SETER - A CENTRALIZED ERROR-HANDLING PACKAGE FOR NCAR GRAPHICS

David J. Kennison
NCAR, P.O. Box 3000, Boulder, Colorado 80307-3000
email: kennison@ncar.ucar.edu (128.117.64.4)

SETER - A CENTRALIZED ERROR-HANDLING PACKAGE FOR NCAR GRAPHICS

David J. Kennison
NCAR, P.O. Box 3000, Boulder, Colorado 80307-3000
email: kennison@ncar.ucar.edu (128.117.64.4)

1.0 INTRODUCTION

The routines described in this document are used for error-handling by NCAR Graphics utilities. They are slightly modified versions of routines from the PORT library. The basic error-handling routine is as follows:

- **SETER** - when an NCAR Graphics utility detects an error, it calls this routine with three arguments: an error message, an error number, and a flag indicating whether the error is “fatal” or “recoverable”. The following actions are taken:

  If the error is fatal, SETER prints the error message, calls a dump routine, and terminates execution of the user’s program. (The default dump routine does nothing; a user-written version may be supplied to do something more useful.)

  If the error is recoverable, and if the user has **not** put SETER into recovery mode, a recoverable error is handled in the same way as a fatal one, but the dump routine is not called. If the user **has** put SETER into recovery mode, SETER sets an internal error flag equal to the error number, saves the error message, and returns control to the user. In this case, the user is expected to test for the occurrence of the recoverable-error condition, take suitable remedial action, and turn off SETER’s internal error flag; if the user does **not** do this, then, on the next call to SETER, both error messages (the remembered one and the one from that next call to SETER) are printed and execution is terminated.

The following routines are used to enter and/or leave recovery mode:

- **ENTSR** is normally called to enter recovery mode. The internal recovery-mode flag is given a new value; the previous value of the flag is returned to the caller.
- **RETSR** is normally called to leave recovery mode; it just resets the internal recovery-mode flag to a user-specified value, normally one returned by an earlier call to ENTSR.

The following routines are of use during recovery from an error:

- **EPRIN**, when called, prints the error message associated with the internal error flag, if that flag is non-zero. If the internal error flag is zero, EPRIN does nothing.

- **NERRO** is a function whose value is that of the internal error flag of SETER. The value will be non-zero if and only if a recoverable error has occurred and the flag has not yet been cleared by the user.
- **SEMESS** is a function of type CHARACTER whose value is either a requested portion of the error message for the error that caused the internal error flag to be set non-zero, if that flag **is** non-zero, or just blanks, otherwise.
- **ICLOEM** is a function whose value is the “actual” length of the error message defined by its single (character-string) argument (ignoring blanks on the right end of the string).
- **ERROF** turns off the internal error flag of SETER and should be called only after remedial action has been taken to recover from an error.

The following routine exists only to be replaced by the user:

- **FDUM** is the dump routine that is called by SETER when it is handling a fatal error. The default version of this routine just RETURNs; a user-written version may be supplied to do something more useful.

The following routine is provided for use in any routine that calls another routine which can, directly or indirectly, call SETER:

- **ICFELL** is a function of type INTEGER whose value is that of the internal error flag at the time ICFELL is entered. There are two arguments; when the value of ICFELL is non-zero, these arguments specify how the current internal error flag and error message are to be modified.

2.0 ROUTINES

This section contains detailed descriptions of all the error-handling routines.

2.1 SETER (MESSG,NERRF,ILEVL)

2.1.1 Usage

The FORTRAN statement “CALL SETER (MESSG,NERRF,ILEVL) is used to tell the error-handling package that an error has occurred. Exactly what happens as a result of such a call depends not only on the values of the arguments, but on the settings of two internal variables: the error flag, which says whether or not there was a prior error, and the recovery-mode flag, which says whether or not recovery mode is in effect.

- If the internal error flag is 0 (no prior error) and the internal recovery-mode flag is 2 (recovery mode not set, which is the default), SETER prints the error message MESSG, calls FDUM if and only if ILEVL = 2, and then STOPs.

