UNIVERSITY OF EDINBURGH
Edinburgh Regional Computing Centre

User Note 84

(February 1986)

Title:

EMAS-3: Multi-language calls to IMP procedures

Author: Keith Yarwood

Contact: Advisory service

Software Support Category: n/a

Synopsis

This Note describes Subsystem procedures which are designed to assist IMP procedures which may be called from IMP, FORTRAN, PASCAL or C.

Keywords

EMAS-3, IMP, mixed language, multi-language, parameters, SSENTER, SSEXIT

Introduction

In EMAS-3 the languages other than IMP in general pass their parameters by name (reference). For most variable-types the data passed comprises a single word (32 bits) containing the address of the variable, or start of the variable's store area. For string (character) variables the parameter comprises a double-word of which the first contains codes describing the representation of the string variable in the calling language and the second contains the address of the first byte of the variable.

The Subsystem procedures SSENTER, SSEXIT (and SSENTER2, SSEXIT2) are provided for use by IMP programs to enable them easily to acquire integer, real and string parameters from any of the languages and to pass data back in those parameters on return if required.

In detail the types of parameters currently handled are:

			Type-code
integer	(*4)	32-bit	i
real	(*4)	32-bit	e
longreal	(*8)	64-bit	d
string(character)			s

Specification of SSENTER, SSEXIT, SSENTER2, SSEXIT2

Two sets of procedures are provided: SSENTER, SSEXIT for programs which handle integer and string parameters only and SSENTER2, SSEXIT2 for programs which in addition may require to handle real parameters. The specifications for these procedures are as follows:

```
%externalroutinespec SSENTER %alias "S#SSENTER" %c
               (%stringname STRING1, %integername INT1, %stringname TEMPLATE)

%externalroutinespec SSEXIT %alias "S#SSEXIT" %c
               (%stringname STRING1, %integername INT1, %stringname TEMPLATE)

%externalroutinespec SSENTER2 %alias "S#SSENTER2" %c
               (%stringname STRING1, %integername INT1, %real REAL1,
                  %longreal LONGREAL1, %stringname TEMPLATE)

%externalroutinespec SSEXIT2 %alias "S#SSEXIT2" %c
               (%stringname STRING1, %integername INT1, %real REAL1,
                  %longreal LONGREAL1, %stringname TEMPLATE)
```

Declarations in the IMP procedure

An IMP procedure which is to be accessed from IMP, FORTRAN, PASCAL or C declares local variables which are to receive the initial contents of the parameter variables. All parameters of the called routine must be of type %name (reference) because that is the standard form for languages other than IMP. When SSENTER is called, data are moved from the parameters into the locals. Conversely, when SSEXIT is called, values are returned as required from the IMP program's local variables into the calling program's variables. These operations are performed under the control of the characters in the string parameter TEMPLATE of the SSENTER and SSEXIT routines.

The positioning of the declarations for the variables to receive the input parameter data is important. There should be one variable of appropriate size (i.e. type) declared in the called program to correspond to each of the called procedure's parameters. The variables should be local variables and should be at the outermost level of the called procedure. All variables of a given type must be declared consecutively and in the same order in which they appear in the parameter list of the called routine (irrespective of intervening parameters of differing types). For example if the called routine has specification:

```
%externalroutine IMPRT (%integername A, %realname X, %integername B,
             %stringname S, %realname Y, %stringname T)
```

then the local declarations should be

```
%integer A1, B1
%real X1, Y1
%string(255) S1, T1
```

[The order of presentation of the groups of declarations of a given type is not important.]

Calling the SSENTER, SSEXIT procedures

Calls of SSENTER, SSEXIT are made by the IMP routine respectively at entry to and exit from the IMP routine. The *first* parameter of each is the first locally-declared integer variable which is to receive/offer integer parameter data. The *second* parameter of each is the first locally-declared string variable which is to receive/offer string parameter data. The *third* parameter of each is the TEMPLATE string which determines the actions to be performed on entry and exit. In the template string, pairs of characters indicate the type-code of the parameter as given in the table above (first character of the pair) and whether the parameter is read-only(r), write-only(w) or both read and write(b) (second character of the pair) by the called procedure. By "read" we mean here that the parameter is an input parameter to the called procedure. Similarly "write" means that the parameter is an output parameter.

Calls of SSENTER2, SSEXIT2 are similar: the five parameters are the first local integer, the first local string, the first local real, the first local longreal and the template string.

Examples

Here is an example of the use of SSENTER, and SSEXIT:

```
   %externalroutine ACCESSIBLE FROM IMP AND FORTRAN %c
            (%integername AXIS, NUMBER, %stringname TEXT,
               %integername X, Y, %stringname ANNOTATE)

   %integer AXISVALUE, NUMBERVALUE, XVALUE, YVALUE
   %string(255) TEXTVALUE, ANNOTATEVALUE

      SSENTER(TEXTVALUE, AXISVALUE, "irirsribibsw")
         ! The above call moves parameter data from AXIS, NUMBER, X, Y
         ! respectively to AXISVALUE, NUMBERVALUE, XVALUE, YVALUE and
         ! from TEXT to TEXTVALUE.
                  :
                  :
                  :
      SSEXIT(TEXTVALUE, AXISVALUE, "irirsribibsw")
         ! The above call moves data from XVALUE, YVALUE to parameters
         ! to parameters X, Y and from ANNOTATEVALUE to ANNOTATE.

   %end; ! ACCESSIBLE FROM IMP AND FORTRAN
```

In this example, SSENTER causes AXISVALUE, NUMBERVALUE, XVALUE, YVALUE to be loaded with the initial values of the parameters AXIS, NUMBER, X, Y, and TEXT to be loaded with the initial value of parameter TEXT. ANNOTATEVALUE is not loaded because the sixth pair (sw) of the TEMPLATE indicates that ANNOTATE is a write-only (output) parameter to routine ACCESSIBLE FROM IMP AND FORTRAN.

Similarly, the call of SSEXIT causes the calling program's variables X, Y to be set from XVALUE, YVALUE, and ANNOTATE from ANNOTATEVALUE.

Here is an example of the use of SSENTER2, SSEXIT2 in which only integer and longreal parameters are to be passed to and from the calling program:

```
%externalroutinespec FMULT(%integername I, J, %longrealname X, Y)

%integer IVALUE, JVALUE
%longreal XVALUE, YVALUE

    SSENTER2(""{unused string param}, IVALUE, 0{unused real param},
        XVALUE, "IRIRDBDB")
              :
              :
    SSEXIT2("", IVALUE, 0, XVALUE, "IRIRDBDB")

%end; ! of FMULT
```

Note that in a given called program, the calls of SSENTER, SSEXIT (and SSENTER2, SSEXIT2) have the same parameters. If the called routines does not require a particular parameter-type, it is not necessary to make any local declaration of that type, and the calls of SSENTER, SSEXIT should be given a zero (or null) value as the corresponding "first-local-variable" parameter.

It is important that TEMPLATE should correctly describe the called routine's parameters, which **MUST** be only of types integername, stringname. Any errors which can be detected will cause an error message to output stream 0 followed by a call of %STOP. The characters in TEMPLATE may be in upper or lower case.