

DEFINE PROTOTYPE

DEFINE PROTOTYPE

$\left\{ \begin{array}{l} \text{prototype-name} \\ \text{[FOR] VARIABLE } \text{prototype-variable-name} \end{array} \right\}$	$\left[\begin{array}{l} \text{UNKNOWN} \\ \text{signature-clause} \\ \text{same-clause} \end{array} \right]$
--	---

END-PROTOTYPE

Function

The prototype definition may be used to specify a signature according to a certain function call. For each function call, the return type must be known, as well as the kind of the function call (VARIABLE). Therefore, this data must be available for each function call. If any of this data is missing, the prototype keyword must be used inside the function call reference. If there is a parameter definition inside the prototype, the parameter values of the function call are compared with the parameters of the prototype definition. If the parameters should not be checked, use the UNKNOWN keyword inside the define data parameter statement of the prototype definition.

prototype-name

The prototype-name must follow the same rules as those used for defining user-defined variables (with one exception: prototype-names may contain periods "."). The prototype-name is completely arbitrary. It is not necessary for it to have the same name as a corresponding function definition. The maximum length of the whole prototype-name is 32 characters.

prototype-variable-name

The prototype-variable-name allows you to call functions using variable function names. This is similar to CALLNAT function calls. The prototype-variable-name is the name of an alphanumeric variable containing the real name of the function, which is to be called in the function reference.

signature-clause

```
[prototype-return-data-definition]

  DEFINE DATA
    {
      PARAMETER UNKNOWN
      {
        PARAMETER { USING parameter-data-area } ...
      }
      END-DEFINE
    }
  END-PROTOTYPE


```

The signature-clause looks like a certain function call. Normally, the prototype agrees with the function definition. But it does not need to be exactly the same. So it is possible to omit the parameter data and to set the keyword UNKNOWN instead. In this case, there will be no parameter checking at compilation time. The type of the return value must be set in every case. If no return value is defined, there is no assignment allowed from the function call to a variable. If no signature is specified inside a prototype definition (signature is UNKNOWN), the corresponding signature of a function call must be specified using the keyword PT.

prototype-return-data-definition

```


  RETURNS [variable-name] { (format-length)
    ( { A } ) DYNAMIC
  }


```

Definition of the type of the return variable. The RETURNS clause is nearly the same as the RETURNS clause of the function definition except that the extension "BY VALUE" is not allowed in the prototype definition. The type of the return value must be known at compilation time.

same-clause

```


  SAME AS [ PROTOTYPE ] { prototype-name
    prototype-variable-name
  }


```

The same clause may be used in order to use signatures previously defined to define a new prototype.

Example

This is a prototype definition of a function named "GET-FIRST-BYTE". Using the following prototype, the function "GET-FIRST-BYTE" can be called as a symbolic function call: GET-FIRST-BYTE (<#A>).

```
/* DPROX1: Example 1 for DEFINE PROTOTYPE
/*
DEFINE DATA LOCAL
1 #A(A10) INIT <'abcdefghijkl'>
END-DEFINE

DEFINE PROTOTYPE GET-FIRST-BYTE
  RETURNS (A1)
  DEFINE DATA PARAMETER
    1 PARM1(A10)
  END-DEFINE
END-PROTOTYPE

WRITE GET-FIRST-BYTE(<#A>)
END
```

The following Natural code contains the prototype definition of a function, in this case "GET-FIRST-BYTE". In order to be able to call the function dynamically, the name of the function must be stored inside the alphanumeric variable #A. The variable #A must be defined as an alphanumeric variable inside the DEFINE DATA statement before it may be used.

```
/* DPROX2: Example 2 for DEFINE PROTOTYPE
/*
DEFINE DATA LOCAL
1 FUNCTION-NAME(A32) INIT<'GET-FIRST-BYTE'>
1 #A(A10) INIT <'abcdefghijkl'>
END-DEFINE

DEFINE PROTOTYPE VARIABLE FUNCTION-NAME
  RETURNS (A1)
  DEFINE DATA PARAMETER
    1 PARM1(A10)
  END-DEFINE
END-PROTOTYPE

WRITE FUNCTION-NAME(<#A>)
END
```