SUN/261.3

Starlink Project
Starlink User Note 261.3

D.S. Berry & Malcolm J. Currie

7th October July 2019

IRQ — Handling of QUALITY in NDFs
Version 5.0
User's Guide

Abstract

This library is a set of Fortran routines for manipulation of quality information within NDFs. In particular it uses names that will be more memorable than bits to assign and set quality attributes of data values within an NDF.

Contents

1 Introduction to QUALITY

A QUALITY structure is one of the standard components of an NDF structure, and is described fully in SUN/33. Briefly, if an NDF has a QUALITY component which is in a *defined* state, then each pixel within the NDF DATA component has a corresponding value in the QUALITY component. Currently, each QUALITY value consists of an unsigned byte (*i.e.* 8 bits). In Fortran the least-significant bit is usually called Bit 0 and the most significant bit is usually called Bit 7. Within the IRQ package the least-significant bit is called Bit 1 and the most-significant bit is called Bit 8. Each bit within the QUALITY value can be used to indicate if the corresponding pixel in the DATA component holds some specific *quality*. For instance, Bit 3 of the QUALITY component may be used to indicate if any DATA pixels are saturated. A particular pixel in the QUALITY component would have Bit 3 set (*i.e.* equal to 1) if the corresponding DATA pixel is saturated, or cleared (*i.e.* equal to 0) if the corresponding DATA pixel is not saturated.

Another option for flagging saturated data is to replace saturated DATA pixel values with a 'bad' (or 'magic') value. This has the disadvantage that the datum is permanently destroyed by being flagged, and also there is no distinction between data values thst are set bad because of the fact they were saturated, and pixels set bad for any other reason.

SUN/33 doesn't specify how the facilities of the QUALITY component are to be used, and many possibilities exist. Obviously some co-ordination between applications is needed so that different applications interpret the QUALITY values in a consistent manner ({emphi.e. using the above example, later applications must know that Bit 3 is a saturation flag). The IRQ package provides a set of routines for doing this.

You may be are wondering about the name IRQ. The library was originally developed for the IRAS90 package, and its subroutine libraries had IR prefix.

2 Introduction to the facilities provided by the IRQ library

IRQ provides a system for handling Boolean qualities (*i.e.* qualities that are either held or not held by each DATA pixel). From the point of view of an application, each defined quality is identified by a *quality name* rather than by a bit number. Information about these quality names is stored in an NDF extension, so that later applications can determine which pixels within the NDF hold a given quality (or combination of qualities).

Within IRQ, each defined quality name is usually associated with a bit in the QUALITY array, and this bit is set if the corresponding DATA pixels are assigned the specified quality. A typical application need know nothing about which QUALITY bit is associated with which quality name. For instance, if *every* pixel in an NDF holds a certain quality (or alternatively, if *no* pixels hold the quality), then it is not necessary to reserve a bit in the QUALITY array to represent the quality. Instead, a single Boolean scalar value can be stored with the quality name in the NDF extension. This scalar is set to `.TRUE.` if *all* pixels hold the quality, and `.FALSE.` if *no* pixels hold the quality. In this way, the number of defined quality names can sometimes exceed the number of bits in the QUALITY component. The handling of such situations is done within IRQ and is completely invisible to the calling application.

IRQ provides the following facilities.

- Add quality name definitions (and associated descriptive comments) to an NDF. There is a limit to the number of quality names which may be defined within an NDF. The exact number depends on how many quality names require a QUALITY bit, but it will always be at least eight.
- Remove quality name definitions from an NDF.
- List all quality names defined within an NDF.
- Assign a given quality to selected pixels.
- Remove a given quality from selected pixels.
- Set 'bad' those pixels of a supplied array which hold a certain combination of qualities.

3 A Set of Four Typical IRQ Applications

The KAPPA package contains a set of four typical IRQ applications, which give an indication of the benefits which IRQ can provide for the user. The A-TASK documentation for these four applications is included in SUN/95, and they are briefly described in this section.

3.1 SETQUAL

SETQUAL creates quality name definitions and stores them within a specified extension of an NDF. It also assigns a specified quality to a sub-set of the pixels with the NDF. In its simplest mode, the user provides an NDF, and a string to use as a quality name (such as `SATURATED`). If this quality name is not already defined within the NDF, then it is added to the list of defined quality names, together with a user-supplied comment describing the quality. The user also provides another NDF to be used as a 'mask'. The 'bad' pixels within the mask NDF define the pixels which are to be assigned the given quality. For instance, a user may have an image containing saturated pixels. If he wants these pixels flagged without being permanently destroyed, he could proceed as follows.

(1) Produce a mask NDF from the original NDF by setting all DATA pixels above the saturation value to the 'bad' value.

(2) Run SETQUAL on the original NDF, giving some quality name such as `SATURATED`, and specifying the mask created in the previous step. This leaves the DATA component of the original NDF unchanged, but assigns the quality `SATURATED` to all the pixels which correspond to 'bad' pixels in the mask.

Alternatively, instead of using a mask, the pixels to which the quality is assigned may be specified by an explicit list of pixel indices stored in a text file.

3.2 REMQUAL

The REMQUAL application removes quality name definitions from an NDF. This may be necessary if there is no room for any more quality name definitions within an NDF. In this case the user may choose to remove some unimportant quality name definitions to make room for new, more-important quality names. REMQUAL can also remove *all* quality names information from an NDF. This can be useful if for any reason the quality-name information becomes corrupted.

3.3 SHOWQUAL

The SHOWQUAL application displays all currently defined quality names within an NDF, together with the associated descriptive comments. Optionally, the number of pixels that hold each quality can be displayed.

3.4 QUALTOBAD

The QUALTOBAD application sets selected pixels within an NDF to the 'bad' value on the basis of the pixel's quality. In effect, QUALTOBAD performs the reverse operation of SETQUAL For instance, using the example of saturated data described above, the user may remove some varying background surface from his original data, and then want to set pixels which were saturated in the original data to the 'bad' value . In this case he would run QUALTOBAD specifying the background removed NDF as input, and specifying a *quality expression* of `SATURATED`. This would cause all pixels which hold the quality `SATURATED` to be set bad in the output NDF.

The 'quality expression' can be more complex than a single quality name. In fact, a quality expression can consist of several quality names combined together using the usual operators of Boolean algebra. For instance, if the quality expression `SATURATED.AND..NOT.(SOURCE_A .OR. SOURCE_B)` was given to QUALTOBAD, then pixels would be set 'bad' only if they had the quality `SATURATED`, but did not have either of the qualities `SOURCE_A` or `SOURCE_B`.

4 Quality Names

A quality name must contain 15 or fewer characters. Any leading blanks are removed from supplied quality names, and they are converted to upper case before being stored in the NDF. Quality names may contain embedded blanks, but may not contain full-stop (`"."`) characters. Certain names are reserved and may not be used. These are `ANY`, `IRQ_BAD_SLOT` and `IRQ_FREE_SLOT`.

5 Quality Expressions

A 'Quality Expression' consists of a set of quality names combined together using Boolean operators into a legal Boolean expression. See Section 6.6 for a description of the use of quality expressions within IRQ. In the following, the symbols `A` and `B` are used to represent two qualities. These can be considered as Boolean values; `true` if a pixel holds the quality, and `false` otherwise. The supported Boolean operators are listed below.

.AND. - The expression `(A.AND.B)` is true if and only if both `A` and `B` are true.

.OR. - The expression `(A.OR.B)` is true if and only if either `A` or `B` is true.

.XOR. - The expression `(A.XOR.B)` is true if and only if either `A` is true and `B` is false, or `A` is false and `B` is true.

.EQV. - The expression `(A.EQV.B)` is true if and only if either `A` is true and `B` is true, or A is false and `B` is false.

.NOT. - The expression `(.NOT.A)` is true if and only if `A` is false.

In addition to the above operators, the Boolean constants `.FALSE.` and `.TRUE.` can be included within a quality expression. Expressions may contain several levels of nested parentheses.

The precedence of these operators decreases in the following order; .NOT., .AND., .OR., .XOR., .EQV. (the final two have equal precedence). In an expression such as `(A.XOR.B.EQV.C.XOR.D)` in which all operators have equal precedence, the evaluation proceeds from left to right., *i.e.* the expression is evaluated as (((A.XOR.B).EQV.C).XOR.D). If there is any doubt about the order in which an expression will be evaluated, parentheses should be used to ensure the required order of evaluation.

Some attempts are made to simplify a quality expression to reduce the run time needed to evaluate the expression for every pixel.

Quality expressions can be up to 254 characters long, and must not contain more than forty symbols (Boolean operators, constants, or quality names).

6 Using IRQ routines

This section gives a brief outline of the IRQ routines which are available to perform some common tasks. The specific details required to use these routines are not included here but can be found in the subroutine specifications contained in Appendix C.

6.1 Constants and Error Values

The IRQ package has associated with it various symbolic constants. These values consist of a name of up to five characters prefixed by "IRQ__" (note the *double* underscore), and can be made available to an application by including the following line at the start of the routines which uses them:

```
INCLUDE 'IRQ_PAR'
```

This assumes that the IRQ library has been installed as part of the UNIX Starlink Software Collection.

The values thus defined are described in the following sections, and also in the subroutine specifications. Another set of symbolic constants is made available by the statements

```
INCLUDE 'IRQ_ERR'
```

These values have the same format of those contained in IRQ_PAR, put define various error conditions which can be generated within the IRQ package. Applications can compare the *STATUS* argument with these values to check for specific error conditions. These values are described in Appendix G.

6.2 Initialising an NDF for use with IRQ

Certain HDS structures must be created within an NDF before the NDF can be used by IRQ. These structures hold information describing the currently defined quality names within the NDF. Routine IRQ_FIND can be used to see if such structures exist within an NDF (see below). If no such structure yet exists within an NDF, routine IRQ_NEW must be called to create the structure. This structure is held in a specified NDF extension. IRQ_NEW returns an array of HDS locators which must be passed to subsequent IRQ routines. One of these locators points to a cloned copy of the NDF identifier. All access to the NDF by subsequent IRQ routines is achieved through this cloned identifier. Once access to the quality names information is no longer required, the resources used by these HDS locators (including the cloned NDF identifier) should be annulled by calling IRQ_RLSE. Note, routines IRQ_ANNUL and IRQ_CLOSE play no part in releasing resources used for accessing quality name information. These two routines are only used for releasing resources used to store *compiled quality expressions* (see Section 6.6). IRQ_RLSE is the only routine which needs to be called to release resources used for accessing *quality names*.

