

INTERNATIONAL STANDARD

ISO/IEC
12088-4

First edition
1995-12-15

Information technology — Computer graphics and image processing — Image processing and interchange — Application program interface language bindings —

Part 4:

C

*Technologies de l'information — Infographie et traitement de l'image —
Traitement et échange de l'image — Liants de langage d'interface de
programme d'application —*

Partie 4: C



Reference number
ISO/IEC 12088-4:1995(E)

Contents

1 Scope	1
2 Normative references	2
3 The C language binding of the Image Processing and Interchange Standard	3
3.1 Classification and designation	3
3.2 Functions versus macros	3
3.3 Implications of the language	4
3.3.1 Character strings	4
3.3.2 Implementation dependencies	4
3.3.3 Data object repository	4
3.4 Identifier mapping	5
3.5 Return values	6
3.6 Header files	6
3.7 Memory management	7
3.8 Error handling	9
3.8.1 Application defined error handlers	9
3.8.2 Function identification	9
3.8.3 Error presentation	9
3.9 Virtual register support	10
3.10 Convenience functions	10
3.11 Program examples	10
4 Tables	11
4.1 Abbreviations	11
4.1.1 Table of abbreviations	11
4.1.2 Abbreviation policy in construction of identifiers	14
4.2 Function names	15
4.2.1 Alphabetical by function name	15
4.2.2 Alphabetical by bound name	27
5 Data type definitions	39
5.1 Mapping of data types	39
5.2 Environment data type definitions	40
5.2.1 External physical image data types	40
5.2.2 Basic parameter data types	41
5.2.2.1 IPI-PIKS parameter data types	41
5.2.2.2 IPI-IIF parameter data types	41
5.2.3 Data object identifiers	42
5.2.3.1 IPI-PIKS object identifiers	42
5.2.3.2 IPI-IIF object identification	43
5.3 Implementation dependent data type definitions	44
5.4 Implementation independent data type definitions	45
5.4.1 Enumerated data type definitions	45
5.4.1.1 IPI-PIKS enumerated data type definitions	45
5.4.1.2 IPI-IIF enumerated data type definitions	50

5.4.2 Union data type definitions.....	51
5.4.2.1 IPI-PIKS union data type definitions.....	51
5.4.2.2 IPI-IIF union type data definitions.....	66
5.4.3 Structure data type definitions.....	68
5.4.3.1 IPI-PIKS structures.....	68
5.4.3.2 IPI-IIF structures.....	83
6 Macro definitions.....	87
6.1 Unbounded option and status codes.....	87
6.1.1 IPI-PIKS unbounded option and status codes.....	87
6.1.2 IPI-IIF unbounded option and status codes.....	101
6.2 IPI-IIF syntax descriptors.....	103
6.2.1 IPI-IIF syntax entity types.....	103
6.2.2 IPI-IIF syntax components.....	107
6.3 Function descriptors.....	123
6.3.1 IPI-PIKS function descriptors.....	123
6.3.2 IPI-PIKS convenience function descriptors.....	135
6.3.3 IPI-IIF function descriptors.....	136
6.3.4 IPI-IIF convenience function descriptors.....	137
6.4 Error descriptors.....	138
6.4.1 IPI-PIKS errors.....	138
6.4.2 IPI-IIF errors.....	143
6.4.3 System errors.....	145
6.4.4 Binding-specific errors.....	145
6.5 Implementation-specific data lengths.....	146
6.6 Repository object descriptors.....	147
6.6.1 Repository impulse response function arrays.....	147
6.6.2 Repository dither arrays.....	149
6.6.3 Repository colour conversion matrices.....	149
7 C functional elements.....	155
7.1 Notational conventions.....	155
7.2 IPI-PIKS functional element prototypes.....	156
7.3 IPI-IIF functional element prototypes.....	273
8 Convenience functions.....	282
8.1 IPI-PIKS convenience functions.....	282
8.1.1 Image preparation functions.....	282
8.1.1.1 prepare_colour_image.....	283
8.1.1.2 prepare_monochrome_image.....	284
8.1.1.3 create_unbounded_image_copy.....	285
8.1.2 ROI creation functions.....	287
8.1.2.1 generate_2d_roi_rectangular.....	288
8.1.2.2 generate_roi_coordinate.....	289
8.1.2.3 generate_roi_elliptical.....	290
8.1.2.4 generate_roi_polygon.....	291
8.1.2.5 generate_roi_rectangular.....	293
8.1.2.6 prepare_2d_roi_rectangular.....	294
8.1.2.7 prepare_roi.....	295
8.1.3 Tuple generation functions.....	296
8.1.3.1 generate_nd_1_tuple.....	297
8.1.3.2 generate_nd_3_tuple.....	298
8.1.3.3 generate_nd_4_tuple.....	299
8.1.3.4 generate_nd_5_tuple.....	300
8.1.3.5 generate_rd_3_tuple.....	301
8.1.3.6 generate_rd_4_tuple.....	302
8.1.3.7 generate_rd_5_tuple.....	303

Contents

8.1.3.8 generate_sd_1_tuple.....	304
8.1.3.9 generate_sd_3_tuple.....	305
8.1.3.10 generate_sd_4_tuple.....	306
8.1.3.11 generate_sd_5_tuple.....	307
8.1.4 Tiled image import and export.....	308
8.1.4.1 tiled_image_export.....	308
8.1.4.2 tiled_image_import.....	309
8.2 IPI-IIF convenience functions.....	310
8.2.1 Attach entity functions.....	310
8.2.1.1 attach.....	310
8.2.1.2 attach_sequence.....	310
8.2.1.3 attach_sequence_end.....	311
8.2.2 Create identifier function.....	311
8.2.3 Get entity component functions.....	312
8.2.3.1 get_component.....	312
8.2.3.2 get_sequence_component.....	312
8.2.4 Get entity value functions.....	313
8.2.4.1 get_entity_boolean.....	313
8.2.4.2 get_entity_integer.....	314
8.2.4.3 get_entity_real.....	314
8.2.4.4 get_entity_string.....	314
8.2.4.5 get_sequence_boolean.....	314
8.2.4.6 get_sequence_integer.....	315
8.2.4.7 get_sequence_real.....	315
8.2.4.8 get_sequence_string.....	315
8.2.4.9 get_sequence_end.....	316
8.2.5 Put entity value functions.....	317
8.2.5.1 put_entity_boolean.....	318
8.2.5.2 put_entity_integer.....	318
8.2.5.3 put_entity_real.....	318
8.2.5.4 put_entity_string.....	319
8.2.5.5 put_sequence_boolean.....	319
8.2.5.6 put_sequence_integer.....	319
8.2.5.7 put_sequence_real.....	320
8.2.5.8 put_sequence_string.....	320
Annexes	
A Memory management.....	321
A.1 Introduction.....	321
A.2 Functions that import the application data to the implementation memory.....	323
A.3 Functions that export the implementation data to the application memory.....	325
B Macros, data types in compilation order and external functions.....	327
B.1 IPI-PIKS macros, types and functions.....	327
B.1.1 Macro definitions.....	327
B.1.2 Data types in compilation order.....	367
B.1.3 Functions.....	390
B.1.4 Convenience functions.....	475
B.2 IPI-IIF macros, types and functions.....	480
B.2.1 Macro definitions.....	480
B.2.2 Data types in compilation order.....	499
B.2.3 Functions.....	504
B.2.4 Convenience functions.....	510

C Sample programs.....	513
C.1 IPI-PIKS application examples.....	513
C.1.1 Application use of IPI-PIKS for memory-managed image import/export	514
C.1.2 Histogram generation.....	519
C.1.3 Region of interest control.....	523
C.1.4 Simulated unsharp mask operation.....	526
C.1.5 Demonstration of asynchronous and chained application.....	531
C.1.5.1 Image blend by synchronous, unchained functional representation.....	531
C.1.5.2 Image blend by asynchronous functional representation.....	533
C.1.5.3 Image blend by chained functional representation.....	537
C.2 IPI-IIF application example.....	541
D Macros for short function identifiers.....	550
Index.....	562

IECNORM.COM : Click to view the full PDF of ISO/IEC 12088-4:1995

List of figures

1 - Buffered memory management of InImportImage(...)	324
2 - Buffered memory management of InExportImage(...)	326

List of tables

1 Data type prefixes	5
2 Abbreviations	11
3 Function names alphabetical by function name	15
4 Function names alphabetical by bound name	27
5 Data types	39
6 IPI-PIKS function descriptors	123
7 IPI-PIKS convenience function descriptors	135
8 IPI-IIF function descriptors	136
9 IPI-IIF convenience function descriptors	137
10 IPI-PIKS error descriptors	138
11 IPI-IIF error descriptors	143
12 System error descriptors	145
13 Binding-specific error descriptors	145
14 IPI-PIKS data lengths	146
15 Repository impulse response function array descriptors	147
16 Repository dither array indices	149
17 Repository colour conversion matrix indices	149

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft international Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 12088-4 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 24, *Computer graphics and image processing*.

ISO/IEC 12088 consists of the following part, under the general title *Information technology — Computer graphics and image processing — Image processing and interchange — Application program interface language bindings*:

— Part 4: C

Other parts may follow.

Annexes A to D of this part of ISO/IEC 12088 are for information only.

Introduction

The Image Processing and Interchange (IPI) functional specification, ISO/IEC 12087, upon which this binding is based, emerged as an International Standard in 1994. It consists of three parts: Part 1: Common Architecture for Imaging (IPI-CAI), Part 2: Programmer's Imaging Kernel System (IPI-PIKS) Application Program Interface and Part 3: Image Interchange Facility (IPI-IIF).

The functional description of ISO/IEC 12088 is specified in a language independent manner and needs to be embedded in language dependent layers (language bindings) for use with particular programming languages.

The purpose of this part of ISO/IEC 12088 is to define a standard binding for the Image Processing and Interchange Standard in the C programming language.

IECNORM.COM : Click to view the full PDF of ISO/IEC 12088-4:1995

Information technology — Computer graphics and image processing — Image processing and interchange — Application program interface language bindings —

Part 4:

C

1 Scope

ISO/IEC 12087 consists of the three parts which define the functional aspects of this part of ISO/IEC 12088. The Common Architecture for Imaging (IPI-CAI) defines the overall architecture. The Programmer's Imaging Kernel System (IPI-PIKS) and the Image Interchange Facility (IPI-IIF) each specify a language independent, image processing Application Program Interface (API) within the Image Processing and Interchange Standard. Either API may be implemented independently or both may be combined in one implementation. For integration into a programming language, IPI-PIKS and IPI-IIF APIs are embedded in a language dependent layer obeying the particular conventions of that language. This part of ISO/IEC 12088 specifies such a language dependent layer for the C language.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 12088. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 12088 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

[1] ISO/IEC 12087-1:1995, *Information technology - Computer graphics and image processing - Image Processing and Interchange (IPI) - Functional specification - Part 1: Common architecture for imaging*.

[2] ISO/IEC 12087-2:1994, *Information technology - Computer graphics and image processing - Image Processing and Interchange (IPI) - Functional specification - Part 2: Programmer's Imaging kernel system application programme interface*.

[3] ISO/IEC 12087-3:1995, *Information technology - Computer graphics and image processing - Image Processing and Interchange (IPI) - Functional specification - Part 3: Imaging Interchange Facility (IIF)*.

[4] ISO/IEC 9899:1990, *Programming Languages - C*.

3 The C language binding of the Image Processing and Interchange Standard

3.1 Classification and designation

This part of ISO/IEC 12088 incorporates the rules of conformance defined in the Image Processing and Interchange Standard (ISO/IEC 12087) for implementations, with those additional requirements specifically defined for C language implementations of the Standard.

The following criteria shall determine conformance of an implementation of this part of ISO/IEC 12088:

In order to conform, development of a C language implementation of the Image Processing and Interchange Standard shall implement one or more specific profiles as specified in the functional specification of ISO/IEC 12087. Independent implementation of the Programmer's Imaging Kernel System (Part 2 of ISO/IEC 12087 including relevant portions of Part 1) or the Image Interchange Facility (Part 3 of ISO/IEC 12087 including relevant portions of Part 1) is allowed.

The implementation shall make visible all of the declarations in the C binding specified in this part of ISO/IEC 12088 for that same level of ISO/IEC 12087 and all lower levels and for a specific level of the C language. Thus, for example, the syntax of the function names shall be precisely as specified in the binding and parameters shall be of the data types stated in the binding.

This part of ISO/IEC 12088 supports the conformance profiles defined in the functional specification of ISO/IEC 12087.

3.2 Functions versus macros

An implementation may substitute macros for functions. Implementation by macro should result in the same effect as implementation by function including any side effects.

3.3 Implications of the language

3.3.1 Character strings

The C language represents character strings as an array of characters terminated by the null character (i.e., '\0'). Therefore, the null character is not usable as a printable character.

3.3.2 Implementation dependencies

There are a number of implementation issues which will affect the portability of application programs employing this part of ISO/IEC 12087. The application programmer should follow accepted practices for ensuring the portability of C language programs to avoid introducing problems when recompiling the application on another system.

This part of ISO/IEC 12088 attempts to avoid dependencies on computer specific types which may vary among computers. Clause 5 recommends type mappings for common numeric types, data object identifiers, and precision to be used throughout this C language binding.

3.3.3 Data object repository

The Image Processing and Interchange Standard provides a repository of data objects for routinely used objects. These objects are defined in ISO/IEC 12087. Access to these objects for their assignment to IPI-PIKS operators is by IPI-PIKS-internal identifiers which may be acquired by the execution of a *return_repository_id* function presented in 7.2 of this part of ISO/IEC 12088.

The repository object identifiers are defined in 6.5. Each repository object name begins with the characters "IR_".

3.4 Identifier mapping

The IPI-PIKS-API and IPI-IIF-API function names are mapped to C identifiers beginning with the sentinel "I" followed by a lower case name prefix which identifies the return parameter type. Words and phrases used in the function names are concatenated with and beyond the prefix and may be abbreviated in the name representation. The acronyms and the first letter of name words and abbreviations are in upper case and the remainder are in lower case. For example, the identifier of the IPI-PIKS *import_image* function, which returns an IPI-PIKS internal object identifier, is *InImportImage*.

Data type element names are mapped to C identifiers beginning with the sentinel "I." The sentinel is followed by a two-part prefix, which delineates the usage and type of the element, followed by a name. These usage and data type designators, see Table 1, include "m" for external physical image pixel representation, "p" for parameter representation and "d" for internal representation. For some data types, the type designator and name may be merged, e.g., *Ipint*, the integer basic type. Acronyms, words and phrases used in the type names are in lower case, often abbreviated in the representation, and separated with the underscore character, "_". For example, the union of pixel data types used for parameters is *Ipsparameter pixel*.

Enumerated data type constants and macro names are mapped to C identifiers beginning with the sentinel "I" followed by a word or abbreviation of the entity to which the identifier belongs and the name of the identifier in upper case with acronyms, abbreviations or words separated with the underscore character, "_". For example the macro name for Boolean false is *IBOOL_FALSE*.

Parameters are mapped to C identifiers beginning with a prefix which identifies the type and/or usage. Words and phrases used in the parameter and variable names are concatenated with and beyond the prefix and may be abbreviated in the representation. The first letter of name words and abbreviations are in upper case and the remainder are in lower case. Acronyms are presented in upper case. For example, the name of the function parameter which denotes the identifier of a destination ROI object is *nDestROI*.

The prefix symbols used in this part of ISO/IEC 12088 are listed in Table 1.

Table 1 - Data type prefixes

Prefix	Meaning
a	array
b	Boolean
c	character
d	internal data type
e	enumerated data type
f	function
i	integer
m	external image data type
n	identifier
p	parameter type
r	real
s	structure
t	pointer
u	unsigned integer
v	void
x	complex
z	zero terminated string

The C language binding of the Image Processing and Interchange Standard

The C language standard requires that compilers recognize internal identifiers which are distinct in at least 31 characters. That standard also requires that external identifiers (i.e., those seen by the linker) be recognized to a minimum of six characters, independent of case. Implementations which must run in environments which accept less than 31 characters in external identifiers must include a set of macros (#defines) in the header file which equate the long names as defined in this document to a set of shorter names which conform to the requirements of the implementation. An example of these shorter 6-character names can be found in Annex D

3.5 Return values

Functional elements which perform a status test within the IPI-PIKS internal domain, and do not affect destination data, output the Boolean result of the test as return value.

Destination identifiers or values are output as return values from elements which include destination objects or values, respectively, as part of their parameter list, even if the identifier itself is input to the element. If more than one destination object is identified by a function, no return value is output. Return values are void from elements which do not provide destination objects or values.

If the destination identifier, output as a return value, is generated by the functional element, the parameter represented by the return value is not included in the argument list of the functional element. Conversely, if the same parameter appears in the argument list, the return value is a replication of an input of the destination identifier, though the contents of the object may be modified by the function.

3.6 Header files

The C language provides a mechanism to allow external files to be included in a compilation. Clause 5 of this part of ISO/IEC 12088 describes the data types that shall be defined in the piks.h and iif.h which should be included in any application program that intends to use IPI-PIKS and/or IPI-IIF, respectively, via the C language binding. Additional implementation-dependent items may be placed in these files if needed and the files may be concatenated as one header file. These items should start with the sentinel "I", as far as applicable.

The files piks.h and iif.h shall also contain external prototypes for all IPI-PIKS and IPI-IIF C language functions. For example, the prototype for the IPI-PIKS function *InResize* would appear as follows:

```
extern Idnimage InResize (                /* OUT destination image identifier */
    Idnimage      nSourceImage,          /* source image identifier */
    Idnimage      nDestImage,            /* destination image identifier */
    Idntuple      nDestSize              /* destination size ND 5-tuple identifier */
);
```

Examples of piks.h and iif.h which utilize suggested data types are presented in Annex B.

3.7 Memory management

IPI-PIKS requires internal memory space, denoted by identifiers containing the letter "d" in their prefixes (immediately following the sentinel) as they appear in this part of ISO/IEC 12088. Internal memory requirements may exist in a memory domain independent of application memory or may be acquired from application memory as provided in a specific implementation. ISO/IEC 12087 and this part of ISO/IEC 12088 do not provide functionality to support the allocation and management of IPI-PIKS internal memory should it be acquired from the application domain.

Both IPI-IIF and IPI-PIKS require external memory space, denoted by identifiers containing the letter "p" in their prefixes (immediately following the sentinel) as they appear in this part of ISO/IEC 12088. External memory requirements must be acquired from application memory space. The application shall allocate the external memory for the data provided to and returned by the implementation of the Image Processing and Interchange Standard. The application will allocate C structures and pass the appropriate identifiers of those structures to the requiring functional element, which will then place information into or receive information from the structures. However, the application may provide data buffered in partial amounts for large data objects input to the functions. Also, some functions return variable length data, the length of which is not known a priori and can not be computed by the application.

If the memory space requirement is small, even if variable in length, the structure and sufficient space must be provided by the application. As example, a B-tuple for colour image data precision specification will contain no more than four data elements and, as such, is a small structure. A B-tuple for spectral image data precision specification may contain many elements.

Functions which may receive buffered data and functions which return variable lengths of sizable data or are considered to return potentially large amounts of fixed-length data are managed such that the application can submit or receive buffered portions of the data. The data portion, or buffer, is a subset of the implementation's internal data structure and is called the application's buffer. This allows the application to process the implementation's data, considered as a single dimension array, in a piecewise manner rather than all at once.

The application allocates the memory for its buffer and passes an identifier of that buffer to the implementation. The implementation places the subset of data into the buffer according to the specifications included among the function's parameters. In order to support this policy of memory management, three additional parameters have been added to functions which may transfer buffered data:

- a) **iApplBufferSize:** An integer input parameter which specifies the length of the application's buffer, dimensioned in terms of the number of data elements of data type specified for the input or return. The positive value of **iApplBufferSize** indicates the element count of the data which can be contained in the application's buffer. A value of 0 is valid as input for this parameter and is useful to interrogate the overall data size available for output or available in the implementation for input of each buffered transfer operation. A negative value is valid for this parameter and is interpreted as a signal for complete transfer of the data, the application being responsible to provide adequate space for the complete data transfer from implementation to application or the complete data requirement for transfer from application to implementation.
- b) **uStartPoint:** An unsigned integer input parameter which is an element index offset into the implementation's data space represented as a 1D array of data type specified for the external data being transferred. **uStartPoint** of 0 represents the first element of both the implementation's and application's data. Otherwise, **uStartPoint** is the index which indicates the element offset in the implementation's data space where data transfer is to be initiated to or from the beginning of the application's buffer. The transfer of data continues to the extent of the application's buffer or until there are no more items to be transferred, whichever is least.

The C language binding of the Image Processing and Interchange Standard

- c) **tuDataSize**: An output parameter which is a pointer to an unsigned integer value. As directed by this indirection parameter, the implementation transfers a value which is the size of the data remaining beyond the given **uStartPoint** less the amount of data transferred during execution of the function. The size is specified in terms of elements of the external data type specified for the data to be transferred into or out of the implementation's data space.

Errors that can be invoked during execution of the functions which utilize this memory management policy are listed in 6.4.4. Failures to correctly transfer data as specified are reported as appropriate error designators by all using functions except inquiry functions. Inquiry functions do not enter into any error condition but provide parametric return to indicate the validity of the data transfer.

A more detailed description of memory management is provided in Annex A and sample programming is presented in Annex C.

3.8 Error handling

The error handling mechanism specified for IPI APIs employs methodology similar to the error handling specified in the graphics standards. This mechanism is described in Parts 2 and 3 of ISO/IEC 12087.

3.8.1 Application defined error handlers

An application can define an IPI binding-specific error handling function for the implementation and access this via the following utility function which has been added to the functions of IPI-PIKS and IPI-IIF in this part of ISO/IEC 12088:

```
void *ItfSetErrorHandler (                                /* OUT old error handling function pointer */  
    void (*ItfNewErrorHandler)() /* new error handler function pointer */  
);
```

Application-defined error handling functions shall accept the same arguments as the standard error handler. In IPI-PIKS, this function is not chainable and requires synchronous operation. It shall be provided in all implementation profiles.

Initially, the implementation will invoke its standard error handler whenever an error occurs. After a call of *ItfSetErrorHandler*, the implementation will invoke the user-defined error function, *ItfNewErrHandler*, whenever an error is detected. The pointer to the previous error handler is returned to the application for later use.

If *ItfNewErrHandler* is not a valid pointer, the error handling will automatically be done by the standard error handler.

3.8.2 Function identification

This part of ISO/IEC 12088 defines descriptors for the identification of functional elements in 6.3. These constant values are the same as those specified in Parts 2 and 3 of ISO/IEC 12087 for functions which appear in those documents.

Function identifier values defined in ISO/IEC 12087 are assigned herein except where functions have been added in this part of ISO/IEC 12088 or have been subdivided to include versions which permit virtual register usage for return value capture and parameter specification. These additional functions are assigned identifier values sequentially following the function identifier values given in ISO/IEC 12087.

Each function identification descriptor begins with the characters "IF_" or "IC_", the latter for convenience functions.

3.8.3 Error presentation

This part of ISO/IEC 12088 defines descriptors for the error conditions in 6.4. Error descriptor values defined in ISO/IEC 12087 are assigned herein.

Each error descriptor begins with the characters "IE_".

3.9 Virtual register support

Virtual registers can be used to replace the parameters and receive the return values of functions. This is necessary for parameter alteration when these functions are operating in chains. To support the substitution of virtual registers as function arguments and recipients of return values, this part of ISO/IEC 12088 adds a separate version of each chainable function, named with a suffix of VReg.

3.10 Convenience functions

Specification of common operations which require the execution of several functions bound in this part of ISO/IEC 12088 are grouped into convenience functions for application users who desire to reduce the amount of programming from such details. A number of these functions are presented in clause 8 where they are normative to assure portability.

3.11 Program examples

Example programs of IPI application are presented in Annex C. These illustrate some of the common operations and uses of the fundamental functionality bound in this part of ISO/IEC 12088.

4 Tables

4.1 Abbreviations

4.1.1 Table of abbreviations

Only words and phrases which are abbreviated appear in this table.

The abbreviations may be used for

comments (acronyms only),
data types,
enumerated constants,
function names,
macro names,
parameter names,
text.

Table 2 - Abbreviations

Abbreviation	Word or Phrase
1D	one dimension
2D	two dimension
2nd	second
3D	three dimension
4D	four dimension
5D	five dimension
A	appearance
API	Application Program Interface
appl	application
arg	argument
arith	arithmetic
ASN.1	Abstract Syntax Notation One
asynch	asynchronous
attr	attribute
B	image spectral band size
BD	Boolean, internal
BI	Boolean, pixel type
bool	Boolean
BP	Boolean, parameter
C	colourmetric
CAI	Common Architecture for Imaging
CCIR	Comite Consultatif International des Radiocommunications
ccw	counterclockwise
CD	complex, internal
CF	complex, external
char	character
chc	choice
chr	chrominance
chrom	chromaticity
CIE	Commision Internationale de l'Eclairage

Table 2 - Abbreviations (continued)

Abbreviation	Word or Phrase
cmp	component
CMY	cyan, magenta, yellow colour coordinates
CMYK	cyan, magenta, yellow, black colour coordinates
COLR	colour image structure
conv	conversion
CP	complex, parameter
CS	character string
def	defined, definition
deriv	derivative
dest	destination
detect	detection
diff	difference, differencing
dilate	dilation
DOR	designated object reference
EBU	European Broadcasting Union
enum	enumerated
equal	equalization
fund	fundamental
Gauss	Gaussian
GEN	generic image structure
grad	gradient
hist	histogram
id	identifier
ID	internal identifier
ID_ARRAY	composite image identifier array
ID_LIST	composite image identifier list
ID_RECORD	composite image identifier record
IEC	International Electrotechnical Commission
IIF	Image Interchange Facility
impl	implementation
int	integer
interp	interpolated, interpolation
IHS	intensity, hue, saturation colour coordinates
IP	parameter identifier
IPI	Image Processing and Interchange
ISO	International Organization for Standardization
JBIG	Joint Bi-level Image Experts Group
JPEG	Joint Photographic Experts Group
LAB	$L^*a^*b^*$ colour coordinates
Laplace	Lapacian
lin	linear
lowleft	lower left
lowright	lower right
lsb	least significant bit
lum	luminance
LUT	lookup table
LUV	$L^*u^*v^*$ colour coordinates
manip	manipulation

Table 2 - Abbreviations (continued)

Abbreviation	Word or Phrase
MON	monochrome image structure
MPEG	Moving Pictures Experts Group
msb	most significant bit
nbhood	neighbourhood
nbr	neighbour
ND	unsigned integer, internal
NI	unsigned integer, pixel type
nonlin	nonlinear
NP	unsigned integer, parameter
NTSC	National Television Systems Committee
orth	orthogonal
OUT	output parameter or structure
PIKS	Programmer's Imaging Kernel System
pixel	picture element
RD	real, internal
RF	real floating point, external
RGB	red, green, blue colour coordinates
ROI	region of interest
RP	real, parameter
SD	signed integer, internal
sdev	standard deviation
sep	separated
seq	sequence
SI	signed integer, pixel type
SMPTE	Society of Motion Picture and Television Engineers
SP	signed integer, parameter
SPE	spectral image structure
stat	statistical
synch	synchronous
T	image temporal size
TC	temporal colour structure
TEM	temporal image structure
templ	template
TI	fixed point integer, pixel type
trunc	truncated
TS	temporal spectral structure
uint	unsigned integer
unbnd	unbounded
upleft	upper left
upright	upper right
UVW	UVW colour coordinates
VC	volumetric colour image
VOL	volumetric image structure
VReg	virtual register
VS	volumetric-spectral image structure
VT	volumetric-temporal image structure
VTC	volumetric-temporal colour image structure
VTs	volumetric-temporal-spectral image structure

Table 2 - Abbreviations (concluded)

Abbreviation	Word or Phrase
X	image horizontal size
XYZ	XYZ colour coordinates
Y	image vertical size
YCBCR	YCbCr colour coordinates
YIQ	YIQ colour coordinates
YUV	Yuv colour coordinates
YXY	Yxy colour coordinates
Z	image depth size

4.1.2 Abbreviation policy in construction of identifiers

In the construction of the names for data types, structures, macros, functions, arguments, etc., the following policy is applied:

- 1) Abbreviations and acronyms are defined and applied only when needed to avoid name lengths in excess of 31 characters.
- 2) When a word is abbreviated or a compound term is converted to an acronym, only one abbreviation applies for that word or one acronym for that term and all names which would utilize the word or term, contain only the abbreviation or acronym.
- 3) The plural of an expression is constructed by adding an "s" after the abbreviation: so, for example, "attribute" is abbreviated to "attr" and "attributes" is abbreviated to "attrrs". If an expression is mapped to NULL, so will be its plural.

4.2 Function names

Function names presented in ISO/IEC 12087 are bound in this part of ISO/IEC 12088 according to construction rules presented in 3.4. Function names are related to bound names in the tables of the following clauses. Functions of IPI-PIKS and IPI-IIF are both included in tables 3 and 4.

The existence of functions within an implementation is directed by a profile level specified for that implementation in ISO/IEC 12087. IPI-PIKS defines four profiles for differing levels of implementation which, in increasing order of capability, are foundation, technical, scientific and full. In addition, the inclusion of the IPI-PIKS *input_object* and *output_object* functions is required only if the IPI-PIKS implementation is IPI-IIF capable. Functions appearing in a lower profile must be included in subsequent levels.

IPI-IIF defines three profiles for differing levels of implementation: however, all IPI-IIF functions are required in all IPI-IIF profiles. The inclusion of *export_iif_data_to_piks* and *import_iif_data_from_piks* functions is required only if the IPI-IIF implementation is IPI-PIKS capable.