- If the internal error flag is 0 (no prior error) and the internal recovery-mode flag is 1 (recovery mode set), then what happens depends on the value of ILEVL: if ILEVL = 2 (fatal error), SETER prints the error message MESSG, calls FDUM, and STOPs, but, if ILEVL = 1 (recoverable error), SETER resets the internal error flag = NERRF, remembers the error message MESSG, and RETURNs. In the latter case, the user is expected to detect the fact that an error has occurred, take the necessary remedial action, and call ERROF to turn the internal error flag off.
- If the internal error flag is non-zero (prior error), then the value of the internal recovery-mode flag will be 1 (recovery mode set); in this case, it doesn't make any difference what the value of ILEVL is: SETER prints both the remembered error message from the prior error and the new error message from MESSG, calls FDUM, and STOPs.

2.1.2 Arguments

MESSG (an input variable or constant of type CHARACTER) is an error message describing the error that has occurred. This should be of the form "XXXXXX - TEXT DESCRIBING THE ERROR", where "XXXXXX" is the name of the routine in which the error occurred, and it should not be more than 113 characters long.

NERRF (an input expression of type INTEGER) is a non-zero error number in the range from -999 to +9999.

ILEVL (an input expression of type INTEGER) is the "error level" - either a 1 or a 2, implying that the error described by MESSG and NERRF is recoverable or fatal, respectively.

2.2 ENTSR (IROLD,IRNEW)

2.2.1 Usage

The FORTRAN statement "CALL ENTSR (IROLD,IRNEW)" is normally used to enter recovery mode and save the previous value of the internal recovery-mode flag, but it can also be used to exit from recovery mode and save the previous value of the flag, or to just get the value of the flag, without changing it.

If recovery mode is turned off by a call to ENTSR at a time when the internal error flag is non-zero, this is treated as a fatal error; the error message is printed, the dump routine FDUM is called, and a STOP is executed.

2.2.2 Arguments

IROLD (an output variable of type INTEGER) receives the old value of the internal flag that indicates whether recovery mode is in effect or not. In the former case, the returned value will be a 1; in the latter case, it will be a 2. Normally, the value returned is saved for a later call to RETSR.

IRNEW (an input expression of type INTEGER) specifies what is to be done to the internal flag that indicates whether recovery mode is in effect or not. If IRNEW is a 1 or a 2, it becomes the new value of the internal recovery-mode flag: the value 1 turns recovery mode on and the value 2 turns it off. Other non-zero values are illegal, but, if IRNEW is zero, the state of the internal recovery-mode flag is not changed

2.3 RETSR (IROLD)

2.3.1 Usage

The FORTRAN statement "CALL RETSR (IROLD)" is normally used to return the internal flag of SETER that says whether or not recovery mode is in effect to the state that it was in before some past call to ENTSR.

If recovery mode is turned off by a call to RETSR at a time when the internal error flag is non-zero, this is treated as a fatal error; the error message is printed, the dump routine FDUM is called, and a STOP is executed.

2.3.2 Arguments

IROLD (an input expression of type INTEGER) specifies a desired value for the internal flag that indicates whether recovery mode is in effect or not. The value 1 turns recovery mode on and the value 2 turns it off. Normally, one uses a value of IROLD that was previously returned by a call to ENTSR.

2.4 EPRIN

2.4.1 Usage

If SETER's internal error flag is non-zero, the FORTRAN statement "CALL EPRIN" prints the remembered error message associated with the non-zero value. Otherwise, EPRIN does nothing.

2.4.2 Arguments

None.

2.5 NERRO (NERRF)

2.5.1 Usage

The expression “NERRO(NERRF)” has the value of the internal error flag of SETER. If its value is non-zero, this indicates that a prior recoverable error occurred and has not yet been cleared. The argument NERRF is given the same value as the function reference; this is useful in some situations.

2.5.2 Arguments

NERRF (an output variable of type INTEGER) receives the same value that is returned as the value of the function itself.

2.6 SEMESS (ITRIM)

2.6.1 Usage

If the internal error flag of SETER is non-zero, the value of the character expression “SEMESS(ITRIM)” is the remembered error message associated with that error flag, trimmed as indicated by the value of the input quantity ITRIM. Otherwise, it is a string of blanks. In any routine that references SEMESS, it must be declared to be of type CHARACTER*113. Assume that the current error message is as follows:

```
'CPCLAM/AREDAM - AREA-MAP ARRAY OVERFLOW'
```

Then, one could use SEMESS as follows:

```
SEMESS(0)='CPCLAM/AREDAM - AREA-MAP ARRAY OVERFLOW'
SEMESS(1)='AREDAM - AREA-MAP ARRAY OVERFLOW'
SEMESS(2)='AREA-MAP ARRAY OVERFLOW'
```

When ITRIM = 0 or less, no trimming is done; the whole error message is returned. When ITRIM = 1, prepended routine names and the associated slashes are trimmed off. When ITRIM = 2 or greater, all leading routine names are trimmed off; just the error description is returned.