6.3 Using previously initialised NDFs within IRQ

If the NDF has been initialised for use by IRQ, then the structure holding the quality names information must be found before other IRQ routines can be called. Routine IRQ_FIND does this. This routine looks through all the NDF extensions until a suitable structure is found. If no such structure is found the status value `IRQ__NOQNI` is returned. If more than one extension contains such a structure, then the status value `IRQ__MULT` is returned. If the routine runs successfully, then an array of HDS locators is returned similar to the array returned by IRQ_NEW. Once access to the quality names information is no longer required, the resources used by these HDS locators (including the cloned NDF identifier) should be annulled by calling IRQ_RLSE.

6.4 Accessing the quality names information stored in an NDF

To access the quality names information stored within an NDF, a set of HDS locators to the information must first be obtained by calling either IRQ_NEW or IRQ_FIND (see above). Once this has been done the information can be read, modified, or added to. IRQ_ADDQN adds a new quality name definition to an NDF (so long as there is room for it within the structures and the QUALITY component). IRQ_CHKQN checks to see if a given name is defined. IRQ_GETQN searches for a specified quality name, and returns various items of information about it (such as which bit of the QUALITY component it is assigned to). IRQ_NUMQN returns the number of defined quality names. IRQ_NXTQN returns the next defined quality name. Repeated calls to IRQ_NXTQN can be made to get a list of all the quality names defined within an NDF. IRQ_REMQN removes the definition of a quality name from an NDF, so long as the quality name has not been flagged as 'read-only' (see IRQ_RWQN).

6.5 Assigning and removing qualities to and from NDF pixels

Once quality names have been defined, they can be assigned to selected pixels within the NDF. There are two ways of specifying which pixels are 'selected'. Routine IRQ_SETQM requires a mask image to be provided in which the 'bad' pixels define the pixels in the NDF DATA component to which the quality is to be assigned. Routine IRQ_SETQL requires a list of pixel indices to be provided which defines the pixels in the NDF DATA component to which the quality is to be assigned. The quality of the NDF pixels which are *not* selected is left unchanged. Thus, if an unselected pixel already has the specified quality, these routines will not *remove* the quality from those pixels.

The routines IRQ_RESQM and IRQ_RESQL, are complementary to IRQ_SETQM and IRQ_SETQL. Instead of ensuring that the selected pixels hold the specified quality, these routines ensure that the selected pixels *do not* hold the specified quality. Again, the quality of unselected pixels is left unchanged.

Routine IRQ_SETQ ensures that *all* pixels in an NDF have a specified quality, and routine IRQ_RESQ ensures that *no* pixels in an NDF have a specified quality.

The routine IRQ_CNTQ will count the number of bits set in each bit plane in the QUALITY component.

6.6 Finding NDF pixels which satisfy a quality expression

If a subset of pixels are to be operated on by an application, the application will usually obtain a quality expression from the user which defines the pixels to be operated on. That is to say, pixels which have qualities which do not satisfy the quality expression would not usually be used by the application. The quality expression is usually obtained from the user using the ADAM parameter system. It is then passed to IRQ_COMP to be compiled. If any problems are detected with the expression (such as syntax errors, or undefined quality names) IRQ_COMP will report an error, and it is then up to the applications to decide what to do. Typically, it will flush the error, and reprompt the user for a new quality expression. The routine IRQ_SYNTX will check a quality expression for syntax errors without actually compiling it or checking that the referenced quality names are defined.

If IRQ_COMP successfully compiles the quality expression, it returns an identifier for the compiled quality expression. This identifier is actually a pointer to a temporary HDS structure holding information about the quality expression. The identifier is passed to IRQ_SBAD which locates all NDF pixels that have qualities that satisfy the quality expression. The corresponding pixels in an array supplied to IRQ_SBAD are set 'bad'. The other pixels are left unchanged. Alternatively, pixels with qualities which *do not* satisfy the quality expression can be set bad in the supplied array. Note, the array supplied to IRQ_SBAD must be the same shape and size as the NDF supplied to IRQ_FIND or IRQ_NEW. If an NDF section was used, then the array supplied to IRQ_SBAD must be the same shape and size as the NDF section, *not* the base NDF.

Up to ten compiled quality expressions can be active at once. The identifiers for each compiled quality expression should be annulled when it is no longer needed by calling IRQ_ANNUL. IRQ_CLOSE should be called when all compiled quality expressions have been finished with.

7 Compiling and Linking with IRQ

This section describes how to compile and link applications which use IRQ subroutines, on UNIX systems. It is assumed that the IRQ library is installed as part of the Starlink Software Collection.

7.1 Standalone Applications

Standalone applications which use IRQ_ routines may be linked by including execution of the command "irq_link" on the compiler command line. Thus, to compile and link a Fortran application called "prog", the following might be used:

```
% f77 -I$STARLINK_DIR/include prog.f -L$STARLINK_DIR/lib `irq_link` -o prog
```

Note the use of backward quote characters, which cause the "irq_link" command to be executed and its result substituted into the compiler command.

7.2 ADAM Applications

Users of the ADAM programming environment (SG/4) should use the **alink** command (SUN/144) to compile and link applications, and can access the IRQ_ library by including execution of the command `irq_link_adam` on the command line, as follows:

```
% alink adamprog.f `irq_link_adam`
```

where `adamprog.f` is the Fortran source file for the A-TASK. Again note the use of opening apostrophies (`) instead of the more usual closing apostrophy (') in the above **alink** command.

To build a program written in C (instead of Fortran), simply name the source file `adamprog.c`, instead of `adamprog.f`.

A Routine Descriptions

IRQ_ADDQN(LOCS, QNAME, DEFLT, COMMNT, STATUS)
Define a new quality name.

IRQ_ANNUL(IDQ, STATUS)
Annul an identifier for a compiled quality expression.

IRQ_CHKQN(LOCS, QNAME, THERE, STATUS)
Check that a specified quality name is defined.

IRQ_CLOSE(STATUS)
Close down the compiled quality expression identifier system.

IRQ_CNTQ(LOCS, SIZE, SET, STATUS)
Count the number of pixels with each bit set in the QUALITY component.

IRQ_CNTQ8(LOCS, SIZE, SET, STATUS)
*Count the number of pixels with each bit set in the QUALITY component - INTEGER*8 interface.*

IRQ_COMP(LOCS, SIZE, INFO, QEXP, UNDEF, NUNDEF, ERRPNT, IDQ, STATUS)
Compile a quality expression.

CALL IRQ_DELET(INDF, STATUS)
Delete all quality-name information from an NDF.

IRQ_FIND(INDF, LOCS, XNAME, STATUS)
Find a structure containing quality names information.

IRQ_FXBIT(LOCS, QNAME, BIT, SET, FIXBIT, STATUS)
Assign a fixed bit number to a quality name.

IRQ_GETQN(LOCS, QNAME, FIXED, VALUE, BIT, COMMNT, STATUS)
Get information about a specified quality name.

CALL IRQ_GETQX(PARAM, QEXP, STATUS)
Get a quality expression from the user and check for syntax errors.

IRQ_NEW(INDF, XNAME, LOCS, STATUS)
Create a structure to hold quality names information.

IRQ_NUMQN(LOCS, NAMES, STATUS)
Return the number of defined quality names.

IRQ_NXTQN(LOCS, CONTXT, QNAME, FIXED, VALUE, BIT, COMMNT, DONE, STATUS)
Return information about the next defined quality name.

IRQ_RBIT(LOCS, QNAME, BIT, STATUS)
Reserve a bit number for a given quality name.

IRQ_REMQN(LOCS, QNAME, STATUS)
Remove the definition of a quality name.

IRQ_RESQ(LOCS, QNAME, STATUS)
Ensure no pixels hold a specified quality.

IRQ_RESQL(LOCS, LISTED, QNAME, NDIM, NCOORD, LIST, SET, STATUS)
Ensure pixels selected by a list do not hold a specified quality.

IRQ_RESQL8(LOCS, LISTED, QNAME, NDIM, NCOORD, LIST, SET, STATUS)
*Ensure pixels selected by a list do not hold a specified quality - INTEGER*8 interface.*

IRQ_RESQM(LOCS, BAD, QNAME, SIZE, MASK, SET, STATUS)
Ensure pixels selected by a mask do not hold a specified quality.

IRQ_RESQM8(LOCS, BAD, QNAME, SIZE, MASK, SET, STATUS)
*Ensure pixels selected by a mask do not hold a specified quality - INTEGER*8 interface.*

IRQ_RLSE(LOCS, STATUS)
Release the resource used to locate quality name information.

IRQ_RWQN(LOCS, QNAME, SET, NEWVAL, OLDVAL, STATUS)
Get and/or set the read-only flag for a quality name.

IRQ_SBADx(IDQ, HELD, SIZE, VEC, ALLBAD, NOBAD, STATUS)
Set pixels bad which satisfy a given quality expression.

IRQ_SETQ(LOCS, QNAME, STATUS)
Ensure all pixels hold a specified quality.

IRQ_SETQL(LOCS, LISTED, QNAME, NDIM, NCOORD, LIST, SET, STATUS)
Ensure pixels selected by a list hold a specified quality.

IRQ_SETQL8(LOCS, LISTED, QNAME, NDIM, NCOORD, LIST, SET, STATUS)
*Ensure pixels selected by a list hold a specified quality - INTEGER*8 interface.*

IRQ_SETQM(LOCS, BAD, QNAME, SIZE, MASK, SET, STATUS)
Ensure pixels selected by a mask hold a specified quality.

IRQ_SETQM8(LOCS, BAD, QNAME, SIZE, MASK, SET, STATUS)
*Ensure pixels selected by a mask hold a specified quality - INTEGER*8 interface.*

IRQ_SYNTX(QEXP, ERRPNT, STATUS)
Check a quality expression for syntax errors.

B Classified List

B.1 Gaining Access to Quality Name Information Within an NDF

IRQ_FIND(INDF, LOCS, XNAME, STATUS)
Find a structure containing quality names information.

IRQ_NEW(INDF, XNAME, LOCS, STATUS)
Create a structure to hold quality names information.

IRQ_RLSE(LOCS, STATUS)
Release the resource used to locate quality name information.

B.2 Storing, Retrieving and Deleting Quality Names

IRQ_ADDQN(LOCS, QNAME, DEFLT, COMMNT, STATUS)
Define a new quality name.

IRQ_CHKQN(LOCS, QNAME, THERE, STATUS)
Check that a specified quality name is defined.

