Subject: .ANDTHEN. and .ORELSE. operators
From: Van Snyder
Reference: 03-258r1, section 2.8.2

Number

TBD

Title

.ANDTHEN. and .ORELSE. operators.

Submitted By

J3

Status

For consideration.

Basic Functionality

Provide logical AND and OR operators that force short-circuit evaluation, rather than merely allowing it.

Rationale

The standard presently allows a processor to short-circuit evaluation of logical expressions. For example, in `A .AND. B`, the processor is allowed not to evaluate `B` if `A` is false. It is sometimes desirable, however, to *require* that the processor not evaluate `B` if `A` is false, as opposed simply to *allowing* it not to. Here's an example:

```
if ( present(x) .and. x /= 0 ) ...
```

One can't *depend* on the processor not trying to evaluate `x /= 0` if `x` is not present.

To support this desire, add an `.ANDTHEN.` operator, the semantics of which require the processor to evaluate the first operand first, and then prohibit it from evaluating the second operand if the first is false. The example becomes:

```
if ( present(x) .andthen. x /= 0 ) ...
```

Similar considerations apply to the .OR. operator, leading to the desire for an .ORELSE. operator, in which the second operand is prohibited to be evaluated if the first is true.

These operators are, of course, even more useful elementally in WHERE statements and constructs. For example

```
where ( x > 0.0 .andthen. log(x) < tol ) ...
```

Estimated Impact

Minor.

Detailed Specification

Provide logical AND and OR operators that force short-circuit evaluation, rather than merely allowing it. One possible spelling is .ANDTHEN. and .ORELSE.

History