Tables 3 and 4 contain the identification of the least profile level in which the listed function is required

4.2.1 Alphabetical by function name

Table 3 - Function names alphabetical by function name

Function Name	Bound Name	Least Profile
abort_asynchronous_execution	IvAbortAsynchExecution	Scientific
accumulator	ItraAccumulator	Foundation
adaptive_histogram_equalization	InAdaptiveHistEqual	Scientific
adaptive_histogram_equalization_vreg	InAdaptiveHistEqualVReg	Full
allocate_chain	InAllocateChain	Full
allocate_chain_vreg	InAllocateChainVReg	Full
allocate_composite_identifier_array	InAllocateIdArray	Scientific
allocate_composite_identifier_array_vreg	InAllocateIdArrayVReg	Full
allocate_composite_identifier_list	InAllocateIdList	Scientific
allocate_composite_identifier_list_vreg	InAllocateIdListVReg	Full
allocate_composite_identifier_record	InAllocateIdRecord	Scientific
allocate_composite_identifier_record_vreg	InAllocateIdRecordVReg	Full
allocate_histogram	InAllocateHist	Foundation
allocate_histogram_vreg	InAllocateHistVReg	Full
allocate_image	InAllocateImage	Foundation
allocate_image_vreg	InAllocateImageVReg	Full
allocate_lookup_table	InAllocateLUT	Foundation
allocate_lookup_table_vreg	InAllocateLUTVReg	Full
allocate_matrix	InAllocateMatrix	Foundation
allocate_matrix_vreg	InAllocateMatrixVReg	Full
allocate_neighbourhood_array	InAllocateNbhoodArray	Foundation
allocate_neighbourhood_array_vreg	InAllocateNbhoodArrayVReg	Full
allocate_pixel_record	InAllocatePixelRecord	Technical
allocate_pixel_record_vreg	InAllocatePixelRecordVReg	Full
allocate_roi	InAllocateROI	Foundation
allocate_roi_vreg	InAllocateROIVReg	Full

Table 3 - Function names alphabetical by function name (continued)

Function Name	Bound Name	Least Profile
allocate_static_array	InAllocateStaticArray	Technical
allocate_static_array_vreg	InAllocateStaticArrayVReg	Full
allocate_tuple	InAllocateTuple	Foundation
allocate_tuple_vreg	InAllocateTupleVReg	Full
allocate_value_bounds	InAllocateValueBounds	Scientific
allocate_value_bounds_vreg	InAllocateValueBoundsVReg	Full
allocate_vreg	InAllocateVReg	Full
allocate_vreg_vreg	InAllocateVRegVReg	Full
alpha_blend_constant	InAlphaBlendConstant	Foundation
alpha_blend_constant_vreg	InAlphaBlendConstantVReg	Full
alpha_blend_variable	InAlphaBlendVariable	Technical
alpha_blend_variable_vreg	InAlphaBlendVariableVReg	Full
array_to_lut	InArrayToLUT	Foundation
array_to_lut_vreg	InArrayToLUTVReg	Full
attach_iif_syntax_entity	InAttachEntity	Foundation
bind_match_point	InBindMatchPoint	Technical
bind_match_point_vreg	InBindMatchPointVReg	Full
bind_roi	InBindROI	Foundation
bind_roi_vreg	InBindROIVReg	Full
bit_shift	InBitShift	Foundation
bit_shift_vreg	InBitShiftVReg	Full
cartesian_to_polar	InCartesianToPolar	Scientific
cartesian_to_polar_vreg	InCartesianToPolarVReg	Full
chain_abort	InChainAbort	Full
chain_abort_vreg	InChainAbortVReg	Full
chain_begin	InChainBegin	Full
chain_delete	InChainDelete	Full
chain_delete_vreg	InChainDeleteVReg	Full
chain_end	InChainEnd	Full
chain_execute	InChainExecute	Full
chain_execute_vreg	InChainExecuteVReg	Full
chain_reload	InChainReload	Full
chain_reload_vreg	InChainReloadVReg	Full
classifier_bayes	InClassifierBayes	Scientific
classifier_bayes_vreg	InClassifierBayesVReg	Full
classifier_nearest_neighbour	InClassifierNearestNbr	Scientific
classifier_nearest_neighbour_vreg	InClassifierNearestNbrVReg	Full
close_iif_gateway	IvCloseGateway	Foundation
close_piks	IvClosePIKS	Foundation
close_piks_emergency	IvClosePIKSEmergency	Foundation
close_port	IvClosePort	Foundation
colour_conversion_linear	InColourConvLin	Foundation
colour_conversion_linear_vreg	InColourConvLinVReg	Full
colour_conversion_matrix	IvColourConvMatrix	Foundation
colour_conversion_matrix_vreg	IvColourConvMatrixVReg	Full
colour_conversion_nonlinear	InColourConvNonlin	Technical
colour_conversion_nonlinear_vreg	InColourConvNonlinVReg	Full

Table 3 - Function names alphabetical by function name (continued)

Function Name	Bound Name	Least Profile
colour_conversion_subtractive	InColourConvSubtractive	Foundation
colour_conversion_subtractive_vreg	InColourConvSubtractiveVReg	Full
colour_lookup_interpolated	InColourLookupInterp	Technical
colour_lookup_interpolated_vreg	InColourLookupInterpVReg	Full
complement	InComplement	Foundation
complement_roi	InComplementROI	Technical
complement_roi_vreg	InComplementROIReg	Full
complement_vreg	InComplementVReg	Full
complex_composition	InComplexComposition	Technical
complex_composition_vreg	InComplexCompositionVReg	Full
complex_conjugate	InComplexConjugate	Technical
complex_conjugate_vreg	InComplexConjugateVReg	Full
complex_decomposition	IvComplexDecomposition	Technical
complex_decomposition_vreg	IvComplexDecompositionVReg	Full
complex_magnitude	InComplexMagnitude	Technical
complex_magnitude_vreg	InComplexMagnitudeVReg	Full
composite_identifier_array_equal	IbIdArrayEqual	Scientific
composite_identifier_array_equal_vreg	InIdArrayEqualVReg	Full
composite_identifier_array_get	InIdArrayGet	Scientific
composite_identifier_array_get_vreg	InIdArrayGetVReg	Full
composite_identifier_array_put	InIdArrayPut	Scientific
composite_identifier_array_put_vreg	InIdArrayPutVReg	Full
composite_identifier_list_empty	IbIdListEmpty	Scientific
composite_identifier_list_empty_vreg	InIdListEmptyVReg	Full
composite_identifier_list_equal	IbIdListEqual	Scientific
composite_identifier_list_equal_vreg	InIdListEqualVReg	Full
composite_identifier_list_get	InIdListGet	Scientific
composite_identifier_list_get_vreg	InIdListGetVReg	Full
composite_identifier_list_insert	InIdListInsert	Scientific
composite_identifier_list_insert_vreg	InIdListInsertVReg	Full
composite_identifier_list_remove	InIdListRemove	Scientific
composite_identifier_list_remove_vreg	InIdListRemoveVReg	Full
composite_identifier_record_equal	IdIdRecordEqual	Scientific
composite_identifier_record_equal_vreg	InIdRecordEqualVReg	Full
composite_identifier_record_get	InIdRecordGet	Scientific
composite_identifier_record_get_vreg	InIdRecordGetVReg	Full
composite_identifier_record_put	InIdRecordPut	Scientific
composite_identifier_record_put_vreg	InIdRecordPutVReg	Full
compress_iif_syntax_entity	InCompressEntity	Foundation
constant_predicate	IbConstantPredicate	Technical
constant_predicate_vreg	InConstantPredicateVReg	Full
convert_array_to_image	InConvertArrayToImage	Foundation
convert_array_to_image_vreg	InConvertArrayToImageVReg	Full
convert_image_datatype	InConvertImageDatatype	Foundation
convert_image_datatype_vreg	InConvertImageDatatypeVReg	Full
convert_image_to_array	InConvertImageToArray	Foundation
convert_image_to_array_vreg	InConvertImageToArrayVReg	Full

Table 3 - Function names alphabetical by function name (continued)

Function Name	Bound Name	Least Profile
convert_image_to_roi	InConvertImageToROI	Technical
convert_image_to_roi_vreg	InConvertImageToROIVReg	Full
convert_roi_to_image	InConvertROIToImage	Foundation
convert_roi_to_image_vreg	InConvertROIToImageVReg	Full
convolve_2d	InConvolve2D	Foundation
convolve_2d_vreg	InConvolve2DVReg	Full
convolve_5d	InConvolve5D	Technical
convolve_5d_vreg	InConvolve5DVReg	Full
copy_iif_syntax_entity	InCopyEntity	Foundation
copy_window	InCopyWindow	Foundation
copy_window_roi	InCopyWindowROI	Technical
copy_window_roi_vreg	InCopyWindowROIvReg	Full
copy_window_vreg	InCopyWindowVReg	Full
create_iif_syntax_entity	InCreateEntity	Foundation
create_tuple	InCreateTuple	Foundation
create_tuple_vreg	InCreateTupleVReg	Full
cross_correlation	InCrossCorrelation	Scientific
cross_correlation_vreg	InCrossCorrelationVReg	Full
deallocate_data_object	IvDeallocateDataObject	Foundation
deallocate_data_object_vreg	IvDeallocateDataObjectVReg	Full
decompress_iif_syntax_entity	IstDecompressEntity	Foundation
define_sub_image	InDefineSubImage	Foundation
define_sub_image_vreg	InDefineSubImageVReg	Full
delete_iif_atomic_sequence_element	InDeleteAtomicSeqElement	Foundation
delete_iif_syntax_entity	IvDeleteEntity	Foundation
detach_iif_syntax_entity	InDetachEntity	Foundation
difference_measures	IrDiffMeasures	Technical
difference_measures_vreg	InDiffMeasuresVReg	Full
diffuse	InDiffuse	Foundation
diffuse_vreg	InDiffuseVReg	Full
dither	InDither	Foundation
dither_vreg	InDitherVReg	Full
draw_pixels	InDrawPixels	Foundation
draw_pixels_roi	InDrawPixelsROI	Technical
draw_pixels_roi_vreg	InDrawPixelsROIvReg	Full
draw_pixels_vreg	InDrawPixelsVReg	Full
dyadic_arithmetic	InDyadicArith	Foundation
dyadic_arithmetic_vreg	InDyadicArithVReg	Full
dyadic_complex	InDyadicComplex	Technical
dyadic_complex_vreg	InDyadicComplexVReg	Full
dyadic_logical	InDyadicLogical	Foundation
dyadic_logical_roi	InDyadicLogicalROI	Technical
dyadic_logical_roi_vreg	InDyadicLogicalROIvReg	Full
dyadic_logical_vreg	InDyadicLogicalVReg	Full
dyadic_predicate	InDyadicPredicate	Foundation
dyadic_predicate_vreg	InDyadicPredicateVReg	Full
edge_detection_orthogonal_gradient	InEdgeDetectOrthGrad	Scientific

Table 3 - Function names alphabetical by function name (continued)

Function Name	Bound Name	Least Profile
edge_detection_orthogonal_gradient_vreg	InEdgeDetectOrthGradVReg	Full
edge_detection_second_derivative	InEdgeDetect2ndDeriv	Scientific
edge_detection_second_derivative_vreg	InEdgeDetect2ndDerivVReg	Full
edge_detection_template_gradient	InEdgeDetectTemplateGrad	Scientific
edge_detection_template_gradient_vreg	InEdgeDetectTemplateGradVReg	Full
equal_predicate	IbEqualPredicate	Technical
equal_predicate_vreg	InEqualPredicateVReg	Full
erosion_dilation_boolean	InErosionDilateBool	Technical
erosion_dilation_boolean_roi	InErosionDilateBoolROI	Technical
erosion_dilation_boolean_roi_vreg	InErosionDilateBoolROIVReg	Full
erosion_dilation_boolean_vreg	InErosionDilateBoolVReg	Full
erosion_dilation_grey	InErosionDilateGrey	Technical
erosion_dilation_grey_vreg	InErosionDilateGreyVReg	Full
error_function_scaling	InErrorFunctionScaling	Scientific
error_function_scaling_vreg	InErrorFunctionScalingVReg	Full
error_handler	IvErrorHandler	Foundation
error_logger	IvErrorLogger	Foundation
error_test	IbErrorTest	Foundation
error_test_vreg	InErrorTestVReg	Full
export_histogram	ItuaExportHist	Foundation
export_iif_data_to_piks	InExportIIFDataToPIKS	Foundation - with PIKS only
export_image	IstExportImage	Foundation
export_lut	IstExportLUT	Foundation
export_matrix	IstExportMatrix	Foundation
export_neighbourhood_array	IstExportNbhoodArray	Foundation
export_roi_array	ItbaExportROIArray	Technical
export_static_array	IstExportStaticArray	Technical
export_tuple	IstExportTuple	Foundation
export_value_bounds	IstExportValueBounds	Scientific
extract_pixel_plane	InExtractPixelPlane	Foundation
extract_pixel_plane_roi	InExtractPixelPlaneROI	Technical
extract_pixel_plane_roi_vreg	InExtractPixelPlaneROIVReg	Full
extract_pixel_plane_vreg	InExtractPixelPlaneVReg	Full
extrema	IvExtrema	Foundation
false_colour	InFalseColour	Scientific
false_colour_vreg	InFalseColourVReg	Full
fill_region	InFillRegion	Technical
fill_region_roi	InFillRegionROI	Technical
fill_region_roi_vreg	InFillRegionROIVReg	Full
fill_region_vreg	InFillRegionVReg	Full
filter_butterworth	InFilterButterworth	Technical
filter_butterworth_vreg	InFilterButterworthVReg	Full
filter_gaussian	InFilterGauss	Technical
filter_gaussian_vreg	InFilterGaussVReg	Full
filter_inverse	InFilterInverse	Scientific
filter_inverse_vreg	InFilterInverseVReg	Full
filter_matched	InFilterMatched	Scientific

Table 3 - Function names alphabetical by function name (continued)

Function Name	Bound Name	Least Profile
filter_matched_vreg	InFilterMatchedVReg	Full
filter_wiener	InFilterWiener	Scientific
filter_wiener_vreg	InFilterWienerVReg	Full
filter_zonal	InFilterZonal	Technical
filter_zonal_vreg	InFilterZonalVReg	Full
filtering_homomorphic	InFilteringHomomorphic	Scientific
filtering_homomorphic_vreg	InFilteringHomomorphicVReg	Full
filtering_linear	InFilteringLin	Technical
filtering_linear_vreg	InFilteringLinVReg	Full
filtering_median	InFilteringMedian	Technical
filtering_median_vreg	InFilteringMedianVReg	Full
filtering_pseudomedian	InFilteringPseudomedian	Scientific
filtering_pseudomedian_vreg	InFilteringPseudomedianVReg	Full
filtering_rank_order	InFilteringRankOrder	Scientific
filtering_rank_order_vreg	InFilteringRankOrderVReg	Full
flip_spin_transpose	InFlipSpinTranspose	Foundation
flip_spin_transpose_roi	InFlipSpinTransposeROI	Foundation
flip_spin_transpose_roi_vreg	InFlipSpinTransposeROI VReg	Full
flip_spin_transpose_vreg	InFlipSpinTransposeVReg	Full
gamma_correction	InGammaCorrection	Technical
gamma_correction_vreg	InGammaCorrectionVReg	Full
generate_and_write_iif_image	InGenerateWriteImage	Foundation
get_colour_pixel	IvGetColourPixel	Foundation
get_colour_pixel_vreg	IvGetColourPixelVReg	Full
get_iif_entity_component	InGetEntityCmp	Foundation
get_iif_syntax_entity_value	IstGetEntityValue	Foundation
get_image_subrange	InGetImageSubrange	Foundation
get_pixel	IstGetPixel	Foundation
get_pixel_array	IstGetPixelArray	Foundation
get_pixel_array_roi	ItbaGetPixelArrayROI	Foundation
get_pixel_array_roi_vreg	InGetPixelArrayROI VReg	Full
get_pixel_record	IstGetPixelRecord	Technical
get_pixel_record_roi	ItbaGetPixelRecordROI	Technical
get_pixel_record_roi_vreg	InGetPixelRecordROI VReg	Full
get_pixel_record_vreg	InGetPixelVReg	Full
get_pixel_roi	IbGetPixelROI	Foundation
histogram_1d	InHist1D	Foundation
histogram_1d_vreg	InHist1DVReg	Full
histogram_2d	InHist2D	Scientific
histogram_2d_vreg	InHist2DVReg	Full
histogram_modification	InHistModification	Scientific
histogram_modification_vreg	InHistModificationVReg	Full
histogram_scaling	InHistScaling	Scientific
histogram_scaling_vreg	InHistScalingVReg	Full
histogram_shape_1d	ItraHistShape1D	Scientific
histogram_shape_2d	ItraHistShape2D	Scientific
hit_or_miss	InHitMiss	Technical
hit_or_miss_roi	InHitMissROI	Technical

Table 3 - Function names alphabetical by function name (continued)

Function Name	Bound Name	Least Profile
hit_or_miss_roi_vreg	InHitMissROIVReg	Full
hit_or_miss_vreg	InHitMissVReg	Full
hough_transform	InHoughTransform	Scientific
hough_transform_vreg	InHoughTransformVReg	Full
image_bar_chart	InImageBarChart	Technical
image_bar_chart_vreg	InImageBarChartVReg	Full
image_constant	InImageConstant	Foundation
image_constant_vreg	InImageConstantVReg	Full
image_gaussian	InImageGauss	Scientific
image_gaussian_vreg	InImageGaussVReg	Full
image_grey_scale	InImageGreyScale	Technical
image_grey_scale_vreg	InImageGreyScaleVReg	Full
image_random_number	InImageRandomNumber	Scientific
image_random_number_vreg	InImageRandomNumberVReg	Full
import_histogram	InImportHist	Foundation
import_histogram_vreg	InImportHistVReg	Full
import_iif_data_from_piks	InImportIIFDataFromPIKS	Foundation - with PIKS only
import_image	InImportImage	Foundation
import_image_vreg	InImportImageVReg	Full
import_lut	InImportLUT	Foundation
import_lut_vreg	InImportLUTVReg	Full
import_matrix	InImportMatrix	Foundation
import_matrix_vreg	InImportMatrixVReg	Full
import_neighbourhood_array	InImportNbhoodArray	Foundation
import_neighbourhood_array_vreg	InImportNbhoodArrayVReg	Full
import_roi_array	InImportROIArray	Technical
import_roi_array_vreg	InImportROIArrayVReg	Full
import_static_array	InImportStaticArray	Technical
import_static_array_vreg	InImportStaticArrayVReg	Full
import_tuple	InImportTuple	Foundation
import_tuple_vreg	InImportTupleVReg	Full
import_value_bounds	InImportValueBounds	Scientific
import_value_bounds_vreg	InImportValueBoundsVReg	Full
impulse_boxcar	InImpulseBoxcar	Scientific
impulse_boxcar_vreg	InImpulseBoxcarVReg	Full
impulse_derivative_of_gaussian	InImpulseDerivGauss	Scientific
impulse_derivative_of_gaussian_vreg	InImpulseDerivGaussVReg	Full
impulse_difference_of_gaussians	InImpulseDiffGauss	Scientific
impulse_difference_of_gaussians_vreg	InImpulseDiffGaussVReg	Full
impulse_elliptical	InImpulseElliptical	Technical
impulse_elliptical_vreg	InImpulseEllipticalVReg	Full
impulse_gaussian	InImpulseGauss	Technical
impulse_gaussian_vreg	InImpulseGaussVReg	Full
impulse_laplacian_of_gaussian	InImpulseLaplaceGauss	Scientific
impulse_laplacian_of_gaussian_vreg	InImpulseLaplaceGaussVReg	Full
impulse_pyramid	InImpulsePyramid	Scientific
impulse_pyramid_vreg	InImpulsePyramidVReg	Full

Table 3 - Function names alphabetical by function name (continued)

Function Name	Bound Name	Least Profile
impulse_rectangular	InImpulseRectangular	Foundation
impulse_rectangular_vreg	InImpulseRectangularVReg	Full
impulse_sinc	InImpulseSinc	Scientific
impulse_sinc_vreg	InImpulseSincVReg	Full
input_object	IsnInputObject	Foundation - with IIF only
input_object_vreg	InInputObjectVReg	Full - with IIF only
inquire_chain_environment	IvInquireChainEnvironment	Full
inquire_chain_status	IvInquireChainStatus	Full
inquire_elements	IvInquireElements	Foundation
inquire_iif_gateway_status	IvInquireGatewayStatus	Foundation
inquire_iif_syntax_component_type_definition	IvCmpTypeDef	Foundation
inquire_iif_syntax_entity_choice	IvInquireEntityChc	Foundation
inquire_iif_syntax_entity_number	IvInquireEntityNumber	Foundation
inquire_iif_syntax_entity_optional	IvInquireEntityOptional	Foundation
inquire_iif_syntax_entity_sequence	IbInquireEntitySeq	Foundation
inquire_iif_syntax_entity_type_definition	IvInquireEntityTypeDef	Foundation
inquire_iif_syntax_entity_validity	IvInquireEntityValidity	Foundation
inquire_image	IvInquireImage	Foundation
inquire_index_assignment	IvInquireIndexAssignment	Technical
inquire_non_image_object	IvInquireNonImageObject	Foundation
inquire_piks_implementation	IvInquirePIKSImpl	Foundation
inquire_piks_status	IvInquirePIKSStatus	Foundation
inquire_port	IvInquirePort	Foundation
inquire_repository	IvInquireRepository	Foundation
inquire_resampling	IvInquireResampling	Foundation
inquire_supported_functionality	IvInquireFunctionality	Foundation
insert_pixel_plane	InInsertPixelPlane	Foundation
insert_pixel_plane_roi	InInsertPixelPlaneROI	Technical
insert_pixel_plane_roi_vreg	InInsertPixelPlaneROIVReg	Full
insert_pixel_plane_vreg	InInsertPixelPlaneVReg	Full
label_objects	InLabelObjects	Scientific
label_objects_vreg	InLabelObjectsVReg	Full
laws_texture_features	InLawsTextureFeatures	Scientific
laws_texture_features_vreg	InLawsTextureFeaturesVReg	Full
level_slice	InLevelSlice	Scientific
level_slice_vreg	InLevelSliceVReg	Full
line_profile	InLineProfile	Scientific
line_profile_vreg	ItraLineProfileVReg	Full
lookup	InLookup	Foundation
lookup_interpolated	InLookupInterp	Technical
lookup_interpolated_vreg	InLookupInterpVReg	Full
lookup_vreg	InLookupVReg	Full
luminance_generation	InLumGeneration	Technical
luminance_generation_vreg	InLumGenerationVReg	Full
markov_spectrum	InMarkovSpectrum	Scientific
markov_spectrum_vreg	InMarkovSpectrumVReg	Full
moments	IvMoments	Foundation

Table 3 - Function names alphabetical by function name (continued)

Function Name	Bound Name	Least Profile
moments_vreg	IvMomentsVReg	Full
monadic_arithmetic	InMonadicArith	Foundation
monadic_arithmetic_vreg	InMonadicArithVReg	Full
monadic_complex	InMonadicComplex	Technical
monadic_complex_vreg	InMonadicComplexVReg	Full
monadic_logical	InMonadicLogical	Foundation
monadic_logical_roi	InMonadicLogicalROI	Technical
monadic_logical_roi_vreg	InMonadicLogicalROIReg	Full
monadic_logical_vreg	InMonadicLogicalVReg	Full
morphic_processor	InMorphicProcessor	Foundation
morphic_processor_roi	InMorphicProcessorROI	Technical
morphic_processor_roi_vreg	InMorphicProcessorROIReg	Full
morphic_processor_vreg	InMorphicProcessorVReg	Full
morphology	InMorphology	Technical
morphology_roi	InMorphologyROI	Technical
morphology_roi_vreg	InMorphologyROIReg	Full
morphology_vreg	InMorphologyVReg	Full
neighbour_count	InNbrCount	Scientific
neighbour_count_vreg	InNbrCountVReg	Full
noise_combination	InNoiseCombination	Scientific
noise_combination_vreg	InNoiseCombinationVReg	Full
open_close	InOpenClose	Scientific
open_close_roi	InOpenCloseROI	Scientific
open_close_roi_vreg	InOpenCloseROIReg	Full
open_close_vreg	InOpenCloseVReg	Full
open_iif_gateway	IvOpenGateway	Foundation
open_piks	IvOpenPIKS	Foundation
open_port	InOpenPort	Foundation
outlier_removal	InOutlierRemoval	Technical
outlier_removal_vreg	InOutlierRemovalVReg	Full
output_object	InOutputObject	Foundation - with IIF only
paint_pixels	InPaintPixels	Technical
paint_pixels_vreg	InPaintPixelsVReg	Full
perimeter_code_generator	ItuaPerimeterCodeGenerator	Scientific
polar_to_cartesian	InPolarToCartesian	Scientific
polar_to_cartesian_vreg	InPolarToCartesianVReg	Full
power_law_scaling	InPowerLawScaling	Technical
power_law_scaling_vreg	InPowerLawScalingVReg	Full
pseudocolour	InPseudocolour	Technical
pseudocolour_vreg	InPseudocolourVReg	Full
put_colour_pixel	InPutColourPixel	Foundation
put_colour_pixel_vreg	InPutColourPixelVReg	Full
put_iif_syntax_entity_value	InPutEntityValue	Foundation
put_image_subrange	ItzPutImageSubrange	Foundation
put_pixel	InPutPixel	Foundation
put_pixel_array	InPutPixelArray	Foundation
put_pixel_array_roi	InPutPixelArrayROI	Technical

Table 3 - Function names alphabetical by function name (continued)

Function Name	Bound Name	Least Profile
put_pixel_array_roi_vreg	InPutPixelArrayROIvReg	Full
put_pixel_array_vreg	InPutPixelArrayVReg	Full
put_pixel_record	InPutPixelRecord	Technical
put_pixel_record_roi	InPutPixelRecordROI	Technical
put_pixel_record_roi_vreg	InPutPixelRecordROIvReg	Full
put_pixel_record_vreg	InPutPixelRecordVReg	Full
put_pixel_roi	InPutPixelROI	Technical
put_pixel_roi_vreg	InPutPixelROIvReg	Full
put_pixel_vreg	InPutPixelVReg	Full
read_and_parse_iif_image	InReadParseImage	Foundation
release_iif_syntax_entity_identifier	IvReleaseEntityId	Foundation
rescale	InRescale	Foundation
rescale_roi	InRescaleROI	Foundation
rescale_roi_vreg	InRescaleROIvReg	Full
rescale_vreg	InRescaleVReg	Full
reset_iif_gateway	IvResetGateway	Foundation
reset_port	IvResetPort	Foundation
resize	InResize	Foundation
resize_roi	InResizeROI	Foundation
resize_roi_vreg	InResizeROIvReg	Full
resize_vreg	InResizeVReg	Full
return_repository_id	InReturnRepositoryId	Foundation
return_repository_id_vreg	InReturnRepositoryIdVReg	Full
roi_coordinate_array	InROICoordinateArray	Technical
roi_coordinate_array_vreg	InROICoordinateArrayVReg	Full
roi_elliptical	InROIElliptical	Technical
roi_elliptical_vreg	InROIEllipticalVReg	Full
roi_polygon	InROIPolygon	Technical
roi_polygon_vreg	InROIPolygonVReg	Full
roi_rectangular	InROIRectangular	Foundation
roi_rectangular_vreg	InROIRectangularVReg	Full
rotate	InRotate	Foundation
rotate_roi	InRotateROI	Technical
rotate_roi_vreg	InRotateROIvReg	Full
rotate_vreg	InRotateVReg	Full
rubber_band_scaling	InRubberBandScaling	Scientific
rubber_band_scaling_vreg	InRubberBandScalingVReg	Full
sequence_average	InSeqAverage	Scientific
sequence_average_vreg	InSeqAverageVReg	Full
sequence_karhunen_loeve	InSeqKarhunenLoeve	Scientific
sequence_karhunen_loeve_vreg	InSeqKarhunenLoeveVReg	Full
sequence_running_measures	InSeqRunningMeasures	Scientific
sequence_running_measures_vreg	InSeqRunningMeasuresVReg	Full
set_error_handler	ItfSetErrorHandler	Foundation
set_globals	IvSetGlobals	Foundation
set_globals_vreg	IvSetGlobalsVReg	Full
set_image_attributes	InSetImageAttrs	Foundation

Table 3 - Function names alphabetical by function name (continued)

Function Name	Bound Name	Least Profile
set_image_attributes_vreg	InSetImageAttrsVReg	Full
set_index_assignment	IvSetIndexAssignment	Technical
set_index_assignment_vreg	IvSetIndexAssignmentVReg	Full
shape_metrics	ItraShapeMetrics	Scientific
spatial_moments_invariant	ItraSpatialMomentsInvariant	Scientific
spatial_moments_scaled	ItraSpatialMomentsScaled	Scientific
split_image	InSplitImage	Foundation
split_image_vreg	InSplitImageVReg	Full
subsample	InSubsample	Foundation
subsample_roi	InSubsampleROI	Technical
subsample_roi_vreg	InSubsampleROIVReg	Full
subsample_vreg	InSubsampleVReg	Full
synchronize	IvSynch	Scientific
template_match	InTemplMatch	Scientific
template_match_vreg	InTemplMatchVReg	Full
threshold	InThreshold	Foundation
threshold_vreg	InThresholdVReg	Full
transform_cosine	InTransformCosine	Technical
transform_cosine_vreg	InTransformCosineVReg	Full
transform_fourier	InTransformFourier	Technical
transform_fourier_vreg	InTransformFourierVReg	Full
transform_hadamard	InTransformHadamard	Scientific
transform_hadamard_vreg	InTransformHadamardVReg	Full
transform_hartley	InTransformHartley	Scientific
transform_hartley_vreg	InTransformHartleyVReg	Full
translate	InTranslate	Foundation
translate_roi	InTranslateROI	Foundation
translate_roi_vreg	InTranslateROIVReg	Full
translate_vreg	InTranslateVReg	Full
unary_integer	InUnaryInteger	Foundation
unary_integer_vreg	InUnaryIntegerVReg	Full
unary_real	InUnaryReal	Technical
unary_real_vreg	InUnaryRealVReg	Full
unsharp_mask	InUnsharpMask	Technical
unsharp_mask_vreg	InUnsharpMaskVReg	Full
value_bounds	InValueBounds	Scientific
value_bounds_vreg	InValueBoundsVReg	Full
vreg_alter	InVRegAlter	Full
vreg_alter_vreg	InVRegAlterVReg	Full
vreg_clear	InVRegClear	Full
vreg_clear_vreg	InVRegClearVReg	Full
vreg_conditional	IvVRegConditional	Full
vreg_conditional_vreg	IvVRegConditionalVReg	Full
vreg_copy	InVRegCopy	Full
vreg_copy_vreg	InVRegCopyVReg	Full
vreg_delete	InVRegDelete	Full
vreg_delete_vreg	InVRegDeleteVReg	Full

Table 3 - Function names alphabetical by function name (concluded)

Function Name	Bound Name	Least Profile
vreg_get	InVRegGet	Full
vreg_set	InVRegSet	Full
vreg_set_vreg	InVRegSetVReg	Full
vreg_wait	InVRegWait	Full
vreg_wait_vreg	InVRegWaitVReg	Full
wallis_statistical_differencing	InWallisStatDiff	Technical
wallis_statistical_differencing_vreg	InWallisStatDiffVReg	Full
warp_control_point	InWarpControlPoint	Technical
warp_control_point_roi	InWarpControlPointROI	Technical
warp_control_point_roi_vreg	InWarpControlPointROIReg	Full
warp_control_point_vreg	InWarpControlPointVReg	Full
warp_lut	InWarpLUT	Technical
warp_lut_roi	InWarpLUTROI	Technical
warp_lut_roi_vreg	InWarpLUTROIReg	Full
warp_lut_vreg	InWarpLUTVReg	Full
warp_polynomial	InWarpPolynomial	Technical
warp_polynomial_roi	InWarpPolynomialROI	Technical
warp_polynomial_roi_vreg	InWarpPolynomialROIReg	Full
warp_polynomial_vreg	InWarpPolynomialVReg	Full
window_level	InWindowLevel	Foundation
window_level_vreg	InWindowLevelVReg	Full
window_statistics	InWindowStatistics	Scientific
window_statistics_vreg	InWindowStatisticsVReg	Full
windowing_function	InWindowingFunction	Scientific
windowing_function_vreg	InWindowingFunctionVReg	Full
z_merge	InZMerge	Technical
z_merge_vreg	InZMergeVReg	Full
zoom	InZoom	Foundation
zoom_roi	InZoomROI	Foundation
zoom_roi_vreg	InZoomROIReg	Full
zoom_vreg	InZoomVReg	Full
3d_slice	In3DSlice	Scientific
3d_slice_vreg	In3DSliceVReg	Full

4.2.2 Alphabetical by bound name

Table 4 - Function names alphabetical by bound name

Bound Name	Function Name	Least Profile
IbConstantPredicate	contsant_predicate	Technical
IbEqualPredicate	equal_predicate	Technical
IbErrorTest	error_test	Foundation
IbGetPixelROI	get_pixel_roi	Foundation
IbIdArrayEqual	composite_identifier_array_equal	Scientific
IbIdListEmpty	composite_identifier_list_empty	Scientific
IbIdListEqual	composite_identifier_list_equal	Scientific
IbInquireEntitySeq	inquire_iif_syntax_entity_sequence	Foundation
IdIdRecordEqual	composite_identifier_record_equal	Scientific
InAdaptiveHistEqual	adaptive_histogram_equalization	Scientific
InAdaptiveHistEqualVReg	adaptive_histogram_equalization_vreg	Full
InAllocateChain	allocate_chain	Full
InAllocateChainVReg	allocate_chain_vreg	Full
InAllocateHist	allocate_histogram	Foundation
InAllocateHistVReg	allocate_histogram_vreg	Full
InAllocateIdArray	allocate_composite_identifier_array	Scientific
InAllocateIdArrayVReg	allocate_composite_identifier_array_vreg	Full
InAllocateIdList	allocate_composite_identifier_list	Scientific
InAllocateIdListVReg	allocate_composite_identifier_list_vreg	Full
InAllocateIdRecord	allocate_composite_identifier_record	Scientific
InAllocateIdRecordVReg	allocate_composite_identifier_record_vreg	Full
InAllocateImage	allocate_image	Foundation
InAllocateImageVReg	allocate_image_vreg	Full
InAllocateLUT	allocate_lookup_table	Foundation
InAllocateLUTVReg	allocate_lookup_table_vreg	Full
InAllocateMatrix	allocate_matrix	Foundation
InAllocateMatrixVReg	allocate_matrix_vreg	Full
InAllocateNbhoodArray	allocate_neighbourhood_array	Foundation
InAllocateNbhoodArrayVReg	allocate_neighbourhood_array_vreg	Full
InAllocatePixelRecord	allocate_pixel_record	Technical
InAllocatePixelRecordVReg	allocate_pixel_record_vreg	Full
InAllocateROI	allocate_roi	Foundation
InAllocateROIVReg	allocate_roi_vreg	Full
InAllocateStaticArray	allocate_static_array	Technical
InAllocateStaticArrayVReg	allocate_static_array_vreg	Full
InAllocateTuple	allocate_tuple	Foundation
InAllocateTupleVReg	allocate_tuple_vreg	Full
InAllocateValueBounds	allocate_value_bounds	Scientific
InAllocateValueBoundsVReg	allocate_value_bounds_vreg	Full
InAllocateVReg	allocate_vreg	Full
InAllocateVRegVReg	allocate_vreg_vreg	Full
InAlphaBlendConstant	alpha_blend_constant	Foundation
InAlphaBlendConstantVReg	alpha_blend_constant_vreg	Full
InAlphaBlendVariable	alpha_blend_variable	Technical
InAlphaBlendVariableVReg	alpha_blend_variable_vreg	Full

Table 4 - Function names alphabetical by bound name (continued)

Bound Name	Function Name	Least Profile
InArrayToLUT	array_to_lut	Foundation
InArrayToLUTVReg	array_to_lut_vreg	Full
InAttachEntity	attach_iif_syntax_entity	Foundation
InBindMatchPoint	bind_match_point	Technical
InBindMatchPointVReg	bind_match_point_vreg	Full
InBindROI	bind_roi	Foundation
InBindROIVReg	bind_roi_vreg	Full
InBitShift	bit_shift	Foundation
InBitShiftVReg	bit_shift_vreg	Full
InCartesianToPolar	cartesian_to_polar	Scientific
InCartesianToPolarVReg	cartesian_to_polar_vreg	Full
InChainAbort	chain_abort	Full
InChainAbortVReg	chain_abort_vreg	Full
InChainBegin	chain_begin	Full
InChainDelete	chain_delete	Full
InChainDeleteVReg	chain_delete_vreg	Full
InChainEnd	chain_end	Full
InChainExecute	chain_execute	Full
InChainExecuteVReg	chain_execute_vreg	Full
InChainReload	chain_reload	Full
InChainReloadVReg	chain_reload_vreg	Full
InClassifierBayes	classifier_bayes	Scientific
InClassifierBayesVReg	classifier_bayes_vreg	Full
InClassifierNearestNbr	classifier_nearest_neighbour	Scientific
InClassifierNearestNbrVReg	classifier_nearest_neighbour_vreg	Full
InColourConvLin	colour_conversion_linear	Foundation
InColourConvLinVReg	colour_conversion_linear_vreg	Full
InColourConvNonlin	colour_conversion_nonlinear	Technical
InColourConvNonlinVReg	colour_conversion_nonlinear_vreg	Full
InColourConvSubtractive	colour_conversion_subtractive	Foundation
InColourConvSubtractiveVReg	colour_conversion_subtractive_vreg	Full
InColourLookupInterp	colour_lookup_interpolated	Technical
InColourLookupInterpVReg	colour_lookup_interpolated_vreg	Full
InComplement	complement	Foundation
InComplementROI	complement_roi	Technical
InComplementROIVReg	complement_roi_vreg	Full
InComplementVReg	complement_vreg	Full
InComplexComposition	complex_composition	Technical
InComplexCompositionVReg	complex_composition_vreg	Full
InComplexConjugate	complex_conjugate	Technical
InComplexConjugateVReg	complex_conjugate_vreg	Full
InComplexMagnitude	complex_magnitude	Technical
InComplexMagnitudeVReg	complex_magnitude_vreg	Full
InCompressEntity	compress_iif_syntax_entity	Foundation
InConstantPredicateVReg	constant_predicate_vreg	Full
InConvertArrayToImage	convert_array_to_image	Foundation
InConvertArrayToImageVReg	convert_array_to_image_vreg	Full

Table 4 - Function names alphabetical by bound name (continued)