CALL IRQ_DELET(INDF, STATUS)
Delete all quality-name information from an NDF

IRQ_FXBIT(LOCS, QNAME, BIT, SET, FIXBIT, STATUS)
Assign a fixed bit number to a quality name.

IRQ_GETQN(LOCS, QNAME, FIXED, VALUE, BIT, COMMNT, STATUS)
Get information about a specified quality name.

IRQ_NUMQN(LOCS, NAMES, STATUS)
Return the number of defined quality names.

IRQ_NXTQN(LOCS, CONTXT, QNAME, FIXED, VALUE, BIT, COMMNT, DONE, STATUS)
Return information about the next defined quality name.

IRQ_RBIT(LOCS, QNAME, BIT, STATUS)
Reserve a bit number for a given quality name.

IRQ_REMQN(LOCS, QNAME, STATUS)
Remove the definition of a quality name.

IRQ_RWQN(LOCS, QNAME, SET, NEWVAL, OLDVAL, STATUS)
Get and/or set the read-only flag for a quality name.

B.3 Handling Quality Expressions

IRQ_ANNUL(IDQ, STATUS)
Annul an identifier for a compiled quality expression.

IRQ_CLOSE(STATUS)
Close down the compiled quality expression identifier system.

IRQ_COMP(LOCS, SIZE, INFO, QEXP, UNDEF, NUNDEF, ERRPNT, IDQ, STATUS)
Compile a quality expression.

IRQ_GETQN(LOCS, QNAME, FIXED, VALUE, BIT, COMMNT, STATUS)
Get information about a specified quality name.

IRQ_SBADx(IDQ, HELD, SIZE, VEC, ALLBAD, NOBAD, STATUS)
Set pixels bad which satisfy a given quality expression.

IRQ_SYNTX(QEXP, ERRPNT, STATUS)
Check a quality expression for syntax errors.

B.4 Assigning Qualities to Selected Pixels

IRQ_RESQ(LOCS, QNAME, STATUS)
Ensure no pixels hold a specified quality.

IRQ_RESQL(LOCS, LISTED, QNAME, NDIM, NCOORD, LIST, SET, STATUS)
Ensure pixels selected by a list do not hold a specified quality.

IRQ_RESQL8(LOCS, LISTED, QNAME, NDIM, NCOORD, LIST, SET, STATUS)
*Ensure pixels selected by a list do not hold a specified quality - INTEGER*8 interface.*

IRQ_RESQM(LOCS, BAD, QNAME, SIZE, MASK, SET, STATUS)
Ensure pixels selected by a mask do not hold a specified quality.

IRQ_RESQM8(LOCS, BAD, QNAME, SIZE, MASK, SET, STATUS)
*Ensure pixels selected by a mask do not hold a specified quality - INTEGER*8 interface.*

IRQ_SETQ(LOCS, QNAME, STATUS)
Ensure all pixels hold a specified quality.

IRQ_SETQL(LOCS, LISTED, QNAME, NDIM, NCOORD, LIST, SET, STATUS)
Ensure pixels selected by a list hold a specified quality.

IRQ_SETQL8(LOCS, LISTED, QNAME, NDIM, NCOORD, LIST, SET, STATUS)
*Ensure pixels selected by a list hold a specified quality - INTEGER*8 interface.*

IRQ_SETQM(LOCS, BAD, QNAME, SIZE, MASK, SET, STATUS)
Ensure pixels selected by a mask hold a specified quality.

IRQ_SETQM8(LOCS, BAD, QNAME, SIZE, MASK, SET, STATUS)
*Ensure pixels selected by a mask hold a specified quality - INTEGER*8 interface.*

B.5 Enquiring Pixel Quality

IRQ_CNTQ(LOCS, SIZE, SET, STATUS)
Count the number of pixels with each bit set in the QUALITY component.

IRQ_CNTQ8(LOCS, SIZE, SET, STATUS)
*Count the number of pixels with each bit set in the QUALITY component - INTEGER*8 interface.*

C Full Routine Specifications

IRQ_ADDQN
Define a new quality name

Description:

This routine adds the quality name specified by QNAME to the NDF specified by LOCS. LOCS must previously have been assigned values by one of the routines IRQ_FIND or IRQ_NEW. If the quality name is already defined, an error is reported. Note, this routine does not reserve a bit in the QUALITY component for the new quality name, it mearly established a default value for the quality which will be used for all pixels in the NDF if no subsequent call to IRQ_SETQL or IRQ_SETQM is made. Note, the string ANY cannot be used as a quality name. Also, quality names may not contain any full stops.

An error is reported if only READ access is available to the NDF.

Invocation:

```
CALL IRQ_ADDQN( LOCS, QNAME, DEFLT, COMMNT, STATUS )
```

Arguments:

LOCS(5) = CHARACTER * (*) (Given)

An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

QNAME = CHARACTER * (*) (Given)

The new quality name to store. The maximum length of this string is given by symbolic constant IRQ__SZQNM which currently has the value 15. Leading spaces are ignored, and the stored name is converted to upper case.

DEFLT = LOGICAL (Given)

If true, then by default all pixels are assumed to hold the quality specified by QNAME. If false, then it is assumed that no pixels hold the quality.

COMMNT = CHARACTER * (*) (Given)

A descriptive comment to store with the quality name. The maximum length of this string is given by symbolic constant IRQ__SZCOM, which currently has the value 50. Any characters beyond this length are ignored.

STATUS = INTEGER (Given and Returned)

The global status.

IRQ_ANNUL
Release an IRQ identifier

Description:

All internal resources used by the specified compiled quality expression identifier (created by IRQ_COMP) are released.

This routine attempts to execute even if STATUS is bad on entry, although no further error report will be made if it subsequently fails under these circumstances.

Invocation:

```
CALL IRQ_ANNUL( IDQ, STATUS )
```

Arguments:

IDQ = INTEGER (Given)
An IRQ identifier for a compiled quality expression.

STATUS = INTEGER (Given and Returned)
The global status.

IRQ_CHKQN
Check a specified quality name to see if it is defined

Description:

This routine searches for a specified quality name in the quality name specified by LOCS. If it is found, THERE is returned true. Otherwise THERE is returned false.

Invocation:

```
CALL IRQ_CHKQN( LOCS, QNAME, THERE, STATUS )
```

Arguments:

LOCS(5) = CHARACTER * (*) (Given)

An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

QNAME = CHARACTER * (*) (Given)

The quality name to search for. Leading blanks are ignored and the search is case-insensitive. The maximum allowed length for quality names is given by symbolic constant IRQ__SZQNM which currently has the value of 15.

THERE = LOGICAL (Returned)

If true, then the quality name is defined within the NDF specified by LOCS. If false, then the quality name is undefined.

STATUS = INTEGER (Given and Returned)

The global status.

IRQ_CLOSE
Close down the IRQ identifier system

Description:

This routine must be called once all use of compiled quality expression identifiers (as generated by IRQ_COMP) has been completed. All internal resources used by any such identifiers currently in use are released. Note, this routine does not release the locators created by IRQ_NEW or IRQ_FIND. IRQ_RLSE must be called to release these locators.

This routine attempts to execute even if STATUS is bad on entry, although no further error report will be made if it subsequently fails under these circumstances.

Invocation:

```
CALL IRQ_CLOSE( STATUS )
```

Arguments:

STATUS = INTEGER (Given and Returned)

The global status.

IRQ_CNTQ
Count the number of pixels which are set in each bit-plane of the QUALITY component

Description:

Each bit plane of the NDF QUALITY component corresponds to a different quality, described by a name stored in the quality names information structure in an NDF extension. A pixel is set in a bit plane of the QUALITY component if the pixel has the quality associated with the bit plane. This routine counts the number of such pixels in each bit plane.

Note, write or update access must be available for the NDF (as set up by routine LPG_ASSOC for instance), and the QUALITY component of the NDF must not be mapped on entry to this routine.

Invocation:

```
CALL IRQ_CNTQ( LOCS, SIZE, SET, STATUS )
```

Arguments:

LOCS(5) = CHARACTER * (*) (Given)

An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

SIZE = INTEGER (Given)

The number of bit planes for which a count of set pixels is required.

SET(SIZE) = INTEGER (Returned)

The number of pixels holding the corresponding quality in each of bit planes 1 to SIZE. The least-significant bit is Bit 1. If SIZE is larger than the number of bit planes in the QUALITY component, the unused elements are set to zero.

STATUS = INTEGER (Given and Returned)

The global status.

IRQ_CNTQ8
Count the number of pixels which are set in each bit-plane of the QUALITY component

Description:

This routine is equivalent to IRQ_CNTQ except that variable SET, is stored in an INTEGER∗8 array rather than INTEGER.

Each bit plane of the NDF QUALITY component corresponds to a different quality, described by a name stored in the quality names information structure in an NDF extension. A pixel is set in a bit plane of the QUALITY component if the pixel has the quality associated with the bit plane. This routine counts the number of such pixels in each bit plane.

Note, write or update access must be available for the NDF (as set up by routine LPG_ASSOC for instance), and the QUALITY component of the NDF must not be mapped on entry to this routine.

Invocation:

```
CALL IRQ_CNTQ8( LOCS, SIZE, SET, STATUS )
```

Arguments:

LOCS(5) = CHARACTER ∗ (∗) (Given)
An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

SIZE = INTEGER (Given)
The number of bit planes for which a count of set pixels is required.

SET(SIZE) = INTEGER∗8 (Returned)
The number of pixels holding the corresponding quality in each of bit planes 1 to SIZE. The least-significant bit is Bit 1. If SIZE is larger than the number of bit planes in the QUALITY component, the unused elements are set to zero.

STATUS = INTEGER (Given and Returned)
The global status.

IRQ_COMP
Compile a quality expression

Description:

All the quality names referenced in the given quality expression (QEXP) are identified. If all quality names referenced in QEXP are defined within the NDF specified in LOCS, then the quality expression is 'compiled', i.e. converted into a form that can be used by IRQ_SBADx. The compiled quality expression is identified by the returned IRQ identifier which should be released using IRQ_ANNUL when no longer needed. If any error is reported, then IRQ is returned set to the value IRQ__NOID.

If any quality names referenced in the quality expression are not defined in the NDF specified by LOCS, they are returned in UNDEF, the number of such undefined quality names is returned in NUNDEF, an error is reported and STATUS is returned with value IRQ__NOQNM. Additionally, if INFO is true, then a message is generated identifying each undefined quality name.

