Common Statement

Description

Global variables, other than pre-defined system variables, are implemented using labeled common blocks. A labeled common block groups a set of global variables using a specified name.

Syntax

Common */NAME/ variable [,variable ...]*

Common *//variable// variable [, variable ...]*

Parameters

The Common statement has the following parameters.

Parameter	Description
/NAME/	A name that can contain any character except for the / (forward slash).
Variable	The variables specified in the variable list retain their values until specifically freed by FreeCommon.

To ensure that the source code that uses labeled common variables stays up to date, put the Common declaration in an insert record. When the labeled common declaration changes, recompile all source that inserts the declaration. When you put a Common statement in Insert, you can use OpenInsight's Impact Analysis feature to determine the affected source code.

The convention for naming labeled common variables is to prefix the name of the common block, or a shortened version of the name, to each variable, with an underscore in between. Also, suffix each variable with an at sign or a percent sign. Also, since adding items to a labeled common block requires recompilation of all code that uses the labeled common, add several extra items labeled "unused" and then replace them with real variable names as additional common variables are needed. For example:

```
common /Report Data/ RD_ReportID@, RD_SelCritera@,| RD_Unused_2@, RD_Unused_1@
```

Notes

Labeled common variables must be defined by a Common statement prior to their use.

Each Common area can contain up to 255 variable identifiers.

Matrices

Matrices must be named and dimensioned by either a labeled Common statement or a Dimension statement, but not both, before a program uses them. The size and number of the dimensions in a matrix will be permanently set the first time the program encounters a labeled Common statement that defines it. Subsequent labeled Common or Dimension statements will not change the dimensions of the matrix.

Examples

```
* Example 1:
Subroutine Test1 (void)
*Set up common
Common /MyGlobals/ MG_message@
*call subroutine that uses common
MG_message@ = "Hello world!"
call Test2()
*destroy common (frees up memory)
freecommon "MyGlobals"
return
* Example 2:
Subroutine Test2 (void)
*Set up common
Common /MyGlobals/ MG_message@
*use common variable
call Msg(@window, MG_message@)
return
```

See also

FreeCommon