PRAMCH

This subroutine is called from ACTOP to check the actual parameter with the correcponding formal parameter and to compile the relevant object program operation which will access this actual parameter for the procedure that is being called.

TABLE 1 gives a list of formal parameter types with the possible actual paramete types. In general, the rules are :-

(i) If the F.p. is called by name, the a.p. must have the same type and kind.

(ii) If the f.p. is a scalar called by value, the actual parameter may be a scalar, expression, array subscript, constant or type procedure call.

(iii) If the f.p. is a label called by value, the actual parameter may be a label or switch subscript.

(iv) Procedures, strings and switches and parameters of formal procedures may be called by name only.

The choice is further complicated, in the case of scalars, by the fact that an actual parameter may itself be a formal parameter. Table 2 gives the possibilities for this case.

The subroutine uses the following variables:-

PROCPO : namelist address of procedure using this actual parameter.

PRMCOU ; number of actual parameter in the procedure call.

I: namelist address of actual parameter.

TYPBOX : expression type (real or integer)

V : 0 if called by name, 1 if called by value.

The first job of PRAMCH is to recognise whether the parameter is of a formal procedure or not, then W locations 7 to 10 are set up the formal entry, V formal, type of formal and type difference (actual type - formal type). Next in the cases PRAMCH (0) and PRAMCH (3) when parameter checking words are compiled W14 is set with the check word, then W11 is set with a bit pattern according to W7, the formal entry, which later helps with the processing of

identifiers, non-formal function designators and array subscripts.

Little remains now except the somewhat tedious syntatical checking and to compile the parameter call. After this is compiled the parameter checking word if applicable and the routine exits to) or comma.

ERRORS

FAIL 5; fp is not label when ap is a switch subscript.
fp is not called by value when ap is a switch subscript.
fp is not scalar when ap is procedure call or array subscript.
fp is not scalar when ap is expression.
fp is not called by value when ap is expression.
ap is integer constant when fp is non integer by name.
ap is Boolean constant when fp is non Boolean.
ap is real constant when fp is non real by name.
ap is wrong type array when fp is array by value.
illegal ap called by value.
ap is not label when fp is label by value.
ap is not Boolean when fp is Boolean scalar.
wrong type of formal parameter scalar used as ap when fp is called by value.
wrong type of formal parameter used as ap when fp is not scalar.

FAIL 51; too many actual parameters.

FAIL 94; unrecognised formal type.

FAIL 108; parameter of formal procedure called by value.

TABLE 1

F.P.	Actual Parameter	Formal Parameter	Name or Value	Actual Parameter Operation
scalar	Integer identifier/constant	Integer	N	Take address or
	Real " "	Real	N	Take address or
	Boolean " "	Boolean	N	Take constant address
array	Integer array	Integer array	N	
	Real "	Real "	N	Take address of array map
	Boolean "	Boolean "	N	
procedure	Blank procedure	Blank procedure	N	
	Integer "	Integer "	N	Take address of procedure
	Real "	Real "	N	block
	Boolean "	Boolean "	N	
switch	Switch	Switch	N	Take constant area address
string	String	String	N	Take address of actual string
label	Label	Label	N	Take label address
	Label	Label	V	Take label address
	Switch subscript	Label	V	Index label address (INDS)
array	Integer array	Integer array	V	Take address of array map
	Real "	Real "	V	
	Boolean "	Boolean "	V	
scalar	Integer/Real constant	(Integer	V	Take value and compile type
	" " "	(Real	V	conversion if necessary.
	Boolean constant	Boolean	V	Take value.

TABLE 1 continued

F.P.	Actual Parameter	Formal Parameter	Name or Value	Actual Parameter Operation
scalar	expression	(Integer	V)	Compile type conversion if
	"	(Real	V)	necessary.
	"	(Boolean	V)	
	Integer/Real identifier	(Integer	V)	Take value and compile type
	" " "	(Real	V)	conversion if necessary.
	Boolean identifier	Boolean	V	Take value.
	Integer/Real array subscript	(Integer	V)	Index value and compile type
	" " " "	(Real	V)	conversion if necessary.
	Boolean array subscript	Boolean	V	Index value.
	Integer/Real procedure call	(Integer	V)	Compile type conversion
	" " " "	(Real	V)	if necessary.
	Boolean procedure call	Boolean	V	

Actual Parameters which are Formal Parameter Scalars

Example

```
real  procedure  P (t) ;  value t;  integer t;
      begin real R;
       R := t

      end
       .
       .
       .
       .
      procedure Q (a,b,c); value a; integer a,b; real c;
        begin real Y;
          Y := P (a) + P (b) + P (c);
      end
       ;
       .
       .
   Q  (JIM, FRED, BILL) ;
```

The call of procedure Q with actual parameters JIM, FRED and BILL will give value, address, address on the stack since the formal parameters are value a, name b,c. However, the procedure P expects an integer value on the stack, since its formal parameters t is called by value. So the final two parameters are called by Take Formal Value which puts the result in the stack.

In the following table of actual parameter operations, procedure Q would be SOURCE and procedure P would be DESTINATION.

TABLE 2

SOURCE		DESTINATION		Actual Parameter operation
value	I/R	value	I	Take Formal Value. R to I if necessary
value	I/R	value	R	" " " I to R "
value		name		Take Stack Address (R/IFUN) same type
name	I/R	value	I	Take Result Call by Name R to I if necessary
name	I/R	value	R	Take Result Call by Name I to R if necessary
name		name		Take Formal Address same type