If any of the STATUS values IRQ__BADSY, IRQ__MSOPT or IRQ__MSOPD are returned (all of which correspond to various forms of syntax error in the quality expression, see ID6 appendix E), a pointer to the approximate position of the error within the quality expression is returned in ERRPNT.

Invocation:

```
CALL IRQ_COMP( LOCS, SIZE, INFO, QEXP, UNDEF, NUNDEF, ERRPNT, IDQ, STATUS )
```

Arguments:

LOCS(5) = CHARACTER * (*) (Given)

An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

SIZE = INTEGER (Given)

The size of the UNDEF array. This should be at least equal to the value of the symbolic constant IRQ__QNREF

INFO = LOGICAL (Given)

If set to .TRUE., then messages are produced identifying any undefined quality names.

QEXP = CHARACTER*(*) (Given and Returned)

A quality expression. See ID6 section 5 for details of the allowed formats for quality expressions. On exit, the string is converted to upper case and any leading blanks are removed.

UNDEF(SIZE) = CHARACTER * (*) (Returned)

An array holding any undefined quality names referenced in the quality expression. The array should have at least IRQ__QNREF elements, each element being a string of length IRQ__SZQNM.

NUNDEF = INTEGER (Returned)
The number of undefined quality names referenced in the quality expression.

ERRPNT = INTEGER (Returned)
If any of the STATUS values IRQ__BADSY(" Unrecognised logical operator or constant"), IRQ__MSOPT (" Missing operator") or IRQ__MSOPD (" Missing operand") are returned, then ERRPNT returns the offset within the quality expression at which the error was detected. Note, the offset refers to the returned form of QEXP, not the given form. These will be different if the given form of QEXP has any leading blanks. An offset of zero is returned if none of the errors associated with the above STATUS values occur.

IDQ = INTEGER (Returned)
An IRQ identifier for the compiled quality expression. This identifier can be passed to IRQ_SBADx. This identifier should be annulled using routine IRQ_ANNUL or IRQ_CLOSE when it is no longer needed. If an error is reported, then an invalid identifier (equal to IRQ__NOID) is returned.

STATUS = INTEGER (Given and Returned)
The global status.

IRQ_DELET
Delete all quality-name information from an NDF

Description:

A search is made through the extensions contained within the supplied NDF for an HDS structure containing quality-name information. If found, the QUALITY_NAMES structure containing the quality names is deleted.

Invocation:

```
CALL IRQ_DELET( INDF, STATUS )
```

Arguments:

INDF = INTEGER (Given)
The input NDF.

STATUS = INTEGER (Given and Returned)
The global status.

IRQ_FIND
Find quality name information within an NDF

Description:

A search is made through the extensions contained within the supplied NDF for an HDS structure containing quality name information. Such information is held in an HDS object named QUALITY_NAMES (these objects can be created using IRQ_NEW). If no such object is found, an error is reported and the status IRQ__NOQNI is returned (if more than one such object is found, an error is reported and the status IRQ__MULT is returned). The name of the NDF extension in which the object was found is returned in XNAME. An array of five HDS locators is returned which is needed when calling other IRQ routines. The first locator points to a temporary object which holds a cloned identifier for the NDF, the other four point to components of the QUALITY_NAMES structure contained in the NDF. IRQ_RLSE should be called to annul these locators (and the NDF identifier) when no further access to the NDFs quality names information is required.

The LOCS argument returned by this routine specifies the NDF which will be operated on by subsequent IRQ routines. Specifically, LOCS determines the bounds of the NDF. Care should therefore be taken that subsequent calls to IRQ routines refer to the NDF specified by the INDF argument to this routine, and not for instance to a section of the NDF which will in general have different bounds.

Invocation:

```
CALL IRQ_FIND( INDF, LOCS, XNAME, STATUS )
```

Arguments:

INDF = INTEGER (Given)
The input NDF.

LOCS(5) = CHARACTER * (*) (Returned)
A set of HDS locators as described above. The character variables supplied for this argument should have a declared length equal to symbolic constant DAT__SZLOC. These locator are annuled by calling IRQ_RLSE.

XNAME = CHARACTER * (*) (Returned)
The name of the NDF extension in which the quality name information was found. The character variable supplied for this argument should have a declared length equal to symbolic constant DAT__SZNAM.

STATUS = INTEGER (Given and Returned)
The global status.

IRQ_FXBIT
Assign a fixed bit number to a quality name

Description:

This routine associates a fixed bit number with a specified quality name. Normally, IRQ manages the allocation of bit numbers to named qualities, but this routine allows the calling application to specify which bit is to be used for a given quality.

By default, a QUALITY-array bit is associated with a quality name only if some pixels hold the quality and some do not hold the quality (i.e. there is a mix of values). Otherwise, a flag is stored in the QUALITY_NAMES structure indicating this, and any quality bit previously associated with the quality name is released for re-use.

This default behaviour is changed by calling this routine. The specified bit number will continue to be associated with the quality name even if all pixels do, or do not, hold the quality.

An error will be returned if the named quality is already associated with a different bit number when this routine is called. An error will also be reported if the specified bit number is already associated with a different quality name.

Invocation:

```
CALL IRQ_FXBIT( LOCS, QNAME, BIT, SET, FIXBIT, STATUS )
```

Arguments:

LOCS(5) = CHARACTER * (*) (Given)

An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

QNAME = CHARACTER * (*) (Given)

The quality name to use. Leading blanks are ignored and the search is case-insensitive. The maximum allowed length for quality names is given by symbolic constant IRQ__SZQNM which currently has the value of 15.

BIT = INTEGER (Given)

The bit number to use. The least significant bit is Bit 1, not Bit 0. If a value below 0 or above 8 is supplied, the properties of the quality name are left unchanged, but the FIXBIT value is still returned.

FIXBIT = LOGICAL (Returned)

Returned .TRUE. if the specified quality name had a fixed bit number on entry to this routine, and .FALSE. otherwise.

STATUS = INTEGER (Given and Returned)

The global status.

IRQ_GETQN
Search for a specified quality name

Description:

This routine searches for a specified quality name in the quality name specified by LOCS, and returns information related to the quality name. If the quality name is not defined then an error is reported and STATUS returned equal to IRQ__NOQNM.

Invocation:

```
CALL IRQ_GETQN( LOCS, QNAME, FIXED, VALUE, BIT, COMMNT, STATUS )
```

Arguments:

LOCS(5) = CHARACTER * (*) (Given)

An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

QNAME = CHARACTER * (*) (Given)

The quality name to search for. Leading blanks are ignored and the search is case-insensitive. The maximum allowed length for quality names is given by symbolic constant IRQ__SZQNM which currently has the value of 15.

FIXED = LOGICAL (Returned)

If true, then the quality is either held by all pixels, or by no pixels. In this case the quality may not have a corresponding bit in the QUALITY component. If false, then some pixels have the quality and some do not, as indicated by the corresponding bit in the QUALITY component.

VALUE = LOGICAL (Returned)

If FIXED is true, then VALUE specifies whether all pixels hold the quality (VALUE = .TRUE.), or whether no pixels hold the quality (VALUE = .FALSE.). If FIXED is false, then VALUE is indeterminate.

BIT = INTEGER (Returned)

BIT holds the corresponding bit number in the QUALITY component. The least-significant bit is called Bit 1 (not Bit 0). If there is no corresponding bit, a value of zero is returned, and FIXED is returned .TRUE.

COMMNT = CHARACTER * (*) (Returned)

The descriptive comment which was stored with the quality name. The supplied character variable should have a declared length given by symbolic constant IRQ__SZCOM.

STATUS = INTEGER (Given and Returned)

The global status.

IRQ_GETQX
Get a quality expression from the user and check for syntax errors

Description:

A string is obtained from the environment using the supplied ADAM parameter. This string is check to see if it has correct syntax for a quality expression (no checks are made to ensure that the quality names referenced within the expression are defined within any specific NDF). If a syntax error is detected, the quality expression is displayed with an exclamation mark under the position at which the syntax error was detected. If any problem is found with the supplied expression, a new value is obtained from the environment. The returned string is converted to upper case and leading blanks are removed.

Invocation:

```
CALL IRQ_GETQX( PARAM, QEXP, STATUS )
```

Arguments:

PARAM = CHARACTER * (*) (Given)
The name of an ADAM parameter of type LITERAL.

QEXP = CHARACTER * (*) (Returned)
The returned quality expression.

STATUS = INTEGER (Given and Returned)
The global status.

IRQ_NEW
Create a new structure to hold quality name information within an NDF extension

Description:

An HDS object (named QUALITY_NAMES) is created to hold quality name information within the specified NDF extension. An error is reported if the NDF extension does not exist. If the extension does exist, an array of five HDS locators is returned which is needed when calling other IRQ routines. The first locator points to a temporary object which holds a cloned identifier for the NDF, the other four point to components of the QUALITY_NAMES structure contained in the NDF. IRQ_RLSE should be called to annul these locators (and the NDF identifier) when no further access to the NDFs quality names information is required.

The QUALITY component of the NDF is reset to an undefined state by this routine. Therefore, the QUALITY component should not be mapped for access prior to calling this routine.

The LOCS argument returned by this routine specifies the NDF which will be operated on by subsequent IRQ routines. Specifically, LOCS determines the bounds of the NDF. Care should therefore be taken that subsequent calls to IRQ routines refer to the NDF specified by the INDF argument to this routine, and not for instance to a section of the NDF which will in general have different bounds.

Note, an error is reported if only READ access is available to the NDF.

Invocation:

```
CALL IRQ_NEW( INDF, XNAME, LOCS, STATUS )
```

Arguments:

INDF = INTEGER (Given)
The NDF identifier.

XNAME = CHARACTER * (*) (Given)
The name of the NDF extension in which the quality name information is to be stored. If this extension does not exist then an error is reported.

LOCS(5) = CHARACTER * (*) (Returned)
A set of HDS locators as described above. The character variables supplied for this argument should have a declared length equal to symbolic constant DAT__SZLOC. These locator are annulled by calling IRQ_RLSE.

STATUS = INTEGER (Given and Returned)
The global status.

IRQ_NUMQN
Return number of defined quality names

Description:

The number of quality names defined in the NDF specified by LOCS is returned.

Invocation:

```
CALL IRQ_NUMQN( LOCS, NAMES, STATUS )
```

Arguments:

LOCS(5) = CHARACTER * (*) (Given)

An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

NAMES = INTEGER (Returned)

The number of quality names defined in the structure located by LOCS.

STATUS = INTEGER (Given and Returned)

The global status.

