

Function Call

```
call-name ( < [ prototype-cast ] [ parameter ] [ , [ parameter ] ] ... > )
```

Related Topics: [DEFINE FUNCTION](#) | [DEFINE PROTOTYPE](#)

General Description

Using the function call, the user can call user defined functions which are defined inside special function objects. There are two different ways to call a function.

Symbolic Function Call

When using the first method, the symbolic function call, the user specifies exactly the name of the function to be executed at run time.

Function Call Using a Variable

When using the second method, the name of the desired function definition is stored inside an alphanumeric variable. At runtime, Natural jumps into the corresponding function definition, the name of which has been stored inside the variable.

In order to identify these two kinds of function calls, a corresponding prototype definition must be specified. Additionally, the prototype may contain the whole signature of the function definition. If no signature has been given, the function call must contain a PT clause in order to specify the missing parts of the signature. Therefore, the VARIABLE keyword of such a prototype specified inside the PT clause has no effect. For variable function calls, there must be a valid prototype with the same name of the alphanumeric variable containing the function name.

If no prototype can be assigned to the function call, a special prototype-cast is necessary in order to define the return format/length at compilation time. The prototype-cast and the parameter list must be enclosed in pointy brackets and parentheses, as displayed in the syntax diagram.

If you want to use the variable method, you must define a prototype with the same name as the variable-name using the keyword VARIABLE.

Example:

```
DEFINE PROTOTYPE VARIABLE variable-name
```