Subject: Re: Common blocks
Posted by chase on Fri, 05 May 1995 07:00:00 GMT
View Forum Message <> Reply to Message

>>>> > "Phil" == Phil <phil@peace.med.ohio-state.edu> writes:
In article <PHIL.95May2201303@peace.med.ohio-state.edu> phil@peace.med.ohio-state.edu (Phil) writes:

Phil> In article <D7z1G2.5JI@midway.uchicago.edu> rivers@cars3.uchicago.edu (Mark Rivers) writes:

Phil> Like trying to compile other functions/pros that use the said variables!

A common block that is declared at the main level is not within the scope of compiled procedures or functions unless it is declared within those procedures or functions. If a common block is declared within a procedure/function then variable name conflict can be avoided simply by giving unique names for each variable in the common block definition for that routine. (A new IDL behavior for common blocks is that when no variable names are given the variables assume the names of those used in the first compiled procedure or function to define the common block).

In regards to one of the earlier posts, once a common block is declared at the main level I do not believe that it can be removed.

I find that common blocks are very handy for implementing a scope for global variables. In contrast, system variables are not limited in their global scope.

I have found many benefits by using a single variable in my common blocks. This variable is an anonymous structure. In the fields of this structure I put all the variables that would normally be placed individually in the common block. This has two advantages:

1) The structure can be redefined. For example, additional fields can be added to the structure. IDL does not let you redefine the number of elements in a common block within the life of the IDL session. To change the common block one has to restart IDL. Redefining the structure does not require restarting IDL. Additionally, I can add additional fields to the structure without affecting the older routines that access (but do not rewrite) previously defined fields in this structure. Sometimes this avoids having to edit alot of procedures.

2) This structure as a global variable helps to avoid namespace conflicts with other variables that might otherwise occur if the

fields had instead been declared as individual variables in the common block. This benefit is similar to C++ class static variables and modules in other languages.

The disadvantage of this method is when storing very large variable arrays in the structure fields. IDL does not have the ability of taking the address of fields or subarrarys. This means that any field or subarray expression must copy the data. This may be undesirable in certain circumstances. Because of this, I was forced in one particular program to use an additional variable in the common block to hold an extremely large array.

In general, when global variables are needed I have found a single structure in a common block useful because of the above benefits.

Chris Chase

--

==========
Bldg 24-E188
The Applied Physics Laboratory
The Johns Hopkins University
Laurel, MD 20723-6099
(301)953-6000 x8529
chris.chase@jhuapl.edu