IRQ_NXTQN
Return the next quality name

Description:

This routine returns the next quality name defined in the NDF specified by LOCS, together with supplementary information. The next quality name is determined by the value of CONTXT. If CONTXT is zero on entry then the first quality name is returned. On exit, CONTXT is set to a value which indicates where the next quality name is stored within the NDF. This value can be passed to a subsequent call to this routine to retrieve information about the next quality name.

Invocation:

```
CALL IRQ_NXTQN( LOCS, CONTXT, QNAME, FIXED, VALUE, BIT, COMMNT, DONE, STATUS
)
```

Arguments:

LOCS(5) = CHARACTER * (*) (Given)

An array of 5 HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

CONTXT = INTEGER (Given and Returned)

The context of the current call. This should be set to zero before the first call to this routine, and then left unchanged between subsequent calls.

QNAME = CHARACTER * (*) (Returned)

The next quality name. The character variable supplied for this argument should have a declared length equal to the symbolic constant IRQ__SZQNM.

FIXED = LOGICAL (Returned)

If true, then the quality is either held by all pixels, or by no pixels. In this case the quality may not have a corresponding bit in the QUALITY component. If false, then some pixels have the quality and some do not, as indicated by the corresponding bit in the QUALITY component.

VALUE = LOGICAL (Returned)

If FIXED is true, then VALUE specifies whether all pixels hold the quality (VALUE = .TRUE.), or whether no pixels hold the quality (VALUE = .FALSE.). If FIXED is false, then VALUE is indeterminate.

BIT = INTEGER (Returned)

BIT holds the corresponding bit number in the QUALITY component. The least-significant bit is called Bit 1 (not Bit 0). A value of zero is returned if the quality has no associated bit in the quality array. In this case, the FIXED argument will indicate if all pixels do, or do not, hold the quality.

COMMNT = CHARACTER ∗ (∗) (Returned)
The descriptive comment which was stored with the quality name. The supplied character variable should have a declared length given by symbolic constant IRQ__SZCOM.

DONE = LOGICAL (Returned)
Returned true if this routine is called when no more names remain to be returned.

STATUS = INTEGER (Given and Returned)
The global status.

IRQ_RBIT
Reserve a bit number for a given quality name

Description:

If the supplied quality name already has a bit number associated with it, the bit number is returned. Otherwise, the next available plane in the quality array is asigned to the quality name, and its bit number is returned.

Note, write or update access must be available for the NDF (as set up by routine LPG_ASSOC for instance).

Invocation:

```
CALL IRQ_RBIT( LOCS, QNAME, BIT, STATUS )
```

Arguments:

LOCS(5) = CHARACTER * (*) (Given)

An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

QNAME = CHARACTER * (*) (Given)

The quality name. This quality name must be defined in the NDF specified by LOCS. Name definitions can be added to the NDF using routine IRQ_ADDQN.

BIT = INTEGER (Returned)

The bit number used by the quality name within the quality array. Note, the least-significant bit is Bit 1, not Bit 0.

STATUS = INTEGER (Given and Returned)

The global status.

IRQ_REMQN
Remove the definition of a specified quality name

Description:

The specified quality name is removed from the NDF specified by LOCS. Any associated bit in the QUALITY array is freed for future use. If the name is not defined an error is reported. A value of ANY for the quality names causes all defined quality names to be removed.

Note, an error is reported if only read access is available to the NDF, or if the quality name has been flagged as read-only using routine IRQ_RWQN.

Invocation:

```
CALL IRQ_REMQN( LOCS, QNAME, STATUS )
```

Arguments:

LOCS(5) = CHARACTER * (*) (Given)

An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

QNAME = CHARACTER * (*) (Given)

The quality name to remove, or 'ANY' if all quality names are to be removed.

STATUS = INTEGER (Given and Returned)

The global status.

IRQ_RESQ
Remove a given quality from all pixels in the NDF

Description:

The quality specified by QNAME is removed from all pixels in the NDF specified by LOCS (LOCS should be obtained either by calling IRQ_FIND or IRQ_NEW). An error is reported if the quality name is undefined within the NDF.

Note, write or update access must be available for the NDF (as set up by routine LPG_ASSOC for instance).

Invocation:

```
CALL IRQ_RESQ( LOCS, QNAME, STATUS )
```

Arguments:

LOCS(5) = CHARACTER * (*) (Given)

An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

QNAME = CHARACTER * (*) (Given)

The quality name to be removed from all pixels in the NDF. This quality name must be defined in the NDF specified by LOC. Name definitions can be added to the NDF using routine IRQ_ADDQN.

STATUS = INTEGER (Given and Returned)

The global status.

IRQ_RESQL
Remove a quality from a list of pixels, leaving unlisted pixels unchanged

Description:

The quality specified by QNAME is removed from all pixels included in (or, if LISTED is false, not included in) the supplied list of pixel indices. The quality of other pixels is left unaltered. The quality name must be defined in the NDF specified by LOCS (LOCS should be obtained either by calling IRQ_FIND or IRQ_NEW). An error is reported if the quality name is undefined.

Note, write or update access must be available for the NDF (as set up by routine LPG_ASSOC for instance), and the QUALITY component must not be mapped on entry to this routine.

Invocation:

```
CALL IRQ_RESQL( LOCS, LISTED, QNAME, NDIM, NCOORD, LIST, SET, STATUS )
```

Arguments:

LOCS(5) = CHARACTER * (*) (Given)

An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

LISTED = LOGICAL (Given)

If true, then the quality is removed from all pixels included in the list given by LIST. If false, then the quality is removed from all pixels not included in the list given by LIST.

QNAME = CHARACTER * (*) (Given)

The quality name to be removed from the selected pixels. This quality name must be defined in the NDF specified by LOC. Name definitions can be added to the NDF using routine IRQ_ADDQN.

NDIM = INTEGER (Given)

The number of values required to specify a pixel position (i.e. the number of dimensions in the NDF).

NCOORD = INTEGER (Given)

The number of pixels included in the input list.

LIST(NDIM, NCOORD) = INTEGER (Given)

The list of pixel indices. Any indices which lie outside the bounds of the NDF are ignored.

SET = INTEGER (Returned)

The number of pixels which hold the quality.

STATUS = INTEGER (Given and Returned)

The global status.

IRQ_RESQL8
Remove a quality from a list of pixels, leaving unlisted pixels unchanged

Description:

This routine is equivalent to IRQ_RESQL except that variables LIST, NCOORD and SET are stored in INTEGER∗8 values rather than INTEGER.

The quality specified by QNAME is removed from all pixels included in (or, if LISTED is false, not included in) the supplied list of pixel indices. The quality of other pixels is left unaltered. The quality name must be defined in the NDF specified by LOCS (LOCS should be obtained either by calling IRQ_FIND or IRQ_NEW). An error is reported if the quality name is undefined.

Note, write or update access must be available for the NDF (as set up by routine LPG_ASSOC for instance), and the QUALITY component must not be mapped on entry to this routine.

Invocation:

```
CALL IRQ_RESQL8( LOCS, LISTED, QNAME, NDIM, NCOORD, LIST, SET, STATUS )
```

Arguments:

LOCS(5) = CHARACTER ∗ (∗) (Given)
An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

LISTED = LOGICAL (Given)
If true, then the quality is removed from all pixels included in the list given by LIST. If false, then the quality is removed from all pixels not included in the list given by LIST.

QNAME = CHARACTER ∗ (∗) (Given)
The quality name to be removed from the selected pixels. This quality name must be defined in the NDF specified by LOC. Name definitions can be added to the NDF using routine IRQ_ADDQN.

NDIM = INTEGER (Given)
The number of values required to specify a pixel position (i.e. the number of dimensions in the NDF).

NCOORD = INTEGER∗8 (Given)
The number of pixels included in the input list.

LIST(NDIM, NCOORD) = INTEGER∗8 (Given)
The list of pixel indices. Any indices which lie outside the bounds of the NDF are ignored.

SET = INTEGER∗8 (Returned)
The number of pixels which hold the quality.

STATUS = INTEGER (Given and Returned)
The global status.

IRQ_RESQM
Remove a quality from pixels selected using a mask image, leaving unselected pixels unchanged

Description:

The quality specified by QNAME is removed from all NDF pixels which either do (or, if BAD is false, do not) correspond to 'bad' pixels in the input mask array. The quality of all other pixels is left unchanged. The quality name must be defined in the NDF specified by LOCS (LOCS should be obtained either by calling IRQ_FIND or IRQ_NEW). An error is reported if the quality name is undefined.

Note, write or update access must be available for the NDF (as set up by routine LPG_ASSOC for instance), and the QUALITY component of the NDF must not be mapped on entry to this routine.

Invocation:

```
CALL IRQ_RESQM( LOCS, BAD, QNAME, SIZE, MASK, SET, STATUS )
```

Arguments:

LOCS(5) = CHARACTER * (*) (Given)

An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

BAD = LOGICAL (Given)

If true, then the quality is removed from all NDF pixels corresponding to 'bad' pixels in the mask. If false, then the quality is removed from all NDF pixels corresponding to pixels which are not 'bad' in the mask.

QNAME = CHARACTER * (*) (Given)

The quality name to be removed from the selected pixels. This quality name must be defined in the NDF specified by LOC. Name definitions can be added to the NDF using routine IRQ_ADDQN.

SIZE = INTEGER (Given)

The total number of pixels in the MASK array.

MASK(SIZE) = REAL (Given)

A vector which defines the pixels from which the quality specified by QNAME is to be removed. It is assumed that this vector corresponds pixel-for-pixel with the vectorised NDF as supplied to routine IRQ_FIND or IRQ_NEW.

SET = INTEGER (Returned)

The number of pixels in the NDF which hold the quality.

STATUS = INTEGER (Given and Returned)

The global status.

IRQ_RESQM8
Remove a quality from pixels selected using a mask image, leaving unselected pixels unchanged

Description:

This routine is equivalent to IRQ_RESQM except that arguments SET and SIZE are stored in INTEGER∗8 variables instead of INTEGER.

The quality specified by QNAME is removed from all NDF pixels which either do (or, if BAD is false, do not) correspond to 'bad' pixels in the input mask array. The quality of all other pixels is left unchanged. The quality name must be defined in the NDF specified by LOCS (LOCS should be obtained either by calling IRQ_FIND or IRQ_NEW). An error is reported if the quality name is undefined.

Note, write or update access must be available for the NDF (as set up by routine LPG_ASSOC for instance), and the QUALITY component of the NDF must not be mapped on entry to this routine.