Bound Name	Function Name	Least Profile
InConvertImageDatatype	convert_image_datatype	Foundation
InConvertImageDatatypeVReg	convert_image_datatype_vreg	Full
InConvertImageToArray	convert_image_to_array	Foundation
InConvertImageToArrayVReg	convert_image_to_array_vreg	Full
InConvertImageToROI	convert_image_to_roi	Technical
InConvertImageToROIVReg	convert_image_to_roi_vreg	Full
InConvertROIToImage	convert_roi_to_image	Foundation
InConvertROIToImageVReg	convert_roi_to_image_vreg	Full
InConvolve2D	convolve_2d	Foundation
InConvolve2DVReg	convolve_2d_vreg	Full
InConvolve5D	convolve_5d	Technical
InConvolve5DVReg	convolve_5d_vreg	Full
InCopyEntity	copy_iif_syntax_entity	Foundation
InCopyWindow	copy_window	Foundation
InCopyWindowROI	copy_window_roi	Technical
InCopyWindowROIVReg	copy_window_roi_vreg	Full
InCopyWindowVReg	copy_window_vreg	Full
InCreateEntity	create_iif_syntax_entity	Foundation
InCreateTuple	create_tuple	Foundation
InCreateTupleVReg	create_tuple_vreg	Full
InCrossCorrelation	cross_correlation	Scientific
InCrossCorrelationVReg	cross_correlation_vreg	Full
InDefineSubImage	define_sub_image	Foundation
InDefineSubImageVReg	define_sub_image_vreg	Full
InDeleteAtomicSeqElement	delete_iif_atomic_sequence_element	Foundation
InDetachEntity	detach_iif_syntax_entity	Foundation
InDiffMeasuresVReg	difference_measures_vreg	Full
InDiffuse	diffuse	Foundation
InDiffuseVReg	diffuse_vreg	Full
InDither	dither	Foundation
InDitherVReg	dither_vreg	Full
InDrawPixels	draw_pixels	Foundation
InDrawPixelsROI	draw_pixels_roi	Technical
InDrawPixelsROIVReg	draw_pixels_roi_vreg	Full
InDrawPixelsVReg	draw_pixels_vreg	Full
InDyadicArith	dyadic_arithmetic	Foundation
InDyadicArithVReg	dyadic_arithmetic_vreg	Full
InDyadicComplex	dyadic_complex	Technical
InDyadicComplexVReg	dyadic_complex_vreg	Full
InDyadicLogical	dyadic_logical	Foundation
InDyadicLogicalROI	dyadic_logical_roi	Technical
InDyadicLogicalROIVReg	dyadic_logical_roi_vreg	Full
InDyadicLogicalVReg	dyadic_logical_vreg	Full
InDyadicPredicate	dyadic_predicate	Foundation
InDyadicPredicateVReg	dyadic_predicate_vreg	Full
InEdgeDetectOrthGrad	edge_detection_orthogonal_gradient	Scientific
InEdgeDetectOrthGradVReg	edge_detection_orthogonal_gradient_vreg	Full

Table 4 - Function names alphabetical by bound name (continued)

Bound Name	Function Name	Least Profile
InEdgeDetect2ndDeriv	edge_detection_second_derivative	Scientific
InEdgeDetect2ndDerivVReg	edge_detection_second_derivative_vreg	Full
InEdgeDetectTemplateGrad	edge_detection_template_gradient	Scientific
InEdgeDetectTemplateGradVReg	edge_detection_template_gradient_vreg	Full
InEqualPredicateVReg	equal_predicate_vreg	Full
InErosionDilateBool	erosion_dilation_boolean	Technical
InErosionDilateBoolROI	erosion_dilation_boolean_roi	Technical
InErosionDilateBoolROIVReg	erosion_dilation_boolean_roi_vreg	Full
InErosionDilateBoolVReg	erosion_dilation_boolean_vreg	Full
InErosionDilateGrey	erosion_dilation_grey	Technical
InErosionDilateGreyVReg	erosion_dilation_grey_vreg	Full
InErrorFunctionScaling	error_function_scaling	Scientific
InErrorFunctionScalingVReg	error_function_scaling_vreg	Full
InErrorTestVReg	error_test_vreg	Full
InExportIIFDataToPIKS	export_iif_data_to_piks	Foundation - with PIKS only
InExtractPixelPlane	extract_pixel_plane	Foundation
InExtractPixelPlaneROI	extract_pixel_plane_roi	Technical
InExtractPixelPlaneROIVReg	extract_pixel_plane_roi_vreg	Full
InExtractPixelPlaneVReg	extract_pixel_plane_vreg	Full
InFalseColour	false_colour	Scientific
InFalseColourVReg	false_colour_vreg	Full
InFillRegion	fill_region	Technical
InFillRegionROI	fill_region_roi	Technical
InFillRegionROIVReg	fill_region_roi_vreg	Full
InFillRegionVReg	fill_region_vreg	Full
InFilterButterworth	filter_butterworth	Technical
InFilterButterworthVReg	filter_butterworth_vreg	Full
InFilterGauss	filter_gaussian	Technical
InFilterGaussVReg	filter_gaussian_vreg	Full
InFilteringHomomorphic	filtering_homomorphic	Scientific
InFilteringHomomorphicVReg	filtering_homomorphic_vreg	Full
InFilteringLin	filtering_linear	Technical
InFilteringLinVReg	filtering_linear_vreg	Full
InFilteringMedian	filtering_median	Technical
InFilteringMedianVReg	filtering_median_vreg	Full
InFilteringPseudomedian	filtering_pseudomedian	Scientific
InFilteringPseudomedianVReg	filtering_pseudomedian_vreg	Full
InFilteringRankOrder	filtering_rank_order	Scientific
InFilteringRankOrderVReg	filtering_rank_order_vreg	Full
InFilterInverse	filter_inverse	Scientific
InFilterInverseVReg	filter_inverse_vreg	Full
InFilterMatched	filter_matched	Scientific
InFilterMatchedVReg	filter_matched_vreg	Full
InFilterWiener	filter_wiener	Scientific
InFilterWienerVReg	filter_wiener_vreg	Full
InFilterZonal	filter_zonal	Technical
InFilterZonalVReg	filter_zonal_vreg	Full

Table 4 - Function names alphabetical by bound name (continued)

Bound Name	Function Name	Least Profile
InFlipSpinTranspose	flip_spin_transpose	Foundation
InFlipSpinTransposeROI	flip_spin_transpose_roi	Foundation
InFlipSpinTransposeROIReg	flip_spin_transpose_roi_vreg	Full
InFlipSpinTransposeVReg	flip_spin_transpose_vreg	Full
InGammaCorrection	gamma_correction	Technical
InGammaCorrectionVReg	gamma_correction_vreg	Full
InGenerateWriteImage	generate_and_write_iif_image	Foundation
InGetEntityCmp	get_iif_entity_component	Foundation
InGetImageSubrange	get_image_subrange	Foundation
InGetPixelArrayROIReg	get_pixel_array_roi_vreg	Full
InGetPixelRecordROIReg	get_pixel_record_roi_vreg	Full
InGetPixelVReg	get_pixel_record_vreg	Full
InHist1D	histogram_1d	Foundation
InHist1DVReg	histogram_1d_vreg	Full
InHist2D	histogram_2d	Scientific
InHist2DVReg	histogram_2d_vreg	Full
InHistModification	histogram_modification	Scientific
InHistModificationVReg	histogram_modification_vreg	Full
InHistScaling	histogram_scaling	Scientific
InHistScalingVReg	histogram_scaling_vreg	Full
InHitMiss	hit_or_miss	Technical
InHitMissROI	hit_or_miss_roi	Technical
InHitMissROIReg	hit_or_miss_roi_vreg	Full
InHitMissVReg	hit_or_miss_vreg	Full
InHoughTransform	hough_transform	Scientific
InHoughTransformVReg	hough_transform_vreg	Full
InIdArrayEqualVReg	composite_identifier_array_equal_vreg	Full
InIdArrayGet	composite_identifier_array_get	Scientific
InIdArrayGetVReg	composite_identifier_array_get_vreg	Full
InIdArrayPut	composite_identifier_array_put	Scientific
InIdArrayPutVReg	composite_identifier_array_put_vreg	Full
InIdListEmptyVReg	composite_identifier_list_empty_vreg	Full
InIdListEqualVReg	composite_identifier_list_equal_vreg	Full
InIdListGet	composite_identifier_list_get	Scientific
InIdListGetVReg	composite_identifier_list_get_vreg	Full
InIdListInsert	composite_identifier_list_insert	Scientific
InIdListInsertVReg	composite_identifier_list_insert_vreg	Full
InIdListRemove	composite_identifier_list_remove	Scientific
InIdListRemoveVReg	composite_identifier_list_remove_vreg	Full
InIdRecordEqualVReg	composite_identifier_record_equal_vreg	Full
InIdRecordGet	composite_identifier_record_get	Scientific
InIdRecordGetVReg	composite_identifier_record_get_vreg	Full
InIdRecordPut	composite_identifier_record_put	Scientific
InIdRecordPutVReg	composite_identifier_record_put_vreg	Full
InImageBarChart	image_bar_chart	Technical
InImageBarChartVReg	image_bar_chart_vreg	Full
InImageConstant	image_constant	Foundation

Table 4 - Function names alphabetical by bound name (continued)

Bound Name	Function Name	Least Profile
InImageConstantVReg	image_constant_vreg	Full
InImageGauss	image_gaussian	Scientific
InImageGaussVReg	image_gaussian_vreg	Full
InImageGreyScale	image_grey_scale	Technical
InImageGreyScaleVReg	image_grey_scale_vreg	Full
InImageRandomNumber	image_random_number	Scientific
InImageRandomNumberVReg	image_random_number_vreg	Full
InImportHist	import_histogram	Foundation
InImportHistVReg	import_histogram_vreg	Full
InImportIIFDataFromPIKS	import_iif_data_from_piks	Foundation - with PIKS only
InImportImage	import_image	Foundation
InImportImageVReg	import_image_vreg	Full
InImportLUT	import_lut	Foundation
InImportLUTVReg	import_lut_vreg	Full
InImportMatrix	import_matrix	Foundation
InImportMatrixVReg	import_matrix_vreg	Full
InImportNbhoodArray	import_neighbourhood_array	Foundation
InImportNbhoodArrayVReg	import_neighbourhood_array_vreg	Full
InImportROIArray	import_roi_array	Technical
InImportROIArrayVReg	import_roi_array_vreg	Full
InImportStaticArray	import_static_array	Technical
InImportStaticArrayVReg	import_static_array_vreg	Full
InImportTuple	import_tuple	Foundation
InImportTupleVReg	import_tuple_vreg	Full
InImportValueBounds	import_value_bounds	Scientific
InImportValueBoundsVReg	import_value_bounds_vreg	Full
InImpulseBoxcar	impulse_boxcar	Scientific
InImpulseBoxcarVReg	impulse_boxcar_vreg	Full
InImpulseDerivGauss	impulse_derivative_of_gaussian	Scientific
InImpulseDerivGaussVReg	impulse_derivative_of_gaussian_vreg	Full
InImpulseDiffGauss	impulse_difference_of_gaussians	Scientific
InImpulseDiffGaussVReg	impulse_difference_of_gaussians_vreg	Full
InImpulseElliptical	impulse_elliptical	Technical
InImpulseEllipticalVReg	impulse_elliptical_vreg	Full
InImpulseGauss	impulse_gaussian	Technical
InImpulseGaussVReg	impulse_gaussian_vreg	Full
InImpulseLaplaceGauss	impulse_laplacian_of_gaussian	Scientific
InImpulseLaplaceGaussVReg	impulse_laplacian_of_gaussian_vreg	Full
InImpulsePyramid	impulse_pyramid	Scientific
InImpulsePyramidVReg	impulse_pyramid_vreg	Full
InImpulseRectangular	impulse_rectangular	Foundation
InImpulseRectangularVReg	impulse_rectangular_vreg	Full
InImpulseSinc	impulse_sinc	Scientific
InImpulseSincVReg	impulse_sinc_vreg	Full
InInputObjectVReg	input_object_vreg	Full - with IIF only
InInsertPixelPlane	insert_pixel_plane	Foundation
InInsertPixelPlaneROI	insert_pixel_plane_roi	Technical

Table 4 - Function names alphabetical by bound name (continued)

Bound Name	Function Name	Least Profile
InInsertPixelPlaneROIReg	insert_pixel_plane_roi_vreg	Full
InInsertPixelPlaneVReg	insert_pixel_plane_vreg	Full
InLabelObjects	label_objects	Scientific
InLabelObjectsVReg	label_objects_vreg	Full
InLawsTextureFeatures	laws_texture_features	Scientific
InLawsTextureFeaturesVReg	laws_texture_features_vreg	Full
InLevelSlice	level_slice	Scientific
InLevelSliceVReg	level_slice_vreg	Full
InLineProfile	line_profile	Scientific
InLookup	lookup	Foundation
InLookupInterp	lookup_interpolated	Technical
InLookupInterpVReg	lookup_interpolated_vreg	Full
InLookupVReg	lookup_vreg	Full
InLumGeneration	luminance_generation	Technical
InLumGenerationVReg	luminance_generation_vreg	Full
InMarkovSpectrum	markov_spectrum	Scientific
InMarkovSpectrumVReg	markov_spectrum_vreg	Full
InMonadicArith	monadic_arithmetic	Foundation
InMonadicArithVReg	monadic_arithmetic_vreg	Full
InMonadicComplex	monadic_complex	Technical
InMonadicComplexVReg	monadic_complex_vreg	Full
InMonadicLogical	monadic_logical	Foundation
InMonadicLogicalROI	monadic_logical_roi	Technical
InMonadicLogicalROIReg	monadic_logical_roi_vreg	Full
InMonadicLogicalVReg	monadic_logical_vreg	Full
InMorphicProcessor	morphic_processor	Foundation
InMorphicProcessorROI	morphic_processor_roi	Technical
InMorphicProcessorROIReg	morphic_processor_roi_vreg	Full
InMorphicProcessorVReg	morphic_processor_vreg	Full
InMorphology	morphology	Technical
InMorphologyROI	morphology_roi	Technical
InMorphologyROIReg	morphology_roi_vreg	Full
InMorphologyVReg	morphology_vreg	Full
InNbrCount	neighbour_count	Scientific
InNbrCountVReg	neighbour_count_vreg	Full
InNoiseCombination	noise_combination	Scientific
InNoiseCombinationVReg	noise_combination_vreg	Full
InOpenClose	open_close	Scientific
InOpenCloseROI	open_close_roi	Scientific
InOpenCloseROIReg	open_close_roi_vreg	Full
InOpenCloseVReg	open_close_vreg	Full
InOpenPort	open_port	Foundation
InOutlierRemoval	outlier_removal	Technical
InOutlierRemovalVReg	outlier_removal_vreg	Full
InOutputObject	output_object	Foundation - with IIF only
InPaintPixels	paint_pixels	Technical
InPaintPixelsVReg	paint_pixels_vreg	Full

Table 4 - Function names alphabetical by bound name (continued)

Bound Name	Function Name	Least Profile
InPolarToCartesian	polar_to_cartesian	Scientific
InPolarToCartesianVReg	polar_to_cartesian_vreg	Full
InPowerLawScaling	power_law_scaling	Technical
InPowerLawScalingVReg	power_law_scaling_vreg	Full
InPseudocolour	pseudocolour	Technical
InPseudocolourVReg	pseudocolour_vreg	Full
InPutColourPixel	put_colour_pixel	Foundation
InPutColourPixelVReg	put_colour_pixel_vreg	Full
InPutEntityValue	put_iif_syntax_entity_value	Foundation
InPutPixel	put_pixel	Foundation
InPutPixelArray	put_pixel_array	Foundation
InPutPixelArrayROI	put_pixel_array_roi	Technical
InPutPixelArrayROIVReg	put_pixel_array_roi_vreg	Full
InPutPixelArrayVReg	put_pixel_array_vreg	Full
InPutPixelRecord	put_pixel_record	Technical
InPutPixelRecordROI	put_pixel_record_roi	Technical
InPutPixelRecordROIVReg	put_pixel_record_roi_vreg	Full
InPutPixelRecordVReg	put_pixel_record_vreg	Full
InPutPixelROI	put_pixel_roi	Technical
InPutPixelROIVReg	put_pixel_roi_vreg	Full
InPutPixelVReg	put_pixel_vreg	Full
InReadParseImage	read_and_parse_iif_image	Foundation
InRescale	rescale	Foundation
InRescaleROI	rescale_roi	Foundation
InRescaleROIVReg	rescale_roi_vreg	Full
InRescaleVReg	rescale_vreg	Full
InResize	resize	Foundation
InResizeROI	resize_roi	Foundation
InResizeROIVReg	resize_roi_vreg	Full
InResizeVReg	resize_vreg	Full
InReturnRepositoryId	return_repository_id	Foundation
InReturnRepositoryIdVReg	return_repository_id_vreg	Full
InROICoordinateArray	roi_coordinate_array	Technical
InROICoordinateArrayVReg	roi_coordinate_array_vreg	Full
InROIElliptical	roi_elliptical	Technical
InROIEllipticalVReg	roi_elliptical_vreg	Full
InROIPolygon	roi_polygon	Technical
InROIPolygonVReg	roi_polygon_vreg	Full
InROIRectangular	roi_rectangular	Foundation
InROIRectangularVReg	roi_rectangular_vreg	Full
InRotate	rotate	Foundation
InRotateROI	rotate_roi	Technical
InRotateROIVReg	rotate_roi_vreg	Full
InRotateVReg	rotate_vreg	Full
InRubberBandScaling	rubber_band_scaling	Scientific
InRubberBandScalingVReg	rubber_band_scaling_vreg	Full
InSeqAverage	sequence_average	Scientific

Table 4 - Function names alphabetical by bound name (continued)

Bound Name	Function Name	Least Profile
InSeqAverageVReg	sequence_average_vreg	Full
InSeqKarhunenLoeve	sequence_karhunen_loeve	Scientific
InSeqKarhunenLoeveVReg	sequence_karhunen_loeve_vreg	Full
InSeqRunningMeasures	sequence_running_measures	Scientific
InSeqRunningMeasuresVReg	sequence_running_measures_vreg	Full
InSetImageAttrs	set_image_attributes	Foundation
InSetImageAttrsVReg	set_image_attributes_vreg	Full
InSplitImage	split_image	Foundation
InSplitImageVReg	split_image_vreg	Full
InSubsample	subsample	Foundation
InSubsampleROI	subsample_roi	Technical
InSubsampleROIVReg	subsample_roi_vreg	Full
InSubsampleVReg	subsample_vreg	Full
InTemplMatch	template_match	Scientific
InTemplMatchVReg	template_match_vreg	Full
InThreshold	threshold	Foundation
InThresholdVReg	threshold_vreg	Full
InTransformCosine	transform_cosine	Technical
InTransformCosineVReg	transform_cosine_vreg	Full
InTransformFourier	transform_fourier	Technical
InTransformFourierVReg	transform_fourier_vreg	Full
InTransformHadamard	transform_hadamard	Scientific
InTransformHadamardVReg	transform_hadamard_vreg	Full
InTransformHartley	transform_hartley	Scientific
InTransformHartleyVReg	transform_hartley_vreg	Full
InTranslate	translate	Foundation
InTranslateROI	translate_roi	Foundation
InTranslateROIVReg	translate_roi_vreg	Full
InTranslateVReg	translate_vreg	Full
InUnaryInteger	unary_integer	Foundation
InUnaryIntegerVReg	unary_integer_vreg	Full
InUnaryReal	unary_real	Technical
InUnaryRealVReg	unary_real_vreg	Full
InUnsharpMask	unsharp_mask	Technical
InUnsharpMaskVReg	unsharp_mask_vreg	Full
InValueBounds	value_bounds	Scientific
InValueBoundsVReg	value_bounds_vreg	Full
InVRegAlter	vreg_alter	Full
InVRegAlterVReg	vreg_alter_vreg	Full
InVRegClear	vreg_clear	Full
InVRegClearVReg	vreg_clear_vreg	Full
InVRegCopy	vreg_copy	Full
InVRegCopyVReg	vreg_copy_vreg	Full
InVRegDelete	vreg_delete	Full
InVRegDeleteVReg	vreg_delete_vreg	Full
InVRegSet	vreg_set	Full
InVRegSetVReg	vreg_set_vreg	Full

Table 4 - Function names alphabetical by bound name (continued)

Bound Name	Function Name	Least Profile
InVRegWait	vreg_wait	Full
InVRegWaitVReg	vreg_wait_vreg	Full
InWallisStatDiff	wallis_statistical_differencing	Technical
InWallisStatDiffVReg	wallis_statistical_differencing_vreg	Full
InWarpControlPoint	warp_control_point	Technical
InWarpControlPointROI	warp_control_point_roi	Technical
InWarpControlPointROIVReg	warp_control_point_roi_vreg	Full
InWarpControlPointVReg	warp_control_point_vreg	Full
InWarpLUT	warp_lut	Technical
InWarpLUTROI	warp_lut_roi	Technical
InWarpLUTROIVReg	warp_lut_roi_vreg	Full
InWarpLUTVReg	warp_lut_vreg	Full
InWarpPolynomial	warp_polynomial	Technical
InWarpPolynomialROI	warp_polynomial_roi	Technical
InWarpPolynomialROIVReg	warp_polynomial_roi_vreg	Full
InWarpPolynomialVReg	warp_polynomial_vreg	Full
InWindowingFunction	windowing_function	Scientific
InWindowingFunctionVReg	windowing_function_vreg	Full
InWindowLevel	window_level	Foundation
InWindowLevelVReg	window_level_vreg	Full
InWindowStatistics	window_statistics	Scientific
InWindowStatisticsVReg	window_statistics_vreg	Full
InZMerge	z_merge	Technical
InZMergeVReg	z_merge_vreg	Full
InZoom	zoom	Foundation
InZoomROI	zoom_roi	Foundation
InZoomROIVReg	zoom_roi_vreg	Full
InZoomVReg	zoom_vreg	Full
In3DSlice	3d_slice	Scientific
In3DSliceVReg	3d_slice_vreg	Full
IrDiffMeasures	difference_measures	Technical
IsnInputObject	input_object	Foundation - with IIF only
IstDecompressEntity	decompress_iif_syntax_entity	Foundation
IstExportImage	export_image	Foundation
IstExportLUT	export_lut	Foundation
IstExportMatrix	export_matrix	Foundation
IstExportNbhoodArray	export_neighbourhood_array	Foundation
IstExportStaticArray	export_static_array	Technical
IstExportTuple	export_tuple	Foundation
IstExportValueBounds	export_value_bounds	Scientific
IstGetEntityValue	get_iif_syntax_entity_value	Foundation
IstGetPixel	get_pixel	Foundation
IstGetPixelArray	get_pixel_array	Foundation
IstGetPixelRecord	get_pixel_record	Technical
ItbaExportROIArray	export_roi_array	Technical
ItbaGetPixelArrayROI	get_pixel_array_roi	Foundation
ItbaGetPixelRecordROI	get_pixel_record_roi	Technical

Table 4 - Function names alphabetical by bound name (continued)

Bound Name	Function Name	Least Profile
ItfSetErrorHandler	set_error_handler	Foundation
ItraAccumulator	accumulator	Foundation
ItraHistShape1D	histogram_shape_1d	Scientific
ItraHistShape2D	histogram_shape_2d	Scientific
ItraLineProfileVReg	line_profile_vreg	Full
ItraShapeMetrics	shape_metrics	Scientific
ItraSpatialMomentsInvariant	spatial_moments_invariant	Scientific
ItraSpatialMomentsScaled	spatial_moments_scaled	Scientific
ItuaExportHist	export_histogram	Foundation
ItuaPerimeterCodeGenerator	perimeter_code_generator	Scientific
ItzPutImageSubrange	put_image_subrange	Foundation
IvAbortAsynchExecution	abort_asynchronous_execution	Scientific
IvCloseGateway	close_iif_gateway	Foundation
IvClosePIKS	close_piks	Foundation
IvClosePIKSEmergency	close_piks_emergency	Foundation
IvClosePort	close_port	Foundation
IvCmpTypeDef	inquire_iif_syntax_component_type_definition	Foundation
IvColourConvMatrix	colour_conversion_matrix	Foundation
IvColourConvMatrixVReg	colour_conversion_matrix_vreg	Full
IvComplexDecomposition	complex_decomposition	Technical
IvComplexDecompositionVReg	complex_decomposition_vreg	Full
IvDeallocateDataObject	deallocate_data_object	Foundation
IvDeallocateDataObjectVReg	deallocate_data_object_vreg	Full
IvDeleteEntity	delete_iif_syntax_entity	Foundation
IvErrorHandler	error_handler	Foundation
IvErrorLogger	error_logger	Foundation
IvExtrema	extrema	Foundation
IvGetColourPixel	get_colour_pixel	Foundation
IvGetColourPixelVReg	get_colour_pixel_vreg	Full
IvInquireChainEnvironment	inquire_chain_environment	Full
IvInquireChainStatus	inquire_chain_status	Full
IvInquireElements	inquire_elements	Foundation
IvInquireEntityChc	inquire_iif_syntax_entity_choice	Foundation
IvInquireEntityNumber	inquire_iif_syntax_entity_number	Foundation
IvInquireEntityOptional	inquire_iif_syntax_entity_optional	Foundation
IvInquireEntityTypeDef	inquire_iif_syntax_entity_type_definition	Foundation
IvInquireEntityValidity	inquire_iif_syntax_entity_validity	Foundation
IvInquireFunctionality	inquire_supported_functionality	Foundation
IvInquireGatewayStatus	inquire_iif_gateway_status	Foundation
IvInquireImage	inquire_image	Foundation
IvInquireIndexAssignment	inquire_index_assignment	Technical
IvInquireNonImageObject	inquire_non_image_object	Foundation
IvInquirePIKSImpl	inquire_piks_implementation	Foundation
IvInquirePIKSStatus	inquire_piks_status	Foundation
IvInquirePort	inquire_port	Foundation
IvInquireRepository	inquire_repository	Foundation
IvInquireResampling	inquire_resampling	Foundation

Table 4 - Function names alphabetical by bound name (concluded)

Bound Name	Function Name	Least Profile
IvMoments	moments	Foundation
IvMomentsVReg	moments_vreg	Full
IvOpenGateway	open_iif_gateway	Foundation
IvOpenPIKS	open_piks	Foundation
IvReleaseEntityId	release_iif_syntax_entity_identifier	Foundation
IvResetGateway	reset_iif_gateway	Foundation
IvResetPort	reset_port	Foundation
IvSetGlobals	set_globals	Foundation
IvSetGlobalsVReg	set_globals_vreg	Full
IvSetIndexAssignment	set_index_assignment	Technical
IvSetIndexAssignmentVReg	set_index_assignment_vreg	Full
IvSynch	synchronize	Scientific
IvVRegConditional	vreg_conditional	Full
IvVRegConditionalVReg	vreg_conditional_vreg	Full
IvVRegGet	vreg_get	Full

5 Data type definitions

5.1 Mapping of data types

The architecture of the Image Processing and Interchange Standard, ISO/IEC 12087-1, defines sets of abstract data types. This clause gives the mapping from those data types to the data defined later in this part of 12088.

The numerous types defined in IPI Parts 1, 2 and 3 and the association with the application data are bound as identified in Table 5. IPI-PIKS identifies internal data types which are implementation-dependent and may not be of the same type as the external representation of related data types. Most parameter and application data types are common to both IPI-IIF and IPI-PIKS. The precision of application data is specified by the application to the implementation. The precision of internal data is given by ISO/IEC 12087 to require at least 8 bits for non-negative integers, 16 bits for signed integers, 32 bits for real values and 32 bits for each component, real and imaginary, of complex value representations.

Table 5 - Data types

Data Type	IPI-PIKS Internal		Parameters		External Image	
	Functional	C Binding	Functional	C Binding	Functional	C Binding
null	NULL	NULL	NULL	NULL		
Boolean	BD		BP	Ipbool	BI	Imbool
enumerated			EP	enum	enumerated	enum
non-negative integer	ND		NP	Ipuint	NI	Imuint
signed integer	SD		SP	Ipint	SI	Imint
fixed point integer					TI	Imfixed
real arithmetic	RD		RP	Ipfloat (PIKS) Ipdouble (IIF)	RF	Imfloat
complex arithmetic	CD		CP	Ipscomplex	CF	Imcomplex
character string	CS		CS	char		
data object identifier	ID	Idnxxx	IP	Ipnxxx		

Real and complex IPI-PIKS internal data types are not required in an implementation of the foundation profile of IPI-PIKS.

The only IPI-PIKS internal data types represented in this binding are NULL, identifiers available to the application for referencing objects and identifiers returned to the application for assignment of objects to functional elements. Other internal IPI-PIKS data types are left as implementation dependent. Data object identifiers, designated Idnxxx and Ipnxxx, tabulated above, are assigned pertinent type names defined in 5.2.3.

5.2 Environment type definitions

The data types defined in this section allow for the ease of porting application programs between different environments. These types are used as the basis for other data types declared in this part of ISO/IEC 12088.

5.2.1 External physical image data types

Each of the external physical image data types shall be defined for pixel representation by the implementation as a specific type suitable for containing their appropriate values used by the implementation. Suggest:

typedef	unsigned char	Imbool;	/* BI data type	*/
typedef	enum {names}	Imenum_name;	/* enumerated data type	*/
typedef	unsigned char	Imuint;	/* NI data type	*/
typedef	int	Imint;	/* SI data type	*/
typedef	int	Imfixed;	/* TI representation of real data type	*/
typedef	float	Imfloat;	/* RF float representation of real data type	*/

typedef struct {				
Imfloat	xReal;	/* real part		*/
Imfloat	xImaginary;	/* imaginary part		*/
} Imscomplex;		/* CF data type		*/

It is suggested that the Boolean external physical image data type, BI, be represented in bitmap form, one pixel per bit, eight pixels per byte. As optioned to import/export utilities, pixels may be ordered from the most significant bit or least significant bit of the image data bytes.

The enumerated pixel data type is specific to the IPI-IIF. An implementation shall provide individually named (Imenum_name) declarations, each with a list of unique enumerated names, for each collection of enumerated pixel types supported.

The fixed point integer data type is unique to IPI-PIKS. This can be represented by the signed integer data type in IPI-IIF.

The real and complex data types are not required in an implementation of the foundation profile of IPI-PIKS.

5.2.2 Basic parameter data types

5.2.2.1 IPI-PIKS parameter data types

Each of these data types shall be defined for an IPI-PIKS implementation as a specific type suitable for containing their appropriate values used by the implementation. Suggest:

```
typedef      unsigned char    Ipbool;          /* BP data type          */
typedef      unsigned long int Ipuint;         /* NP data type          */
typedef      int              Ipint;           /* SP data type          */
typedef      float            Ipfloat;         /* RP real data type     */

typedef struct {
    Ipfloat      xReal;          /* real part              */
    Ipfloat      xImaginary;     /* imaginary part         */
} Ipscomplex; /* CP data type          */
```

5.2.2.2 IPI-IIF parameter data types

Each of these data types shall be defined for an IPI-IIF implementation as a specific type suitable for containing their appropriate values used by the implementation. Suggest:

```
typedef      unsigned char    Ipbool;          /* BP data type          */
typedef      unsigned long int Ipuint;         /* NP data type          */
typedef      int              Ipint;           /* SP data type          */
typedef      double           Ipdouble;        /* RP real data type     */

typedef struct {
    Ipdouble      xReal;          /* real part              */
    Ipdouble      xImaginary;     /* imaginary part         */
} Ipscomplex; /* CP data type          */
```

5.2.3 Data object identifiers

5.2.3.1 IPI-PIKS object identifiers

Each of these data types shall be defined by the implementation as a type suitable for containing their appropriate representations used by the implementation. Suggest:

typedef	void	*Idnchain;	/* IPI-PIKS internal chain object	*/
typedef	void	*Idnid_array;	/* IPI-PIKS internal image id array	*/
typedef	void	*Idnid_list;	/* IPI-PIKS internal image id list	*/
typedef	void	*Idnid_record;	/* IPI-PIKS internal image id record	*/
typedef	void	*Idnhist;	/* IPI-PIKS internal histogram arrays	*/
typedef	void	*Idnimage;	/* IPI-PIKS internal image objects	*/
typedef	void	*Idnlut;	/* IPI-PIKS internal lookup tables	*/
typedef	void	*Idnmatrix;	/* IPI-PIKS internal matrix objects	*/
typedef	void	*Idnnbhood;	/* IPI-PIKS internal neighbourhood arrays	*/
typedef	void	*Idnpixel_record;	/* IPI-PIKS internal pixel record	*/
typedef	void	*Idnrepository;	/* IPI-PIKS repository object	*/
typedef	void	*Idnroi;	/* IPI-PIKS internal region of interest objects	*/
typedef	void	*Idnstatic;	/* IPI-PIKS internal static arrays	*/
typedef	void	*Idntuple;	/* IPI-PIKS internal tuples	*/
typedef	void	*Idnbounds;	/* IPI-PIKS internal value bounds lists	*/
typedef	void	*Idnvreg;	/* IPI-PIKS internal virtual register objects	*/
typedef	void	*Ipnarg;	/* external function reloadable arguments	*/
typedef	FILE	*Ipnerror;	/* external error file	*/
typedef	void	*Ipnport;	/* external input/output port	*/
typedef	void	(*Iptferror_function)	/* error handler function pointer	*/
	(Ipoint Ipoint Idnchain Ipnerror Ipsynchronicity_state	iCode, iElement, nChain, nError, eSynchronicityState);		

5.2.3.2 IPI-IIF object identification

Each of these data types shall be defined by the implementation as a specific type suitable for containing their appropriate representation used by the implementation. Suggest:

```
typedef      void      *Ipndata;          /* data field entity identifier      */
typedef      void      *Ipnentity;        /* syntax entity identifier          */
typedef      FILE      *Ipnerror;         /* external error file               */
typedef      void      *Ipnport;          /* external port                     */
typedef      void      *Ipnport_name;     /* external port name                */
typedef      void      (*Iptferror_function_iif) /* error handler function pointer */
(Ipint
 Ipnentity
 Ipint
 Ipnerror
 iCode,
 nEntity,
 iElement,
 nError);
```

5.3 Implementation dependent data type definitions

Part 1 of ISO/IEC 12087 describes an IPI-PIKS specific architecture which identifies an image data object, image object descriptors and non-image data objects. These objects are internal to the IPI-PIKS domain and, as such, are private to IPI-PIKS and remain implementation dependent.