2.6.2 Arguments

ITRIM (an input expression of type INTEGER) says what part of the error message is to be returned, as detailed above in the section called “Usage”.

2.7 ICLOEM (MESSG)

2.7.1 Usage

The value of the expression “ICLOEM(MESSG)” is the actual length of the non-blank portion of the input character variable MESSG. Thus, “MESSG(1:ICLOEM(MESSG))” is the non-blank part of MESSG. (It is assumed that the non-blank part of MESSG is left-justified in the character string, with blank fill to the right, as is the case for all NCAR Graphics error messages in calls to SETER.)

2.7.2 Arguments

MESSG (an input variable or constant of type CHARACTER) is most likely an error message returned by a reference to the function SEMESS.

2.8 ERROF

2.8.1 Usage

The FORTRAN statement “CALL ERROF” turns off (zeroes) SETER’s internal error flag.

2.8.2 Arguments

None.

2.9 FDUM

2.9.1 Usage

This subroutine is not intended to be called by the user. It is called by SETER when a fatal error is being processed and a STOP is about to be executed. The default version of FDUM does nothing. A user-supplied replacement version may take some other (system-dependent or program-dependent) action which would be helpful in diagnosing what may have caused the error to occur.

2.9.2 Arguments

None.

2.10 ICFELL (MESSG,NERRF)

2.10.1 Usage

ICFELL (which stands for “I Check For Errors on Lower Level”) is used to check for the occurrence of a recoverable error in a lower-level routine and (perhaps) to update the current internal error message and error flag.

The old value of the error flag is returned as the value of the function ICFELL. If that value is zero, nothing else has been done. If the value of ICFELL is non-zero, the following actions have been taken:

- If MESSG is blank, the current error message has not been changed.
- If MESSG is non-blank and its length is 6 or less, it should be the name of the routine referencing ICFELL; the current error message has been altered by prepending first a slash and then MESSG. (Note that, if an error occurs several levels deep, the effect of using ICFELL in this manner at each level is, effectively, to generate traceback information in the error message.)
- If MESSG is non-blank and its length is 7 or greater, its value has become the new value of the current error message and the previous error message has been printed. This is intended for use at the beginning of an NCAR Graphics routine to check for an outstanding error that the user has not recovered from and to ensure that the message for the outstanding error gets printed.
- If the expression NERRF has the value zero, the current error flag has not been changed.
- If the expression NERRF has a non-zero value, the value of the current error flag has been made equal to that value.

An example: Assume that the routine “A” calls the routine “B” and that “B” detects an error and calls SETER with error number “32” and error message 'B - ERROR HAS OCCURRED'. If recovery mode is **not** in effect, SETER prints the error message and STOPs; if recovery mode **is** in effect, control returns from “SETER” to “B” and thence to “A”. At that point, the statement

```
IF (ICFELL('A',13).NE.0) RETURN
```

detects the fact that an error has occurred in “B” and results in a return from “A” to whatever routine called it. It also changes the current error message to read 'A/B - ERROR HAS OCCURRED' and changes the error number from “32” to “13”.

Another example: Assume that the NCAR Graphics routine “A” is called when recovery mode is set and that it detects an error, calls SETER, and RETURNs to the user. If the user neglects to check the error state and calls the routine “B” next, the statement

```
IF (ICFELL('B - UNCLEARED PRIOR ERROR',1).NE.0) RETURN
```

ensures that the error message from routine “A” will be printed, that it will be replaced by an error message referring to the routine “B”, and that “B” won’t do anything else.

2.10.2 Arguments

MESSG (an input variable or constant of type CHARACTER) is a character string saying what is to be done to the current error message (when an error has occurred). If MESSG is blank, no change is to be made in the current error message. If the length of MESSG is six characters or fewer, a slash is prepended to the current error message and then MESSG is prepended to that. If the length of MESSG is seven characters or more, MESSG is a complete new error message to be substituted for the current one.; it should be of the form “XXXXXX - TEXT DESCRIBING THE ERROR”, where “XXXXXX” is the name of the routine in which the error occurred, and it should not be more than 113 characters long.

