Introduction to Dynamic Variables and Fields

This chapter covers the following topics:

- Purpose of Dynamic Variables
- Definition of Dynamic Variables
- Value Space Currently Used for a Dynamic Variable
- Size Limitation Check
- Allocating/Freeing Memory Space for a Dynamic Variable

Purpose of Dynamic Variables

In that the maximum size of large data structures (for example, pictures, sounds, videos) may not exactly be known at application development time, Natural additionally provides for the definition of alphanumeric and binary variables with the attribute `DYNAMIC`. The value space of variables which are defined with this attribute will be extended dynamically at execution time when it becomes necessary (for example, during an assignment operation: `#picture1 := #picture2`). This means that large binary and alphanumeric data structures may be processed in Natural without the need to define a limit at development time. The execution-time allocation of dynamic variables is of course subject to available memory restrictions. If the allocation of dynamic variables results in an insufficent memory condition being returned by the underlying operating system, the `ON ERROR` statement can be used to intercept this error condition; otherwise, an error message will be returned by Natural.

The Natural system variable `*LENGTH` can be used obtain the length (in terms of code units) of the value space which is currently used for a given dynamic variable. For A and B formats, the size of one code unit is 1 byte. For U format, the size of one code unit is 2 bytes (UTF-16). Natural automatically sets `*LENGTH` to the length of the source operand during assignments in which the dynamic variable is involved. `*LENGTH(`*`field`*`)` therefore returns the length (in terms of code units) currently used for a dynamic Natural field or variable.

If the dynamic variable space is no longer needed, the `REDUCE` or `RESIZE` statements can be used to reduce the space used for the dynamic variable to zero (or any other desired size). If the upper limit of memory usage is known for a specific dynamic variable, the `EXPAND` statement can be used to set the space used for the dynamic variable to this specific size.

If a dynamic variable is to be initialized, the `MOVE ALL UNTIL` statement should be used for this purpose.

Definition of Dynamic Variables

Because the actual size of large alphanumeric and binary data structures may not be exactly known at application development time, the definition of *dynamic* variables of format A, B or U can be used to manage these structures. The dynamic allocation and extension (reallocation) of large variables is

transparent to the application programming logic. Dynamic variables are defined without any length. Memory will be allocated either implicitly at execution time, when the dynamic variable is used as a target operand, or explicitly with an EXPAND or RESIZE statement.

Dynamic variables can only be defined in a DEFINE DATA statement using the following syntax:

```
level variable-name ( A ) DYNAMIC
level variable-name ( B ) DYNAMIC
level variable-name ( U ) DYNAMIC
```

Restrictions:

The following restrictions apply to a dynamic variable:

- A redefinition of a dynamic variable is not allowed.
- A dynamic variable may not be contained in a REDEFINE clause.

Value Space Currently Used for a Dynamic Variable

The length (in terms of code units) of the currently used value space of a dynamic variable can be obtained from the system variable *LENGTH. *LENGTH is set to the (used) length of the source operand during assignments automatically.

Warning:
Due to performance considerations, the storage area that is allocated to hold the value of the dynamic variable may be larger than the value of *LENGTH (used size available to the programmer). You should not rely on the storage that is allocated beyond the used length as indicated by *LENGTH: it may be released at any time, even if the respective dynamic variable is not accessed. It is not possible for the Natural programmer to obtain information about the currently allocated size. This is an internal value.

*LENGTH(*field*) returns the used length (in terms of code units) of a dynamic Natural field or variable. For A and B formats, the size of one code unit is 1 byte. For U format, the size of one code unit is 2 bytes (UTF-16). *LENGTH may be used only to get the currently used length for dynamic variables.

Size Limitation Check

Profile Parameter USIZE

For dynamic variables, a size limitation check at compile time is not possible because no length is defined for dynamic variables. The size of user buffer area (USIZE) indicates the size of the user buffer in virtual memory. The user buffer contains all data dynamically allocated by Natural. If a dynamic variable is allocated or extended at execution time and the USIZE limitation is exceeded, an error message will be returned.

Allocating/Freeing Memory Space for a Dynamic Variable

The statements EXPAND, REDUCE and RESIZE are used to explicitly allocate and free memory space for a dynamic variable.

Syntax:

```
EXPAND [SIZE OF] DYNAMIC [VARIABLE] operand1 TO operand2
REDUCE [SIZE OF] DYNAMIC [VARIABLE] operand1 TO operand2
RESIZE [SIZE OF] DYNAMIC [VARIABLE] operand1 TO operand2
```

- where *operand1* is a dynamic variable and *operand2* is a non-negative numeric size value.

EXPAND

Function

The EXPAND statement is used to increase the allocated length of the dynamic variable (*operand1*) to the specified length (*operand2*).

Changing the Specified Size

The length currently used (as indicated by the Natural system variable *LENGTH, see above) for the dynamic variable is not modified.

If the specified length (*operand2*) is less than the allocated length of the dynamic variable, the statement will be ignored.

REDUCE

Function

The REDUCE statement is used to reduce the allocated length of the dynamic variable (*operand1*) to the specified length (*operand2*).

The storage allocated for the dynamic variable (*operand1*) beyond the specified length (*operand2*) may be released at any time, when the statement is executed or at a later time.

Changing the Specified Length

If the length currently used (as indicated by the Natural system variable *LENGTH, see above) for the dynamic variable is greater than the specified length (*operand2*), the system variable *LENGTH of this dynamic variable is set to the specified length. The content of the variable is truncated, but not modified.

If the given length is larger than the currently allocated storage of the dynamic variable, the statement will be ignored.

RESIZE

Function

The RESIZE statement adjusts the currently allocated length of the dynamic variable (*operand1*) to the specified length (*operand2*).

Changing the Specified Length

If the specified length is smaller then the used length (as indicated by the Natural system variable *LENGTH, see above) of the dynamic variable, the used length is reduced accordingly.

If the specified length is larger than the currently allocated length of the dynamic variable, the allocated length of the dynamic variable is increased. The currently used length (as indicated by the system variable *LENGTH) of the dynamic variable is not affected and remains unchanged.

If the specified length is the same as the currently allocated length of the dynamic variable, the execution of the RESIZE statement has no effect.