IECNORM.COM : Click to view the full PDF of ISO/IEC 12088-4:1995

5.4 Implementation independent data type definitions

Enumerated, union and structure data types are specified in support of the API functions defined in ISO/IEC 12087. They are constructed according to the naming rules described in 3.4 and presented in the following clauses.

5.4.1 Enumerated data type definitions

Bounded option parameters identified in ISO/IEC 12087 are represented in this part of ISO/IEC 12088 as enumerated data types. Enumerated data types are specified for IPI-PIKS and IPI-IIF separately though they may be combined in one implementation as presented.

5.4.1.1 IPI-PIKS enumerated data type definitions

BINARY EVENT

```
typedef enum {
    IBINARY_EVENT_ON,           /* yes/on          */
    IBINARY_EVENT_OFF,          /* no/off           */
} Ipebinary_event;
```

COMPLEX CONVERSION OPTIONS

```
typedef enum {
    ICOMPLEX_REAL_IMAGINARY,    /* real/imaginary to/from complex */
    ICOMPLEX_MAGNITUDE_PHASE,   /* magnitude/phase to/from complex */
} Ipecomplex_conv;
```

COMPLEX MAGNITUDE OPTIONS

```
typedef enum {
    ICOMPLEX_MAGNITUDE,         /* magnitude output */
    ICOMPLEX_SQUARED,           /* magnitude squared output */
} Ipemagnitude;
```

DATA VALIDITY INDICATOR

```
typedef enum {
    IVALIDITY_VALID,            /* data is valid and available */
    IVALIDITY_INVALID,          /* data is not valid */
} Ipevalidity;
```

DERIVATIVE OPTION

```
typedef enum {
    IDERIV_HORIZONTAL, /* horizontal derivative */
    IDERIV_VERTICAL, /* vertical derivative */
} Ipederiv;
```

DIMENSION OPTION

```
typedef enum {
    IDIMENSION_1D, /* one dimension */
    IDIMENSION_2D, /* two dimensions */
    IDIMENSION_3D, /* three dimensions */
    IDIMENSION_4D, /* four dimensions */
    IDIMENSION_5D, /* five dimensions */
} Ipedimension;
```

EROSION & DILATION, OPERATION OPTIONS

```
typedef enum {
    IOPERATION_EROSION, /* erosion */
    IOPERATION_DILATE, /* dilation */
} Ipeerosion;
```

FLIP, SPIN, TRANSPOSE OPTIONS

```
typedef enum {
    IFLIP_TOP_BOTTOM, /* top-to-bottom flip */
    IFLIP_LEFT_RIGHT, /* left-to-right flip */
    ISPIN_90_DEGREES_CCW, /* 90 degrees counterclockwise spin */
    ISPIN_180_DEGREES_CCW, /* 180 degrees counterclockwise spin */
    ISPIN_270_DEGREES_CCW, /* 270 degrees counterclockwise spin */
    ITRANSPOSE_UPLIFT_LOWRIGHT, /* transpose about upper left to lower right */
    ITRANSPOSE_LOWLEFT_UPRIGHT, /* transpose about lower left to upper right */
} Ipeflip;
```

FREQUENCY SHIFT OPTIONS

```
typedef enum {
    IFREQUENCY_SHIFT_ORIGIN, /* constant term at origin */
    IFREQUENCY_SHIFT_CENTRE, /* constant term centre */
} Ipefrequency_shift;
```

GAMMA CORRECTION OPTIONS

```
typedef enum {  
    ICORRECT_LIN_GAMMA,          /* linear to gamma correction */  
    ICORRECT_GAMMA_LIN          /* gamma to linear correction */  
} Ipegamma;
```

INDEX ASSIGNMENT CODES

```
typedef enum {  
    IINDEX_HORIZONTAL,          /* horizontal dimension */  
    IINDEX_VERTICAL,           /* vertical dimension */  
    IINDEX_DEPTH,              /* depth dimension */  
    IINDEX_TEMPORAL,           /* temporal dimension */  
    IINDEX_BAND                 /* band dimension */  
} Ipeindex;
```

LEVEL SLICE BACKGROUND OPTIONS

```
typedef enum {  
    IBACKGROUND_ZERO,          /* zero background */  
    IBACKGROUND_IMAGE          /* image background */  
} Ipebackground;
```

MODES, ERROR REPORT

```
typedef enum {  
    IERROR_REPORT_ACTIVE,       /* error reporting active */  
    IERROR_REPORT_INACTIVE      /* error reporting inactive */  
} Ipeerror_report_mode;
```

MODES, INDEX ASSIGNMENT

```
typedef enum {  
    IINDEX_ASSIGNMENT_ON,       /* index assignment operative */  
    IINDEX_ASSIGNMENT_OFF       /* index assignment inoperative */  
} Ipeindex_mode;
```

MODES, MATCH POINT

```
typedef enum {  
    IMATCH_POINT_ON,            /* match point adjustment operative */  
    IMATCH_POINT_OFF            /* match point adjustment inoperative */  
} Ipematch_mode;
```

MODES, ROI CONTROL

```
typedef enum {
    IROI_CONTROL_ON,           /* ROI control operative */
    IROI_CONTROL_OFF,          /* ROI control inoperative */
} Iperoi_control_mode;
```

MODES, ROI PROCESS

```
typedef enum {
    IROI_PROCESS_ON,           /* ROI processing operative */
    IROI_PROCESS_OFF,          /* ROI processing inoperative */
} Iperoi_process_mode;
```

MODES, SYNCHRONICITY

```
typedef enum {
    ISYNCHRONICITY_MODE_SYNCH, /* synchronous mode */
    ISYNCHRONICITY_MODE_ASYNC, /* asynchronous mode */
} Ipesynchronicity_mode;
```

OPEN CLOSE OPTIONS

```
typedef enum {
    IOPERATION_OPEN,           /* open option */
    IOPERATION_CLOSE,          /* close option */
} Ipeopen_close;
```

POLARITY OPTION

```
typedef enum {
    IPOLARITY_TRUE,            /* TRUE state */
    IPOLARITY_FALSE,           /* FALSE state */
} Ipepolarity;
```

PROCESSING MODE OPTION

```
typedef enum {
    IPROCESS_MODE_0,           /* processing mode 0 */
    IPROCESS_MODE_1,           /* processing mode 1 */
    IPROCESS_MODE_2,           /* processing mode 2 */
    IPROCESS_MODE_3,           /* processing mode 3 */
    IPROCESS_MODE_4,           /* processing mode 4 */
} Ipeprocess;
```

REFLECTION OPTION

```
typedef enum {  
    IREFLECTION_NO,           /* no reflection */  
    IREFLECTION_YES          /* reflection */  
} Ipereflection;
```

ROTATION PERMISSION OPTION

```
typedef enum {  
    IROTATION_NO,            /* no rotation */  
    IROTATION_YES           /* rotation */  
} Iperotation;
```

STATES, CHAIN

```
typedef enum {  
    ICHAIN_BUILD,           /* building chain state */  
    ICHAIN_EXECUTE,        /* executing chain state */  
    ICHAIN_NOT              /* not in any chain state */  
} Ipechain_state;
```

STATES, ERROR

```
typedef enum {  
    IERROR_SUBSTATE,       /* error substate condition exists */  
    IERROR_NO              /* no error condition */  
} Ipeerror_state;
```

STATES, PIKS OPERATIONAL

```
typedef enum {  
    IPIKS_OPERATION_CLOSED, /* PIKS closed */  
    IPIKS_OPERATION_OPEN    /* PIKS open */  
} Ipepiks_operation_state;
```

STATES, SYNCHRONICITY

```
typedef enum {  
    ISYNCHRONICITY_STATE_SYNCH, /* synchronous state */  
    ISYNCHRONICITY_STATE_ASYNCH /* asynchronous state */  
} Ipesynchronicity_state;
```

TRANSFORM DIRECTION

```
typedef enum {  
    ITRANSFORM_FORWARD,          /* forward transform */  
    ITRANSFORM_INVERSE,         /* inverse transform */  
} Itransform;
```

VIRTUAL REGISTER CHOICE

```
typedef enum {  
    IVREG_CHC_DIRECT,           /* direct parameter specification */  
    IVREG_CHC_VREG,            /* virtual register representation */  
} Ipevreg_chc;
```

VIRTUAL REGISTER STATE

```
typedef enum {  
    IVREG_STATE_CLEARED,        /* virtual register cleared */  
    IVREG_STATE_SET,           /* virtual register set */  
} Ipevreg_state;
```

5.4.1.2 IPI-IIF enumerated data type definitions

DATA VALIDITY INDICATOR

```
typedef enum {  
    IVALIDITY_VALID,           /* data is valid and available */  
    IVALIDITY_INVALID,        /* data is not valid */  
} Ipevalidity;
```

NOTE - Referenced DATA_VALID is bound as IVALIDITY_VALID and DATA_INVALID is bound as IVALIDITY_INVALID.

5.4.2 Union data type definitions

Unions of various data types are used for specifying parameters which may be presented in different forms. The implementation and application are responsible for the utilization of the appropriate data type for each combinable usage.

5.4.2.1 IPI-PIKS union data type definitions

ARRAY IDENTIFIER

```
typedef union {
    Idnmatrix          nMatrix;          /* matrix identifier          */
    Idnnbhood          nNbhood;         /* neighbourhood array identifier */
    Idnstatic          nStatic;         /* static array identifier      */
} Ipsarray_id;
```

EXTERNAL PHYSICAL IMAGE PIXEL DATA TYPES

```
typedef union {
    Imbool          *tbPixel;          /* external Boolean pixel value pointer */
    Imuint          *tuPixel;          /* external unsigned integer pixel value pointer */
    Imint           *tiPixel;          /* external signed integer pixel pointer */
    Imfixed         *trPixelFixed;     /* external fixed point pixel pointer    */
    Imfloat         *trPixelFloat;     /* external floating point pixel pointer  */
    Imscomplex      *txPixel;          /* external complex pixel value pointer   */
} Ipspiks_pixel_types;
```

INQUIRE NON-IMAGE INFORMATION

```
typedef union {
    Ipsattrs_id_array *tsIdArray;      /* identifier array attributes pointer */
    Ipsattrs_id_record *tsIdRecord;    /* identifier record attributes pointer */
    Ipsattrs_hist      *tsHist;        /* histogram attributes pointer        */
    Ipsattrs_lut        *tsLUT;        /* lookup table attributes pointer     */
    Ipsattrs_matrix     *tsMatrix;     /* matrix attributes pointer           */
    Ipsattrs_nbhood     *tsNbhood;     /* neighbourhood array attributes pointer */
    Ipsattrs_pixel_record *tsPixel;    /* pixel record attributes pointer     */
    Ipsattrs_roi_array  *tsROIArray;   /* ROI array attributes pointer        */
    Ipsattrs_roi_coordinate *tsROICoordinate; /* ROI coordinate attributes pointer */
    Ipsattrs_roi_elliptical *tsROIElliptical; /* ROI elliptical attributes pointer */
    Ipsattrs_roi_generic *tsROIGeneric; /* ROI generic attributes pointer      */
    Ipsattrs_roi_polygon *tsROIPolygon; /* ROI polygon attributes pointer      */
    Ipsattrs_roi_rectangular *tsROIRectangular; /* ROI rectangular attributes pointer */
    Ipsattrs_static     *tsStatic;     /* static array attributes pointer     */
    Ipsattrs_tuple      *tsTuple;      /* tuple attributes pointer            */
    Ipsattrs_bounds     *tsBounds;     /* value bounds collection attributes pointer */
    Ipsattrs_vreg       *tsVReg;       /* virtual register attributes pointer  */
} Ipsattrs_non_image;
```

INQUIRE NON-IMAGE OBJECTS

```
typedef union {
    Idnid_array      nIdArray;      /* composite identifier array identifier */
    Idnid_list       nIdList;       /* composite identifier list identifier */
    Idnid_record     nIdRecord;     /* composite identifier record identifier */
    Idnhist          nHist;         /* histogram identifier */
    Idnlut           nLUT;          /* lookup table identifier */
    Idnmatrix        nMatrix;       /* matrix identifier */
    Idnnbhood        nNbhood;       /* neighbourhood array identifier */
    Idnpixel_record  nPixel;        /* pixel record identifier */
    Idnroi           nROI;          /* region of interest identifier */
    Idnstatic        nStatic;       /* static array identifier */
    Idntuple         nTuple;        /* tuple identifier */
    Idnbounds        nBounds;       /* value bounds collection identifier */
    Idnvreg          nVReg;         /* virtual register identifier */
} Ipsnon_image_object;
```

INTERNAL OBJECT IDENTIFIERS

```
typedef union {
    Idnchain         nChain;        /* chain identifier */
    Idnid_array      nIdArray;      /* composite identifier array identifier */
    Idnid_list       nIdList;       /* composite identifier list identifier */
    Idnid_record     nIdRecord;     /* composite identifier record identifier */
    Idnhist          nHist;         /* histogram identifier */
    Idnimage         nImage;        /* image identifier */
    Idnlut           nLUT;          /* lookup table identifier */
    Idnmatrix        nMatrix;       /* matrix identifier */
    Idnnbhood        nNbhood;       /* neighbourhood array identifier */
    Idnpixel_record  nPixel;        /* pixel record identifier */
    Idnrepository    nRepository;   /* repository identifier */
    Idnroi           nROI;          /* region of interest identifier */
    Idnstatic        nStatic;       /* static array identifier */
    Idntuple         nTuple;        /* tuple identifier */
    Idnbounds        nBounds;       /* value bounds collection identifier */
    Idnvreg          nVReg;         /* virtual register identifier */
} Ipsobject;
```

PARAMETERS, ARITHMETIC DATA TYPES

```
typedef union {
    Ipuint           *tuArith;      /* unsigned integer value pointer */
    Ipiint           *tiArith;      /* signed integer value pointer */
    Ipfloating        *trArithFloat; /* floating real value pointer */
} Ipsparameter_arith;
```

PARAMETERS, BASIC DATA TYPES

```
typedef union {
    Ipbool          *tbParameter;    /* Boolean value pointer */
    Ipuint          *tuParameter;    /* non-negative integer value pointer */
    Ipint           *tiParameter;    /* signed integer value pointer */
    Ipfloat         *trParameterFloat; /* floating real value pointer */
    Ipscomplex      *txParameter;    /* complex data pointer */
    char            *tcParameter;    /* character pointer */
} Ipsparameter_basic;
```

PARAMETERS, COLOUR DATA TYPES

```
typedef union {
    Ipbool          *tbColour;       /* Boolean value pointer */
    Ipuint          *tuColour;       /* non-negative integer value pointer */
    Ipint           *tiColour;       /* signed integer value pointer */
    Ipfloat         *trColourFloat;  /* floating real value pointer */
} Ipsparameter_colour;
```

PARAMETERS, EDGE DETECTION DATA TYPES

```
typedef union {
    Ipbool          *tbEdge;         /* Boolean output value pointer */
    Ipfloat         *trEdgeFloat;    /* floating real output value pointer */
} Ipsparameter_edge;
```

PARAMETERS, LOGICAL DATA TYPES

```
typedef union {
    Ipbool          *tbLogical;      /* Boolean value pointer */
    Ipuint          *tuLogical;      /* unsigned integer value pointer */
    Ipint           *tiLogical;      /* signed integer value pointer */
} Ipsparameter_logical;
```

PARAMETERS, NUMERIC DATA TYPES

```
typedef union {
    Ipuint          *tuNumeric;      /* unsigned integer value pointer */
    Ipint           *tiNumeric;      /* signed integer value pointer */
    Ipfloat         *trNumericFloat; /* floating real value pointer */
    Ipscomplex      *txNumeric;      /* complex data pointer */
} Ipsparameter_numeric;
```

PARAMETERS, PIXEL DATA TYPES

```
typedef union {
    Ipbool          *tbPixel;          /* Boolean value pointer          */
    Ipuint          *tuPixel;          /* unsigned integer value pointer */
    Ipoint          *tiPixel;          /* signed integer value pointer   */
    Ipfloa          *trPixelFloat;     /* floating real value pointer    */
    Ipscomplex      *txPixel;          /* complex data pointer           */
} Ipsparameter_pixel;
```

PARAMETERS, VIRTUAL REGISTER VALUE DATA TYPES

```
typedef union {
    Ipsobject       snVReg;            /* PIKS internal identifier structure */
    void            *tVReg;            /* external object pointer           */
    Ipsparameter_basic stVReg;         /* basic parameter type value pointer structure */
} Ipsparameter_vreg;
```

VIRTUAL REGISTER ALTERNATIVE ARGUMENTS IDENTIFIER

```
typedef union {
    Ipnarg          sArgs;             /* argument parameter collection identifier */
    Idnvreg         nVReg;             /* virtual register identifier             */
} Ipsarg_vreg;
```

VIRTUAL REGISTER ALTERNATIVE ARITHMETIC DATA TYPE

```
typedef union {
    Ipsparameter_arith stArith;        /* arithmetic parameter pointer union */
    Idnvreg            nVReg;          /* virtual register identifier         */
} Ipsparameter_arith_vreg;
```

VIRTUAL REGISTER ALTERNATIVE ARRAY IDENTIFIER

```
typedef union {
    Ipsarray_id      snArrayId;        /* matrix, neighbourhood or static array identifier */
    Idnvreg          nVReg;            /* virtual register identifier                     */
} Ipsarray_id_vreg;
```

VIRTUAL REGISTER ALTERNATIVE BACKGROUND OPTION

```
typedef union {
    Ipebackground    eOption;          /* level slice background option */
    Idnvreg          nVReg;            /* virtual register identifier    */
} Ipsbackground_vreg;
```

VIRTUAL REGISTER ALTERNATIVE BASIC PARAMETER DATA TYPE

```
typedef union {  
    Ipsparameter_basic    stParameter;    /* basic parameter pointer union    */  
    Idnvreg                nVReg;         /* virtual register identifier      */  
} Ipsparameter_basic_vreg;
```

VIRTUAL REGISTER ALTERNATIVE BP ARRAY

```
typedef union {  
    Ipbool                *tbaBPValue;    /* BP array data pointer          */  
    Idnvreg                nVReg;         /* virtual register identifier      */  
} Ipsabool_vreg;
```

VIRTUAL REGISTER ALTERNATIVE BP PARAMETER

```
typedef union {  
    Ipbool                bBPValue;       /* BP parameter value            */  
    Idnvreg                nVReg;         /* virtual register identifier      */  
} Ipsbool_vreg;
```

VIRTUAL REGISTER ALTERNATIVE CHAIN IDENTIFIER

```
typedef union {  
    Idnchain                nChain;        /* chain identifier              */  
    Idnvreg                nVReg;         /* virtual register identifier      */  
} Ipschain_vreg;
```

VIRTUAL REGISTER ALTERNATIVE CHAR STRING PARAMETER

```
typedef union {  
    char                *tzChar;          /* character string pointer      */  
    Idnvreg                nVReg;         /* virtual register identifier      */  
} Ipschar_vreg;
```

VIRTUAL REGISTER ALTERNATIVE COLOUR PARAMETER DATA TYPE

```
typedef union {  
    Ipsparameter_colour    stColour;      /* colour parameter pointer union */  
    Idnvreg                nVReg;         /* virtual register identifier      */  
} Ipsparameter_colour_vreg;
```

VIRTUAL REGISTER ALTERNATIVE COMPLEX CONVERSION OPTION

```
typedef union {
    Ipecomplex_conv    eOption;          /* complex conversion option */
    Idnvreg            nVReg;            /* virtual register identifier */
} Ipscomplex_conv_vreg;
```

VIRTUAL REGISTER ALTERNATIVE COMPOSITE ID ARRAY ATTRIBUTES

```
typedef union {
    Ipsattrs_id_array  *tsAttrsArray;    /* composite id array attributes pointer */
    Idnvreg            nVReg;            /* virtual register identifier */
} Ipsattrs_id_array_vreg;
```

VIRTUAL REGISTER ALTERNATIVE COMPOSITE ID ARRAY IDENTIFIER

```
typedef union {
    Idnid_array        nIdArray;         /* composite id array identifier */
    Idnvreg            nVReg;            /* virtual register identifier */
} Ipsid_array_vreg;
```

VIRTUAL REGISTER ALTERNATIVE COMPOSITE ID LIST IDENTIFIER

```
typedef union {
    Idnid_list         nIdList;          /* composite id list identifier */
    Idnvreg            nVReg;            /* virtual register identifier */
} Ipsid_list_vreg;
```

VIRTUAL REGISTER ALTERNATIVE COMPOSITE ID RECORD ATTRIBUTES

```
typedef union {
    Ipsattrs_id_record *tsAttrsRecord;   /* composite id record attributes pointer */
    Idnvreg            nVReg;            /* virtual register identifier */
} Ipsattrs_id_record_vreg;
```

VIRTUAL REGISTER ALTERNATIVE COMPOSITE ID RECORD IDENTIFIER

```
typedef union {
    Idnid_record       nIdRecord;        /* composite id record identifier */
    Idnvreg            nVReg;            /* virtual register identifier */
} Ipsid_record_vreg;
```

VIRTUAL REGISTER ALTERNATIVE CP PARAMETER

```
typedef union {
    Ipscomplex          sxData;          /* complex data value          */
    Idnvreg             nVReg;          /* virtual register identifier */
} Ipscomplex_vreg;
```

VIRTUAL REGISTER ALTERNATIVE DERIVATIVE OPTION

```
typedef union {
    Ipederiv            eDirection;      /* direction option           */
    Idnvreg             nVReg;          /* virtual register identifier */
} Ipsderiv_vreg;
```

VIRTUAL REGISTER ALTERNATIVE DIMENSION OPTION

```
typedef union {
    Ipdimension         eDimension;      /* dimension option           */
    Idnvreg             nVReg;          /* virtual register identifier */
} Ipsdimension_vreg;
```

VIRTUAL REGISTER ALTERNATIVE DRAW PIXELS IMAGE DATA

```
typedef union {
    Ipsdraw_pixels      *tsaPixel;      /* pixel position and value   */
    Idnvreg             nVReg;          /* virtual register identifier */
} Ipsdraw_pixels_vreg;
```

VIRTUAL REGISTER ALTERNATIVE DRAW PIXELS ROI DATA

```
typedef union {
    Ipsdraw_pixels_roi  *tsaPixel;      /* pixel position and value   */
    Idnvreg             nVReg;          /* virtual register identifier */
} Ipsdraw_pixels_roi_vreg;
```

VIRTUAL REGISTER ALTERNATIVE EDGE PARAMETER DATA TYPE

```
typedef union {
    Ipsparameter_edge   stEdge;          /* edge output parameter pointer union */
    Idnvreg             nVReg;          /* virtual register identifier          */
} Ipsparameter_edge_vreg;
```

VIRTUAL REGISTER ALTERNATIVE EROSION OPTION

```
typedef union {
    Ipeerosion          eErodeDilate;    /* erosion or dilation option */
    Idnvreg             nVReg;           /* virtual register identifier */
} Ipserosion_vreg;
```

VIRTUAL REGISTER ALTERNATIVE EXTERNAL IMAGE PIXEL DATA TYPES

```
typedef union {
    Ipspiks_pixel_types stPixel;          /* external physical image pixel data pointer */
    Idnvreg             nVReg;           /* virtual register identifier */
} Ipspiks_pixel_types_vreg;
```

VIRTUAL REGISTER ALTERNATIVE FLIP, SPIN, TRANSPOSE OPTION

```
typedef union {
    Ipeflip             eOption;          /* flip, spin, transpose option */
    Idnvreg             nVReg;           /* virtual register identifier */
} Ipsflip_vreg;
```

VIRTUAL REGISTER ALTERNATIVE FREQUENCY SHIFT OPTION

```
typedef union {
    Ipefrequency_shift eOption;          /* frequency shift option */
    Idnvreg             nVReg;           /* virtual register identifier */
} Ipsfrequency_shift_vreg;
```

VIRTUAL REGISTER ALTERNATIVE GAMMA CONVERSION OPTION

```
typedef union {
    Ipegamma            eOption;          /* gamma conversion option */
    Idnvreg             nVReg;           /* virtual register identifier */
} Ipsgamma_vreg;
```

VIRTUAL REGISTER ALTERNATIVE HISTOGRAM ATTRIBUTES

```
typedef union {
    Ipsattrs_hist       *tsAttrs;         /* histogram attributes pointer */
    Idnvreg             nVReg;           /* virtual register identifier */
} Ipsattrs_hist_vreg;
```

VIRTUAL REGISTER ALTERNATIVE HISTOGRAM IDENTIFIER

```
typedef union {  
    Idnhist          nHist;          /* histogram identifier          */  
    Idnvreg          nVReg;          /* virtual register identifier    */  
} Ipshist_vreg;
```

VIRTUAL REGISTER ALTERNATIVE IMAGE ATTRIBUTES

```
typedef union {  
    Ipsattrs_image   *tsAttr;        /* image attributes pointer      */  
    Idnvreg          nVReg;          /* virtual register identifier    */  
} Ipsattrs_image_vreg;
```

VIRTUAL REGISTER ALTERNATIVE IMAGE IDENTIFIER

```
typedef union {  
    Idnimage         nImage;         /* image identifier              */  
    Idnvreg          nVReg;          /* virtual register identifier    */  
} Ipsimage_vreg;
```

VIRTUAL REGISTER ALTERNATIVE INDEX OPTION

```
typedef union {  
    Ipeindex         eIndex;         /* image index assignment        */  
    Idnvreg          nVReg;          /* virtual register identifier    */  
} Ipsindex_vreg;
```

VIRTUAL REGISTER ALTERNATIVE LOGICAL PARAMETER DATA TYPE

```
typedef union {  
    Ipsparameter_logical stLogical; /* logical parameter pointer union */  
    Idnvreg          nVReg;          /* virtual register identifier    */  
} Ipsparameter_logical_vreg;
```

VIRTUAL REGISTER ALTERNATIVE LOOKUP TABLE ATTRIBUTES

```
typedef union {  
    Ipsattrs_lut      *tsAttr;        /* lookup table attributes pointer */  
    Idnvreg          nVReg;          /* virtual register identifier    */  
} Ipsattrs_lut_vreg;
```

VIRTUAL REGISTER ALTERNATIVE LOOKUP TABLE IDENTIFIER

```
typedef union {
    Idnlut          nLUT;          /* lookup table identifier */
    Idnvreg         nVReg;         /* virtual register identifier */
} Ipslut_vreg;
```

VIRTUAL REGISTER ALTERNATIVE MAGNITUDE OPTION

```
typedef union {
    Ipemagnitude    eMagnitude;   /* magnitude options */
    Idnvreg         nVReg;         /* virtual register identifier */
} Ipsmagnitude_vreg;
```

VIRTUAL REGISTER ALTERNATIVE MATRIX ATTRIBUTES

```
typedef union {
    Ipsattrs_matrix *tsAttr;      /* matrix attributes pointer */
    Idnvreg         nVReg;         /* virtual register identifier */
} Ipsattrs_matrix_vreg;
```

VIRTUAL REGISTER ALTERNATIVE MATRIX IDENTIFIER

```
typedef union {
    Idnmatrix       nMatrix;      /* matrix identifier */
    Idnvreg         nVReg;         /* virtual register identifier */
} Ipsmatrix_vreg;
```

VIRTUAL REGISTER ALTERNATIVE NEIGHBOURHOOD ARRAY ATTRIBUTES

```
typedef union {
    Ipsattrs_nbhood *tsAttr;      /* neighbourhood array attributes pointer */
    Idnvreg         nVReg;         /* virtual register identifier */
} Ipsattrs_nbhood_vreg;
```

VIRTUAL REGISTER ALTERNATIVE NEIGHBOURHOOD ARRAY IDENTIFIER

```
typedef union {
    Idnnbhood       nNbhood;      /* neighbourhood array identifier */
    Idnvreg         nVReg;         /* virtual register identifier */
} Ipsnbhood_vreg;
```

VIRTUAL REGISTER ALTERNATIVE NEIGHBOURHOOD ARRAY IDENTIFIER SET

```
typedef union {  
    Idnnbhood          *tnaNbhood;    /* set of neighbourhood array identifiers */  
    Idnvreg            nVReg;         /* virtual register identifier */  
} Ipsanbhood_vreg;
```

VIRTUAL REGISTER ALTERNATIVE NP ARRAY DATA

```
typedef union {  
    Ipuint             *tuaNPValue;   /* NP parameter array pointer */  
    Idnvreg            nVReg;         /* virtual register identifier */  
} Ipsauint_vreg;
```

VIRTUAL REGISTER ALTERNATIVE NP PARAMETER

```
typedef union {  
    Ipuint             uNPValue;      /* NP parameter value */  
    Idnvreg            nVReg;         /* virtual register identifier */  
} Ipsuint_vreg;
```

VIRTUAL REGISTER ALTERNATIVE NUMERIC PARAMETER DATA TYPE

```
typedef union {  
    Ipsparameter_numeric stNumeric;   /* numeric parameter pointer union */  
    Idnvreg            nVReg;         /* virtual register identifier */  
} Ipsparameter_numeric_vreg;
```

VIRTUAL REGISTER ALTERNATIVE OBJECT IDENTIFIER

```
typedef union {  
    Ipsobject          snObject;      /* object identifier union */  
    Idnvreg            nVReg;         /* virtual register identifier */  
} Ipsobject_vreg;
```

VIRTUAL REGISTER ALTERNATIVE OPEN CLOSE OPTION

```
typedef union {  
    Ipeopen_close      eOption;       /* open close operation option */  
    Idnvreg            nVReg;         /* virtual register identifier */  
} Ipsopen_close_vreg;
```

VIRTUAL REGISTER ALTERNATIVE PIKS MODES

```
typedef union {
    Ipspiks_modes      sMode;          /* PIKS modes settings          */
    Idnvreg            nVReg;          /* virtual register identifier   */
} Ipspiks_modes_vreg;
```

VIRTUAL REGISTER ALTERNATIVE PIXEL PARAMETER DATA TYPE

```
typedef union {
    Ipsparameter_pixel stPixel;        /* pixel parameter pointer union */
    Idnvreg            nVReg;          /* virtual register identifier   */
} Ipsparameter_pixel_vreg;
```

VIRTUAL REGISTER ALTERNATIVE PIXEL RECORD ATTRIBUTES

```
typedef union {
    Ipsattrs_pixel_record *tsAttrs;    /* pixel record attributes pointer */
    Idnvreg            nVReg;          /* virtual register identifier   */
} Ipsattrs_pixel_record_vreg;
```

VIRTUAL REGISTER ALTERNATIVE POLARITY OPTION

```
typedef union {
    Ipepolarity        ePolarity;      /* polarity option              */
    Idnvreg            nVReg;          /* virtual register identifier   */
} Ipspolarity_vreg;
```

VIRTUAL REGISTER ALTERNATIVE PORT IDENTIFIER

```
typedef union {
    Ipnport            nPort;          /* input/output port identifier  */
    Idnvreg            nVReg;          /* virtual register identifier   */
} Ipsport_vreg;
```

VIRTUAL REGISTER ALTERNATIVE REFLECTION OPTION

```
typedef union {
    Ipereflection      eOption;        /* reflection option            */
    Idnvreg            nVReg;          /* virtual register identifier   */
} Ipsreflection_vreg;
```

VIRTUAL REGISTER ALTERNATIVE ROI ATTRIBUTES

```
typedef union {  
    Ipsattrs_roi      *tsAttr;      /* ROI attributes pointer */  
    ldnvreg           nVReg;        /* virtual register identifier */  
} Ipsattrs_roi_vreg;
```

VIRTUAL REGISTER ALTERNATIVE ROI IDENTIFIER

```
typedef union {  
    Idnroi            nROI;          /* ROI identifier */  
    ldnvreg           nVReg;        /* virtual register identifier */  
} Ipsroi_vreg;
```

VIRTUAL REGISTER ALTERNATIVE ROTATION OPTION

```
typedef union {  
    Iperotation       eOption;       /* rotation option */  
    ldnvreg           nVReg;        /* virtual register identifier */  
} Ipsrotation_vreg;
```

VIRTUAL REGISTER ALTERNATIVE RP FLOAT ARRAY

```
typedef union {  
    Ipfloat           *traRPFloat;   /* float RP parameter pointer */  
    ldnvreg           nVReg;        /* virtual register identifier */  
} Ipsafloat_vreg;
```

VIRTUAL REGISTER ALTERNATIVE RP FLOAT PARAMETER

```
typedef union {  
    Ipfloat           rRPFloat;      /* RP parameter value */  
    ldnvreg           nVReg;        /* virtual register identifier */  
} Ipsfloat_vreg;
```

VIRTUAL REGISTER ALTERNATIVE SP PARAMETER

```
typedef union {
    Ipint                iSPValue;    /* SP parameter value          */
    Idnvreg              nVReg;       /* virtual register identifier  */
} Ipsint_vreg;
```

VIRTUAL REGISTER ALTERNATIVE STATIC ARRAY ATTRIBUTES

```
typedef union {
    Ipsattrs_static      *tsAttrs;    /* static array attributes pointer */
    Idnvreg              nVReg;       /* virtual register identifier     */
} Ipsattrs_static_vreg;
```

VIRTUAL REGISTER ALTERNATIVE STATIC ARRAY IDENTIFIER

```
typedef union {
    Idnstatic            nStatic;     /* static array identifier        */
    Idnvreg              nVReg;       /* virtual register identifier     */
} Ipsstatic_vreg;
```

VIRTUAL REGISTER ALTERNATIVE TRANSFORM OPTION

```
typedef union {
    Ipetransform         eTransform;  /* transform option              */
    Idnvreg              nVReg;       /* virtual register identifier     */
} Ipstransform_vreg;
```

VIRTUAL REGISTER ALTERNATIVE TUPLE ATTRIBUTES

```
typedef union {
    Ipsattrs_tuple       *tsAttrs;    /* tuple attributes pointer       */
    Idnvreg              nVReg;       /* virtual register identifier     */
} Ipsattrs_tuple_vreg;
```

VIRTUAL REGISTER ALTERNATIVE TUPLE IDENTIFIER

```
typedef union {
    Idntuple             nTuple;      /* tuple identifier              */
    Idnvreg              nVReg;       /* virtual register identifier     */
} Ipstuple_vreg;
```

VIRTUAL REGISTER ALTERNATIVE VALUE BOUNDS ATTRIBUTES

```
typedef union {  
    Ipsattrs_bounds    *tsAttrs;    /* value bounds collection attributes pointer */  
    Idnvreg            nVReg;       /* virtual register identifier */  
} Ipsattrs_bounds_vreg;
```