NERRF (an input expression of type INTEGER) has a value in the range from -999 to +9999. If this value is zero, no change is made in the internal error flag; otherwise, it is the desired new value of that flag (when an error has occurred).

3.0 USING SETER IN NCAR GRAPHICS

There are specific conventions for the use of SETER within NCAR Graphics, as follows:

- All detectable errors shall be recoverable, in the sense described above. (That is, in every call to SETER, the final argument shall be a 1, rather than a 2.) This is by request of the folks doing NCAR Interactive, who consider STOPs in the utilities undesirable. The idea is to let the user decide what is to be done about the various error conditions.
- Whenever an NCAR Graphics routine calls a lower-level routine that might detect an error and call SETER, it should subsequently use ICFELL to check the error state; if a recoverable error has occurred, it should first do required clean-up chores, if possible, and then pass control back to the routine that called it. In all such uses of ICFELL, the first argument should be the name of the routine referencing ICFELL and the second argument should be a new number for the error, reflecting the position of the reference to the lower-level routine in the upper-level routine.
- Any NCAR Graphics routine that can be called by a user and that can potentially yield a call to SETER must immediately check the error state and, if that error state is non-zero, return control without doing anything else. This is most conveniently done using a reference to ICFELL; see the second example in the “Usage” section of the description of ICFELL. All such references should have a first argument of the form 'XXXXXX - UNCLEARED PRIOR ERROR', where “XXXXXX” is the name of the routine in which the reference occurs, and a second argument equal to “1”.
- It is recommended that, within a given utility routine, the error numbers in references to SETER and ICFELL should start at 1 and increment by 1. These numbers generally have no intrinsic meaning in and of themselves: they are merely intended to allow a consultant to find the reference that generated a given error.

- NCAR Graphics routines are not required to turn recovery mode on before calling a lower-level routine that might call SETER (which was the convention in the PORT library, as described in the PORT document). Instead, the assumption is that it is the responsibility of the **user** of NCAR Graphics to set recovery mode if he/she desires to do recovery. Since, by default, recovery mode is turned off, **all** NCAR Graphics calls to SETER will be treated as fatal: the error message will be printed and execution will be terminated. Once the user turns recovery mode on, however, **no** NCAR Graphics error will be treated in this way except for one that the user fails to recover from.

Note: These conventions are being adopted as of December 2, 1993, and represent a goal for the future. The current situation is somewhat muddled: In some utilities, all SETER calls are fatal ones. In other utilities, some SETER calls are fatal and some are not. In other utilities, no SETER calls are fatal. In general, errors at a lower level are not detected and passed back up the call chain. Users have complained (and rightly so) that error recovery is, in general, not possible; observance of these conventions should help to fix the situation.

Further note: As of March 30, 1994, the situation has improved markedly. CONPACK and all utilities referenced by it have been updated to follow the guidelines given above and work is proceeding on other utilities.

There is one sticky area in which questions remains to be answered: Sometimes, when an error condition occurs during execution of a utility routine that has changed the internal state of GKS or SPPS, it has not been possible to restore the state of those packages to exactly what it was before that routine was entered. In some cases, better bookkeeping would allow restoration to be done; in other cases, though, restoration would involve calling a routine that could generate a call to SETER, which would cause a STOP. In the latter cases, I have not yet worked out a good solution. For the moment, therefore, the situation is this: if you call an NCAR Graphics utility with recovery mode turned on and, when you get control back, you find that an error has occurred, you must be prepared to deal with the possibility that at least the following things might been changed: 1) the current SET call; 2) the current polyline color index; 3) the current polymarker color index; 4) the current text color index; 5) the current fill area color index; 6) the current dash pattern.

4.0 EXAMPLES

The NCAR Graphics examples named “tseter” and “arex02” illustrate the use of SETER and associated routines; the former demonstrates minimal functioning of SETER and the latter demonstrates how to use the package in a real situation: to recover from area-map array overflow errors in the package AREAS.

Note: An example named “xxxxxx”may be obtained, on a Unix system on which NCAR Graphics is installed, by typing the command “ncargex xxxxxx”. A file called “xxxxxx.f” will be copied into the current working directory, compiled, and executed. If a metafile is produced, it will have the name “xxxxxx.ncgm”. The command “ncargex -n xxxxxx” may be used to get the source file “xxxxxx.f” without compiling or executing it.