Invocation:

```
CALL IRQ_RESQM8( LOCS, BAD, QNAME, SIZE, MASK, SET, STATUS )
```

Arguments:

LOCS(5) = CHARACTER ∗ (∗) (Given)
An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

BAD = LOGICAL (Given)
If true, then the quality is removed from all NDF pixels corresponding to 'bad' pixels in the mask. If false, then the quality is removed from all NDF pixels corresponding to pixels which are not 'bad' in the mask.

QNAME = CHARACTER ∗ (∗) (Given)
The quality name to be removed from the selected pixels. This quality name must be defined in the NDF specified by LOC. Name definitions can be added to the NDF using routine IRQ_ADDQN.

SIZE = INTEGER∗8 (Given)
The total number of pixels in the MASK array.

MASK(SIZE) = REAL (Given)
A vector which defines the pixels from which the quality specified by QNAME is to be removed. It is assumed that this vector corresponds pixel-for-pixel with the vectorised NDF as supplied to routine IRQ_FIND or IRQ_NEW.

SET = INTEGER∗8 (Returned)
The number of pixels in the NDF which hold the quality.

STATUS = INTEGER (Given and Returned)
The global status.

IRQ_RLSE
Release a temporary structure created by IRQ_NEW or IRQ_FIND

Description:

This routine releases the resources reserved by a call to IRQ_NEW or IRQ_FIND. The cloned NDF identifier held in LOCS(1) is annulled, and then all the five HDS locators in LOCS are annulled. If no defined quality names exist within the NDF, then the structure used to hold such names is deleted and the QUALITY component of the NDF is reset to an undefined state.

Note, this routine attempts to execute even if STATUS is set on entry, although no further error report will be made if it subsequently fails under these circumstances.

Invocation:

```
CALL IRQ_RLSE( LOCS, STATUS )
```

Arguments:

LOCS(5) = CHARACTER * (*) (Given)

An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

STATUS = INTEGER (Given and Returned)

The global status.

IRQ_RWQN
Get and/or set the read-only flag for a quality name

Description:

This routine returns the current value of the read-only flag associated with a quality name, and optionally assigns a new value to the flag.

If the read-only flag is set for a quality name, any attempt to remove the quality name using IRQ_REMQN will result in an error being reported.

Invocation:

```
CALL IRQ_RWQN( LOCS, QNAME, SET, NEWVAL, OLDVAL, STATUS )
```

Arguments:

LOCS(5) = CHARACTER * (*) (Given)

An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

QNAME = CHARACTER * (*) (Given)

The quality name to use. Leading blanks are ignored and the search is case-insensitive. The maximum allowed length for quality names is given by symbolic constant IRQ__SZQNM which currently has the value of 15.

SET = LOGICAL (Given)

If true, then the read-only flag for the quality name will be set to the value supplied in NEWVAL. Otherwise, the current value of the flag will be left unchanged.

NEWVAL = LOGICAL (Given)

The new value for the read-only flag. Only accessed if SET is true.

OLDVAL = LOGICAL (Returned)

The value of the read-only flag on entry to this routine. If the old value is of no interest, it is safe to supply the same variable for OLDVAL as for NEWVAL since OLDVAL is updated after NEWVAL is used.

STATUS = INTEGER (Given and Returned)

The global status.

IRQ_SBADx
Set pixels 'bad' which satisfy a given quality expression

Description:

IRQ_COMP should be called before this routine to produce the compiled quality expression identified by IDQ. The QUALITY component of the NDF to which the quality expression refers (see IRQ_COMP argument LOCS) is mapped as a one-dimensional vector. The supplied array VEC must correspond pixel-for-pixel with the mapped QUALITY vector. All pixels which hold a QUALITY satisfying the quality expression are found. If HELD is true, then the corresponding pixels in VEC are set to the 'bad' value (other pixels are left unaltered). If HELD is false, the corresponding pixels in VEC are left as they are, but all the other pixels in VEC are set to the 'bad' value. ALLBAD and NOBAD indicate if the output VEC values are either all bad or all good.

Note, if the QUALITY component of the NDF is mapped for WRITE or UPDATE access on entry to this routine, an error is reported.

Invocation:

```
CALL IRQ_SBADx( IDQ, HELD, SIZE, VEC, ALLBAD, NOBAD, STATUS )
```

Arguments:

IDQ = INTEGER (Given)

An identifier for a compiled quality expression, produced by routine IRQ_COMP. This identifier determines the NDF to which the expression refers.

HELD = LOGICAL (Given)

If true then those VEC pixels which hold a quality satisfying the supplied quality expression are set 'bad'. Otherwise, those pixels which don't hold such a quality are set 'bad'.

SIZE = INTEGER (Given)

The total number of pixels in VEC. An error is reported if this is not the same as the total number of pixels in the NDF determined by IDQ.

VEC(SIZE) = ? (Given and Returned)

The data to be set 'bad', depending on the corresponding quality values stored in the NDF. It must be the same size as the NDF, and must correspond pixel-for-pixel with the vectorised NDF. Pixels which are not explicitly set 'bad' by this routine retain the values they had on entry.

ALLBAD = LOGICAL (Returned)

Returned true if all pixels in VEC are returned with 'bad' values, and false if any returned pixel values are not 'bad'.

NOBAD = LOGICAL (Returned)

Returned true if no pixels in VEC are returned with 'bad' values. False if any 'bad' pixel values are returned.

STATUS = INTEGER (Given and Returned)
The global status.

Notes:

- There is a routine for each numeric data type: replace "`x`" in the routine name by D, R, I, W, UW, B or UB as appropriate. The VEC array supplied to the routine must have the data type specified.

IRQ_SBADx
Set pixels 'bad' which satisfy a given quality expression

Description:

This function is equivalent to IRQ_SBADx except that argument SIZE is held in an INTEGER88 instead of a 4-byte INTEGER.

IRQ_COMP should be called before this routine to produce the compiled quality expression identified by IDQ. The QUALITY component of the NDF to which the quality expression refers (see IRQ_COMP argument LOCS) is mapped as a one-dimensional vector. The supplied array VEC must correspond pixel-for-pixel with the mapped QUALITY vector. All pixels which hold a QUALITY satisfying the quality expression are found. If HELD is true, then the corresponding pixels in VEC are set to the 'bad' value (other pixels are left unaltered). If HELD is false, the corresponding pixels in VEC are left as they are, but all the other pixels in VEC are set to the 'bad' value. ALLBAD and NOBAD indicate if the output VEC values are either all bad or all good.

Note, if the QUALITY component of the NDF is mapped for WRITE or UPDATE access on entry to this routine, an error is reported.

Invocation:

```
CALL IRQ_SBAD8x( IDQ, HELD, SIZE, VEC, ALLBAD, NOBAD, STATUS )
```

Arguments:

IDQ = INTEGER (Given)

An identifier for a compiled quality expression, produced by routine IRQ_COMP. This identifier determines the NDF to which the expression refers.

HELD = LOGICAL (Given)

If true then those VEC pixels which hold a quality satisfying the supplied quality expression are set 'bad'. Otherwise, those pixels which don't hold such a quality are set 'bad'.

SIZE = INTEGER*8 (Given)

The total number of pixels in VEC. An error is reported if this is not the same as the total number of pixels in the NDF determined by IDQ.

VEC(SIZE) = ? (Given and Returned)

The data to be set 'bad', depending on the corresponding quality values stored in the NDF. It must be the same size as the NDF, and must correspond pixel-for-pixel with the vectorised NDF. Pixels which are not explicitly set 'bad' by this routine retain the values they had on entry.

ALLBAD = LOGICAL (Returned)

Returned true if all pixels in VEC are returned with 'bad' values, and false if any returned pixel values are not 'bad'.

NOBAD = LOGICAL (Returned)
Returned true if no pixels in VEC are returned with 'bad' values. False if any 'bad' pixel values are returned.

STATUS = INTEGER (Given and Returned)
The global status.

Notes:

- There is a routine for each numeric data type: replace "x" in the routine name by D, R, I, W, UW, B or UB as appropriate. The VEC array supplied to the routine must have the data type specified.

IRQ_SETQ
Assign a given quality to all pixels in the NDF

Description:

The quality specified by QNAME is assigned to all pixels in the NDF specified by LOCS (LOCS should be obtained either by calling IRQ_FIND or IRQ_NEW). An error is reported if the quality name is undefined within the NDF.

Note, write or update access must be available for the NDF (as set up by routine LPG_ASSOC for instance).

Invocation:

```
CALL IRQ_SETQ( LOCS, QNAME, STATUS )
```

Arguments:

LOCS(5) = CHARACTER * (*) (Given)

An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

QNAME = CHARACTER * (*) (Given)

The quality name to assign to all pixels in the NDF. This quality name must be defined in the NDF specified by LOC. Name definitions can be added to the NDF using routine IRQ_ADDQN.

STATUS = INTEGER (Given and Returned)

The global status.

IRQ_SETQL
Assign a given quality to a list of pixels, leaving unlisted pixels unchanged

Description:

The quality specified by QNAME is assigned to all pixels included (or, if LISTED is false, not included) in the supplied list of pixel indices. The quality of other pixels is left unaltered. The quality name must be defined in the NDF specified by LOCS (LOCS should be obtained either by calling IRQ_FIND or IRQ_NEW). An error is reported if the quality name is undefined.

Note, write or update access must be available for the NDF (as set up by routine LPG_ASSOC for instance), and the QUALITY component must not be mapped on entry to this routine.

Invocation:

```
CALL IRQ_SETQL( LOCS, LISTED, QNAME, NDIM, NCOORD, LIST, SET, STATUS )
```

Arguments:

LOCS(5) = CHARACTER * (*) (Given)

An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

LISTED = LOGICAL (Given)

If true, then the quality is assigned to all pixels included in the list given by LIST. If false, then the quality is assigned to all pixels not included in the list given by LIST.

QNAME = CHARACTER * (*) (Given)

The quality name to assign to the selected pixels. This quality name must be defined in the NDF specified by LOC. Name definitions can be added to the NDF using routine IRQ_ADDQN.

NDIM = INTEGER (Given)

The number of values required to specify a pixel position (i.e. the number of dimensions in the NDF).

NCOORD = INTEGER (Given)

The number of pixels included in the input list.

LIST(NDIM, NCOORD) = INTEGER (Given)

The list of pixel indices. Any indices which lie outside the bounds of the NDF are ignored.

SET = INTEGER (Returned)

The number of pixels in the NDF which hold the quality.

STATUS = INTEGER (Given and Returned)

The global status.