VIRTUAL REGISTER ALTERNATIVE VALUE BOUNDS IDENTIFIER

```
typedef union {  
    Idnbounds          nBounds;     /* value bounds identifier */  
    Idnvreg            nVReg;       /* virtual register identifier */  
} Ipsbounds_vreg;
```

VIRTUAL REGISTER ALTERNATIVE VIRTUAL REGISTER ATTRIBUTES

```
typedef union {  
    Ipsattrs_vreg      *tsAttrs;    /* virtual register attributes pointer */  
    Idnvreg            nVReg;       /* virtual register identifier */  
} Ipsattrs_vreg_vreg;
```

VIRTUAL REGISTER ALTERNATIVE VIRTUAL REGISTER VALUE DATA TYPES

```
typedef union {  
    Ipsparameter_vreg  sValue;      /* virtual register value data types union */  
    Idnvreg            nVReg;       /* virtual register identifier */  
} Ipsparameter_vreg_vreg;
```

5.4.2.2 IPI-IIF union data type definitions

COMPRESSION PARAMETERS

```
typedef union {
    Ipsgroup_3      *tsGroup3;      /* Group 3 facsimile parameters pointer */
    Ipsgroup_4      *tsGroup4;      /* Group 4 facsimile parameters pointer */
    Ipsjbig          *tsJBIG;       /* JBIG parameters pointer */
    Ipsjpeg          *tsJPEG;       /* JPEG parameters pointer */
    Ipsmpeg_1       *tsMPEG;       /* MPEG parameters pointer */
} Ipscompress;
```

ENTITY PARAMETER TYPES

```
typedef union {
    Ipndata          nld;           /* parameter field identifier */
    Ipbool           *tbParameter;  /* BP parameter pointer */
    Ipint            *tiParameter;  /* SP parameter pointer */
    Ipdouble         *trParameterDouble; /* RP parameter pointer */
    char            *tzParameter;  /* CS parameter pointer */
} Ipsentity_value;
```

IIF PIXEL DATA TYPES

```
typedef union {
    Imbool           *tbPixel;      /* Boolean pixel pointer */
    Imuint           *tuPixel;      /* unsigned integer or enumerated pixel pointer */
    Imint            *tiPixel;      /* signed integer or enumerated pixel pointer */
    Imfloat          *trPixelFloat; /* floating real pixel pointer */
    Imscomplex       *txPixel;      /* complex pixel pointer */
    char            *tzPixel;      /* character string or enumerated pixel pointer */
} Ipsiif_pixel_types;
```

JPEG COMPRESSION LOSS TYPES

```
typedef union {
    Ipslossless      *tsLossless;   /* pointer to lossless mode parameters */
    Ipslossy         *tsLossy;     /* pointer to lossy mode parameters */
} Ipsloss;
```

JPEG COMPRESSION SCAN MODE TYPES

```
typedef union {  
    lpslossy_option    *tsLossy;        /* pointer to optional lossy mode    */  
    lps Huffman        *tsHuffman;      /* pointer to optional Huffman mode  */  
    lpsarith           *tsArith;        /* pointer to optional arithmetic mode*/  
    lpslossy_huffman   *tsLossyHuffman; /* pointer to optional lossy & Huffman*/  
    lpslossy_arith     *tsLossyArith;   /* pointer to optional lossy & arithmetic*/  
} Ipsscan_mode;
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 12088-4:1995

5.4.3 Structure data type definitions

Structures are defined for the representation of data objects and for the convenient grouping of parameters required by various functional elements.

5.4.3.1 IPI-PIKS structures

ATTRIBUTES, COMPOSITE IDENTIFIER ARRAY

```
typedef struct {
    Ipoint          uArrayLength;      /* identifier array length */
} Ipsattrs_id_array;
```

ATTRIBUTES, COMPOSITE IDENTIFIER RECORD

```
typedef struct {
    Ipoint          uRecordLength;     /* identifier array record length */
    char            **tatzNames;       /* pointer to array of name string pointers */
} Ipsattrs_id_record;
```

ATTRIBUTES, HISTOGRAM

```
typedef struct {
    Ipoint          uHistSize;         /* histogram size */
    Ipfloor         rLowerBound;       /* lower amplitude bound */
    Ipfloor         rUpperBound;      /* upper amplitude bound */
} Ipsattrs_hist;
```

ATTRIBUTES, IMAGE

```
typedef struct {
    Idntuple        nImageSize;        /* image sizes ND 5-tuple identifier */
    Idntuple        nBandType;         /* band data types SD B-tuple identifier */
    Idntuple        nBandPrecision;    /* band precision ND B-tuple identifier */
    Ipfloor         rXWhite;           /* X tristimulus value white point */
    Ipfloor         rYWhite;           /* Y tristimulus value white point */
    Ipfloor         rZWhite;           /* Z tristimulus value white point */
    Ipoint          iImageStructureOption; /* image structure option */
    Ipoint          iColourOption;     /* colour space option */
} Ipsattrs_image;
```

ATTRIBUTES, LOOKUP TABLE

```
typedef struct {
    Ipuint          uTableEntries;    /* number of table entries      */
    Ipuint          uTableBands;      /* number of table bands        */
    Ipuint          iInputTypeOption; /* table input data type (internal) option */
    Ipuint          iOutputTypeOption; /* table output data type (internal) option */
} Ipsattrs_lookup;
```

ATTRIBUTES, MATRIX

```
typedef struct {
    Ipuint          uColumnSize;      /* matrix column size          */
    Ipuint          uRowSize;         /* matrix row size              */
    Ipuint          iTypeOption;      /* matrix data type (internal) option */
} Ipsattrs_matrix;
```

ATTRIBUTES, NEIGHBOURHOOD ARRAY

```
typedef struct {
    Idntuple        nArraySize;      /* array size ND 5-tuple identifier */
    Idntuple        nKeyPixel;       /* array key pixel SD 5-tuple identifier */
    Ipuint          iScaleFactor;     /* array scale factor              */
    Ipuint          iLabelOption;     /* array semantic label option      */
    Ipuint          iTypeOption;     /* array data type (internal) option */
} Ipsattrs_neighborhood;
```

NOTE - The key pixel position requires signed integer (SD) tuple type representation.

ATTRIBUTES, PIXEL RECORD

```
typedef struct {
    Ipuint          uRecordLength;    /* pixel record length          */
    Idntuple        nTypeOption;     /* band data types SD B-tuple identifier */
} Ipsattrs_pixel_record;
```

ATTRIBUTES, ROI

```
typedef struct {
    Idntuple        nROIsize;        /* ROI size ND 5-tuple identifier */
    Ipuint          iStructureOption; /* ROI structure option            */
    Ipepolarity     ePolarityOption; /* ROI polarity option             */
} Ipsattrs_roi;
```

ATTRIBUTES, ROI ARRAY

```
typedef struct {
    Idntuple          nROISize;          /* ROI array size ND 5-tuple identifier */
} Ipsattrrs_roi_array;
```

ATTRIBUTES, ROI COORDINATE

```
typedef struct {
    Idntuple          nROISize;          /* ROI coordinate size ND 5-tuple identifier */
    Ipoint            uCollectionSize;    /* number of coordinates */
    Ipepolarity        ePolarityOption;  /* ROI polarity option */
} Ipsattrrs_roi_coordinate;
```

ATTRIBUTES, ROI ELLIPTICAL

```
typedef struct {
    Idntuple          nROISize;          /* ROI elliptical array size ND 5-tuple identifier */
    Idntuple          nEllipseCentre;    /* ellipse centre RD 5-tuple identifier */
    Idntuple          nEllipseLength;    /* ellipse axial lengths RD 5-tuple identifier */
    Idntuple          nIndexManip;       /* index manipulation CS 5-tuple identifier */
    Ipedimension      eDimensionOption;  /* ellipse dimension option */
    Ipepolarity        ePolarityOption;  /* ROI polarity option */
} Ipsattrrs_roi_elliptical;
```

ATTRIBUTES, ROI GENERIC

```
typedef struct {
    Idntuple          nROISize;          /* ROI generic size ND 5-tuple identifier */
} Ipsattrrs_roi_generic;
```

ATTRIBUTES, ROI POLYGON

```
typedef struct {
    Idntuple          nROISize;          /* ROI polygon size ND 5-tuple identifier */
    Ipoint            uPolygonArraySize; /* number of polygon array entries */
    Idntuple          nIndexManip;       /* index manipulation CS 5-tuple identifier */
    Ipepolarity        ePolarityOption;  /* ROI polarity option */
} Ipsattrrs_roi_polygon;
```

ATTRIBUTES, ROI RECTANGULAR

```
typedef struct {
    Idntuple          nROISize;          /* ROI rectangular size ND 5-tuple identifier */
    Idntuple          nRectangleStart;    /* rectangle start position ND 5-tuple identifier */
    Idntuple          nRectangleEnd;      /* rectangle end position ND 5-tuple identifier */
    Idntuple          nIndexManip;        /* index manipulation CS 5-tuple identifier */
    Ipedimension      eDimensionOption;   /* rectangle dimension option */
    Ipepolarity        ePolarityOption;   /* ROI polarity option */
} Ipsattrs_roi_rectangular;
```

ATTRIBUTES, STATIC ARRAY

```
typedef struct {
    Idntuple          nArraySize;         /* static array size ND 5-tuple identifier */
    Ipin              iLabelOption;       /* static array semantic label option */
    Ipin              iTypeOption;        /* static array data type (internal) option */
} Ipsattrs_static;
```

ATTRIBUTES, TUPLE

```
typedef struct {
    Ipuin             uTupleSize;         /* tuple size */
    Ipin              iTypeOption;        /* tuple data type (internal) option */
} Ipsattrs_tuple;
```

ATTRIBUTES, VALUE BOUNDS COLLECTION

```
typedef struct {
    Ipuin             uBoundsSize;        /* collection size */
    Ipsparameter_arith stLowerBound;      /* lower amplitude bound pointer union */
    Ipsparameter_arith stUpperBound;      /* upper amplitude bound pointer union */
    Ipin              iTypeOption;        /* pixel data type (parameter) option */
} Ipsattrs_bounds;
```

ATTRIBUTES, VIRTUAL REGISTER

```
typedef struct {
    Ipbbool           bImmediate;         /* immediate mode flag */
    Ipin              iTypeOption;        /* virtual register data type (vreg) option */
} Ipsattrs_vreg;
```

BINDINGS, IMAGE

```
typedef struct {
    Idntuple      nMatch;          /* match point SD 5-tuple */
    Idnroi        nROI;           /* ROI identifier */
    Idntuple      nOffset;        /* ROI offset SD 5-tuple */
} Ipsbind_image;
```

DRAW PIXELS IMAGE COLLECTION ELEMENT

```
typedef struct {
    Ipuint        uHorizontal;    /* horizontal coordinate value */
    Ipuint        uVertical;      /* vertical coordinate value */
    Ipuint        uDepth;         /* depth coordinate value */
    Ipuint        uTemporal;      /* temporal coordinate value */
    Ipuint        uBand;          /* band coordinate value */
    union {
        Ipbool    bValue;
        Ipuint    uValue;
        Ipuint    iValue;
    } sPixel;
} Ipsdraw_pixels;
```

DRAW PIXELS ROI COLLECTION ELEMENT

```
typedef struct {
    Ipuint        uHorizontal;    /* horizontal coordinate value */
    Ipuint        uVertical;      /* vertical coordinate value */
    Ipuint        uDepth;         /* depth coordinate value */
    Ipuint        uTemporal;      /* temporal coordinate value */
    Ipuint        uBand;          /* band coordinate value */
    Ipbool        bPixelValue;    /* ROI pixel value */
} Ipsdraw_pixels_roi;
```

INQUIRE PIKS IMPLEMENTATION INFORMATION

```
typedef struct {
    char          zProfile[IL_PROFILE]; /* conformance profile label */
    Idntuple      nDimensionLimit;      /* maximum dimension sizes identifier */
    Ipuint        iNDPrecision;         /* storage precision of ND data type */
    Ipuint        iSDPrecision;         /* storage precision of SD data type */
    Ipuint        iRDPrecision;         /* storage precision of RD data type */
    char          zRD[IL_REAL];         /* implementation method of RD data type */
    char          *tzInformation;        /* implementation information pointer */
    Ipbinary_event eAsynch;             /* asynchronous support */
} Ipspiks_impl;
```

PIKS MODES

```
typedef struct {
    Iperoi_control_mode    eROIControlMode;    /* PIKS ROI control mode */
    Iperoi_process_mode    eROIProcessMode;    /* PIKS ROI processing mode */
    Ipematch_mode          eMatchPointMode;    /* PIKS match point control mode */
    Ipeindex_mode          eIndexAssignmentMode; /* PIKS index assignment mode */
    Ipesynchronicity_mode  eSynchronicityMode; /* PIKS synchronicity mode */
    Ipeerror_report_mode   eErrorReportMode;   /* PIKS error report mode */
    Ipeint                 iImageResampleMode; /* PIKS image resampling selection mode */
    Ipeint                 iROIResampleMode;   /* PIKS ROI resampling selection */
} Ispiks_modes;
```

PIKS STATES

```
typedef struct {
    Ipepiks_operation_state eOperationState;    /* PIKS operational state */
    Ipechain_state          eChainState;        /* PIKS chain state */
    Ipeerror_state          eErrorState;        /* PIKS error state */
    Ipesynchronicity_state  eSynchronicityState; /* PIKS synchronicity state */
} Ispiks_states;
```

VIRTUAL REGISTER ARITHMETIC DATA TYPE CHOICE

```
typedef struct {
    Ipevreg_chc            eChc;                /* data source choice option */
    Ipsparameter_arith_vreg sData;              /* data source */
} Ipsparameter_arith_vreg_chc;
```

VIRTUAL REGISTER BACKGROUND OPTION CHOICE

```
typedef struct {
    Ipevreg_chc            eChc;                /* data source choice option */
    Ipsbackground_vreg      sOption;            /* level slice background option source */
} Ipsbackground_vreg_chc;
```

VIRTUAL REGISTER BASIC PARAMETER DATA TYPE CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipsparameter_basic_vreg sData;      /* basic parameter data source */
} Ipsparameter_basic_vreg_chc;
```

VIRTUAL REGISTER BP ARRAY CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipsabool_vreg        sData;        /* Boolean data array source */
} Ipsabool_vreg_chc;
```

VIRTUAL REGISTER BP PARAMETER CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipsbool_vreg         sData;        /* Boolean data value source */
} Ipsbool_vreg_chc;
```

VIRTUAL REGISTER CHAR DATA CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipschar_vreg         sCharacter;    /* character source */
} Ipschar_vreg_chc;
```

VIRTUAL REGISTER COLOUR PARAMETER CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipsparameter_colour_vreg sData;    /* colour data parameter source */
} Ipsparameter_colour_vreg_chc;
```

VIRTUAL REGISTER COMPLEX CONVERSION OPTION CHOICE

```
typedef struct {  
    Ipevreg_chc          eChc;          /* data source choice option */  
    Ipscomplex_conv_vreg sOption;       /* option source */  
} Ipscomplex_conv_vreg_chc;
```

VIRTUAL REGISTER COMPOSITE ID ARRAY ATTRIBUTES CHOICE

```
typedef struct {  
    Ipevreg_chc          eChc;          /* data source choice option */  
    Ipsattrs_id_array_vreg sAttrs;      /* composite id array attributes source */  
} Ipsattrs_id_array_vreg_chc;
```

VIRTUAL REGISTER COMPOSITE ID RECORD ATTRIBUTES CHOICE

```
typedef struct {  
    Ipevreg_chc          eChc;          /* data source choice option */  
    Ipsattrs_id_record_vreg sAttrs;     /* composite id record attributes source */  
} Ipsattrs_id_record_vreg_chc;
```

VIRTUAL REGISTER CP PARAMETER CHOICE

```
typedef struct {  
    Ipevreg_chc          eChc;          /* data source choice option */  
    Ipscomplex_vreg      sData;         /* complex data value source */  
} Ipscomplex_vreg_chc;
```

VIRTUAL REGISTER DERIVATIVE OPTION CHOICE

```
typedef struct {  
    Ipevreg_chc          eChc;          /* data source choice option */  
    Ipsderiv_vreg        sOption;       /* derivative option source */  
} Ipsderiv_vreg_chc;
```

VIRTUAL REGISTER DIMENSION OPTION CHOICE

```
typedef struct {  
    Ipevreg_chc          eChc;          /* data source choice option */  
    Ipsdimension_vreg    sOption;       /* dimension option source */  
} Ipsdimension_vreg_chc;
```

VIRTUAL REGISTER DRAW PIXELS IMAGE DATA CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipsdraw_pixels_vreg  sData;         /* draw pixels image data source */
} Ipsdraw_pixels_vreg_chc;
```

VIRTUAL REGISTER DRAW PIXELS ROI DATA CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipsdraw_pixels_roi_vreg sData;      /* draw pixels ROI data source */
} Ipsdraw_pixels_roi_vreg_chc;
```

VIRTUAL REGISTER EDGE PARAMETER CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipsparameter_edge_vreg sData;      /* edge parameter data source */
} Ipsparameter_edge_vreg_chc;
```

VIRTUAL REGISTER EROSION OPTION CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipserosion_vreg      sOption;       /* erosion option source */
} Ipserosion_vreg_chc;
```

VIRTUAL REGISTER EXTERNAL IMAGE PIXEL DATA TYPES CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipspiks_pixel_types_vreg sData;     /* external image pixel data source */
} Ipspiks_pixel_types_vreg_chc;
```

VIRTUAL REGISTER FLIP, SPIN, TRANSPOSE OPTION CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipsflip_vreg         sOption;       /* flip, spin, transpose option source */
} Ipsflip_vreg_chc;
```

VIRTUAL REGISTER FREQUENCY SHIFT OPTION CHOICE

```
typedef struct {  
    Ipevreg_chc          eChc;          /* data source choice option */  
    Ipsfrequency_shift_vreg sOption;     /* frequency shift option source */  
} Ipsfrequency_shift_vreg_chc;
```

VIRTUAL REGISTER GAMMA CONVERSION OPTION CHOICE

```
typedef struct {  
    Ipevreg_chc          eChc;          /* data source choice option */  
    Ipsgamma_vreg        sOption;       /* gamma conversion option source */  
} Ipsgamma_vreg_chc;
```

VIRTUAL REGISTER HISTOGRAM ATTRIBUTES CHOICE

```
typedef struct {  
    Ipevreg_chc          eChc;          /* data source choice option */  
    Ipsattrs_hist_vreg   sAttrs;        /* histogram attributes source */  
} Ipsattrs_hist_vreg_chc;
```

VIRTUAL REGISTER IMAGE ATTRIBUTES CHOICE

```
typedef struct {  
    Ipevreg_chc          eChc;          /* data source choice option */  
    Ipsattrs_image_vreg  sAttrs;        /* image attributes source */  
} Ipsattrs_image_vreg_chc;
```

VIRTUAL REGISTER INDEX OPTION CHOICE

```
typedef struct {  
    Ipevreg_chc          eChc;          /* data source choice option */  
    Ipsindex_vreg        sOption;       /* index assignment option source */  
} Ipsindex_vreg_chc;
```

VIRTUAL REGISTER LOGICAL PARAMETER CHOICE

```
typedef struct {  
    Ipevreg_chc          eChc;          /* data source choice option */  
    Ipsparameter_logical_vreg sAttrs;   /* logical data value source */  
} Ipsparameter_logical_vreg_chc;
```

VIRTUAL REGISTER LOOKUP TABLE ATTRIBUTES CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipsattrs_lut_vreg    sAttrs;        /* lookup table attributes source */
} Ipsattrs_lut_vreg_chc;
```

VIRTUAL REGISTER MAGNITUDE OPTION CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipsmagnitude_vreg    sOption;       /* magnitude option source */
} Ipsmagnitude_vreg_chc;
```

VIRTUAL REGISTER MATRIX ATTRIBUTES CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipsattrs_matrix_vreg sAttrs;        /* matrix attributes source */
} Ipsattrs_matrix_vreg_chc;
```

VIRTUAL REGISTER NEIGHBOURHOOD ARRAY ATTRIBUTES CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipsattrs_nbhood_vreg sAttrs;        /* neighbourhood array attributes source */
} Ipsattrs_nbhood_vreg_chc;
```

VIRTUAL REGISTER NEIGHBOURHOOD ARRAY SET CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipsanbhood_vreg      sArrays;       /* neighbourhood array set source */
} Ipsanbhood_vreg_chc;
```

VIRTUAL REGISTER NP ARRAY DATA CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipsaint_vreg         sData;         /* NP data array source */
} Ipsaint_vreg_chc;
```

VIRTUAL REGISTER NP DATA CHOICE

```
typedef struct {  
    Ipevreg_chc          eChc;          /* data source choice option */  
    Ipsuint_vreg         sData;         /* NP data value source */  
} Ipsuint_vreg_chc;
```

VIRTUAL REGISTER NUMERIC PARAMETER CHOICE

```
typedef struct {  
    Ipevreg_chc          eChc;          /* data source choice option */  
    Ipsparameter_numeric_vreg sData;    /* numeric parameter data source */  
} Ipsparameter_numeric_vreg_chc;
```

VIRTUAL REGISTER OPEN CLOSE OPTION CHOICE

```
typedef struct {  
    Ipevreg_chc          eChc;          /* data source choice option */  
    Ipsopen_close_vreg   sData;         /* open close option source */  
} Ipsopen_close_vreg_chc;
```

VIRTUAL REGISTER PIKS MODES CHOICE

```
typedef struct {  
    Ipevreg_chc          eChc;          /* data source choice option */  
    Ipspiks_modes_vreg   sOption;       /* PIKS modes option source */  
} Ipspiks_modes_vreg_chc;
```

VIRTUAL REGISTER PIXEL PARAMETER CHOICE

```
typedef struct {  
    Ipevreg_chc          eChc;          /* data source choice option */  
    Ipsparameter_pixel_vreg sData;      /* pixel parameter source */  
} Ipsparameter_pixel_vreg_chc;
```

VIRTUAL REGISTER PIXEL RECORD ATTRIBUTES CHOICE

```
typedef struct {  
    Ipevreg_chc          eChc;          /* data source choice option */  
    Ipsattrs_pixel_record_vreg sAttrs;  /* pixel record attributes source */  
} Ipsattrs_pixel_record_vreg_chc;
```

VIRTUAL REGISTER POLARITY OPTION CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipspolarity_vreg     sPolarity;      /* polarity option source */
} Ipspolarity_vreg_chc;
```

VIRTUAL REGISTER REFLECTION OPTION CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipsreflection_vreg    sOption;       /* reflection option source */
} Ipsreflection_vreg_chc;
```

VIRTUAL REGISTER ROI ATTRIBUTES CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipsattrs_roi_vreg     sAttrs;       /* ROI attributes source */
} Ipsattrs_roi_vreg_chc;
```

VIRTUAL REGISTER ROTATION OPTION CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipsrotation_vreg      sOption;      /* rotate option source */
} Ipsrotation_vreg_chc;
```

VIRTUAL REGISTER RP FLOAT ARRAY CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipsafloat_vreg       sData;         /* RP data array source */
} Ipsafloat_vreg_chc;
```

VIRTUAL REGISTER RP FLOAT DATA CHOICE

```
typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option */
    Ipsfloat_vreg        sData;         /* RP data value source */
} Ipsfloat_vreg_chc;
```

VIRTUAL REGISTER SP DATA CHOICE

```
typedef struct {  
    lpevreg_chc          eChc;          /* data source choice option */  
    lpsint_vreg          sData;         /* SP data source */  
} lpsint_vreg_chc;
```

VIRTUAL REGISTER STATIC ARRAY ATTRIBUTES CHOICE

```
typedef struct {  
    lpevreg_chc          eChc;          /* data source choice option */  
    lpsattrs_static_vreg saAttrs;       /* static array attributes source */  
} lpsattrs_static_vreg_chc;
```

VIRTUAL REGISTER TRANSFORM OPTION CHOICE

```
typedef struct {  
    lpevreg_chc          eChc;          /* data source choice option */  
    lpstransform_vreg    sOption;       /* transform option source */  
} lpstransform_vreg_chc;
```

VIRTUAL REGISTER TUPLE ATTRIBUTES CHOICE

```
typedef struct {  
    lpevreg_chc          eChc;          /* data source choice option */  
    lpsattrs_tuple_vreg  saAttrs;       /* tuple attributes source */  
} lpsattrs_tuple_vreg_chc;
```

VIRTUAL REGISTER VALUE BOUNDS ATTRIBUTES CHOICE

```
typedef struct {  
    lpevreg_chc          eChc;          /* data source choice option */  
    lpsattrs_bounds_vreg saAttrs;       /* value bounds attributes source */  
} lpsattrs_bounds_vreg_chc;
```

VIRTUAL REGISTER VIRTUAL REGISTER ATTRIBUTES CHOICE

```
typedef struct {  
    lpevreg_chc          eChc;          /* data source choice option */  
    lpsattrs_vreg_vreg   saAttrs;       /* virtual register attributes source */  
} lpsattrs_vreg_vreg_chc;
```

VIRTUAL REGISTER VIRTUAL REGISTER VALUE DATA TYPES CHOICE

```
typedef struct {  
    lpevreg_chc          eChc;          /* data source choice option */  
    lpsparameter_vreg_vreg sValue;      /* virtual register virtual register data source */  
} lpsparameter_vreg_vreg_chc;
```

5.4.3.2 IPI-IIF structures

GROUP 3 FACSIMILE COMPRESSION PARAMETERS

```
typedef struct {
    Ipoint          iMode;          /* compression mode          */
} Ipsgroup_3;
```

GROUP 4 FACSIMILE COMPRESSION PARAMETERS

```
typedef struct {
    Ipoint          iClass;         /* classification            */
} Ipsgroup_4;
```

JBIG COMPRESSION PARAMETERS

```
typedef struct {
    Ipoint          uPlanes;        /* number of bit planes      */
    Ipoint          bArrangementFlag; /* progressive data arrangement flag */
    Ipoint          uInitialLayer;   /* initial layer to be transmitted */
    Ipoint          uFinalLayer;     /* final layer to be transmitted   */
    Ipoint          bInitialFlag;    /* initial layer template flag    */
    Ipoint          bOrderFlag;      /* transmission order high to low flag */
    Ipoint          uLines;          /* number of lines per stripe    */
    Ipoint          uStripes;        /* number of stripes            */
    Ipoint          uHorizontal;     /* maximum horizontal offset for template */
    Ipoint          uVertical;       /* maximum vertical offset for template */
} Ipsjbig;
```

JPEG COMPRESSION FRAME PARAMETERS

```
typedef struct {
    Ipoint          uScans;          /* number of scans           */
    Ipoint          *tuaScanComponents; /* pointer to list of components in each scan */
    Ipoint          bEntropyFlag;    /* arithmetic entropy coding flag */
    Ipoint          bLosslessFlag;   /* lossless or lossy mode selection */
    Ipsloss         sLossInformation; /* lossless or lossy mode information */
    Ipsscan_option  *tuaScanOptions; /* pointer to array of scan optional parameters */
} Ipsframe;
```

JPEG COMPRESSION HIERARCHICAL PARAMETERS

```
typedef struct {
    Ipoint          uFrames;         /* number of frames          */
    Ipshierarchical_frame *tsaFrame; /* hierarchical frame content structures pointer */
} Ipshierarchical;
```

JPEG COMPRESSION HIERARCHICAL FRAME PARAMETERS

```
typedef struct {
    Ipbool      bDifferential;    /* differential frame          */
    Ipuint      uLines;          /* number of lines in frame    */
    Ipuint      uSamples;        /* number of samples per line  */
    Ipuint      *tuaComponents;   /* pointer to list of components in frame */
} Ipsierarchical_frame;
```

JPEG COMPRESSION LOSSLESS PARAMETERS

```
typedef struct {
    Ipuint      uPredictor;       /* scan predictor              */
    Ipuint      uTransform;       /* point transform value       */
} Ipslossless;
```

JPEG COMPRESSION LOSSY PARAMETERS

```
typedef struct {
    Ipbool      bMode;           /* extended mode flag          */
    Ipbool      bProgressive;    /* extended progressive mode flag or NULL */
    Ipuint      *tuaSelector;    /* quantization table selector */
    Ipsscan     *tuaComponent;   /* pointer to scan component parameters */
} Ipslossy;
```

JPEG COMPRESSION PARAMETERS

```
typedef struct {
    Ipbool      bHierarchicalMode; /* hierarchical mode flag          */
    Ipsierarchical sFrames;        /* hierarchical frame parameters or NULL */
    Ipsframe     *tsFrameInformation; /* pointer to frame information */
} Ipsjpeg;
```

JPEG COMPRESSION SCAN PARAMETERS

```
typedef struct {
    Ipuint      uDCEntropy;       /* scan component DC entropy table selector */
    Ipuint      uACEntropy;       /* scan component AC entropy table selector */
    Ipuint      uStart;          /* start of spectral selection              */
    Ipuint      uEnd;            /* end of spectral selection                */
    Ipuint      uHigh;           /* successive approximation bit high        */
    Ipuint      uLow;            /* successive approximation bit low         */
} Ipsscan;
```

JPEG COMPRESSION SCAN OPTION ARITHMETIC PARAMETERS

```
typedef struct {  
    Ipuint          uArithDest;      /* arithmetic coding table destination identifier */  
    Ipuint          uTable;          /* DC or lossless arithmetic entropy table value */  
} Ipsarith;
```

JPEG COMPRESSION SCAN OPTION HUFFMAN PARAMETERS

```
typedef struct {  
    Ipuint          uHuffman;        /* Huffman table destination identifier */  
    Ipuint          uaLosslessTable[32]; /* Huffman DC or lossless table */  
} Ipshuffman;
```

JPEG COMPRESSION SCAN OPTION LOSSY PARAMETERS

```
typedef struct {  
    Ipuint          uPrecision;      /* quantization table element precision */  
    Ipuint          uDest;           /* quantization table destination identifier */  
    Ipuint          uaElements[64];  /* quantization table elements */  
} Ipslossy_option;
```

JPEG COMPRESSION SCAN OPTION LOSSY ARITHMETIC PARAMETERS

```
typedef struct {  
    Ipuint          uArithId;        /* arithmetic coding table destination identifier */  
    Ipuint          uConditioning;    /* AC conditioning table value */  
} Ipslossy_arith;
```

JPEG COMPRESSION SCAN OPTION LOSSY HUFFMAN PARAMETERS

```
typedef struct {  
    Ipuint          uLossyHuffman;    /* Huffman table destination identifier */  
    Ipuint          uaLossyTable[272]; /* AC Huffman table */  
} Ipslossy_huffman;
```

JPEG COMPRESSION SCAN OPTION PARAMETERS

```
typedef struct {  
    Ipsscan_mode    sScanOption;      /* scan optional parameters */  
    Ipuint          uRestart;          /* restart interval segment length */  
} Ipsscan_option;
```

MPEG-1 COMPRESSION PARAMETERS

