Rocket software

Rocket Uniface Library 10.4

Example: Structs as Parameters

This example consists of four functions that demonstrate how references to a Struct created in one function or operation are passed to another using IN, OUT or INOUT parameters.

The `struct` data type can be used for parameters in ProcScript functions and operations.

When Struct parameters are passed between functions or partner operations, it is not the Struct that is passed, but a reference to it. When they are passed to public operations, they are passed by value. For more information, see Passing Struct Parameters.

1. STRUCT_AS_PARAMS

The STRUCT_AS_PARAMS function sets up the example, performs the calls to the other functions, and examines the result of the calls.

```
function STRUCT_AS_PARAMS
variables
  struct vStructAsIn
  struct vStructAsOut
  struct vStructPitFall
endvariables

; Display entry header in the message frame
  call printHeader("STRUCT_AS_PARAMS")

; Create the Struct
  [1] vStructAsIn = $newstruct

; --- Calling STRUCT_PARAMS_IN_DO ---
; Entry STRUCT_PARAMS_IN_DO manipulates the Struct and returns
; without passing anything back.

; Examine the Struct before calling STRUCT_PARAMS_IN_DO
  putmess "(Empty) Struct before call to entry STRUCT_PARAMS_IN_DO:"
  putmess vStructAsIn->$dbgstring

; Pass the Struct to entry STRUCT_PARAMS_IN_DO
  [2] call STRUCT_PARAMS_IN_DO(vStructAsIn)

; Examine the result
  putmess "vStructAsIn after entry call:"
  [3] putmess vStructAsIn->$dbgstring

; --- Calling STRUCT_PARAMS_OUT_DO ---
; Entry STRUCT_PARAMS_OUT_DO creates a Struct passes it back
; as an OUT parameter

; Display the Struct referenced by vStructAsOut
  putmess "(Uninitialized, unassigned) Struct before entry call:"
  [4] putmess "  vStructAsOut->$dbgstring returns an empty string: %\
   %%(vStructAsOut->$dbgstring)%%%"
```

```
  [5] call STRUCT_PARAMS_OUT_DO(vStructAsOut)

  ; Examine the Struct
  putmess "vStructAsOut after entry call:"
  [6] putmess vStructAsOut->$dbgstring

; --- Calling STRUCT_PARAMS_PITFALL ---
; Problem: Call an entry with struct vStructPitFall IN parameter
;          before creating a Struct

; Solution #1: Explicitly create a Struct here:
; vStructPitFall = $newstruct

 [7] call STRUCT_PARAMS_PITFALL_DO(vStructPitFall)

; Examine the result
    putmess "Struct manipulations done by the called module are not visible"
    putmess "if the Struct is (1)not created in the calling module, or is "
    putmess "(2)not returned by the called module."
    putmess "  vStructPitFall->$dbgstring:  %%(vStructPitFall->$dbgstring)%%%"

end ;- function STRUCT_AS_PARAMS
```

1. The Struct variable `vStructAsIn` is created as an empty Struct.

   ```
   (Empty) Struct before call to function STRUCT_PARAMS_IN_DO:
    [] = ""
   ```

2. In calling the function `STRUCT_PARAMS_IN_DO`, the Struct is passed as a parameter.
3. function `STRUCT_PARAMS_IN_DO` manipulates the Struct and returns without passing anything back. However, the function `STRUCT_AS_PARAMS` references the same Struct and can therefore examine the result.

   ```
   vStructAsIn after function call: [] = ""
    [aValue] = "1111"
   ```

4. The Struct variable `vStructAsOut` is not initialized or assigned, so it returns an empty string, not an empty Struct.
5. In calling the function `STRUCT_PARAMS_OUT_DO`, the (empty) variable `vStructAsOut` is passed as a parameter.
6. The function `STRUCT_PARAMS_OUT_DO` creates a Struct and passes a reference to the Struct it just created in its OUT parameter.

   The variable `vStructAsOut`now contains the Struct created and returned as a parameter by `STRUCT_PARAMS_OUT_DO`:

   ```
   vStructAsOut after function call: []
    [aValue] = "2222"
   ```

7. When passing a Struct parameter, the Struct must be created by the module that passes the Struct. This sample provides two solutions in the comments—Solution #1 is in the calling function, Solution #2 is in the called function.
8. This implicitly creates the Struct if it does not already exist. In calling function `STRUCT_PARAMS_PITFALL_DO`, the Struct variable `vStructPitFall` has not been initialized.

9. As a result, Struct manipulations done by the called function are not visible:

```
Struct manipulations done by the called module are not visible
 if the Struct is (1)not created in the calling module, or is
 (2)not returned by the called module.
 vStructPitFall->$dbgstring:
```

However, if either Solution #1 or #2 has been applied, the result is visible:

```
Struct manipulations done by the called module are not visible
 if the Struct is (1)not created in the calling module, or is
 (2)not returned by the called module.
 vStructPitFall->$dbgstring: []
 [aValue] = "3333"
```

2. STRUCT_PARAMS_IN_DO

The function `STRUCT_PARAMS_IN_DO` takes a Struct as IN parameter and sets its value.

```
function STRUCT_PARAMS_IN_DO
params
 struct pStruct: IN
endparams
 pStruct->aValue = "1111"
end ;- function STRUCT_PARAMS_IN_DO
```

3. STRUCT_PARAMS_OUT_DO

The function `STRUCT_PARAMS_OUT_DO` creates a Struct and passes it back as an OUT parameter

```
function STRUCT_PARAMS_OUT_DO
params
 struct pStruct: OUT
endparams
; Create a Struct with one member.
[1] pStruct = $newstruct
 pStruct->aValue = "2222"
end ;- function STRUCT_PARAMS_OUT_DO
```

1. This instruction is redundant. If it is omitted, the next statement, which assigns a value to the struct variable, implicitly creates the Struct.

4. STRUCT_PARAMS_PITFALL_DO

The function `STRUCT_PARAMS_PITFALL_DO` takes a Struct as IN parameter and sets its value. It demonstrates how to avoid the pitfall of forgetting to create a Struct before passing a reference to that Struct to another function.

If no Struct is provided in the IN parameter, as Struct is implicitly created when a value is assigned to the parameter. Because the calling module (`STRUCTS_AS_PARAMS`) did not create the Struct, it cannot access it after it has been created by `STRUCT_PARAMS_PITFALL_DO` .

Two solutions are provided in the comments of the `STRUCTS_AS_PARAMS` and `STRUCT_PARAMS_PITFALL_DO`:

- Solution #1: Create the Struct in the calling function `STRUCTS_AS_PARAMS`
- Solution #2: Use INOUT parameters in the called function `STRUCT_PARAMS_PITFALL_DO`.

```
function STRUCT_PARAMS_PITFALL_DO
params
 struct pStruct : IN
 ; Solution #2: Use INOUT parameters
 ; struct pStruct : INOUT
endparams
 ; Add a member to the Struct (which is implicitly created if it doesn't exist)
 pStruct->aValue = "3333"

end ;- function STRUCT_PARAMS_PITFALL_DO
```