IRQ_SETQL8
Assign a given quality to a list of pixels, leaving unlisted pixels unchanged

Description:

This routine is equivalent to IRQ_SETQL except that variables LIST, NCOORD and SET are stored in INTEGER∗8 values rather than INTEGER.

The quality specified by QNAME is assigned to all pixels included (or, if LISTED is false, not included) in the supplied list of pixel indices. The quality of other pixels is left unaltered. The quality name must be defined in the NDF specified by LOCS (LOCS should be obtained either by calling IRQ_FIND or IRQ_NEW). An error is reported if the quality name is undefined.

Note, write or update access must be available for the NDF (as set up by routine LPG_ASSOC for instance), and the QUALITY component must not be mapped on entry to this routine.

Invocation:

```
CALL IRQ_SETQL8( LOCS, LISTED, QNAME, NDIM, NCOORD, LIST, SET, STATUS )
```

Arguments:

LOCS(5) = CHARACTER ∗ (∗) (Given)
An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

LISTED = LOGICAL (Given)
If true, then the quality is assigned to all pixels included in the list given by LIST. If false, then the quality is assigned to all pixels not included in the list given by LIST.

QNAME = CHARACTER ∗ (∗) (Given)
The quality name to assign to the selected pixels. This quality name must be defined in the NDF specified by LOC. Name definitions can be added to the NDF using routine IRQ_ADDQN.

NDIM = INTEGER (Given)
The number of values required to specify a pixel position (i.e. the number of dimensions in the NDF).

NCOORD = INTEGER∗8 (Given)
The number of pixels included in the input list.

LIST(NDIM, NCOORD) = INTEGER∗8 (Given)
The list of pixel indices. Any indices which lie outside the bounds of the NDF are ignored.

SET = INTEGER∗8 (Returned)
The number of pixels in the NDF which hold the quality.

STATUS = INTEGER (Given and Returned)
The global status.

IRQ_SETQM
Assign a quality to pixels selected using a mask image, leaving unselected pixels unchanged

Description:

The quality specified by QNAME is assigned to all NDF pixels which either do (or, if BAD is false, do not) correspond to 'bad' pixels in the input mask array. The quality of all other pixels is left unchanged. The quality name must be defined in the NDF specified by LOCS (LOCS should be obtained either by calling IRQ_FIND or IRQ_NEW). An error is reported if the quality name is undefined.

Note, write or update access must be available for the NDF (as set up by routine LPG_ASSOC for instance), and the QUALITY component of the NDF must not be mapped on entry to this routine.

Invocation:

```
CALL IRQ_SETQM( LOCS, BAD, QNAME, SIZE, MASK, SET, STATUS )
```

Arguments:

LOCS(5) = CHARACTER * (*) (Given)

An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

BAD = LOGICAL (Given)

If true, then the quality is assigned to all NDF pixels corresponding to 'bad' pixels in the mask. If false, then the quality is assigned to all NDF pixels corresponding to pixels which are not 'bad' in the mask.

QNAME = CHARACTER * (*) (Given)

The quality name to assign to the selected pixels. This quality name must be defined in the NDF specified by LOC. Name definitions can be added to the NDF using routine IRQ_ADDQN.

SIZE = INTEGER (Given)

The total number of pixels in the MASK array.

MASK(SIZE) = REAL (Given)

A vector which defines the pixels to which the quality specified by QNAME is to be assigned. It is assumed that this vector corresponds pixel-for-pixel with the vectorised NDF supplied to IRQ_NEW or IRQ_FIND.

SET = INTEGER (Returned)

The number of pixels in the NDF which hold the quality.

STATUS = INTEGER (Given and Returned)

The global status.

IRQ_SETQM8
Assign a quality to pixels selected using a mask image, leaving unselected pixels unchanged

Description:

This routine is equivalent to IRQ_SETQM except that arguments SET and SIZE are stored in INTEGER∗8 variables instead of INTEGER.

The quality specified by QNAME is assigned to all NDF pixels which either do (or, if BAD is false, do not) correspond to 'bad' pixels in the input mask array. The quality of all other pixels is left unchanged. The quality name must be defined in the NDF specified by LOCS (LOCS should be obtained either by calling IRQ_FIND or IRQ_NEW). An error is reported if the quality name is undefined.

Note, write or update access must be available for the NDF (as set up by routine LPG_ASSOC for instance), and the QUALITY component of the NDF must not be mapped on entry to this routine.

Invocation:

```
CALL IRQ_SETQM8( LOCS, BAD, QNAME, SIZE, MASK, SET, STATUS )
```

Arguments:

LOCS(5) = CHARACTER ∗ (∗) (Given)
An array of five HDS locators. These locators identify the NDF and the associated quality name information. They should have been obtained using routine IRQ_FIND or routine IRQ_NEW.

BAD = LOGICAL (Given)
If true, then the quality is assigned to all NDF pixels corresponding to 'bad' pixels in the mask. If false, then the quality is assigned to all NDF pixels corresponding to pixels which are not 'bad' in the mask.

QNAME = CHARACTER ∗ (∗) (Given)
The quality name to assign to the selected pixels. This quality name must be defined in the NDF specified by LOC. Name definitions can be added to the NDF using routine IRQ_ADDQN.

SIZE = INTEGER∗8 (Given)
The total number of pixels in the MASK array.

MASK(SIZE) = REAL (Given)
A vector which defines the pixels to which the quality specified by QNAME is to be assigned. It is assumed that this vector corresponds pixel-for-pixel with the vectorised NDF supplied to IRQ_NEW or IRQ_FIND.

SET = INTEGER∗8 (Returned)
The number of pixels in the NDF which hold the quality.

STATUS = INTEGER (Given and Returned)
The global status.

IRQ_SYNTX
Check the syntax of a quality expression

Description:

The syntax of the supplied quality expression is checked, and an error is reported if a syntax error is detected. If any of the STATUS values IRQ__BADSY, IRQ__MSOPT or IRQ__MSOPD are returned (all of which correspond to various forms of syntax error in the quality expression, see ID6 appendix E), a pointer to the approximate position of the error within the quality expression is returned in ERRPNT. Note, in order for a quality expression to compile successfully (using IRQ_COMP), it must not only contain no syntax errors, but must also contain no undefined quality names. IRQ_SYNTX cannot check for undefined quality names.

Invocation:

```
CALL IRQ_SYNTX( QEXP, ERRPNT, STATUS )
```

Arguments:

QEXP = CHARACTER∗(∗) (Given and Returned)

A quality expression. See ID6 section 5 for details of the allowed formats for quality expressions. On exit, the string is converted to upper case and any leading blanks are removed.

ERRPNT = INTEGER (Returned)

If any of the STATUS values IRQ__BADSY(" Unrecognised logical operator or constant"), IRQ__MSOPT (" Missing operator") or IRQ__MSOPD (" Missing operand") are returned, then ERRPNT returns the offset within the quality expression at which the error was detected. Note, the offset refers to the returned form of QEXP, not the given form. These will be different if the given form of QEXP has any leading blanks. An offset of zero is returned if none of the errors associated with the above STATUS values occur.

STATUS = INTEGER (Given and Returned)

The global status.

D HDS Data Structures

The IRQ package uses several different HDS data structures. These are described in this appendix. HDS *names* are indicated by being placed within square brackets ([ ]), and HDS *types* are indicated by being placed within angled brackets (< >).

D.1 Quality names information stored in an NDF

Information describing the quality names which are defined within an NDF is stored in a structure called [**QUALITY_NAMES**], with HDS *type* <**QUALITY_NAMES**>. This structure can be stored in any extension within the NDF. The components of this structure are shown in Table 1.

Table 1: Components of a <**QUALITY_NAMES**> structure

Component Name	TYPE	Brief Description
[**QUAL**]	<**IRQ_QUAL**>	Vector of quality name definitions
[**LAST_USED**]	<**_INTEGER**>	Highest used index within [**QUAL**]
[**NFREE**]	<**_INTEGER**>	No. of un-used cells within [**QUAL**]
[**FREE**(NFREE)]	<**_INTEGER**>	Vector holding indices of un-used cells in [**QUAL**]

The [**QUAL**] component is a vector in which each cell is a structure holding various items of information needed to define a single quality name. The array has an initial size of 8 but is increased if necessary. There will usually be some unused cells within [**QUAL**], and the other components of the [**QUALITY_NAMES**] structure listed in Table 1 are used to locate these unused cells. In particular, [**FREE**] is a vector which holds the indices of all unused cells within [**QUAL**]. These indices may appear in any order within in [**FREE**].

The set of five HDS locators returned by routines IRQ_NEW and IRQ_FIND include locators to each of the four components of the [**QUALITY_NAMES**] structure, together with a locator for a temporary HDS structure holding a single <**_INTEGER**> scalar used to store the cloned NDF identifier.

The information describing each individual quality name is stored in a single cell of the [**QUAL**] vector. The components within each cell of this array are listed in Table 2.

If *all* pixels hold a given quality, or if *no* pixels hold the quality, then a true value is stored for [**FIXED**]. A true value is stored for [**VALUE**] if *all* pixels hold the quality, and a false value if *no* pixels hold the quality. In either case, [**BIT**] is ignored since no QUALITY bit needs to be reserved for the quality name, thus allowing more than eight quality names to be defined simultaneously.

If some pixels *do* hold the quality but some *do not*, then a false value is stored for [**FIXED**] and [**VALUE**] is ignored. In this case, a QUALITY bit is reserved to represent the quality and its bit number (in the range 1 to 8) is stored in [**BIT**].

Table 2: Components of a <**IRQ_QUAL**> structure

Component Name	TYPE	Brief Description
[**NAME**]	<**_CHAR*15**>	A quality name (maximum of
[**FIXED**]	<**_LOGICAL**>	True if all pixels are in the same state
[**VALUE**]	<**_LOGICAL**>	The state of all pixels, if fixed
[**BIT**]	<**_INTEGER**>	QUALITY bit used to store this quality
[**COMMENT**]	<**_CHAR***>	A descriptive comment for the quality

D.2 Temporary structures used to hold compiled quality expressions

When routine IRQ_COMP is called to compile a quality expression, the resulting information (known as a 'compiled quality expression') is stored in a temporary HDS structure. Up to ten compiled quality expressions can exist simultaneously, each being stored in one cell of an array of temporary structures. The identifier returned by IRQ_COMP is just an index within this array. Each cell of the array has an HDS *name* of [**QEXP**] and an HDS *type* of <**QEXP**>, and contains the components listed in Table 3.