```
typedef struct {  
    Ipuint          uIntraframeInterval; /* intraframe interval */  
    Ipuint          uFrameInterval;     /* predicted frame interval */  
    Ipuint          uPictureRate;       /* picture rate */  
} Ipsmpeg_1;
```

6 Macro definitions

6.1 Unbounded option and status codes

Numerous options and status returns of functional elements are specified or identified by macros for IPI-PIKS and IPI-IIF in order to permit the future addition of choices.

6.1.1 IPI-PIKS unbounded option and status codes

BIT SHIFT OPTIONS

#define IBIT_SHIFT_LEFT_OVERFLOW	1 /* left overflow shift	*/
#define IBIT_SHIFT_RIGHT_OVERFLOW	2 /* right overflow shift	*/
#define IBIT_SHIFT_LEFT_BARREL	3 /* left barrel shift	*/
#define IBIT_SHIFT_RIGHT_BARREL	4 /* right barrel shift	*/
#define IBIT_SHIFT_LEFT_ARITH	5 /* left arithmetic shift	*/
#define IBIT_SHIFT_RIGHT_ARITH	6 /* right arithmetic shift	*/

BOOLEAN VALUES

#define IBOOL_FALSE	0 /* false Boolean value	*/
#define IBOOL_TRUE	1 /* true Boolean value	*/

CHAIN STATUS CODES

#define ICHAIN_NONEXISTENT	0 /* no chain exists for specified identifier	*/
#define ICHAIN_UNSTARTED	1 /* execution has not been requested	*/
#define ICHAIN_SCHEDULED	2 /* execution requested, not begun	*/
#define ICHAIN_RUNNING	3 /* execution has begun and is active	*/
#define ICHAIN_IDLING	4 /* execution begun, elements inactive	*/
#define ICHAIN_ABORTED	5 /* execution was explicitly aborted	*/
#define ICHAIN_FAILURE	6 /* execution terminated with failure	*/
#define ICHAIN_COMPLETE	7 /* execution completed naturally	*/

COLOUR CONVERSION, DIRECTION OPTIONS

#define ICOLOUR_DIRECTION_RGB_CMY	1 /* RGB to CMY option	*/
#define ICOLOUR_DIRECTION_RGB_CMYK	2 /* RGB to CMYK option	*/
#define ICOLOUR_DIRECTION_CMY_RGB	3 /* CMY to RGB option	*/
#define ICOLOUR_DIRECTION_CMYK_RGB	4 /* CMYK to RGB option	*/

COLOUR CONVERSION, FUNCTION INPUT AND OUTPUT OPTIONS

#define ICOLOUR_CONV_RGB	1 /* RGB	*/
#define ICOLOUR_CONV_IHS	2 /* IHS	*/
#define ICOLOUR_CONV_YXY	3 /* Yxy	*/
#define ICOLOUR_CONV_YUV	4 /* Yuv	*/
#define ICOLOUR_CONV_LAB	5 /* L*a*b*	*/
#define ICOLOUR_CONV_LUV	6 /* L*u*v*	*/

COLOUR CONVERSION, METHOD OPTIONS

#define ICOLOUR_METHOD_SIMPLE	1 /* Simple conversion	*/
#define ICOLOUR_METHOD_IMPL_DEPENDENT	2 /* Implementation dependent conversion	*/

COLOUR CONVERSION, RGB COLOUR OPTIONS

#define ICOLOUR_RGB_UNCALIBRATED	1 /* uncalibrated	*/
#define ICOLOUR_RGB_CIE_SPECTRAL	2 /* CIE spectral	*/
#define ICOLOUR_RGB_LIN_NTSC	3 /* linear NTSC	*/
#define ICOLOUR_RGB_LIN_EBU	4 /* linear EBU	*/
#define ICOLOUR_RGB_LIN_SMPTE	5 /* linear SMPTE	*/
#define ICOLOUR_RGB_LIN_CCIR	6 /* linear CCIR	*/

COLOUR SPACE OPTIONS

#define ICOLOUR_NONE	1 /* unspecified	*/
#define ICOLOUR_NON_STANDARD_RGB	2 /* non-standard RGB	*/
#define ICOLOUR_LIN_CCIR_D65_RGB	3 /* linear CCIR illuminant D65 RGB	*/
#define ICOLOUR_LIN_CIE_E_RGB	4 /* linear CIE illuminant E RGB	*/
#define ICOLOUR_LIN_EBU_C_RGB	5 /* linear EBU illuminant C RGB	*/
#define ICOLOUR_LIN_EBU_D65_RGB	6 /* linear EBU illuminant D65 RGB	*/
#define ICOLOUR_LIN_NTSC_C_RGB	7 /* linear NTSC illuminant C RGB	*/
#define ICOLOUR_LIN_NTSC_D65_RGB	8 /* linear NTSC illuminant D65 RGB	*/
#define ICOLOUR_LIN_SMPTE_D65_RGB	9 /* linear SMPTE illuminant D65 RGB	*/
#define ICOLOUR_GAMMA_CCIR_D65_RGB	10 /* gamma CCIR illuminant D65 RGB	*/
#define ICOLOUR_GAMMA_EBU_C_RGB	11 /* gamma EBU illuminant C RGB	*/
#define ICOLOUR_GAMMA_EBU_D65_RGB	12 /* gamma EBU illuminant D65 RGB	*/
#define ICOLOUR_GAMMA_NTSC_C_RGB	13 /* gamma NTSC illuminant C RGB	*/
#define ICOLOUR_GAMMA_NTSC_D65_RGB	14 /* gamma NTSC illuminant D65 RGB	*/
#define ICOLOUR_GAMMA_SMPTE_D65_RGB	15 /* gamma SMPTE illuminant D65 RGB	*/
#define ICOLOUR_LUM_CHR_EBU_C_YUV	16 /* EBU illuminant C YUV	*/
#define ICOLOUR_LUM_CHR_EBU_D65_YUV	17 /* EBU illuminant D65 YUV	*/
#define ICOLOUR_LUM_CHR_NTSC_C_YIQ	18 /* NTSC illuminant C YIQ	*/
#define ICOLOUR_LUM_CHR_NTSC_D65_YIQ	19 /* NTSC illuminant D65 YIQ	*/
#define ICOLOUR_LUM_CHR_SMPTE_D65_YCBCR	20 /* SMPTE illuminant D65 YCbCr	*/
#define ICOLOUR_CIE_XYZ	21 /* CIE XYZ	*/
#define ICOLOUR_CIE_UVW	22 /* CIE UVW	*/
#define ICOLOUR_CIE_YXY	23 /* CIE Yxy	*/
#define ICOLOUR_CIE_YUV	24 /* CIE Yuv	*/
#define ICOLOUR_CIE_LAB	25 /* CIE L*a*b*	*/
#define ICOLOUR_CIE_LUV	26 /* CIE L*u*v*	*/
#define ICOLOUR_IHS	27 /* IHS	*/
#define ICOLOUR_CMY	28 /* CMY	*/
#define ICOLOUR_CMYK	29 /* CMYK	*/

COMPOSITE IMAGE IDENTIFIER LIST GET POINTER OPTIONS

#define IID_LIST_CURRENT	1 /* current position	*/
#define IID_LIST_PREDECESSOR	2 /* predecessor position	*/
#define IID_LIST_SUCCESSOR	3 /* successor position	*/
#define IID_LIST_HEAD	4 /* head position	*/
#define IID_LIST_TAIL	5 /* tail position	*/

COMPOSITE IDENTIFIER LIST INSERT POINTER OPTIONS

#define IID_LIST_BEFORE	1 /* insert before current pointer position	*/
#define IID_LIST_AFTER	2 /* insert after current pointer position	*/

CONVOLUTION 2D OPTIONS

#define ICONVOLVE_UPPER_LEFT	1 /* upper left corner justified	*/
#define ICONVOLVE_ENCLOSED	2 /* enclosed array	*/
#define ICONVOLVE_KEY_ZERO	3 /* key pixel. zero exterior	*/
#define ICONVOLVE_KEY_REFLECTED	4 /* key pixel. reflected exterior	*/

DATA TRANSFER OPTIONS

#define ITRANSFER_IIF	1 /* IIF as source	*/
#define ITRANSFER_APPL	2 /* application as source	*/

DATA TRANSFER SYNTAX OPTIONS

#define ISYNTAX_ASN1	1 /* ASN.1 notation	*/
#define ISYNTAX_PRIVATE	2 /* private notation	*/

DATA TYPE (EXTERNAL PIXEL) OPTIONS

#define IPIXEL_BI_MSB_PLANAR	1 /* Boolean planar data from msb	*/
#define IPIXEL_BI_MSB_INTERLEAVED	2 /* Boolean interleaved data from msb	*/
#define IPIXEL_BI_LSB_PLANAR	3 /* Boolean planar data from lsb	*/
#define IPIXEL_BI_LSB_INTERLEAVED	4 /* Boolean interleaved data from lsb	*/
#define IPIXEL_NI_PLANAR	5 /* non-negative integer planar data	*/
#define IPIXEL_NI_INTERLEAVED	6 /* non-negative integer interleaved data	*/
#define IPIXEL_SI_PLANAR	7 /* signed integer planar data	*/
#define IPIXEL_SI_INTERLEAVED	8 /* signed integer interleaved data	*/
#define IPIXEL_TI_PLANAR	9 /* fixed point integer planar data	*/
#define IPIXEL_TI_INTERLEAVED	10 /* fixed point integer interleaved data	*/
#define IPIXEL_RF_PLANAR	11 /* arithmetic real planar data	*/
#define IPIXEL_RF_INTERLEAVED	12 /* arithmetic real interleaved data	*/
#define IPIXEL_CF	13 /* complex data	*/

DATA TYPE (INTERNAL) OPTIONS

#define IDATA_TYPE_INTERNAL_BD	1 /* Boolean	*/
#define IDATA_TYPE_INTERNAL_ND	2 /* unsigned integer	*/
#define IDATA_TYPE_INTERNAL_SD	3 /* signed integer	*/
#define IDATA_TYPE_INTERNAL_RD	4 /* real arithmetic	*/
#define IDATA_TYPE_INTERNAL_CD	5 /* complex arithmetic	*/
#define IDATA_TYPE_INTERNAL_CS	6 /* character	*/

DATA TYPE (PARAMETER) OPTIONS

#define IDATA_TYPE_PARAMETER_BP	1 /* Boolean	*/
#define IDATA_TYPE_PARAMETER_NP	2 /* unsigned integer	*/
#define IDATA_TYPE_PARAMETER_SP	3 /* signed integer	*/
#define IDATA_TYPE_PARAMETER_RP	4 /* real arithmetic	*/
#define IDATA_TYPE_PARAMETER_CP	5 /* complex arithmetic	*/
#define IDATA_TYPE_PARAMETER_CS	6 /* character	*/

DATA TYPE (VIRTUAL REGISTER) OPTIONS

#define IDATA_TYPE_VREG_ID	1 /* internal identifier	*/
#define IDATA_TYPE_VREG_IP	2 /* external identifier	*/
#define IDATA_TYPE_VREG_BP	3 /* Boolean	*/
#define IDATA_TYPE_VREG_NP	4 /* unsigned integer	*/
#define IDATA_TYPE_VREG_SP	5 /* signed integer	*/
#define IDATA_TYPE_VREG_RP	6 /* real arithmetic	*/
#define IDATA_TYPE_VREG_CP	7 /* complex arithmetic	*/
#define IDATA_TYPE_VREG_CS	8 /* character	*/

DIFFERENCE MEASURES OPTIONS

#define IDIFF_MEASURE_SQUARED	1 /* squares difference	*/
#define IDIFF_MEASURE_NORMALIZED	2 /* normalized squares difference	*/

DIMENSION OPTIONS

#define IDIMENSION_VOL	1 /* volume image dimensions	*/
#define IDIMENSION_TEM	2 /* temporal image dimensions	*/
#define IDIMENSION_SPE	3 /* spectral image dimensions	*/

DISTANCE MEASURE OPTIONS

#define IDISTANCE_MEASURE_EUCLIDEAN	1 /* Euclidean	*/
#define IDISTANCE_MEASURE_CITY_BLOCK	2 /* City block	*/

DYADIC, ARITHMETIC COMBINATION OPTIONS

#define IDYADIC_ABSOLUTE_DIFF	1 /* absolute value difference	*/
#define IDYADIC_ADDITION	2 /* addition	*/
#define IDYADIC_ADDITION_SCALED	3 /* addition, scaled	*/
#define IDYADIC_ARCTANGENT	4 /* arctangent ratio	*/
#define IDYADIC_DIVISION	5 /* division	*/
#define IDYADIC_MAXIMUM	6 /* maximum	*/
#define IDYADIC_MINIMUM	7 /* minimum	*/
#define IDYADIC_MULTIPLICATION	8 /* multiplication	*/
#define IDYADIC_SUBTRACTION	9 /* subtraction	*/
#define IDYADIC_SUBTRACTION_SCALED	10 /* subtraction, scaled	*/

DYADIC, COMPLEX COMBINATION OPTIONS

#define IDYADIC_COMPLEX_ADDITION	1 /* addition	*/
#define IDYADIC_COMPLEX_DIVISION	2 /* division	*/
#define IDYADIC_COMPLEX_MULTIPLICATION	3 /* multiplication	*/
#define IDYADIC_COMPLEX_SUBTRACTION	4 /* subtraction	*/

DYADIC, LOGICAL COMBINATION OPTIONS

#define IDYADIC_LOGICAL_AND	1 /* bitwise AND	*/
#define IDYADIC_LOGICAL_NAND	2 /* bitwise NAND	*/
#define IDYADIC_LOGICAL_NOR	3 /* bitwise NOR	*/
#define IDYADIC_LOGICAL_OR	4 /* bitwise OR	*/
#define IDYADIC_LOGICAL_XOR	5 /* bitwise XOR	*/
#define IDYADIC_LOGICAL_INTERSECTION	6 /* Boolean intersection	*/
#define IDYADIC_LOGICAL_UNION	7 /* Boolean union	*/

DYADIC, PREDICATE COMBINATION OPTIONS

#define IDYADIC_PREDICATE_GREATER	1 /* greater than	*/
#define IDYADIC_PREDICATE_GREATER_EQUAL	2 /* greater than or equal to	*/
#define IDYADIC_PREDICATE_LESS	3 /* less than	*/
#define IDYADIC_PREDICATE_LESS_EQUAL	4 /* less than or equal to	*/
#define IDYADIC_PREDICATE_EQUAL	5 /* equal to	*/
#define IDYADIC_PREDICATE_NOT_EQUAL	6 /* not equal to	*/

EDGE DETECTION OPTIONS

#define IEDGE_DETECT_GRAD	1 /* edge gradient	*/
#define IEDGE_DETECT_THRESHOLD	2 /* thresholded edge gradient	*/
#define IEDGE_DETECT_QUANTIZED	3 /* quantized edge angle	*/

EDGE DETECTION, ORTHOGONAL OPTIONS

#define IEDGE_ORTH_EDGE_GRADIENT	1 /* edge gradient	*/
#define IEDGE_ORTH_THRESHOLD	2 /* thresholded edge gradient	*/
#define IEDGE_ORTH_EDGE_ANGLE	3 /* edge angle	*/
#define IEDGE_ORTH_ANGLE_QUANTIZED	4 /* quantized edge angle	*/

EROSION & DILATION PROCESSING OPTIONS

#define IEROSION_ARRAY_NON_ROTATE	1 /* enclosed array, non-rotated element	*/
#define IEROSION_ARRAY_ROTATE	2 /* enclosed array, rotated element	*/
#define IEROSION_KEY_NON_ROTATE	3 /* key pixel, non-rotated element	*/
#define IEROSION_KEY_ROTATE	4 /* key pixel, rotated structuring element	*/

FILTER (BUTTERWORTH) TYPE OPTIONS

#define IFILTER_BUTTERWORTH_LOW_PASS	1 /* low-pass	*/
#define IFILTER_BUTTERWORTH_HIGH_PASS	2 /* high-pass	*/

FILTER (ZONAL) TYPE OPTIONS

#define IFILTER_ZONAL_LOW_PASS	1 /* low-pass filter	*/
#define IFILTER_ZONAL_HIGH_PASS_TERM	2 /* high-pass filter with constant term	*/
#define IFILTER_ZONAL_HIGH_PASS_NO_TERM	3 /* high-pass filter without constant. term	*/

FILTERING PROCESS OPTIONS

#define IFILTERING_ENCLOSED_ARRAY	1 /* enclosed array	*/
#define IFILTERING_KEY_PIXEL	2 /* key pixel	*/

HISTOGRAM COMPUTATION OPTION

#define IHIST_LIMIT_SPECIFIED	1 /* specified limits	*/
#define IHIST_LIMIT_EXTREMA	2 /* image extrema limits	*/

HISTOGRAM MODIFICATION OPTIONS

#define IHIST_MODIFICATION_CUBE_ROOT	1 /* cube root	*/
#define IHIST_MODIFICATION_EXPONENTIAL	2 /* exponential	*/
#define IHIST_MODIFICATION_LOGARITHMIC	3 /* logarithmic	*/
#define IHIST_MODIFICATION_RAYLEIGH	4 /* Rayleigh	*/
#define IHIST_MODIFICATION_UNIFORM	5 /* uniform	*/

HIT OR MISS TRANSFORMATION OPTIONS

#define IHIT_MISS_BRANCH_DETECT	1 /* branch detect	*/
#define IHIT_MISS_BRIDGE	2 /* bridge	*/
#define IHIT_MISS_CONVEX_HULL_FILL	3 /* convex hull fill	*/
#define IHIT_MISS_DIAGONAL_FILL	4 /* diagonal fill	*/
#define IHIT_MISS_DILATE_8_NEIGHBOUR	5 /* dilate, eight-neighbour	*/
#define IHIT_MISS_END_DETECT	6 /* end detect	*/
#define IHIT_MISS_END_DETECT_EAST	7 /* end detect, east	*/
#define IHIT_MISS_END_DETECT_NORTH	8 /* end detect, north	*/
#define IHIT_MISS_END_DETECT_NORTHEAST	9 /* end detect, northeast	*/
#define IHIT_MISS_END_DETECT_NORTHWEST	10 /* end detect, northwest	*/
#define IHIT_MISS_END_DETECT_SOUTH	11 /* end detect, south	*/
#define IHIT_MISS_END_DETECT_SOUTHEAST	12 /* end detect, southeast	*/
#define IHIT_MISS_END_DETECT_SOUTHWEST	13 /* end detect, southwest	*/
#define IHIT_MISS_END_DETECT_WEST	14 /* end detect, west	*/
#define IHIT_MISS_ERODE_8_NEIGHBOUR	15 /* erode, eight-neighbour	*/
#define IHIT_MISS_FATTEN	16 /* fatten	*/
#define IHIT_MISS_H_BREAK	17 /* H break	*/
#define IHIT_MISS_HOLE_DETECT	18 /* hole detect	*/
#define IHIT_MISS_INTERIOR_PIXEL_FILL	19 /* interior pixel fill	*/
#define IHIT_MISS_INTERIOR_PIXEL_REMOVE	20 /* interior pixel remove	*/
#define IHIT_MISS_ISOLATED_PIXEL_DETECT	21 /* isolated pixel detect	*/
#define IHIT_MISS_ISOLATED_PIXEL_REMOVE	22 /* isolated pixel remove	*/
#define IHIT_MISS_MAJORITY_BLACK	23 /* majority black	*/
#define IHIT_MISS_SPUR_REMOVE	24 /* spur remove	*/

IMAGE, GREY SCALE ORIENTATION OPTIONS

#define IGREY_LEFT_RIGHT	1 /* left-to-right	*/
#define IGREY_RIGHT_LEFT	2 /* right-to-left	*/
#define IGREY_TOP_BOTTOM	3 /* top-to-bottom	*/
#define IGREY_BOTTOM_TOP	4 /* bottom-to-top	*/

IMAGE STRUCTURE OPTIONS

#define IIMAGE_STRUCTURE_MON	1 /* monochrome image	*/
#define IIMAGE_STRUCTURE_VOL	2 /* volume image	*/
#define IIMAGE_STRUCTURE_TEM	3 /* temporal image	*/
#define IIMAGE_STRUCTURE_COLR	4 /* colour image	*/
#define IIMAGE_STRUCTURE_SPE	5 /* spectral image	*/
#define IIMAGE_STRUCTURE_VT	6 /* volume-temporal image	*/
#define IIMAGE_STRUCTURE_VC	7 /* volume-colour image	*/
#define IIMAGE_STRUCTURE_VS	8 /* volume-spectral image	*/
#define IIMAGE_STRUCTURE_TC	9 /* temporal-colour image	*/
#define IIMAGE_STRUCTURE_TS	10 /* temporal-spectral image	*/
#define IIMAGE_STRUCTURE_VTC	11 /* volume-temporal-colour image	*/
#define IIMAGE_STRUCTURE_VTS	12 /* volume-temporal-spectral image	*/
#define IIMAGE_STRUCTURE_GEN	13 /* generic image	*/

IMPULSE, PYRAMID SHAPE OPTIONS

#define IPYRAMID_SMOOTHING	1 /* smoothing	*/
#define IPYRAMID_HORIZONTAL	2 /* horizontal derivative	*/
#define IPYRAMID_VERTICAL	3 /* vertical derivative	*/

IMPULSE ROTATION OPTIONS

#define IIMPULSE_NO_REFLECTION	1 /* no reflection	*/
#define IIMPULSE_REFLECTION	2 /* reflection	*/

LOOKUP TABLE MODES

#define ILOOKUP_1D	1 /* one dimension	*/
#define ILOOKUP_2D	2 /* two dimensions	*/

LUMINANCE RGB COLOR OPTIONS

#define ILUM_LIN_CCIR_D65_RGB	1 /* linear CCIR illuminant D65 RGB	*/
#define ILUM_LIN_CIE_E_RGB	2 /* linear CIE illuminant E RGB	*/
#define ILUM_LIN_EBU_C_RGB	3 /* linear EBU illuminant C RGB	*/
#define ILUM_LIN_EBU_D65_RGB	4 /* linear EBU illuminant D65 RGB	*/
#define ILUM_LIN_NTSC_C_RGB	5 /* linear NTSC illuminant C RGB	*/
#define ILUM_LIN_NTSC_D65_RGB	6 /* linear NTSC illuminant D65 RGB	*/
#define ILUM_LIN_SMPTE_D65_RGB	7 /* linear SMPTE illuminant D65 RGB	*/
#define ILUM_GAMMA_EBU_C_RGB	8 /* gamma EBU illuminant C RGB	*/
#define ILUM_GAMMA_EBU_D65_RGB	9 /* gamma EBU illuminant D65 RGB	*/
#define ILUM_GAMMA_NTSC_C_RGB	10 /* gamma NTSC illuminant C RGB	*/
#define ILUM_GAMMA_NTSC_D65_RGB	11 /* gamma NTSC illuminant D65 RGB	*/
#define ILUM_GAMMA_SMPTE_D65_RGB	12 /* gamma SMPTE illuminant D65 RGB	*/

MARKOV SPECTRUM SYMMETRY OPTIONS

#define IMARKOV_ELLIPTICAL	1 /* elliptical	*/
#define IMARKOV_SEPARABLE	2 /* separable	*/

MONADIC, ARITHMETIC OPTIONS

#define IMONADIC_ADDITION_BY	1 /* addition by constant	*/
#define IMONADIC_ADDITION_BY_SCALED	2 /* addition by constant, scaled	*/
#define IMONADIC_DIVISION_BY	3 /* division by constant	*/
#define IMONADIC_DIVISION_OF	4 /* division of constant	*/
#define IMONADIC_MAXIMUM	5 /* maximum with constant	*/
#define IMONADIC_MINIMUM	6 /* minimum with constant	*/
#define IMONADIC_MULTIPLICATION	7 /* multiplication by constant	*/
#define IMONADIC_SUBTRACTION_BY	8 /* subtraction by constant	*/
#define IMONADIC_SUBTRACTION_BY_SCALED	9 /* subtraction by constant, scaled	*/
#define IMONADIC_SUBTRACTION_OF	10 /* subtraction of constant	*/
#define IMONADIC_SUBTRACTION_OF_SCALED	11 /* subtraction of constant, scaled	*/

MONADIC, COMPLEX OPTIONS

#define IMONADIC_COMPLEX_ADDITION_BY	1 /* addition by constant	*/
#define IMONADIC_COMPLEX_DIVISION_BY	2 /* division by constant	*/
#define IMONADIC_COMPLEX_DIVISION_OF	3 /* division of constant	*/
#define IMONADIC_COMPLEX_MULTIPLICATION	4 /* multiplication by constant	*/
#define IMONADIC_COMPLEX_SUBTRACTION_BY	5 /* subtraction by constant	*/
#define IMONADIC_COMPLEX_SUBTRACTION_OF	6 /* subtraction of constant	*/

MONADIC, LOGICAL OPTIONS

#define IMONADIC_LOGICAL_AND	1 /* bitwise AND	*/
#define IMONADIC_LOGICAL_NAND	2 /* bitwise NAND	*/
#define IMONADIC_LOGICAL_NOR	3 /* bitwise NOR	*/
#define IMONADIC_LOGICAL_OR	4 /* bitwise OR	*/
#define IMONADIC_LOGICAL_XOR	5 /* bitwise XOR	*/
#define IMONADIC_LOGICAL_INTERSECTION	6 /* Boolean intersection	*/
#define IMONADIC_LOGICAL_UNION	7 /* Boolean union	*/

MORPHOLOGICAL OPTIONS

#define IMORPHOLOGICAL_GROW	1 /* end grow	*/
#define IMORPHOLOGICAL_GROW_EAST	2 /* end grow, east	*/
#define IMORPHOLOGICAL_GROW_NORTH	3 /* end grow, north	*/
#define IMORPHOLOGICAL_GROW_NORTHEAST	4 /* end grow, northeast	*/
#define IMORPHOLOGICAL_GROW_NORTHWEST	5 /* end grow, northwest	*/
#define IMORPHOLOGICAL_GROW_SOUTH	6 /* end grow, south	*/
#define IMORPHOLOGICAL_GROW_SOUTHEAST	7 /* end grow, southeast	*/
#define IMORPHOLOGICAL_GROW_SOUTHWEST	8 /* end grow, southwest	*/
#define IMORPHOLOGICAL_GROW_WEST	9 /* end grow, west	*/
#define IMORPHOLOGICAL_EXOSKELETON	10 /* exoskeleton	*/
#define IMORPHOLOGICAL_EXOTHIN	11 /* exothin	*/
#define IMORPHOLOGICAL_SHRINK	12 /* shrink	*/
#define IMORPHOLOGICAL_SKELETON	13 /* skeleton	*/
#define IMORPHOLOGICAL_THICKEN	14 /* thicken	*/
#define IMORPHOLOGICAL_THIN	15 /* thin	*/

NEIGHBOURHOOD ARRAY SEMANTIC LABEL OPTIONS

#define INBHOOD_LABEL_GENERIC	1 /* generic	*/
#define INBHOOD_LABEL_DITHER	2 /* dither	*/
#define INBHOOD_LABEL_IMPULSE	3 /* impulse response	*/
#define INBHOOD_LABEL_MASK	4 /* mask	*/
#define INBHOOD_LABEL_STRUCTURE	5 /* structuring element	*/

OBJECT TYPE OPTIONS

#define IOBJECT_IMAGE	1 /* image	*/
#define IOBJECT_HIST	2 /* histogram	*/
#define IOBJECT_ID_ARRAY	3 /* identifier array	*/
#define IOBJECT_ID_LIST	4 /* identifier list	*/
#define IOBJECT_ID_RECORD	5 /* identifier record	*/
#define IOBJECT_LUT	6 /* lookup table	*/
#define IOBJECT_MATRIX	7 /* matrix	*/
#define IOBJECT_NBHOOD	8 /* neighbourhood array	*/
#define IOBJECT_PIXEL_RECORD	9 /* pixel record	*/
#define IOBJECT_ROI_ARRAY	10 /* ROI, array type	*/
#define IOBJECT_ROI_COORDINATE	11 /* ROI, coordinate array type	*/
#define IOBJECT_ROI_ELLIPTICAL	12 /* ROI, elliptical type	*/
#define IOBJECT_ROI_GENERIC	13 /* ROI, generic type	*/
#define IOBJECT_ROI_POLYGON	14 /* ROI, polygon type	*/
#define IOBJECT_ROI_RECTANGULAR	15 /* ROI, rectangular type	*/
#define IOBJECT_STATIC	16 /* static array	*/
#define IOBJECT_TUPLE	17 /* tuple	*/
#define IOBJECT_VALUE_BOUNDS	18 /* value bounds collection	*/
#define IOBJECT_VREG	19 /* virtual register	*/

PAINT BRUSH OPTIONS

#define IBRUSH_HARD	1 /* hard brush	*/
#define IBRUSH_SOFT	2 /* soft brush	*/

PAINTING OPTIONS

#define IPAIN_T_POINT	1 /* point painting	*/
#define IPAIN_T_LINE	2 /* straight line painting	*/

PROBABILITY DENSITY OPTIONS

#define IPROBABILITY_UNIFORM	1 /* uniform	*/
#define IPROBABILITY_GAUSS	2 /* Gaussian	*/

REPOSITORY CLASS OPTIONS

#define IREPOSITORY_IMPULSE	1 /* impulse response array	*/
#define IREPOSITORY_DITHER	2 /* dither array	*/
#define IREPOSITORY_COLOUR	3 /* colour conversion matrix	*/

RESAMPLING OPTIONS

#define IRESAMPLE_NEAREST_NEIGHBOUR	1 /* nearest neighbour	*/
#define IRESAMPLE_2_BILINEAR	2 /* support 2 bilinear	*/
#define IRESAMPLE_2_LUT	3 /* support 2 lookup table	*/
#define IRESAMPLE_4_TRUNC_SINC	4 /* support 4 truncated sinc	*/
#define IRESAMPLE_4_CUBIC_B_SPLINE	5 /* support 4 cubic B-spline	*/
#define IRESAMPLE_4_CUBIC_CONVOLUTION	6 /* support 4 cubic convolution	*/
#define IRESAMPLE_4_LAGRANGIAN	7 /* support 4 Lagrangian	*/
#define IRESAMPLE_4_LUT	8 /* support 4 lookup table	*/
#define IRESAMPLE_8_TRUNC_SINC	9 /* support 8 truncated sinc	*/
#define IRESAMPLE_8_LUT	10 /* support 8 lookup table	*/

ROI STRUCTURE OPTIONS

#define IROI_ARRAY	1 /* array ROI structure	*/
#define IROI_COORDINATE	2 /* coordinate array ROI structure	*/
#define IROI_ELLIPTICAL	3 /* elliptical ROI structure	*/
#define IROI_GENERIC	4 /* generic ROI structure	*/
#define IROI_POLYGON	5 /* polygon ROI structure	*/
#define IROI_RECTANGULAR	6 /* rectangular ROI structure	*/

ROTATION OPTIONS

#define IROTATION_SPACE_2D	1 /* 2D rotation	*/
#define IROTATION_SPACE_3D	2 /* 3D rotation	*/

SEQUENCE RUNNING MEASURE OPTIONS

#define ISEQ_MEASURE_DIFF	1 /* difference	*/
#define ISEQ_MEASURE_RATIO	2 /* ratio	*/
#define ISEQ_MEASURE_LOGARITHM_RATIO	3 /* logarithm ratio	*/

SPLIT IMAGE OPTIONS

#define ISPLIT_LEFT_LEFT	1 /* split left 1 - left 2	*/
#define ISPLIT_RIGHT_RIGHT	2 /* split right 1 - right 2	*/
#define ISPLIT_TOP_TOP	3 /* split top 1 - top 2	*/
#define ISPLIT_BOTTOM_BOTTOM	4 /* split bottom 1 - bottom 2	*/
#define ISPLIT_LEFT_RIGHT	5 /* split left 1 - right 2	*/
#define ISPLIT_TOP_BOTTOM	6 /* split top 1 - bottom 2	*/

STATIC ARRAY SEMANTIC LABEL OPTIONS

#define ISTATIC_GENERIC	1 /* generic	*/
#define ISTATIC_POWER	2 /* power spectrum	*/
#define ISTATIC_TRANSFER	3 /* transfer function	*/
#define ISTATIC_WINDOWING	4 /* windowing function	*/

UNARY INTEGER CONVERSION OPTIONS

#define IUNARY_INTEGER_ABSOLUTE	1 /* absolute value	*/
#define IUNARY_INTEGER_CUBE	2 /* cube	*/
#define IUNARY_INTEGER_NEGATIVE	3 /* negative	*/
#define IUNARY_INTEGER_SQUARE	4 /* square	*/

UNARY REAL CONVERSION OPTIONS

#define IUNARY_REAL_ABSOLUTE	1 /* absolute value	*/
#define IUNARY_REAL_ARC_COSINE	2 /* arc cosine	*/
#define IUNARY_REAL_ARC_SINE	3 /* arc sine	*/
#define IUNARY_REAL_ARC_TANGENT	4 /* arc tangent	*/
#define IUNARY_REAL_COSECANT	5 /* cosecant	*/
#define IUNARY_REAL_COSINE	6 /* cosine	*/
#define IUNARY_REAL_COTANGENT	7 /* cotangent	*/
#define IUNARY_REAL_CUBE	8 /* cube	*/
#define IUNARY_REAL_CUBE_ROOT	9 /* cube root	*/
#define IUNARY_REAL_EXPONENTIAL	10 /* exponential	*/
#define IUNARY_REAL_LOGARITHM_E	11 /* logarithm, base e	*/
#define IUNARY_REAL_LOGARITHM_10	12 /* logarithm, base 10	*/
#define IUNARY_REAL_LOGARITHM_2	13 /* logarithm, base 2	*/
#define IUNARY_REAL_NEGATIVE	14 /* negative	*/
#define IUNARY_REAL_RECIPROCAL	15 /* reciprocal	*/
#define IUNARY_REAL_SECANT	16 /* secant	*/
#define IUNARY_REAL_SINE	17 /* sine	*/
#define IUNARY_REAL_SQUARE	18 /* square	*/
#define IUNARY_REAL_SQUARE_ROOT	19 /* square root	*/
#define IUNARY_REAL_TANGENT	20 /* tangent	*/

VIRTUAL REGISTER CONDITIONAL RETURN OPTIONS

#define IVREG_CONDITIONAL_TERMINATE	1 /* terminate chain	*/
#define IVREG_CONDITIONAL_PROCEED	2 /* proceed to next chain operator	*/

WINDOW STATISTIC OPTIONS

#define IWINDOW_MEAN	1 /* mean statistic option	*/
#define IWINDOW_SDEV	2 /* standard deviation option	*/
#define IWINDOW_MEDIAN	3 /* median option	*/
#define IWINDOW_MODE	4 /* mode option	*/

WINDOWING OPTIONS

#define IWINDOWING_BARTLETT	1 /* Bartlett	*/
#define IWINDOWING_BLACKMAN	2 /* Blackman	*/
#define IWINDOWING_HAMMING	3 /* Hamming	*/
#define IWINDOWING_HANNING	4 /* Hanning	*/
#define IWINDOWING_RECTANGULAR	5 /* rectangular	*/

6.1.2 IPI-IIF unbounded option and status codes

BOOLEAN VALUES

#define IBOOL_FALSE	0 /* false Boolean value	*/
#define IBOOL_TRUE	1 /* true Boolean value	*/

COMPONENT CLASSES

#define ICOMPONENT_REQUIRED	1 /* syntax component required class	*/
#define ICOMPONENT_OPTIONAL	2 /* syntax component optional class	*/
#define ICOMPONENT_CHC	3 /* syntax component choice class	*/

COMPRESSION ALGORITHM OPTIONS

#define ICOMPRESSION_FAX_G3	1 /* Fax Group 3	*/
#define ICOMPRESSION_FAX_G4	2 /* Fax Group 4	*/
#define ICOMPRESSION_JBIG	3 /* JBIG field	*/
#define ICOMPRESSION_JPEG	4 /* JPEG interchange format	*/
#define ICOMPRESSION_MPEG	5 /* MPEG bitstream	*/

COMPRESSION MODES

#define ICOMPRESSION_UNCOMPRESSED	1 /* uncompressed mode	*/
#define ICOMPRESSION_ONE_DIMENSIONAL	2 /* one-dimensional mode	*/
#define ICOMPRESSION_TWO_DIMENSIONAL	3 /* two-dimensional mode	*/

ENTITY CLASSES

#define IENTITY_SEQ	1 /* entity sequence class	*/
#define IENTITY_CHC	2 /* entity choice class	*/
#define IENTITY_SEQ_OF	3 /* entity sequence of class	*/
#define IENTITY_SET	4 /* entity set class	*/
#define IENTITY_SET_OF	5 /* entity set of class	*/
#define IENTITY_EQUIVALENCE	6 /* entity equivalence class	*/

GATEWAY STATUS

#define IGATEWAY_OPEN_STATUS	1 /* gateway open status	*/
#define IGATEWAY_ERROR_STATUS	2 /* gateway error status	*/
#define IGATEWAY_CLOSED_STATUS	3 /* gateway closed status	*/

INPUT/OUTPUT ACCESS TYPE OPTIONS

#define IACCESS_RANDOM	1 /* random access	*/
#define IACCESS_SEQUENTIAL	2 /* sequential access	*/

INPUT/OUTPUT DIRECTION OPTIONS

#define IDIRECTION_READ	1 /* read direction	*/
#define IDIRECTION_WRITE	2 /* write direction	*/
#define IDIRECTION_READ_WRITE	3 /* read and write direction	*/

INPUT/OUTPUT DOMAIN OPTIONS

#define IDOMAIN_PIKS	1 /* PIKS domain	*/
#define IDOMAIN_NON_PIKS	2 /* non-PIKS domain	*/

INPUT/OUTPUT STATUS

#define ISTATUS_CLEAR	1 /* port clear	*/
#define ISTATUS_PENDING	2 /* port operation pending	*/

PROFILE INDICATORS

#define IPROFILE_FULL	1 /* full profile of the IIF-DF	*/
#define IPROFILE_PIKS	2 /* PIKS profile of the IIF-DF	*/
#define IPROFILE_COLOUR	3 /* colour profile of the IIF-DF	*/

6.2 IPI-IIF syntax descriptors

The IPI-IIF specifies a number of entities and components which are designated by numeric descriptors for convenience.

6.2.1 IPI-IIF syntax entity types

The IPI-IIF syntax entity type descriptors are represented by one to three digit macro values. The syntax entity names are bound with the prefix ISE_.

ENTITIES FOR THE DESCRIPTION OF THE ENTIRE IIF DATA FORMAT

#define ISE_FULL_DATA_FORMAT	1 /* full data format	*/
#define ISE_FORMAT_DESCRIPTOR	2 /* format descriptor	*/
#define ISE_VERSION	3 /* version	*/
#define ISE_PROFILE	4 /* profile	*/
#define ISE_CONTENTS_HEADER	5 /* contents header	*/
#define ISE_CHAR_STRING	6 /* character string	*/
#define ISE_SPECIAL_CHAR_STRING	7 /* special character string	*/
#define ISE_CONTENTS	8 /* contents	*/
#define ISE_CONTENTS_ELEMENT	9 /* contents element	*/
#define ISE_CONTENTS_BODY	10 /* contents body	*/

ENTITIES FOR THE DESCRIPTION OF IMAGES

#define ISE_IMAGE	101 /* image	*/
#define ISE_IMAGE_STRUCTURE	102 /* image structure	*/
#define ISE_COMPOUND_IMAGE_STRUCTURE	103 /* compound image structure	*/
#define ISE_COMPOUND_IMAGE_ARRAY	104 /* compound image array	*/
#define ISE_DIMENSIONALITY	105 /* dimensionality	*/
#define ISE_DIMENSION_DESCRIPTION	106 /* dimension description	*/
#define ISE_ID	107 /* identifier	*/
#define ISE_SERIALIZATION	108 /* serialization	*/
#define ISE_DATA_PLACEMENT	109 /* data placement	*/
#define ISE_COMPOUND_IMAGE_RECORD	110 /* compound image record	*/
#define ISE_RECORD_CMP	111 /* record component	*/
#define ISE_COMPOUND_IMAGE_LIST	112 /* compound image list	*/
#define ISE_COMPOUND_IMAGE_SET	113 /* compound image set	*/
#define ISE_FUND_IMAGE_STRUCTURE	114 /* fundamental image structure	*/
#define ISE_BAND_RECORD	115 /* band record	*/
#define ISE_BAND_RECORD_CMP	116 /* band record component	*/
#define ISE_METRIC_ARRAY	117 /* metric array	*/
#define ISE_METRIC_ARRAY_ELEMENT	118 /* metric array element	*/
#define ISE_PIXEL_STRUCTURE	119 /* pixel structure	*/
#define ISE_ELEMENTARY_PIXEL_STRUCTURE	120 /* elementary pixel structure	*/
#define ISE_PIXEL_BAND_RECORD	121 /* pixel band record	*/
#define ISE_PIXEL_BAND_RECORD_CMP	122 /* pixel band record component	*/

ENTITIES FOR THE DESCRIPTION OF THE REPRESENTATION OF PIXEL VALUES

#define ISE_REFERENCED_UNIT	201 /* referenced unit	*/
#define ISE_DATA_UNIT	202 /* data unit	*/
#define ISE_SUBDIVIDED_DATA_UNIT	203 /* subdivided data unit	*/
#define ISE_SINGLE_DATA_UNIT	204 /* single data unit	*/
#define ISE_BUILTIN_ENCODED_DATA_UNIT	205 /* builtin encoded data unit	*/
#define ISE_BUILTIN_VALUE	206 /* builtin value	*/
#define ISE_COMPLEX_VALUE	207 /* complex value	*/
#define ISE_EXTERNALLY_DEF_DATA_UNIT	208 /* externally defined data unit	*/
#define ISE_COMPRESSED_DATA_UNIT	209 /* compressed data unit	*/
#define ISE_REGISTERED_DATA_UNIT	210 /* registered data unit	*/
#define ISE_EXTERNAL_REFERENCE	211 /* external reference	*/
#define ISE_EXTERNAL_ADDRESS	212 /* external address	*/

ENTITIES FOR THE DESCRIPTION OF IMAGE-RELATED DATA

#define ISE_IMAGE_RELATED_DATA	301 /* image-related data	*/
#define ISE_HIST	302 /* histogram	*/
#define ISE_PARTITION_CLASS	303 /* partition class	*/
#define ISE_LUT	304 /* look up table	*/
#define ISE_ROI	305 /* region of interest	*/
#define ISE_BOOL_ARRAY	306 /* Boolean array	*/
#define ISE_ELLIPSE	307 /* ellipse	*/
#define ISE_INTERVAL_N_DIMENSIONAL	308 /* interval n-dimensional	*/
#define ISE_INTERVAL_1_DIMENSIONAL	309 /* interval 1 dimensional	*/
#define ISE_COORDINATE_N_DIMENSIONAL	310 /* coordinate n-dimensional	*/
#define ISE_SET_OF_COORDINATES	311 /* set of coordinates	*/
#define ISE_NBHOOD_ARRAY	312 /* neighbourhood array	*/
#define ISE_INDEX_N_DIMENSIONAL	313 /* index n-dimensional	*/
#define ISE_STATIC_ARRAY	314 /* static array	*/
#define ISE_FEATURE_LIST	315 /* feature list	*/
#define ISE_COORDINATE_AND_FEATURE	316 /* coordinate and feature	*/
#define ISE_VALUE_BOUNDS_COLLECTION	317 /* value bounds collection	*/
#define ISE_TRANSFORMATION_MATRIX	318 /* transformation matrix	*/
#define ISE_PIXEL_RECORD	319 /* pixel record	*/
#define ISE_PIXEL_RECORD_CMP	320 /* pixel record component	*/
#define ISE_TUPLE	321 /* tuple	*/

ENTITIES FOR THE DESCRIPTION OF IMAGE ATTRIBUTES

#define ISE_IMAGE_ATTR	401 /* image attribute	*/
#define ISE_METRIC_DESCRIPTION	402 /* metric description	*/
#define ISE_DIMENSION_MAPPING	403 /* dimension mapping	*/
#define ISE_MEASUREMENT_UNIT	404 /* measurement unit	*/
#define ISE_DELTA_VECTOR	405 /* delta vector	*/
#define ISE_METRIC_TRANSFORMATION	406 /* metric transformation	*/
#define ISE_DOMAIN	407 /* domain	*/
#define ISE_CHANNEL_CHARACTERISTICS	408 /* channel characteristics	*/
#define ISE_COMPANDOR_DESCRIPTION	409 /* compandor description	*/
#define ISE_COLOUR_REPRESENTATION	410 /* colour representation	*/
#define ISE_STANDARDIZED_SPACE	411 /* standardized space	*/
#define ISE_CIE_XYZ_SPACE	412 /* CIE XYZ space	*/
#define ISE_CIE_YXY_SPACE	413 /* CIE Yxy space	*/
#define ISE_CIE_UVW_SPACE	414 /* CIE UVW space	*/
#define ISE_CIE_YUV_SPACE	415 /* CIE Yuv space	*/
#define ISE_CIE_LAB_SPACE	416 /* CIE Lab space	*/
#define ISE_CIE_LUV_SPACE	417 /* CIE Luv space	*/
#define ISE_CIE_XYZ_COORDINATE	418 /* CIE XYZ coordinate	*/
#define ISE_LIN_RGB_SPACE	419 /* linear RGB space	*/
#define ISE_GAMMA_RGB_SPACE	420 /* gamma RGB space	*/
#define ISE_YIQ_COLOUR_SPACE	421 /* YIQ colour space	*/
#define ISE_YUV_COLOUR_SPACE	422 /* YUV colour space	*/
#define ISE_YCBCR_COLOUR_SPACE	423 /* YCbCr colour space	*/
#define ISE_NON_STANDARDIZED_SPACE	424 /* non-standardized space	*/
#define ISE_NON_STANDARDIZED_RGB	425 /* non-standardized RGB	*/
#define ISE_NON_STANDARDIZED_IHS	426 /* non-standardized IHS	*/
#define ISE_PRIMARIES	427 /* primaries	*/
#define ISE_CIE_XY_COORDINATE	428 /* CIE xy coordinate	*/
#define ISE_NON_STANDARDIZED_CMY	429 /* non-standardized CMY	*/
#define ISE_NON_STANDARDIZED_CMYK	430 /* non-standardized CMYK	*/
#define ISE_NON_STANDARDIZED_N_BAND	431 /* non-standardized N band	*/
#define ISE_COLOUR_BAND	432 /* colour band	*/
#define ISE_TEST_COLOUR	433 /* test colour	*/
#define ISE_PIKS_CONTROL	434 /* PIKS control	*/

ENTITIES FOR THE DESCRIPTION OF IMAGE ANNOTATIONS

#define ISE_IMAGE_ANNOTATION	501 /* image annotation	*/
#define ISE_LOCATION	502 /* location	*/

ENTITIES FOR THE DESCRIPTION OF BASIC DATA OBJECTS

#define ISE_BASIC_DATA_OBJECT	601 /* basic data object	*/
#define ISE_BASIC_DATA_TYPE	602 /* basic data type	*/
#define ISE_COMPOUND_DATA_TYPE	603 /* compound data type	*/
#define ISE_BASIC_ARRAY	604 /* basic array	*/
#define ISE_BASIC_RECORD	605 /* basic record	*/
#define ISE_BASIC_RECORD_CMP	606 /* basic record component	*/
#define ISE_BASIC_LIST	607 /* basic list	*/
#define ISE_BASIC_SET	608 /* basic set	*/
#define ISE_ELEMENTARY_DATA_TYPE	609 /* elementary data type	*/

ENTITIES FOR DATA TYPES

#define ISE_ANY	700 /* Any	*/
#define ISE_BOOL	701 /* Boolean	*/
#define ISE_INTEGER	702 /* integer	*/
#define ISE_BIT_STRING	703 /* bit string	*/
#define ISE_OCTET_STRING	704 /* octet string	*/
#define ISE_OBJECT_ID	706 /* object identifier	*/
#define ISE_REAL	709 /* real	*/
#define ISE_IA5STRING	722 /* ia5string	*/
#define ISE_GENERALIZED_TIME	724 /* generalized time	*/

NOTE - The needed entity "Any" has been added.

ENTITIES OTHER

#define ISE_PIXEL_FIELD_ENCODING	801 /* pixel field encoding	*/
#define ISE_DOR	802 /* designated object reference	*/

6.2.2 IPI-IIF syntax components

The IPI-IIF syntax component descriptors are represented by three to five digit macro values. The least significant two digits represent the component number. The component number is preceded by the syntax entity number. The syntax entity component names are bound with the prefix ISC_.

FULL DATA FORMAT

#define ISC_FORMAT_DESCRIPTOR	100 /* format descriptor	*/
#define ISC_CONTENTS_HEADER	101 /* contents header	*/
#define ISC_CONTENTS	102 /* contents	*/

FORMAT DESCRIPTOR

#define ISC_SELF_IDENTIFICATION	200 /* self identification	*/
#define ISC_VERSION	201 /* version	*/
#define ISC_PROFILE	202 /* profile	*/

VERSION

#define ISC_STANDARD	300 /* standard	*/
#define ISC_PUBLICATION_DATE	301 /* publication date	*/

CONTENTS HEADER

#define ISC_TITLE	500 /* title	*/
#define ISC_OWNER	501 /* owner	*/
#define ISC_DATE_AND_TIME	502 /* date and time	*/
#define ISC_MESSAGE	503 /* message	*/
#define ISC_APPL_DATA	504 /* application data	*/

CHARACTER STRING

#define ISC_STANDARD_CHARS	600 /* standard characters	*/
#define ISC_SPECIAL_CHARS	601 /* special characters	*/

SPECIAL CHARACTER STRING

#define ISC_CHAR_SET_ESCAPE	700 /* character set escape	*/
#define ISC_CHARS	701 /* characters	*/

CONTENTS ELEMENT

#define ISC_PROLOG	900 /* prolog	*/
#define ISC_BODY	901 /* body	*/
#define ISC_EPILOG	902 /* epilog	*/

CONTENTS BODY

#define ISC_IMAGE	1000 /* image	*/
#define ISC_IMAGE_RELATED_DATA	1001 /* image-related data	*/
#define ISC_IMAGE_ATTR	1002 /* image attribute	*/
#define ISC_IMAGE_ANNOTATION	1003 /* image annotation	*/
#define ISC_BASIC_DATA_OBJECT	1004 /* basic data object	*/

IMAGE

#define ISC_IMAGE_STRUCTURE	10100 /* image structure	*/
#define ISC_IMAGE_DATA	10101 /* image data	*/

IMAGE STRUCTURE

#define ISC_STRUCTURE_TYPE	10200 /* structure type	*/
#define ISC_COMPOUND_STRUCTURE	10201 /* compound structure	*/
#define ISC_FUND_STRUCTURE	10202 /* fundamental structure	*/
#define ISC_IMAGE_ATTRS	10203 /* image attributes	*/
	/* image-related data, refer 1001	*/
#define ISC_IMAGE_ANNOTATIONS	10205 /* image annotations	*/
#define ISC_PROCESSING_HISTORY	10206 /* processing history	*/

COMPOUND IMAGE STRUCTURE

#define ISC_COMPOUND_IMAGE_ARRAY	10300 /* compound image array	*/
#define ISC_COMPOUND_IMAGE_RECORD	10301 /* compound image record	*/
#define ISC_COMPOUND_IMAGE_LIST	10302 /* compound image list	*/
#define ISC_COMPOUND_IMAGE_SET	10303 /* compound image set	*/

COMPOUND IMAGE ARRAY

#define ISC_DIMENSIONALITY	10400 /* dimensionality	*/
#define ISC_SERIALIZATION	10401 /* serialization	*/
#define ISC_DATA_PLACEMENT	10402 /* data placement	*/
#define ISC_ELEMENT_STRUCTURE	10403 /* element structure	*/

DIMENSIONALITY

#define ISC_NUMBER_OF_DIMENSIONS	10500 /* number of dimensions	*/
#define ISC_DIMENSION_DESCRIPTIONS	10501 /* dimension descriptions	*/

DIMENSION DESCRIPTION

#define ISC_DIMENSION_ID	10600 /* dimension identifier	*/
#define ISC_LOWER_BOUNDARY	10601 /* lower boundary	*/
#define ISC_UPPER_BOUNDARY	10602 /* upper boundary	*/

COMPOUND IMAGE RECORD

#define ISC_NUMBER_OF_COMPONENTS	11000 /* number of components	*/
	/* data placement, refer 10402	*/
#define ISC_RECORD_CMPS	11002 /* record components	*/

RECORD COMPONENTS

#define ISC_CMP_ID	11100 /* component identifier	*/
#define ISC_CMP_STRUCTURE	11101 /* component structure	*/

COMPOUND IMAGE LIST

#define ISC_NUMBER_OF_ELEMENTS	11200 /* number of elements	*/
	/* data placement, refer 10402	*/
	/* element structure, refer 10403	*/

COMPOUND IMAGE SET

#define ISC_NUMBER_OF_MEMBERS	11300 /* number of members	*/
	/* data placement, refer 10402	*/
#define ISC_MEMBER_STRUCTURE	11302 /* member structure	*/

FUNDAMENTAL IMAGE STRUCTURE

#define ISC_BAND_RECORD	11400 /* band record	*/
#define ISC_METRIC_ARRAY	11401 /* metric array	*/

BAND RECORD

#define ISC_NUMBER_OF_BANDS	11500	/* number of bands	*/
		/* data placement, refer 10402	*/
		/* record components, refer 11002	*/

BAND RECORD COMPONENT

#define ISC_BAND_ID	11600	/* band identifier	*/
		/* component structure, refer 11101	*/

METRIC ARRAY

/* dimensionality, refer 10400	*/
/* serialization, refer 10401	*/
/* data placement, refer 10402	*/
/* element structure, refer 10403	*/

METRIC ARRAY ELEMENT

#define ISC_PIXEL_STRUCTURE	11800	/* pixel structure	*/
		/* band record, refer 11400	*/
		/* metric array, refer 11401	*/

PIXEL STRUCTURE

#define ISC_ELEMENTARY_PIXEL_STRUCTURE	11900	/* elementary pixel structure	*/
#define ISC_COMPOUND_PIXEL_STRUCTURE	11901	/* compound pixel structure	*/

PIXEL BAND RECORD

/* number of components, refer 11000	*/
/* record components, refer 11002	*/

PIXEL BAND RECORD COMPONENT

/* band identifier, refer 11600	*/
/* component structure, refer 11101	*/

REFERENCED UNIT

#define ISC_REFERENCE_LABEL	20100 /* reference label	*/
#define ISC_PIXEL_VALUES	20101 /* pixel values	*/

DATA UNIT

#define ISC_SINGLE_DATA_UNIT	20200 /* single data unit	*/
#define ISC_SUBDIVIDED_DATA_UNIT	20201 /* subdivided data unit	*/
#define ISC_EXTERNAL_DATA_UNIT	20202 /* external data unit	*/

SUBDIVIDED DATA UNIT

#define ISC_NUMBER_OF_PIXELS	20300 /* number of pixels	*/
#define ISC_PARTITIONS	20301 /* partitions	*/

SINGLE DATA UNIT

#define ISC_BUILTIN_ENCODED_DATA_UNIT	20400 /* builtin encoded data unit	*/
#define ISC_EXTERNALLY_DEF_DATA_UNIT	20401 /* externally defined data unit	*/
#define ISC_COMPRESSED_DATA_UNIT	20402 /* compressed data unit	*/
#define ISC_REGISTERED_DATA_UNIT	20403 /* registered data unit	*/

BUILTIN ENCODED DATA UNIT

#define ISC_SEQ_OF_BOOL	20500 /* sequence of Boolean	*/
#define ISC_SEQ_OF_OTHERS	20501 /* sequence of others	*/

BUILTIN VALUES

#define ISC_BOOL	20600 /* Boolean	*/
#define ISC_NON_NEGATIVE_INTEGER	20601 /* non-negative integer	*/
#define ISC_SIGNED_INTEGER	20602 /* signed integer	*/
#define ISC_REAL	20603 /* real	*/
#define ISC_COMPLEX	20604 /* complex	*/
#define ISC_ENUMERATED	20605 /* enumerated	*/

COMPLEX VALUE

	/* real, refer 20603	*/
#define ISC_IMAGINARY	20701 /* imaginary	*/

COMPRESSED DATA UNIT

#define ISC_DATA_REPRESENTATION	20900 /* data representation	*/
#define ISC_DATA	20901 /* data	*/

REGISTERED DATA UNIT

#define ISC_REGISTRATION_ID	21000 /* registration identification	*/
	/* pixel values, refer 20101	*/

EXTERNAL REFERENCE

#define ISC_OBJECT_ADDRESS	21100 /* object address	*/
#define ISC_OBJECT_FORMAT	21101 /* object format	*/
#define ISC_OBJECT_INTERNAL_ID	21102 /* object internal identification	*/

EXTERNAL ADDRESS

#define ISC_STRUCTURED_ADDRESS	21200 /* structured address	*/
#define ISC_CLEARTEXT_ADDRESS	21201 /* cleartext address	*/

IMAGE-RELATED DATA

#define ISC_ID	30100 /* identifier	*/
#define ISC_USAGE	30101 /* usage	*/
#define ISC_DATA_TYPE	30102 /* data type	*/
#define ISC_HIST	30103 /* histogram	*/
#define ISC_LUT	30104 /* lookup table	*/
#define ISC_ROI	30105 /* region of interest	*/
#define ISC_NBHOOD_ARRAY	30106 /* neighbourhood array	*/
#define ISC_STATIC_ARRAY	30107 /* static array	*/
#define ISC_FEATURE_LIST	30108 /* feature list	*/
#define ISC_VALUE_BOUNDS_COLLECTION	30109 /* value bounds collection	*/
#define ISC_MATRIX	30110 /* matrix	*/
#define ISC_PIXEL_RECORD	30111 /* pixel record	*/
#define ISC_TUPLE	30112 /* tuple	*/

HISTOGRAM

#define ISC_NUMBER_OF_CLASSES	30200 /* number of classes	*/
#define ISC_CLASS_DESCRIPTION	30201 /* class description	*/
#define ISC_CLASSES_AND_COUNTS	30202 /* classes and counts	*/

PARTITION CLASS

		/* lower boundary, refer 10601	*/
		/* upper boundary, refer 10602	*/
#define ISC_COUNT	30302	/* count	*/

LOOKUP TABLE

#define ISC_NUMBER_OF_INPUT_DIMENSIONS	30400	/* number of input dimensions	*/
#define ISC_INPUT_DATA_TYPES	30401	/* input data types	*/
#define ISC_NUMBER_OF_OUTPUT_DIMENSIONS	30402	/* number of output dimensions	*/
#define ISC_OUTPUT_DATA_TYPES	30403	/* output data types	*/
#define ISC_NUMBER_OF_ENTRIES	30404	/* number of entries	*/
#define ISC_INTERP_METHOD	30405	/* interpolation method	*/
#define ISC_INPUT_VALUES	30406	/* input values	*/
#define ISC_OUTPUT_VALUES	30407	/* output values	*/

REGION OF INTEREST

#define ISC_ROI_TYPE	30500	/* region of interest type	*/
#define ISC_ARRAY	30501	/* array	*/
#define ISC_ELLIPSE	30502	/* ellipse	*/
#define ISC_RECTANGULAR	30503	/* rectangular	*/
#define ISC_POLYGON	30504	/* polygon	*/
#define ISC_SET_OF_COORDINATES	30505	/* set of coordinates	*/
#define ISC_POLARITY_REVERSED	30506	/* polarity reversed	*/
#define ISC_INDEX_MANIPULATION	30507	/* index manipulation	*/

BOOLEAN ARRAY

		/* dimensionality, refer 10400	*/
		/* serialization, refer 10401	*/
#define ISC_BOOL_VALUES	30602	/* Boolean values	*/

ELLIPSE

#define ISC_ROI_DIMENSIONALITY	30700	/* region of interest dimensionality	*/
#define ISC_ROI_SIZE	30701	/* region of interest size	*/
#define ISC_ELLIPSE_CENTRE	30702	/* ellipse centre	*/
#define ISC_ELLIPSE_AXIS_LENGTH	30703	/* ellipse axis length	*/
#define ISC_ELLIPSE_DIMENSIONALITY	30704	/* ellipse dimensionality	*/
#define ISC_ELLIPTICAL_DIMENSIONS	30705	/* elliptical dimensions	*/

INTERVAL N-DIMENSIONAL

	/* number of dimensions, refer 10500	*/
#define ISC_INTERVALS	30801 /* intervals	*/

INTERVAL 1-DIMENSIONAL

	/* lower boundary, refer 10601	*/
	/* upper boundary, refer 10602	*/

COORDINATE N-DIMENSIONAL

	/* number of dimensions, refer 10500	*/
#define ISC_VECTOR_TYPE	31001 /* vector type	*/
#define ISC_INTEGER_VECTOR	31002 /* integer vector	*/
#define ISC_REAL_VECTOR	31003 /* real vector	*/

SET OF COORDINATES

#define ISC_NUMBER_OF_COORDINATES	31100 /* number of coordinates	*/
#define ISC_PIXEL_COORDINATES	31101 /* pixel coordinates	*/

NEIGHBOURHOOD ARRAY

	/* dimensionality, refer 10400	*/
	/* serialization, refer 10401	*/
	/* element structure, refer 10403	*/
#define ISC_ARRAY_ELEMENTS	31203 /* array elements	*/
#define ISC_SEMANTIC_LABEL	31204 /* semantic label	*/
#define ISC_KEY_PIXEL	31205 /* key pixel	*/
#define ISC_SCALE_FACTOR	31206 /* scale factor	*/

INDEX N-DIMENSIONAL

	/* number of dimensions, refer 31100	*/
#define ISC_INDEX_VECTOR	31301 /* index vector	*/

STATIC ARRAY

/* dimensionality, refer 10400 */
/* serialization, refer 10401 */
/* element structure, refer 10403 */
/* array elements, refer 31203 */
/* semantic label, refer 31204 */

FEATURE LIST

#define ISC_COORDINATES_AND_FEATURES 31501 /* number of coordinates, refer 31100 */
/* coordinates and features */

COORDINATE AND FEATURE

#define ISC_COORDINATE 31600 /* coordinate */
#define ISC_FEATURE 31601 /* feature */

VALUE BOUNDS COLLECTION

#define ISC_PIXEL_DATA_TYPE 31700 /* pixel data type */
/* lower boundary, refer 10601 */
/* upper boundary, refer 10602 */
/* number of coordinates, refer 31100 */
#define ISC_COORDINATE_LIST 31704 /* coordinate list */

TRANSFORMATION MATRIX

#define ISC_MATRIX_SIZE 31800 /* matrix size */
#define ISC_MATRIX_ELEMENTS 31801 /* matrix elements */
#define ISC_INTEGERS 31802 /* integers */
#define ISC_REALS 31803 /* reals */

PIXEL RECORD

/* number of components, refer 11000 */
/* record components, refer 11002 */

PIXEL RECORD COMPONENT

#define ISC_CMP_VALUE 32001 /* identifier, refer 30100 */
/* component value */

TUPLE

	/* number of elements, refer 11200	*/
	/* element structure, refer 10403	*/
#define ISC_VALUES	32102 /* values	*/

IMAGE ATTRIBUTE

#define ISC_METRIC	40100 /* metric	*/
#define ISC_CHANNEL	40101 /* channel	*/
#define ISC_COLOUR	40102 /* colour	*/
#define ISC_CONTROL	40103 /* control	*/
#define ISC_FREEFORM	40104 /* freeform	*/

METRIC DESCRIPTION

#define ISC_COORDINATE_SYSTEM	40200 /* coordinate system	*/
#define ISC_DIMENSION_MAPPINGS	40201 /* dimension mappings	*/
#define ISC_METRIC_TRANSFORM	40202 /* metric transform	*/
#define ISC_DOMAIN	40203 /* domain	*/

DIMENSION MAPPING

	/* dimension identifier, refer 10600	*/
#define ISC_COORDINATE_AXIS	40301 /* coordinate axis	*/
#define ISC_PHYSICAL_MEASUREMENT	40302 /* physical measurement	*/
#define ISC_ORIGIN	40303 /* origin	*/
#define ISC_DELTA_TYPE	40304 /* delta type	*/
#define ISC_SINGLE_DELTA	40305 /* single delta	*/
#define ISC_MULTIPLE_DELTAS	40306 /* multiple deltas	*/
#define ISC_PRECISION	40307 /* precision	*/

MEASUREMENT UNIT

#define ISC_MEASUREMENT	40400 /* measurement	*/
#define ISC_BASIS	40401 /* basis	*/
#define ISC_VERBAL_DESCRIPTION	40402 /* verbal description	*/

METRIC TRANSFORMATION

#define ISC_STANDARD_TRANSFORMATION	40600 /* standard transformation	*/
	/* matrix, refer 30110	*/

DOMAIN

#define ISC_DOMAIN_TYPE	40700 /* domain type	*/
	/* verbal description, refer 40402	*/

CHANNEL CHARACTERISTICS

	/* band identifier, refer 11600	*/
	/* verbal description, refer 40402	*/
	/* precision, refer 40307	*/
#define ISC_CHANNEL_TYPE	40803 /* channel type	*/
#define ISC_CHANNEL_USAGE	40804 /* channel usage	*/
#define ISC_BACKGROUND_VALUE	40805 /* background value	*/
#define ISC_COMPANDOR_DESCRIPTION	40806 /* compandor description	*/
#define ISC_SAMPLING_FUNCTION	40807 /* sampling function	*/
#define ISC_APPL_SPECIFICS	40808 /* application specifics	*/

COMPANDOR DESCRIPTION

#define ISC_FUNCTION_TYPE	40900 /* function type	*/
#define ISC_TRANSFER_FACTOR	40901 /* transfer factor	*/
#define ISC_FUNCTION_PARAMETER	40902 /* function parameter	*/
#define ISC_EXPANDER_FUNCTION	40903 /* expander function	*/
#define ISC_APPL_SPECIFIC	40904 /* application specific	*/

COLOUR REPRESENTATION

#define ISC_COLOUR_SPACE_TYPE	41000 /* colour space type	*/
#define ISC_STANDARDIZED_SPACE	41001 /* standardized space	*/
#define ISC_NON_STANDARDIZED_SPACE	41002 /* non-standardized space	*/
#define ISC_TEST_COLOUR	41003 /* test colour	*/

STANDARDIZED SPACE

#define ISC_CIE_XYZ	41100 /* CIE XYZ space	*/
#define ISC_CIE_YXY	41101 /* CIE Yxy space	*/
#define ISC_CIE_UVW	41102 /* CIE UVW space	*/
#define ISC_CIE_YUV	41103 /* CIE Yuv space	*/
#define ISC_CIE_LAB	41104 /* CIE Lab space	*/
#define ISC_CIE_LUV	41105 /* CIE Luv space	*/
#define ISC_LIN_RGB	41106 /* linear RGB space	*/
#define ISC_GAMMA_RGB	41107 /* gamma RGB space	*/
#define ISC_YIQ_COLOUR_SPACE	41108 /* YIQ colour space	*/
#define ISC_YUV_COLOUR_SPACE	41109 /* YUV colour space	*/
#define ISC_YCBCR_COLOUR_SPACE	41110 /* YCbCr colour space	*/

CIE XYZ SPACE

#define ISC_X_BAND_ID	41200 /* X band identifier	*/
#define ISC_Y_BAND_ID	41201 /* Y band identifier	*/
#define ISC_Z_BAND_ID	41202 /* Z band identifier	*/

CIE Yxy SPACE

#define ISC_YL_BAND_ID	41300 /* Yl band identifier	*/
#define ISC_XC_BAND_ID	41301 /* xc band identifier	*/
#define ISC_YC_BAND_ID	41302 /* yc band identifier	*/

CIE UVW SPACE

#define ISC_U_BAND_ID	41400 /* U band identifier	*/
#define ISC_V_BAND_ID	41401 /* V band identifier	*/
#define ISC_W_BAND_ID	41402 /* W band identifier	*/

CIE Yuv SPACE

/* Y band identifier, refer 41201	*/
/* u band identifier, refer 41400	*/
/* v band identifier, refer 41401	*/

CIE Lab SPACE

#define ISC_L_BAND_ID	41600 /* L band identifier	*/
#define ISC_A_BAND_ID	41601 /* a band identifier	*/
#define ISC_B_BAND_ID	41602 /* b band identifier	*/
#define ISC_WHITE_POINT	41603 /* white point	*/

CIE Luv SPACE

/* L band identifier, refer 41600	*/
/* u band identifier, refer 41400	*/
/* v band identifier, refer 41401	*/
/* white point, refer 41603	*/

CIE XYZ COORDINATE

#define ISC_CIE_X	41800 /* CIE X	*/
#define ISC_CIE_Y	41801 /* CIE Y	*/
#define ISC_CIE_Z	41802 /* CIE Z	*/

LINEAR RGB SPACE

#define ISC_REPRESENTATION	41900 /* representation	*/
#define ISC_ILLUMINANT	41901 /* illuminant	*/
#define ISC_R_BAND_ID	41902 /* R band identifier	*/
#define ISC_G_BAND_ID	41903 /* G band identifier	*/
	/* B band identifier, refer 41602	*/

GAMMA RGB SPACE

/* representation, refer 41900	*/
/* illuminant, refer 41901	*/
/* R band identifier, refer 41902	*/
/* G band identifier, refer 41903	*/
/* B band identifier, refer 41602	*/

YIQ COLOUR SPACE

	/* Y band identifier, refer 41201	*/
#define ISC_I_BAND_ID	42101 /* I band identifier	*/
#define ISC_Q_BAND_ID	42102 /* Q band identifier	*/

YUV COLOUR SPACE

/* Y band identifier, refer 41201	*/
/* U band identifier, refer 41400	*/
/* V band identifier, refer 41401	*/

YCBCR COLOUR SPACE

	/* Y band identifier, refer 41201	*/
#define ISC_CB_BAND_ID	42301 /* Cb band identifier	*/
#define ISC_CR_BAND_ID	42302 /* Cr band identifier	*/

NON-STANDARDIZED SPACE

#define ISC_RGB	42400 /* RGB	*/
#define ISC_IHS	42401 /* IHS	*/
#define ISC_CMY	42402 /* CMY	*/
#define ISC_CMYK	42403 /* CMYK	*/
#define ISC_N_BAND	42404 /* n band	*/

NON-STANDARDIZED RGB

