Sur Mac OS, son codage en binaire sera : 01000010 01001100 01010110 01010010
Ce qui représente une suite de 4 octets qui de gauche à droite correspondent à la valeur décimale :
66, le caractère B de la table ASCII
76, le caractère L de la table ASCII
86, le caractère V de la table ASCII
82, le caractère R de la table ASCII

Sur Windows, ce même entier long s'écrira : 01010010 01010110 01001100 01000010
Ce qui représente une suite de 4 octets qui de gauche à droite correspondent à la valeur décimale :
82, le caractère R de la table ASCII
86, le caractère V de la table ASCII
76, le caractère L de la table ASCII
66, le caractère B de la table ASCII

Dans ce document, tous les exemples utilisent des Blob avec le Byte Ordering du Macintosh

Adressage des octets

Pour un Blob dont la taille est n octets , les octets sont numérotés de 0 à n-1,.
L'offset du Blob qui correspond à l'indice suivant le dernier octet aura la valeur n.
L'adressage adopte la même syntaxe que pour les tableaux, ainsi si l'on veut lire le 5ème octet, on écrira : $vlMonOctet :=MonBlob{6}

Chaîne en pascal

Dans 4D, les variables alpha sont codées en Chaîne Pascal à savoir :
1 octet pour le nombre de caractères, suivi d'autant d'octets qu'il y a de caractères.
Il n'y a pas de marqueur de fin.

Réels

Les numériques sont stockés dans les champs et variables 4e Dimension en tant que nombres à virgule flottante.
Sur les plates-formes Macintosh PPC et Windows, on travaille sur des mots de 64 bits (8 octets), tandis que sur les Macintosh 68K, le chiffre est stocké sur 80 bits (10 octets).

4D a toujours travaillé et enregistré ses réels sur 10 octets.
Pour maintenir la compatibilité des bases avec les versions de 4D pour 68K, 4D continue d'enregistrer les réels sur10 octets. Pour cela, 4D utilise des routines de conversion livrées par Apple, permettant de convertir des nombres à virgule flottante de 8 en 10 octets (et vice-versa).

Le format 64-bits peut contenir 15 chiffres décimaux dont 52-bits significatifs et comme ils sont stockés dans le format 19 chiffres, les 4 derniers chiffres sont essentiellement aléatoires.
Le format à virgule flottante est représenté comme suit :
s = signe (+ ou -) (1 bit)
e = exposant (1bit pour le signe +10 bits pour la valeur)
m = la mantisse, ou partie significative (les chiffres significatifs qui construisent le nombre) (52 bits)
Par exemple, -12 340 000 000 est un chiffre négatif. Si nous le décomposons, il s'écrira comme suit :
- Signe : - (négatif)
- Exposant : +10 (10^10)
- Significatif : 1,234

Padding

Un processeur est plus efficace si les instructions d'adressage du processeur se font sur des adresses paires (multiple de 4 pour Altivec), a cet effet une structure qui occupe un nombre d'octets impairs, se voit ajouter après le dernier octet, un octet supplémentaire non significatif.

Organisation des Blob

Une variable 4D stockée dans un blob sera organisée de la façon suivante :

4 Octets pour Native Variable Vers Blob.
- stockés sous la forme RVLB avec le format Macintosh
- stockés sous la forme BLVR avec le format Windows.

Indique qu'il s'agit d'une variable 4D et sous quel Byte Ordering cette variable est stockée.
1 Octet pour indiquer le type de la variable en fonction du tableau ci-dessus :

S'il s'agit d'une variable du type :

Entier : Il n'y a pas de variable de type entier dans 4D, La directive C_entier () existe bien, mais l'entier ainsi déclaré sera codé sur 4 octets comme un entier long.

Entier Long : 4 octets définissant la valeur de la variable
1 octet pour le padding

Ex : Entier long = 255

Réel : 10 octets définissant la valeur de la variable.
1 octet pour le padding

Ex : Réel = 100,25

Heure : 4 octets définissant le nombre de secondes
1 octet pour le padding

Ex : Heure = †01 :02 ;03† (†01 :02 ;03†=3723 secondes)

Date : 3*2 octets définissant le jour, le mois, l'année
1 octet pour le padding

Ex : Date = !01/05/2000!

Alpha : 2 octets pour la longueur de la chaîne tel que dans la déclaration de typage, suivi d'une chaîne Pascal, c'est-à-dire :

1 octet pour le nombre de caractère
suivi d'un octet pour chaque caractère de la chaîne
Suivi, éventuellement, d'un octet pour le padding

Ex : Chaîne Alpha 20 = "4e Dimension"

Chaîne Alpha 15 = "4eDimension"

Texte : 2 octets pour la longueur de la chaîne
Suivis d'un octet pour chaque caractère de la chaîne
Suivis éventuellement d'un octet pour le padding

Ex : Texte = "4e Dimension"

Image : 4 octets pour la taille
Suivis des octets constituants l'image
Suivis, éventuellement, d'un octet pour le padding

Un tableau 4D stocké dans un blob sera organisé de la façon suivante :

4 Octets pour Native Variable Vers Blob.
- stockés sous la forme RVLB avec le format Macintosh
- stockés sous la forme BLVR avec le format Windows.

1 Octet pour indiquer le type de la variable,
4 octets définissant le nombre d'éléments,
4 octets pour désigner l'élément sélectionné.

Suivi en fonction du type :

Entier : (Il y a des tableaux de type entier dans 4D)
n*(2 octets), où n représente le nombre d'éléments
1 octet pour le padding

Entier Long : n*(4 octets), définissant la valeur de l'entier long
1 octet pour le padding

Réel : n*(10 octets) définissant la valeur de la variable
1 octet pour le padding

Heure : n*(4 octets) définissant le nombre de secondes
éventuellement 1 octet pour le padding

Date : n*(6 octets) définissant le jour, le mois, l'année
1 octet pour le padding

Alpha : n*(2 octets +1 octet +1octet pour chaque caractère de la chaîne)
Suivi éventuellement d'un octet pour le padding

Texte : n*(2 octets +1octet pour chaque caractère de la chaîne)
Suivi éventuellement d'un octet pour le padding

Image : n*(4 octets + les octets constituants l'image)
Suivi éventuellement d'un octet pour le padding

Remarques

Un Blob ne peut être affiché à l'écran, mais tout ou partie des octets qui le composent peuvent être lus et convertis pour être affichés sous forme binaire, hexadécimale, décimale ou affichés avec les caractères ASCII correspondants.
Vous pouvez à cette fin, utiliser l' outil WoodBlob livré avec la NT N°4D-200002-06-FR

On peut ajouter plusieurs variables dans un blob,
En appelant plusieurs fois VARIABLE VERS BLOB(MaVar ;MonBlob ;$vlOffset | *) et en utilisant le même blob dans chacune des instructions, les variables seront stockées les unes à la suite des autres si le paramètre * est passé, ou seront stockéés à la valeur de $vlOffset si ce paramètre est passé.

Dans ce cas $vlOffset (il faut que ce soit une variable et non une valeur passée en dur) prendra la valeur du prochain octet libre, c'est-à-dire le premier octet de l'objet rajouté.
On doit mémoriser cette valeur si l'on veut adresser directement un objet dans la liste des objets stockés dans le blob.