IDL 5.3
More Helpful Tips

From the RSI "Most Frequently Asked Questions" Web Page
http://www.rsinc.com/services/topfaq.cfm

Tips & Tricks for Efficient IDL Programming

Article Name: GENLANG82
RSI Products: IDL
OS Platforms: MACINTOSH; UNIX; VMS; WINDOWS
Function Area: GEN. LANG. - Array/Image Proc.; GEN. LANG. - Objects; GEN. LANG. - Operators; GEN. LANG. - Other
Last Updated By: RSI Technical Support
Last Updated: 01/10/2000

TOPIC:
Programmers can significantly decrease execution time and memory usage by following these tips and tricks for efficient programming:

DISCUSSION:
1. IDL supplies a number of built-in functions and procedures to perform common operations. These system-supplied functions have been optimized and are faster than writing the equivalent operations in IDL with loops and subscripting. In particular, array operations that are performed on the entire array are faster than the equivalent operations performed in a loop on single elements of an array. A common operation is to find the sum of the elements in an array or subarray. Using the IDL supplied TOTAL function is 10 times faster than summing the elements with a FOR loop.

2. The order in which an expression is evaluated can have a significant effect on program speed. Consider the following expression:

```
B = A * 16. / MAX(A)
```

This statement multiplies every element in A by 16 and then divides each element by the value of the maximum element of A. The number of operations required is twice the number of elements in A.

A much faster way of computing the same result is:

```
B = 16./MAX(A) * A
```

3. Try to avoid IF statements. When an IF statement appears in the middle of a loop with each element of an array in a conditional, the loop can be eliminated most of the time by using logical array expressions. For instance consider the following statement:

```
FOR I = 0, (N-1) DO IF B[I] GT 0 THEN A[I] = A[I] + B[I]
```

This can be rewritten as

```
A = A + (B GT 0) * B
```

or, even faster as

```
A = A + (B > 0)
```

4. Use of the WHERE function is much faster than using IF statements of loops:

```
FOR I=0, (N-1) DO IF A[I] LE 0 THEN C= -SQRT(-A[I]) $
    ELSE C[I] = SQRT(A[I])
```

can be replaced with

```
negs = WHERE(A LT 0)
C = SQRT(ABS(A))
C[negs] = -C[negs]
```

5. Eliminate invariant expressions from inside loops.

6. Access large arrays by memory order. Arrays in IDL are row-major order, meaning that the linear order of the data elements proceeds from the first element of the first row through the last element of the first row before beginning on the second row. IDL indexes arrays as (column, row). The upper left-hand element of a matrix is considered to be (0,0). This is the format used by FORTRAN, and is traditionally associated with image processing because it keeps all the elements of a single image scan line together. In contrast C and Visual Basic use column-major order and indexes its data as (row,column).

When an array is larger than or close to the working set size (the amount of physical memory available for the process) it should be accessed in memory order, meaning access it by accessing rows first:

```
FOR X = 0, 511 DO FOR Y = 0, 511 DO ARR[X,Y] = ....
```

is very inefficient because to read the first column 250,000 bytes of data must be read into physical memory. This process must be repeated for each column, requiring the entire array to be read and written almost 512 times. By exchanging the two FOR loops the computing time can be reduced by a factor of 50:

```
FOR Y = 0, 511 DO FOR X = 0, 511 DO ARR[X,Y] = ....
```

7. Setting the NOZERO keyword in BYTARR, COMPLEXARR, DCOMPLEXARR, FLTARR, INTARR, LONARR, MAKE_ARRAY, OBJARR, and PTRARR will prevent

the elements in the resulting array from being set to zero. This saves time, but the array elements contain random values.

8. The REPLICATE_INPLACE procedure updates an existing array by replacing all or selected parts of it with a specified value. This procedure is faster and uses less memory than the IDL function REPLICATE or by direct coding.

9. If the NO_COPY keyword is used in PTR_NEW, VERT_T3D, WIDGET_BASE, WIDGET_BUTTON, WIDGET_CONTROL, WIDGET_DRAW, WIDGET_DROPLIST, WIDGET_LABEL, WIDGET_LIST, WIDGET_SLIDER, WIDGET_TABLE, and WIDGET_TEXT, IDL takes the data from the source and attaches it directly to the destination. However, it has the side effect of causing the source variable to become undefined.

10. Use of the TEMPORARY function minimizes memory use when performing operations on large arrays. TEMPORARY returns the value of its argument as a temporary variable and makes the argument undefined. Assume that A is a large array. To add 1 to each element of A type the following:

```
A = A + l
```

This statements creates a new array for the result of the addition and assigns the result to A before freeing the old allocation of A. Therefore, this operation needs 2*sizeof(A) of memory to perform the operation. The statement

```
A = temporary(A) + 1
```

needs no additional space.