```

/* R band identifier, refer 41902 */
/* G band identifier, refer 41903 */
/* B band identifier, refer 41602 */
/* white point, refer 41603 */
#define ISC_PRIMARIES 42504 /* primaries */

```

NON-STANDARDIZED IHS

```

/* I band identifier, refer 42101 */
#define ISC_H_BAND_ID 42601 /* H band identifier */
#define ISC_S_BAND_ID 42602 /* S band identifier */
/* white point, refer 41603 */
/* primaries, refer 42504 */

```

PRIMARIES

```

#define ISC_R_PRIMARY 42700 /* R primary */
#define ISC_G_PRIMARY 42701 /* G primary */
#define ISC_B_PRIMARY 42702 /* B primary */

```

CIE xy COORDINATE

```

/* CIE x, refer 41800 */
/* CIE y, refer 41801 */

```

NON-STANDARDIZED CMY

```

#define ISC_C_BAND_ID 42900 /* C band identifier */
#define ISC_M_BAND_ID 42901 /* M band identifier */
/* Y band identifier, refer 41201 */
/* verbal description, refer 40402 */

```

NON-STANDARDIZED CMYK

```

/* C band identifier, refer 42900 */
/* M band identifier, refer 42901 */
/* Y band identifier, refer 41201 */
#define ISC_K_BAND_ID 43003 /* K band identifier */
/* verbal description, refer 40402 */

```

NON-STANDARDIZED N BAND

