Rocket software

Rocket Uniface Library 10.4

Example: Using Members after Member Reassignment

This example demonstrates how you can maintain a reference to original Struct members, even after they have been assigned a new value in the original Struct.

```
function MEMBER_REFERENCES
variables
  struct vStruct1, vStruct2
  string string1
endvariables

  call printHeader("MEMBER_REFERENCES") ; display entry header in the message frame

; Use a struct variable to maintain a reference to a collection of members

  ; Build a Struct in which multiple members have the same name, but different values:
  vStruct1->a = "A1"
  vStruct1->a{2} = "A2"
  vStruct1->a{3} = "A3"

  ; Save a reference to the collection of these members:
  vStruct2 = vStruct1->a     [1]

  ; Update the complete member set:
  vStruct1->a = "A-updated"  [2]
  putmess "Updated vStruct1: "
  putmess vStruct1->$dbgString

  ; vStruct2 still points to the original collection of 3 members:
  putmess "vStruct2 has not been updated:" [3]
  putmess  vStruct2->$dbgstring
  ; result:
  ;          [a] = "A1"
  ;          [a] = "A2"
  ;          [a] = "A3"

; Single member nodes        [4]
  ; Build a Struct with one member:
  vStruct1->a = "A"

  ;Save a reference to that member
  vStruct2 = vStruct1->a

  ; Update the original Struct
  vStruct1->a = "A-updated"

  putmess "vStruct1 has been updated:"
  putmess  vStruct1->a->$dbgstring
  putmess "vStruct2 points to the original member:"
  putmess  vStruct2->$dbgstring

end ;- function MEMBER_REFERENCES
```

1. The expression `vStruct1->a` returns a *collection* of Struct members that have the name `a`.

2. Assigning a new value to the collection of members named a, replaces them with one member.The Struct referenced by `vStruct1` now contains only one member called a:

```
==============================
 MEMBER_REFERENCES
==============================
Updated vStruct1:
[]
 [a] = "A-updated"
```

3. However, the Struct collection referenced by `vStruct2` remains unchanged:

```
[a] = "A1"
[a] = "A2"
[a] = "A3"
```

> **Note:** By using a reference to a collection, you can continue to access the original data. In *Example: Struct Collections*, this technique is used to restore a Struct to its previous state.

4. This technique also works for single members. Assigning a new value to the original Struct member, actually creates a new member that overwrites the existing member. The new member gets the same tags as the original member had. For more information, see Example: Tags Inheritance.

```
vStruct1 has been updated:
[a] = "A-updated"
vStruct2 points to the original member:
[a] = "A"
```