Table 3: Components of a <**QEXP**> structure

Component Name	TYPE	Brief Description
[**MASKS**]	<**_INTEGER**>	A vector of bit masks
[**OPCODE**]	<**_INTEGER**>	A vector of instruction codes

Each bit mask held in [**MASKS**] specifies a set of QUALITY bits which are to be tested as part of the evaluation of a quality expression performed by routine IRQ_SBAD. The instruction codes held in [**OPCODE**] represent the operations which must be performed on a "First In - Last Out" stack in order to evaluate a quality expression. The sizes of theses vectors are held in common.

E Examples of Using IRQ

E.1 Adding a new quality name

This example shows a code fragment which adds the quality name SATURATED to an NDF and then assigns the quality to all pixels with value greater than 10.0. An error is reported if the quality name is already in use.

```
*  Include ADAM, IRQ and NDF symbolic constants.
      INCLUDE 'SAE_PAR'
      INCLUDE 'IRQ_PAR'
      INCLUDE 'IRQ_ERR'
      INCLUDE 'NDF_PAR'

*  Declare local INTEGER variables.
      INTEGER NDFIN, NDF2, NDIM, LBND(NDF__MXDIM), NEL,
     :        UBND(NDF__MXDIM), PNT, PNT2, PLACE,
     :        STATUS

*  Declare local LOGICAL variables.
      INTEGER FOUND

*  Declare local CHARACTER variables.
      CHARACTER*(DAT__SZLOC) LOCS(5),XLOC
      CHARACTER*(DAT__SZNAM) XNAME

*  Start an NDF context.
      CALL NDF_BEGIN

*  Obtain an identifier for the input NDF.
      CALL NDF_ASSOC( 'IN', 'READ', NDFIN, STATUS )

*  Attempt to locate any existing quality name
*  information in the input NDF. If such information is
*  found, LOCS is returned holding a set of five HDS
*  locators which identify the NDF and various items of
*  quality information. XNAME is returned holding the
*  name of the NDF extension in which the information
*  was found. If no quality name information is found,
*  then an error is reported.
      CALL IRQ_FIND( NDFIN, LOCS, XNAME, STATUS )

*  If no quality name information was found, annul the
*  error. New quality names information will be set up
*  in the "IRAS" NDF extension.
      IF( STATUS .EQ. IRQ__NOQNI ) THEN
```

```
         CALL ERR_ANNUL( STATUS )

*  If the "IRAS" extension does not exist, create it.
         CALL NDF_XSTAT( NDFIN, 'IRAS', FOUND, STATUS )
         IF( .NOT. FOUND ) THEN
            CALL NDF_XNEW( NDFIN, 'IRAS', 'IRAS', 0, 0,
     :                     XLOC, STATUS )
            CALL DAT_ANNUL( XLOC, STATUS )
         END IF

*  Create a new structure to hold quality information
*  in the "IRAS" NDF extension.
         CALL IRQ_NEW( NDFIN, 'IRAS', LOCS, STATUS )
      END IF

*  Attempt to add the quality name "SATURATED" to the
*  NDF. If the name already exists an error will be
*  reported.
      CALL IRQ_ADDQN( LOCS, 'SATURATED', .FALSE.,
     :  'Pixels with value greater than 10.0', STATUS )

*  Get a temporary NDF which is the same shape as the
*  input NDF.
      CALL NDF_BOUND( NDFIN, NDF__MXDIM, LBND, UBND,
     :                NDIM, STATUS )
      CALL NDF_TEMP( PLACE, STATUS )
      CALL NDF_NEW( '_REAL', NDIM, LBND, UBND, PLACE,
     :              NDF2, STATUS )

*  Map the DATA array, initialising its contents to
*  zero.
      CALL NDF_MAP( NDF2, 'DATA', '_REAL',
     :              'WRITE/ZERO', PNT2, NEL, STATUS )

*  Map the DATA array of the input NDF.
      CALL NDF_MAP( NDFIN, 'DATA', '_REAL', 'READ',
     :              PNT, NEL, STATUS )

*  Set pixels bad in the temporary array which
*  correspond to pixels greater than 10.0 in the input
*  NDF.
      CALL MASKIT( %VAL(PNT), %VAL(PNT2), NEL, 10.0,
     :             STATUS )

*  Assign the quality SATURATED to all the pixels which
*  are bad in the temporary NDF.
      CALL IRQ_SETQM( LOCS, .TRUE., 'SATURATED', NEL,
     :               %VAL( PNT2 ), STATUS )
```

```
*  Release the resources used by IRQ.
      CALL IRQ_RLSE( LOCS, STATUS )

*  End the NDF context.
      CALL NDF_END( STATUS )
```

E.2 Finding pixels which satisfy a quality expression

This example produces a copy of an input NDF in which all pixels which do not satisfy the quality expression ".NOT.(BACKGROUND .OR. SATURATED) " are set bad.

```
*  Attempt to locate any quality name information in
*  the input NDF. If no quality name information is
*  found, then an error is reported.
      CALL IRQ_FIND( NDFIN, LOCS, XNAME, STATUS )         ①

*  Attempt to compile the quality expression.
      QEXP = '.NOT. ( SATURATED .OR. BACKGROUND )'
      CALL IRQ_COMP( LOCS, IRQ__QNREF, .TRUE., QEXP,      ②
     :               UNDEF, NUNDEF, ERRPNT, IDQ, STATUS )

*  Produce a temporary copy of the input NDF.
      CALL NDF_TEMP( PLACE, STATUS )
      CALL NDF_COPY( NDFIN, PLACE, NDF2, STATUS )

*  Map the DATA array for UPDATE access.
      CALL NDF_MAP( NDF2, 'DATA', '_REAL', 'UPDATE',
     :              PNT2, NEL, STATUS )

*  Find all pixels which do not satisfy the quality
*  expression and set them bad in the NDF copy.
      CALL IRQ_SBAD( IDQ, .FALSE., NEL, %VAL( PNT2 ),
     :               ALLBAD, NOBAD, STATUS )

*  Unmap the NDF copy.
      CALL NDF_UNMAP( NDFIN, 'DATA', STATUS )

*  Close down the IRQ identifier system.
      CALL IRQ_CLOSE( STATUS )

*  Release the resources used by IRQ.
      CALL IRQ_RLSE( LOCS, STATUS )
```

Programming notes:

(1) NDFIN is the NDF identifier for the input NDF and should have been obtained previously. If the NDF contains no quality name information, then an error will be reported by IRQ_FIND.

(2) The quality expression must be assigned to a character variable, since the call to IRQ_COMP updates the expression by removing leading blanks, and converting the expression to upper case. An access violation would result if a literal string were supplied as an argument, instead of a character variable. If either of the quality names `SATURATED` or `BACKGROUND` is not defined in the input NDF, then an error will be reported by IRQ_COMP, and the undefined names will be returned in the UNDEF array.

F Packages Called by IRQ

IRQ_ makes calls to the following packages:

CHR_ - The CHR character handling package; see SUN/40.

CMP_ - HDS; see SUN/92.

DAT_ - HDS; see SUN/92.

ERR_ - The Starlink error reporting package; see SUN/104.

MSG_ - The Starlink message reporting package; see SUN/104.

NDF_ - The NDF access package; see SUN/33.

VEC_ - The PRIMDAT package; see SUN/39.

Access to these packages, together with packages called from within these packages, is necessary to use IRQ.

G IRQ Error Codes

IRQ routines can return any STATUS value generated by the subroutine packages which it calls. In addition it can return the following IRQ-specific values.

IRQ__BADBT
Bit value outside range [1,8] supplied.

IRQ__BADDM
Incorrect NDF dimensions supplied.

IRQ__BADNM
An illegal quality name has been given.

IRQ__BADQN
Incomplete QUALITY_NAMES structure found.

IRQ__BADSL
Invalid slot number supplied.

IRQ__BADST
Incomplete slot structure found.

IRQ__BADSY
Unrecognised logical operator or constant in quality expression.

IRQ__CMPLX
Too many symbols in quality expression.

IRQ__INCOM
Supplied vector has different size to the NDF

IRQ__INTER
Internal IRQ error (report to maintainer of IRQ).

IRQ__INVID
Invalid IRQ identifier supplied.

IRQ__IVNDF
Invalid NDF identifier found.

IRQ__LSHRT
Character variable too short.

IRQ__MSDOT
Missing delimiter "." in quality expression.

IRQ__MSOPD
Missing operand in quality expression.

IRQ__MSOPT
Missing or invalid operator in quality expression.

IRQ__MSPAR
Unpaired parentheses in quality expression.

IRQ__MULT
More than one structure found holding quality names information.

IRQ__NOMOR
No identifiers left for compiled quality expressions.

IRQ__NOOPS
No operands can be found.

IRQ__NOQNM
A quality name could not be found.

IRQ__NOQNI
No quality names structure found in the NDF.

IRQ__NOSPA
Can't reclaim space in an instruction array.

IRQ__NOWRT
Write access to the NDF is unavailable.

IRQ__QBAD
BAD values exists in the QUALITY component.

IRQ__QEXPL
Too many characters in quality expression.

IRQ__QIEXS
Extension already contains a quality names structure.

IRQ__QNEXS
Quality name is already defined.

IRQ__QLONG
Quality name too long.

IRQ__QREFS
Too many quality names in quality expression.

IRQ__QUNDF
QUALITY component is undefined.

IRQ__RDONL
An attempt has been made to remove a read-only quality name.

IRQ__STKOV
Evaluation stack overflow.

IRQ__STKUN
Evaluation stack underflow.

IRQ__XBITS
No bits left in QUALITY component.

H Changes Introduced in Version 3.0 of this Document

- It has a separate identity. It was previously bundled into KAPLIBS after being recovered from IRAS90.
- There is documentation.

I Changes Introduced in Version 4.0 of this Document

- The “VALID” component in the quality extension was previously intentionally left with an undefined value. It is now set to the defined value of TRUE.

J Changes Introduced in Version 5.0 of this Document

- The library is no longer restricted to NDFs for which the number of pixel can be represented by a 4-byte signed integer. Routines that have arguments representing pixel counts now have two versions - the original version - which continues to used 4-byte INTEGERs to represent pixel counts - and a new "8-byte" version that uses INTEGER*8 variables for such arguments. The name of each 8-byte function is the same as the name of the original 4-byte function, but with “8” appended to the end. So for instance, “IRQ_CNTQ8” is the 8-byte verssion of “IRQ_CNTQ”. The routine that have 8-byte interfaces are: IRQ_CNTQ, IRQ_RESQL, IRQ_RESQM, IRQ_SETQL, IRQ_SETQM and IRQ_SBAD.