DEFINE SUBROUTINE

```
DEFINE [SUBROUTINE]  subroutine-name
   statement ...
 ⎧ END-SUBROUTINE (structured mode only) ⎫
 ⎩ RETURN  (reporting mode only)         ⎭
```

This chapter covers the following topics:

- Function
- Restrictions
- Syntax Description
- Examples

For an explanation of the symbols used in the syntax diagram, see *Syntax Symbols*.

Related Statements: `CALL` | `CALL FILE` | `CALL LOOP` | `CALLNAT` | `ESCAPE` | `FETCH` | `PERFORM`

Belongs to Function Group: *Invoking Programs and Routines*

Function

The `DEFINE SUBROUTINE` statement is used to define a Natural subroutine. A subroutine is invoked with a `PERFORM` statement.

Inline/External Subroutines

A subroutine may be defined within the object which contains the `PERFORM` statement that invokes the subroutine (inline subroutine); or it may be defined external to the object that contains the `PERFORM` statement (external subroutine). An inline subroutine may be defined before or after the first `PERFORM` statement which references it.

Note:
Although the structuring of a program function into multiple external subroutines is recommended for achieving a clear program structure, please note that a subroutine should always contain a larger function block because the invocation of the external subroutine represents an additional overhead as compared with inline code or subroutines.

Data Available in a Subroutine

Inline Subroutines

No explicit parameters can be passed from the invoking program via the `PERFORM` statement to an internal subroutine.

An inline subroutine has access to the currently established global data area as well as to the local data area used by the invoking program.

External Subroutines

An external subroutine has access to the currently established global data area. In addition, parameters can be passed directly with the `PERFORM` statement from the invoking object to the external subroutine; thus, you may reduce the size of the global data area.

An external subroutine has no access to the local data area defined in the calling program; however, an external subroutine may have its own local data area.

Restrictions

- Any processing loop initiated within a subroutine must be closed before `END-SUBROUTINE` is issued.
- An inline subroutine must not contain another `DEFINE SUBROUTINE` statement (see *Example 1* below).
- An external subroutine (that is, an object of type subroutine) must not contain more than one `DEFINE SUBROUTINE` statement block (see *Example 2* below). However, an external `DEFINE SUBROUTINE` block may contain further inline subroutines (see *Example 1* below).

Example 1

The following construction is possible in an object of type subroutine, but not in any other object (where `SUBR01` would be considered an inline subroutine):

```
...
DEFINE SUBROUTINE SUBR01
   ...
   PERFORM SUBR02
   PERFORM SUBR03
   ...
   DEFINE SUBROUTINE SUBR02
   /* inline subroutine...
   END-SUBROUTINE
 ...
   DEFINE SUBROUTINE SUBR03
   /* inline subroutine...
   END-SUBROUTINE
END-SUBROUTINE
END
```

Example 2 (invalid):

The following construction is *not* allowed in an object of type subroutine:

```
...
DEFINE SUBROUTINE SUBR01
...
END-SUBROUTINE
DEFINE SUBROUTINE SUBR02
...
END-SUBROUTINE
END
```

Syntax Description

Syntax Element	Description
subroutine-name	**Name of Subroutine:** For a subroutine name (maximum 32 characters), the same naming conventions apply as for user-defined variables; see *Naming Conventions for User-Defined Variables* in the *Using Natural* documentation. The subroutine name is independent of the name of the module in which the subroutine is defined (it may but need not be the same).
statement	**Statement(s) to be Executed:** In place of *statement*, you must supply one or several suitable statements, depending on the situation. For an example of a statement, see *Examples* below.

END-SUBROUTINE
RETURN
	End of DEFINE SUBROUTINE Statement:
	
	In structured mode, the subroutine definition is terminated with END-SUBROUTINE.
	
	In reporting mode, RETURN may also be used to terminate a subroutine.

Examples

- Example 1 - Define Subroutine
- Example 2 - Sample Structure for External Subroutine Using GDA Fields

Example 1 - Define Subroutine

```
** Example 'DSREX1S': DEFINE SUBROUTINE (structured mode)
************************************************************************
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 NAME
  2 ADDRESS-LINE  (A20/2)
  2 PHONE
*
1 #ARRAY    (A75/1:4)
```

```
1 REDEFINE #ARRAY
  2 #ALINE  (A25/1:4,1:3)
1 #X        (N2) INIT <1>
1 #Y        (N2) INIT <1>
END-DEFINE
*
FORMAT PS=20
LIMIT 5
FIND EMPLOY-VIEW WITH NAME = 'SMITH'
  MOVE NAME            TO #ALINE (#X,#Y)
  MOVE ADDRESS-LINE(1) TO #ALINE (#X+1,#Y)
  MOVE ADDRESS-LINE(2) TO #ALINE (#X+2,#Y)
  MOVE PHONE           TO #ALINE (#X+3,#Y)
  IF  #Y = 3
    RESET INITIAL #Y
    PERFORM PRINT
  ELSE
    ADD 1 TO #Y
  END-IF
  AT END OF DATA
    PERFORM PRINT
  END-ENDDATA
END-FIND
*
DEFINE SUBROUTINE PRINT
  WRITE NOTITLE (AD=OI) #ARRAY(*)
  RESET #ARRAY(*)
  SKIP 1
END-SUBROUTINE
*
END
```

Output of Program DSREX1S:

```
SMITH                    SMITH                    SMITH
ENGLANDSVEJ 222          3152 SHETLAND ROAD       14100 ESWORTHY RD.
                         MILWAUKEE                MONTERREY
554349                   877-4563                 994-2260

SMITH                    SMITH
5 HAWTHORN               13002 NEW ARDEN COUR
OAK BROOK                SILVER SPRING
150-9351                 639-8963
```

Equivalent reporting-mode example: DSREX1R.

Example 2 - Sample Structure for External Subroutine Using GDA Fields

```
** Example 'DSREX2': DEFINE SUBROUTINE (using GDA fields)
************************************************************************
DEFINE DATA
GLOBAL
  USING DSREX2G
END-DEFINE
*
INPUT 'Enter value in GDA field' GDA-FIELD1
*
*  Call external subroutine in DSREX2S
*
PERFORM DSREX2-SUB
*
END
```

Global Data Area DSREX2G Used by Program DSREX2:

```
1 GDA-FIELD1                       A    2
```

Subroutine DSREX2S Called by Program DSREX2:

```
** Example ’DSREX2S’: SUBROUTINE (external subroutine using global data)
************************************************************************
DEFINE DATA
GLOBAL
  USING DSREX2G
END-DEFINE
*
DEFINE SUBROUTINE DSREX2-SUB
*
  WRITE ’IN SUBROUTINE’ *PROGRAM ’=’ GDA-FIELD1
*
END-SUBROUTINE
*
END
```