NAME

e_reflex -- Input Message error reporting

SYNOPSIS

```
e_already(x,opt)  char *x, *opt;

e_arb(s1,s2,......$n,0)  char *s1, *s2, ....,*$n;

e_backgrd()

e_chl(x,opt)  char *x, *opt;

e_exkw(x,opt)  char *x, *opt;

e_incd(x,y,opt)  char *x, *y, *opt;

e_inckw(x,y,opt)  char *x, *y, *opt;

e_inperr(opt)  char *opt;

e_invc(x,opt)  char *x, *opt;

e_invd(x,y,opt)  char *x, *y, *opt;

e_invkw(x,opt)  char *x, *opt;

e_ip()

e_kw(s1,s2, .....,0)  char *s1, *s2, ..

e_loc(x,opt)  char *x, *opt;

e_lperm(x,opt)  char *x, *opt;

e_misd(x,opt)  char *x, *opt;

e_miskw(x,opt)  char *x, *opt;

e_ng(s1,s2, .....,0)  char *s1, *s2, ..

e_ofc(x,opt)  char *x, *opt;

e_ok()

e_perm(x,opt)  char *x, *opt;

e_pf()

e_punct(x,y,opt)  char *x, *y, *opt;

e_rgerr(x,opt)  char *x, *opt;

e_rlbsy(opt)  char *opt;
```

```
e_rlovld(opt)  char *opt;

e_sched()

e_spinoff()

e_ssys(x,opt)  char *x, *opt;

e_stderr()

e_stdin()

e_stdout()

e_syntax(s1,s2, .....,0)  char *s1, *s2, ..

e_uperm(x,y,opt)  char *x, *y, *opt;
```

DESCRIPTION

Each function (except e_ok , e_ip , and e_pf) is provided as a standard method of responding to reflexive errors in an Input Message (IM) or in a prompted user response error.

e_ok , e_ip , and e_pf are provided for the generation of common, high usage non-error messages.

Each function is listed below with the error message format. Note, however, that the message acknowlegements (OK, NG, etc.) are not preceeded by a newline and will be outputted on the same line as the input command. Some functions require arguments "x" and "y" of type (char *). If non-zero, the string is inserted in the message where indicated by <x> and <y>. Some functions require an argument "opt" of type (char *). If non-zero, the string contained in square brackets will be outputted in the formats below.

Those functions accepting an arbitrary number of arguments "s1", "s2",... of type (char *) require that the last argument be zero.

Each of the routines produces the following output which is directed to file descriptor 2.

```
e_already(x,opt)
 NG
 ALREADY <x> [; <opt>]

e_arb(s1,s2,.....,0)
  s1
s2 s3 ......

e_backgrd()
```

```
 NG
 Command can not executed in the background

e_chl(x,opt)
 ?E
 INVALID CHL: <x> [; <opt>]

e_exkw(x,opt)
 ?E
 EXTRA KEYWORD <x> [; <opt>]

e_incd(x,y,opt)
 ?E
 INCONSISTENT DATA <x>[, WITH <y>][; <opt>]

e_inckw(x,y,opt)
 ?E
 INCONSISTENT KEYWORD <x>[, WITH <y>][; <opt>]

e_inperr(opt)
 ?E
 INPUT ERROR [; <opt>]

e_invc(x,opt)
 ?E
 INVALID CHARACTER <x>[; <opt>]

e_invd(x,y,opt)
 ?E
 INVALID DATA <x> [FOR KEYWORD <y>][; <opt>]

e_invkw(x,opt)
 ?E
 INVALID KEYWORD: <x>[; <opt>]

e_ip()
 IP

e_kw(s1,s2,....,0)
 VALID KEYWORDS: <s1>
           <s2>
             .
             .

e_loc(x,opt)
 ?E
 INVALID LOCATION: <x>[; <opt>]

e_lperm(x,opt)
 ?E
 <x> NOT IN THIS LOCATION[; <opt>]

e_misd(x,opt)
```

```
 ?E
 MISSING DATA [FOR KEYWORD <x>][; <opt>]

e_miskw(x,opt)
 ?E
 MISSING KEYWORD <x>[; <opt>]

e_ng(s1,s2,...,0)
 NG
 <s1> <s2> ....

e_ofc(x,opt)
 ?E
 INVALID OFC: <x>[; <opt>]

e_ok()
 OK

e_perm(x,opt)
 ?E
 USE OF <x> NOT PERMITTED [; <opt>]

e_pf()
 PF

e_punct(x,y,opt)
 ?E
 INVALID PUNCTUATION; '<x>' SHOULD BE '<y>' [; <opt>]

e_rgerr(x,opt)
 ?E
 RANGE ERROR IN <x>[; <opt>]

e_rlbsy(opt)
 RL
 PROGRAM BUSY[; <opt>]

e_rlovld(opt)
 RL
 SYSTEM OVLD[; <opt>]

e_sched()
 NG
 Command can not be scheduled

e_spinoff()
 NG
 Command can not be spunoff

e_ssys(x,opt)
 ?E
 INVALID SUBSYSTEM: <x>[; <opt>]
```

```
e_stderr()
  NG
ERROR OUTPUT OF PROGRAM MUST BE TO A TERMINAL

e_stdin()
  NG
INPUT TO PROGRAM MUST BE FROM A TERMINAL

e_stdout()
  NG
NORMAL OUTPUT OF PROGRAM MUST BE TO A TERMINAL
e_syntax(s1,s2,....,0)
 PROPER FORMAT: <s1>
                <s2>
              .

e_uperm(x,y,opt)
 ?E
 <x> RESTRICTED TO <y>[; <opt>]
```

The e_arb() routine is intended to output reflexive errors of an arbitrary format as described by the strings pointed to by s2,s3,... It is not intended that this routine be used in lieu of other reflexive routines which satisfy a specific need. Note that s1 is to be used as an acknowledgement string. That is, two spaces will be prepended to s1 and a newline will be appended to s1. The string will then be written out before continuing with the remaining arguments. Following the output of s1, the routine will output the following strings on the same line unless a character count of 75 is exceeded or an imbedded newline is encountered. The routine will attempt to break a line between specified strings or at an imbedded space char. Note that the routine will place a space between each string.

FILES

LIBRARY

/lib/libl.a

DIAGNOSTICS

Reports as above. Returns a 0 if successful and a -1 in case of error.

BUGS