Subject: Re: Passing Structures with Pointers with Call_External
Posted by Peter Mason on Wed, 11 Aug 2004 23:44:17 GMT
View Forum Message <> Reply to Message

PeterOut wrote:
...
> Just a follow up question I thought of. Is it possible to return a C
> structure with pointers to IDL and have IDL interpret it correctly?

No, I can't see that working. An array in an IDL structure is "right there" in the structure's data block; it does not hang off a memory-pointer. (...Except for a string scalar which might be considered to be an array of sorts but is really a special kind of thing in IDL - it hangs off a descriptor.) If you're thinking "IDL pointer" rather than memory-pointer then again - you can't. For some reason, RSI have not exposed any functions to work with IDL pointers in external code, and digging around in exports.h doesn't reveal anything interesting. (Note I haven't got IDL6.1 yet so I don't know if this is still the case.)

I sense a certain regret in abandoning the structure approach? :-)
I imagine that you have ancillary frame information that you'd like to keep together with the primary data, or something like that?
If you really would like to keep all this info in a single IDL structure array then I think that you could work with it in a C routine if you don't mind getting your hands a bit dirty. What I'm thinking here is accessing the structure data in a raw sense. Your C CALL_EXTERNAL routine gets passed a memory pointer to the structure array's data block and can interpret it any way that it likes. In your particular example, you could just cast it to an int pointer *IP. You'd have to track which frame you were at. Given that the DATA array has to be the same size in each structure element, you could calculate a frame-size as IFS=IP[0]*IP[1] + 2.
To get at the DATA of frame I, use a float pointer *FDATA and set it to FDATA=(float *)(I*IFS + 2).
If you go with this approach you will have to take great care in how you define the structure in IDL. Your example from the other day just had LONGs and FLOATs so all elements were 4 bytes long and the structure had no padding bytes. If the structure got more complicated then things could get a lot messier.

Cheers
Peter Mason