```
/* number of bands, refer 11500 */
#define ISC_BAND_DESCRIPTIONS 43101 /* band descriptions */
```

COLOUR BAND

```
/* band identifier, refer 11600 */
/* verbal description, refer 40402 */
```

TEST COLOUR

```
/* colour, refer 40102 */
#define ISC_REGION 43301 /* region */
```

PIKS CONTROL

```
/* region of interest, refer 30105 */
#define ISC_ROI_OFFSET 43401 /* region of interest offset */
#define ISC_MATCH_POINT 43402 /* match point */
```

NOTE - The region of interest component is not redefined.

IMAGE ANNOTATION

```
#define ISC_LOCATION 50100 /* location */
#define ISC_CONTENTS_DESCRIPTOR 50101 /* contents descriptor */
#define ISC_NUMERIC_ID 50102 /* numeric identifier */
#define ISC_OBJECT_ID 50103 /* object identifier */
#define ISC_CONTENTS_TYPE 50104 /* contents type */
/* contents, refer 00102 */
#define ISC_REFERENCE_TO_IMAGE_DATA 50106 /* reference to image data */
```

LOCATION

```
#define ISC_POINT 50200 /* point */
/* region, refer 43301 */
```

BASIC DATA OBJECT

```
#define ISC_BASIC_DATA_TYPE 60100 /* basic data type */
/* data, refer 20901 */
```

NOTE - The data component is not redefined.

BASIC DATA TYPE

#define ISC_COMPOUND_DATA_TYPE	60200 /* compound data type	*/
#define ISC_ELEMENTARY_DATA_TYPE	60201 /* elementary data type	*/

COMPOUND DATA TYPE

#define ISC_BASIC_ARRAY	60300 /* basic array	*/
#define ISC_BASIC_RECORD	60301 /* basic record	*/
#define ISC_BASIC_LIST	60302 /* basic list	*/
#define ISC_BASIC_SET	60303 /* basic set	*/

BASIC ARRAY

	/* dimensionality, refer 10400	*/
	/* serialization, refer 10401	*/
#define ISC_ELEMENT_TYPE	60402 /* element type	*/

BASIC RECORD

	/* number of components, refer 11000	*/
#define ISC_BASIC_RECORD_CMPS	60501 /* basic record components	*/

BASIC RECORD COMPONENT

	/* component identifier, refer 11100	*/
#define ISC_CMP_DATA_TYPE	60601 /* component data type	*/

BASIC LIST

	/* number of elements, refer 11200	*/
	/* element type, refer 60402	*/

BASIC SET

	/* number of members, refer 11300	*/
#define ISC_MEMBER_TYPE	60801 /* member type	*/

6.3 Function descriptors

The error functions require a unique mapping of the functions to a set of numbers. In IPI this is accomplished separately in IPI-IIF and IPI-PIKS APIs. Some redundancy of identification appears in ISO/IEC 12087; however, this does not prevent exclusive identification since:

- Individual IPI-IIF and IPI-PIKS inquiry functions are specified for interrogating the functional support provided by separate and combined implementations of IPI-IIF and IPI-PIKS.
- When present in error logs, the error codes are unique for separate or combined implementations of IPI-IIF and IPI-PIKS.

Numeric values assigned to functions presented in the functional specifications of IPI-PIKS and IPI-IIF range up to 999 and their macro names are function names with IF_ as prefix. Numeric values assigned to convenience functions presented in clause 8 range upward from 1000 and their macro names are function names with IC_ as prefix. Some macro names contain abbreviations or have missing words relative to the function names to assure the maximum name length of 31 characters is not exceeded.

6.3.1 IPI-PIKS function descriptors

The IPI-PIKS function descriptor numbers correspond to those defined in ISO/IEC 12087-2. The function, *set error handler*, which is added in this part of ISO/IEC 12088 is assigned the next available sequential function number, 269.

The handling of parameters with virtual registers has been supported with alternate functions added to this part of ISO/IEC 12088. The descriptor numbers for these functions follow from the next available sequential function number, 270.

The function descriptor numbers are presented in Table 6 in alphabetical order by defined function name which is the same as the alphabetical order of function names presented in ISO/IEC 12087-2.

Table 6 - IPI-PIKS function descriptors

Function Descriptor Name	Code	Function Name
#define IF_ABORT_ASYNC_EXECUTION	1	/* abort_asynchronous_execution */
#define IF_ACCUMULATOR	2	/* accumulator */
#define IF_ADAPTIVE_HIST_EQUAL	3	/* adaptive_histogram_equalization */
#define IF_ADAPTIVE_HIST_EQUAL_VREG	270	/* adaptive_histogram_equalization_vreg */
#define IF_ALLOCATE_CHAIN	4	/* allocate_chain */
#define IF_ALLOCATE_CHAIN_VREG	271	/* allocate_chain_vreg */
#define IF_ALLOCATE_ID_ARRAY	5	/* allocate_composite_identifier_array */
#define IF_ALLOCATE_ID_ARRAY_VREG	272	/* allocate_composite_identifier_array_vreg */
#define IF_ALLOCATE_ID_LIST	6	/* allocate_composite_identifier_list */
#define IF_ALLOCATE_ID_LIST_VREG	273	/* allocate_composite_identifier_list_vreg */
#define IF_ALLOCATE_ID_RECORD	7	/* allocate_composite_identifier_record */
#define IF_ALLOCATE_ID_RECORD_VREG	274	/* allocate_composite_identifier_record_vreg */
#define IF_ALLOCATE_HIST	8	/* allocate_histogram */
#define IF_ALLOCATE_HIST_VREG	275	/* allocate_histogram_vreg */
#define IF_ALLOCATE_IMAGE	9	/* allocate_image */
#define IF_ALLOCATE_IMAGE_VREG	276	/* allocate_image_vreg */
#define IF_ALLOCATE_LUT	10	/* allocate_lookup_table */

Table 6 - IPI-PIKS function descriptors (continued)

Function Descriptor Name	Code	Function Name
#define IF_ALLOCATE_LUT_VREG	277	/* allocate_lookup_table_vreg *
#define IF_ALLOCATE_MATRIX	11	/* allocate_matrix *
#define IF_ALLOCATE_MATRIX_VREG	278	/* allocate_matrix_vreg *
#define IF_ALLOCATE_NBHOOD_ARRAY	12	/* allocate_neighbourhood_array *
#define IF_ALLOCATE_NBHOOD_ARRAY_VREG	279	/* allocate_neighbourhood_array_vreg *
#define IF_ALLOCATE_PIXEL_RECORD	13	/* allocate_pixel_record *
#define IF_ALLOCATE_PIXEL_RECORD_VREG	280	/* allocate_pixel_record_vreg *
#define IF_ALLOCATE_ROI	14	/* allocate_roi *
#define IF_ALLOCATE_ROI_VREG	281	/* allocate_roi_vreg *
#define IF_ALLOCATE_STATIC_ARRAY	15	/* allocate_static_array *
#define IF_ALLOCATE_STATIC_ARRAY_VREG	282	/* allocate_static_array_vreg *
#define IF_ALLOCATE_TUPLE	16	/* allocate_tuple *
#define IF_ALLOCATE_TUPLE_VREG	283	/* allocate_tuple_vreg *
#define IF_ALLOCATE_VALUE_BOUNDS	17	/* allocate_value_bounds *
#define IF_ALLOCATE_VALUE_BOUNDS_VREG	284	/* allocate_value_bounds_vreg *
#define IF_ALLOCATE_VREG	18	/* allocate_vreg *
#define IF_ALLOCATE_VREG_VREG	285	/* allocate_vreg_vreg *
#define IF_ALPHA_BLEND_CONSTANT	19	/* alpha_blend_constant *
#define IF_ALPHA_BLEND_CONSTANT_VREG	286	/* alpha_blend_constant_vreg *
#define IF_ALPHA_BLEND_VARIABLE	20	/* alpha_blend_variable *
#define IF_ALPHA_BLEND_VARIABLE_VREG	287	/* alpha_blend_variable_vreg *
#define IF_ARRAY_TO_LUT	21	/* array_to_lut *
#define IF_ARRAY_TO_LUT_VREG	288	/* array_to_lut_vreg *
#define IF_BIND_MATCH_POINT	22	/* bind_match_point *
#define IF_BIND_MATCH_POINT_VREG	289	/* bind_match_point_vreg *
#define IF_BIND_ROI	23	/* bind_roi *
#define IF_BIND_ROI_VREG	290	/* bind_roi_vreg *
#define IF_BIT_SHIFT	24	/* bit_shift *
#define IF_BIT_SHIFT_VREG	291	/* bit_shift_vreg *
#define IF_CARTESIAN_TO_POLAR	25	/* cartesian_to_polar *
#define IF_CARTESIAN_TO_POLAR_VREG	292	/* cartesian_to_polar_vreg *
#define IF_CHAIN_ABORT	26	/* chain_abort *
#define IF_CHAIN_ABORT_VREG	293	/* chain_abort_vreg *
#define IF_CHAIN_BEGIN	27	/* chain_begin *
#define IF_CHAIN_DELETE	28	/* chain_delete *
#define IF_CHAIN_DELETE_VREG	294	/* chain_delete_vreg *
#define IF_CHAIN_END	29	/* chain_end *
#define IF_CHAIN_EXECUTE	30	/* chain_execute *
#define IF_CHAIN_EXECUTE_VREG	295	/* chain_execute_vreg *
#define IF_CHAIN_RELOAD	31	/* chain_reload *
#define IF_CHAIN_RELOAD_VREG	296	/* chain_reload_vreg *
#define IF_CLASSIFIER_BAYES	32	/* classifier_bayes *
#define IF_CLASSIFIER_BAYES_VREG	297	/* classifier_bayes_vreg *
#define IF_CLASSIFIER_NEAREST_NBR	33	/* classifier_nearest_neighbour *
#define IF_CLASSIFIER_NEAREST_NBR_VREG	298	/* classifier_nearest_neighbour_vreg *
#define IF_CLOSE_PIKS	34	/* close_piks *
#define IF_CLOSE_PIKS_EMERGENCY	35	/* close_piks_emergency *

Table 6 - IPI-PIKS function descriptors (continued)

Function Descriptor Name	Code	Function Name
#define IF_COLOUR_CONV_LIN	36	/* colour_conversion_linear */
#define IF_COLOUR_CONV_LIN_VREG	299	/* colour_conversion_linear_vreg */
#define IF_COLOUR_CONV_MATRIX	37	/* colour_conversion_matrix */
#define IF_COLOUR_CONV_MATRIX_VREG	300	/* colour_conversion_matrix_vreg */
#define IF_COLOUR_CONV_NONLIN	38	/* colour_conversion_nonlinear */
#define IF_COLOUR_CONV_NONLIN_VREG	301	/* colour_conversion_nonlinear_vreg */
#define IF_COLOUR_CONV_SUBTRACTIVE	39	/* colour_conversion_subtractive */
#define IF_COLOUR_CONV_SUBTRACTIVE_VREG	302	/* colour_conversion_subtractive_vreg */
#define IF_COLOUR_LOOKUP_INTERP	40	/* colour_lookup_interpolated */
#define IF_COLOUR_LOOKUP_INTERP_VREG	303	/* colour_lookup_interpolated_vreg */
#define IF_COMPLEMENT	41	/* complement */
#define IF_COMPLEMENT_ROI	42	/* complement_roi */
#define IF_COMPLEMENT_ROI_VREG	304	/* complement_roi_vreg */
#define IF_COMPLEMENT_VREG	305	/* complement_vreg */
#define IF_COMPLEX_COMPOSITION	43	/* complex_composition */
#define IF_COMPLEX_COMPOSITION_VREG	306	/* complex_composition_vreg */
#define IF_COMPLEX_CONJUGATE	44	/* complex_conjugate */
#define IF_COMPLEX_CONJUGATE_VREG	307	/* complex_conjugate_vreg */
#define IF_COMPLEX_DECOMPOSITION	45	/* complex_decomposition */
#define IF_COMPLEX_DECOMPOSITION_VREG	308	/* complex_decomposition_vreg */
#define IF_COMPLEX_MAGNITUDE	46	/* complex_magnitude */
#define IF_COMPLEX_MAGNITUDE_VREG	309	/* complex_magnitude_vreg */
#define IF_ID_ARRAY_EQUAL	47	/* composite_identifier_array_equal */
#define IF_ID_ARRAY_EQUAL_VREG	310	/* composite_identifier_array_equal_vreg */
#define IF_ID_ARRAY_GET	48	/* composite_identifier_array_get */
#define IF_ID_ARRAY_GET_VREG	311	/* composite_identifier_array_get_vreg */
#define IF_ID_ARRAY_PUT	49	/* composite_identifier_array_put */
#define IF_ID_ARRAY_PUT_VREG	312	/* composite_identifier_array_put_vreg */
#define IF_ID_LIST_EMPTY	50	/* composite_identifier_list_empty */
#define IF_ID_LIST_EMPTY_VREG	313	/* composite_identifier_list_empty_vreg */
#define IF_ID_LIST_EQUAL	51	/* composite_identifier_list_equal */
#define IF_ID_LIST_EQUAL_VREG	314	/* composite_identifier_list_equal_vreg */
#define IF_ID_LIST_GET	52	/* composite_identifier_list_get */
#define IF_ID_LIST_GET_VREG	315	/* composite_identifier_list_get_vreg */
#define IF_ID_LIST_INSERT	53	/* composite_identifier_list_insert */
#define IF_ID_LIST_INSERT_VREG	316	/* composite_identifier_list_insert_vreg */
#define IF_ID_LIST_REMOVE	54	/* composite_identifier_list_remove */
#define IF_ID_LIST_REMOVE_VREG	317	/* composite_identifier_list_remove_vreg */
#define IF_ID_RECORD_EQUAL	55	/* composite_identifier_record_equal */
#define IF_ID_RECORD_EQUAL_VREG	318	/* composite_identifier_record_equal_vreg */
#define IF_ID_RECORD_GET	56	/* composite_identifier_record_get */
#define IF_ID_RECORD_GET_VREG	319	/* composite_identifier_record_get_vreg */
#define IF_ID_RECORD_PUT	57	/* composite_identifier_record_put */
#define IF_ID_RECORD_PUT_VREG	320	/* composite_identifier_record_put_vreg */
#define IF_CONSTANT_PREDICATE	58	/* constant_predicate */
#define IF_CONSTANT_PREDICATE_VREG	321	/* constant_predicate_vreg */
#define IF_CONVERT_ARRAY_TO_IMAGE	59	/* convert_array_to_image */

Table 6 - IPI-PIKS function descriptors (continued)

Function Descriptor Name	Code	Function Name
#define IF_CONVERT_ARRAY_TO_IMAGE_VREG	322	/* convert_array_to_image_vreg */
#define IF_CONVERT_IMAGE_DATATYPE	60	/* convert_image_datatype */
#define IF_CONVERT_IMAGE_DATATYPE_VREG	323	/* convert_image_datatype_vreg */
#define IF_CONVERT_IMAGE_TO_ARRAY	61	/* convert_image_to_array */
#define IF_CONVERT_IMAGE_TO_ARRAY_VREG	324	/* convert_image_to_array_vreg */
#define IF_CONVERT_IMAGE_TO_ROI	62	/* convert_image_to_roi */
#define IF_CONVERT_IMAGE_TO_ROI_VREG	325	/* convert_image_to_roi_vreg */
#define IF_CONVERT_ROI_TO_IMAGE	63	/* convert_roi_to_image */
#define IF_CONVERT_ROI_TO_IMAGE_VREG	326	/* convert_roi_to_image_vreg */
#define IF_CONVOLVE_2D	64	/* convolve_2d */
#define IF_CONVOLVE_2D_VREG	327	/* convolve_2d_vreg */
#define IF_CONVOLVE_5D	65	/* convolve_5d */
#define IF_CONVOLVE_5D_VREG	328	/* convolve_5d_vreg */
#define IF_COPY_WINDOW	66	/* copy_window */
#define IF_COPY_WINDOW_ROI	67	/* copy_window_roi */
#define IF_COPY_WINDOW_ROI_VREG	329	/* copy_window_roi_vreg */
#define IF_COPY_WINDOW_VREG	330	/* copy_window_vreg */
#define IF_CREATE_TUPLE	68	/* create_tuple */
#define IF_CREATE_TUPLE_VREG	331	/* create_tuple_vreg */
#define IF_CROSS_CORRELATION	69	/* cross_correlation */
#define IF_CROSS_CORRELATION_VREG	332	/* cross_correlation_vreg */
#define IF_DEALLOCATE_DATA_OBJECT	70	/* deallocate_data_object */
#define IF_DEALLOCATE_DATA_OBJECT_VREG	333	/* deallocate_data_object_vreg */
#define IF_DEFINE_SUB_IMAGE	71	/* define_sub_image */
#define IF_DEFINE_SUB_IMAGE_VREG	334	/* define_sub_image_vreg */
#define IF_DIFF_MEASURES	72	/* difference_measures */
#define IF_DIFF_MEASURES_VREG	335	/* difference_measures_vreg */
#define IF_DIFFUSE	73	/* diffuse */
#define IF_DIFFUSE_VREG	336	/* diffuse_vreg */
#define IF_DITHER	74	/* dither */
#define IF_DITHER_VREG	337	/* dither_vreg */
#define IF_DRAW_PIXELS	75	/* draw_pixels */
#define IF_DRAW_PIXELS_ROI	76	/* draw_pixels_roi */
#define IF_DRAW_PIXELS_ROI_VREG	338	/* draw_pixels_roi_vreg */
#define IF_DRAW_PIXELS_VREG	339	/* draw_pixels_vreg */
#define IF_DYADIC_ARITH	77	/* dyadic_arithmetic */
#define IF_DYADIC_ARITH_VREG	340	/* dyadic_arithmetic_vreg */
#define IF_DYADIC_COMPLEX	78	/* dyadic_complex */
#define IF_DYADIC_COMPLEX_VREG	341	/* dyadic_complex_vreg */
#define IF_DYADIC_LOGICAL	79	/* dyadic_logical */
#define IF_DYADIC_LOGICAL_ROI	80	/* dyadic_logical_roi */
#define IF_DYADIC_LOGICAL_ROI_VREG	342	/* dyadic_logical_roi_vreg */
#define IF_DYADIC_LOGICAL_VREG	343	/* dyadic_logical_vreg */
#define IF_DYADIC_PREDICATE	81	/* dyadic_predicate */
#define IF_DYADIC_PREDICATE_VREG	344	/* dyadic_predicate_vreg */
#define IF_EDGE_DETECT_ORTH_GRAD	82	/* edge_detection_orthogonal_gradient */
#define IF_EDGE_DETECT_ORTH_GRAD_VREG	345	/* edge_detection_orthogonal_gradient_vreg */

Table 6 - IPI-PIKS function descriptors (continued)

Function Descriptor Name	Code	Function Name
#define IF_EDGE_DETECT_2ND_DERIV	83	/* edge_detection_second_derivative */
#define IF_EDGE_DETECT_2ND_DERIV_VREG	346	/* edge_detection_second_derivative_vreg */
#define IF_EDGE_DETECT_TEMPL_GRAD	84	/* edge_detection_template_gradient */
#define IF_EDGE_DETECT_TEMPL_GRAD_VREG	347	/* edge_detection_template_gradient_vreg */
#define IF_EQUAL_PREDICATE	85	/* equal_predicate */
#define IF_EQUAL_PREDICATE_VREG	348	/* equal_predicate_vreg */
#define IF_EROSION_DILATE_BOOL	86	/* erosion_dilation_boolean */
#define IF_EROSION_DILATE_BOOL_ROI	87	/* erosion_dilation_boolean_roi */
#define IF_EROSION_DILATE_BOOL_ROI_VREG	349	/* erosion_dilation_boolean_roi_vreg */
#define IF_EROSION_DILATE_BOOL_VREG	350	/* erosion_dilation_boolean_vreg */
#define IF_EROSION_DILATE_GREY	88	/* erosion_dilation_grey */
#define IF_EROSION_DILATE_GREY_VREG	351	/* erosion_dilation_grey_vreg */
#define IF_ERROR_FUNCTION_SCALING	89	/* error_function_scaling */
#define IF_ERROR_FUNCTION_SCALING_VREG	352	/* error_function_scaling_vreg */
#define IF_ERROR_HANDLER	90	/* error_handler */
#define IF_ERROR_LOGGER	91	/* error_logger */
#define IF_ERROR_TEST	92	/* error_test */
#define IF_ERROR_TEST_VREG	353	/* error_test_vreg */
#define IF_EXPORT_HIST	93	/* export_histogram */
#define IF_EXPORT_IMAGE	94	/* export_image */
#define IF_EXPORT_LUT	95	/* export_lut */
#define IF_EXPORT_MATRIX	96	/* export_matrix */
#define IF_EXPORT_NBHOOD_ARRAY	97	/* export_neighbourhood_array */
#define IF_EXPORT_ROI_ARRAY	98	/* export_roi_array */
#define IF_EXPORT_STATIC_ARRAY	99	/* export_static_array */
#define IF_EXPORT_TUPLE	100	/* export_tuple */
#define IF_EXPORT_VALUE_BOUNDS	101	/* export_value_bounds */
#define IF_EXTRACT_PIXEL_PLANE	102	/* extract_pixel_plane */
#define IF_EXTRACT_PIXEL_PLANE_ROI	103	/* extract_pixel_plane_roi */
#define IF_EXTRACT_PIXEL_PLANE_ROI_VREG	354	/* extract_pixel_plane_roi_vreg */
#define IF_EXTRACT_PIXEL_PLANE_VREG	355	/* extract_pixel_plane_vreg */
#define IF_EXTREMA	104	/* extrema */
#define IF_FALSE_COLOUR	105	/* false_colour */
#define IF_FALSE_COLOUR_VREG	356	/* false_colour_vreg */
#define IF_FILL_REGION	106	/* fill_region */
#define IF_FILL_REGION_ROI	107	/* fill_region_roi */
#define IF_FILL_REGION_ROI_VREG	357	/* fill_region_roi_vreg */
#define IF_FILL_REGION_VREG	358	/* fill_region_vreg */
#define IF_FILTER_BUTTERWORTH	108	/* filter_butterworth */
#define IF_FILTER_BUTTERWORTH_VREG	359	/* filter_butterworth_vreg */
#define IF_FILTER_GAUSS	109	/* filter_gaussian */
#define IF_FILTER_GAUSS_VREG	360	/* filter_gaussian_vreg */
#define IF_FILTER_INVERSE	110	/* filter_inverse */
#define IF_FILTER_INVERSE_VREG	361	/* filter_inverse_vreg */
#define IF_FILTER_MATCHED	111	/* filter_matched */
#define IF_FILTER_MATCHED_VREG	362	/* filter_matched_vreg */
#define IF_FILTER_WIENER	112	/* filter_wiener */

Table 6 - IPI-PIKS function descriptors (continued)

Function Descriptor Name	Code	Function Name
#define IF_FILTER_WIENER_VREG	363	/* filter_wiener_vreg *
#define IF_FILTER_ZONAL	113	/* filter_zonal *
#define IF_FILTER_ZONAL_VREG	364	/* filter_zonal_vreg *
#define IF_FILTERING_HOMOMORPHIC	114	/* filtering_homomorphic *
#define IF_FILTERING_HOMOMORPHIC_VREG	365	/* filtering_homomorphic_vreg *
#define IF_FILTERING_LIN	115	/* filtering_linear *
#define IF_FILTERING_LIN_VREG	366	/* filtering_linear_vreg *
#define IF_FILTERING_MEDIAN	116	/* filtering_median *
#define IF_FILTERING_MEDIAN_VREG	367	/* filtering_median_vreg *
#define IF_FILTERING_PSEUDOMEDIAN	117	/* filtering_pseudomedian *
#define IF_FILTERING_PSEUDOMEDIAN_VREG	368	/* filtering_pseudomedian_vreg *
#define IF_FILTERING_RANK_ORDER	118	/* filtering_rank_order *
#define IF_FILTERING_RANK_ORDER_VREG	369	/* filtering_rank_order_vreg *
#define IF_FLIP_SPIN_TRANSPOSE	119	/* flip_spin_transpose *
#define IF_FLIP_SPIN_TRANSPOSE_ROI	120	/* flip_spin_transpose_roi *
#define IF_FLIP_SPIN_TRANSPOSE_ROI_VREG	370	/* flip_spin_transpose_roi_vreg *
#define IF_FLIP_SPIN_TRANSPOSE_VREG	371	/* flip_spin_transpose_vreg *
#define IF_GAMMA_CORRECTION	121	/* gamma_correction *
#define IF_GAMMA_CORRECTION_VREG	372	/* gamma_correction_vreg *
#define IF_GET_COLOUR_PIXEL	122	/* get_colour_pixel *
#define IF_GET_COLOUR_PIXEL_VREG	373	/* get_colour_pixel_vreg *
#define IF_GET_PIXEL	123	/* get_pixel *
#define IF_GET_PIXEL_ROI	124	/* get_pixel_roi *
#define IF_GET_PIXEL_ROI_VREG	374	/* get_pixel_roi_vreg *
#define IF_GET_PIXEL_VREG	375	/* get_pixel_vreg *
#define IF_GET_PIXEL_ARRAY	125	/* get_pixel_array *
#define IF_GET_PIXEL_ARRAY_ROI	126	/* get_pixel_array_roi *
#define IF_GET_PIXEL_ARRAY_ROI_VREG	376	/* get_pixel_array_roi_vreg *
#define IF_GET_PIXEL_ARRAY_VREG	377	/* get_pixel_array_vreg *
#define IF_GET_PIXEL_RECORD	127	/* get_pixel_record *
#define IF_GET_PIXEL_RECORD_ROI	128	/* get_pixel_record_roi *
#define IF_GET_PIXEL_RECORD_ROI_VREG	378	/* get_pixel_record_roi_vreg *
#define IF_GET_PIXEL_RECORD_VREG	379	/* get_pixel_record_vreg *
#define IF_HIST_MODIFICATION	129	/* histogram_modification *
#define IF_HIST_MODIFICATION_VREG	380	/* histogram_modification_vreg *
#define IF_HIST_SCALING	130	/* histogram_scaling *
#define IF_HIST_SCALING_VREG	381	/* histogram_scaling_vreg *
#define IF_HIST_SHAPE_1D	131	/* histogram_shape_1d *
#define IF_HIST_SHAPE_2D	132	/* histogram_shape_2d *
#define IF_HIST_1D	133	/* histogram_1d *
#define IF_HIST_1D_VREG	382	/* histogram_1d_vreg *
#define IF_HIST_2D	134	/* histogram_2d *
#define IF_HIST_2D_VREG	383	/* histogram_2d_vreg *
#define IF_HIT_MISS	135	/* hit_or_miss *
#define IF_HIT_MISS_ROI	136	/* hit_or_miss_roi *
#define IF_HIT_MISS_ROI_VREG	384	/* hit_or_miss_roi_vreg *
#define IF_HIT_MISS_VREG	385	/* hit_or_miss_vreg *

Table 6 - IPI-PIKS function descriptors (continued)

Function Descriptor Name	Code	Function Name
#define IF_HOUGH_TRANSFORM	137	/* hough_transform *
#define IF_HOUGH_TRANSFORM_VREG	386	/* hough_transform_vreg *
#define IF_IMAGE_BAR_CHART	138	/* image_bar_chart *
#define IF_IMAGE_BAR_CHART_VREG	387	/* image_bar_chart_vreg *
#define IF_IMAGE_CONSTANT	139	/* image_constant *
#define IF_IMAGE_CONSTANT_VREG	388	/* image_constant_vreg *
#define IF_IMAGE_GAUSS	140	/* image_gaussian *
#define IF_IMAGE_GAUSS_VREG	389	/* image_gaussian_vreg *
#define IF_IMAGE_GREY_SCALE	141	/* image_grey_scale *
#define IF_IMAGE_GREY_SCALE_VREG	390	/* image_grey_scale_vreg *
#define IF_IMAGE_RANDOM_NUMBER	142	/* image_random_number *
#define IF_IMAGE_RANDOM_NUMBER_VREG	391	/* image_random_number_vreg *
#define IF_IMPORT_HIST	143	/* import_histogram *
#define IF_IMPORT_HIST_VREG	392	/* import_histogram_vreg *
#define IF_IMPORT_IMAGE	144	/* import_image *
#define IF_IMPORT_IMAGE_VREG	393	/* import_image_vreg *
#define IF_IMPORT_LUT	145	/* import_lut *
#define IF_IMPORT_LUT_VREG	394	/* import_lut_vreg *
#define IF_IMPORT_MATRIX	146	/* import_matrix *
#define IF_IMPORT_MATRIX_VREG	395	/* import_matrix_vreg *
#define IF_IMPORT_NBHOOD_ARRAY	147	/* import_neighbourhood_array *
#define IF_IMPORT_NBHOOD_ARRAY_VREG	396	/* import_neighbourhood_array_vreg *
#define IF_IMPORT_ROI_ARRAY	148	/* import_roi_array *
#define IF_IMPORT_ROI_ARRAY_VREG	397	/* import_roi_array_vreg *
#define IF_IMPORT_STATIC_ARRAY	149	/* import_static_array *
#define IF_IMPORT_STATIC_ARRAY_VREG	398	/* import_static_array_vreg *
#define IF_IMPORT_TUPLE	150	/* import_tuple *
#define IF_IMPORT_TUPLE_VREG	399	/* import_tuple_vreg *
#define IF_IMPORT_VALUE_BOUNDS	151	/* import_value_bounds *
#define IF_IMPORT_VALUE_BOUNDS_VREG	400	/* import_value_bounds_vreg *
#define IF_IMPULSE_BOXCAR	152	/* impulse_boxcar *
#define IF_IMPULSE_BOXCAR_VREG	401	/* impulse_boxcar_vreg *
#define IF_IMPULSE_DERIV_GAUSS	153	/* impulse_derivative_of_gaussian *
#define IF_IMPULSE_DERIV_GAUSS_VREG	402	/* impulse_derivative_of_gaussian_vreg *
#define IF_IMPULSE_DIFF_GAUSS	154	/* impulse_difference_of_gaussians *
#define IF_IMPULSE_DIFF_GAUSS_VREG	403	/* impulse_difference_of_gaussians_vreg *
#define IF_IMPULSE_ELLIPTICAL	155	/* impulse_elliptical *
#define IF_IMPULSE_ELLIPTICAL_VREG	404	/* impulse_elliptical_vreg *
#define IF_IMPULSE_GAUSS	156	/* impulse_gaussian *
#define IF_IMPULSE_GAUSS_VREG	405	/* impulse_gaussian_vreg *
#define IF_IMPULSE_LAPLACE_GAUSS	157	/* impulse_laplacian_of_gaussian *
#define IF_IMPULSE_LAPLACE_GAUSS_VREG	406	/* impulse_laplacian_of_gaussian_vreg *
#define IF_IMPULSE_PYRAMID	158	/* impulse_pyramid *
#define IF_IMPULSE_PYRAMID_VREG	407	/* impulse_pyramid_vreg *
#define IF_IMPULSE_RECTANGULAR	159	/* impulse_rectangular *
#define IF_IMPULSE_RECTANGULAR_VREG	408	/* impulse_rectangular_vreg *
#define IF_IMPULSE_SINC	160	/* impulse_sinc *

Table 6 - IPI-PIKS function descriptors (continued)

Function Descriptor Name	Code	Function Name
#define IF_IMPULSE_SINC_VREG	409	/* impulse_sinc_vreg */
#define IF_INPUT_OBJECT	161	/* input_object */
#define IF_INPUT_OBJECT_VREG	410	/* input_object_vreg */
#define IF_INQUIRE_CHAIN_ENVIRONMENT	162	/* inquire_chain_environment */
#define IF_INQUIRE_CHAIN_STATUS	163	/* inquire_chain_status */
#define IF_INQUIRE_ELEMENTS	164	/* inquire_elements */
#define IF_INQUIRE_IMAGE	165	/* inquire_image */
#define IF_INQUIRE_INDEX_ASSIGNMENT	166	/* inquire_index_assignment */
#define IF_INQUIRE_NON_IMAGE_OBJECT	167	/* inquire_non_image_object */
#define IF_INQUIRE_PIKS_IMPL	168	/* inquire_piks_implementation */
#define IF_INQUIRE_PIKS_STATUS	169	/* inquire_piks_status */
#define IF_INQUIRE_REPOSITORY	170	/* inquire_repository */
#define IF_INQUIRE_RESAMPLING	171	/* inquire_resampling */
#define IF_INSERT_PIXEL_PLANE	172	/* insert_pixel_plane */
#define IF_INSERT_PIXEL_PLANE_ROI	173	/* insert_pixel_plane_roi */
#define IF_INSERT_PIXEL_PLANE_ROI_VREG	411	/* insert_pixel_plane_roi_vreg */
#define IF_INSERT_PIXEL_PLANE_VREG	412	/* insert_pixel_plane_vreg */
#define IF_LABEL_OBJECTS	174	/* label_objects */
#define IF_LABEL_OBJECTS_VREG	413	/* label_objects_vreg */
#define IF_LAWS_TEXTURE_FEATURES	175	/* laws_texture_features */
#define IF_LAWS_TEXTURE_FEATURES_VREG	414	/* laws_texture_features_vreg */
#define IF_LEVEL_SLICE	176	/* level_slice */
#define IF_LEVEL_SLICE_VREG	415	/* level_slice_vreg */
#define IF_LINE_PROFILE	177	/* line_profile */
#define IF_LINE_PROFILE_VREG	416	/* line_profile_vreg */
#define IF_LOOKUP	178	/* lookup */
#define IF_LOOKUP_VREG	417	/* lookup_vreg */
#define IF_LOOKUP_INTERP	179	/* lookup_interpolated */
#define IF_LOOKUP_INTERP_VREG	418	/* lookup_interpolated_vreg */
#define IF_LUM_GENERATION	180	/* luminance_generation */
#define IF_LUM_GENERATION_VREG	419	/* luminance_generation_vreg */
#define IF_MARKOV_SPECTRUM	181	/* markov_spectrum */
#define IF_MARKOV_SPECTRUM_VREG	420	/* markov_spectrum_vreg */
#define IF_MOMENTS	182	/* moments */
#define IF_MOMENTS_VREG	421	/* moments_vreg */
#define IF_MONADIC_ARITH	183	/* monadic_arithmetic */
#define IF_MONADIC_ARITH_VREG	422	/* monadic_arithmetic_vreg */
#define IF_MONADIC_COMPLEX	184	/* monadic_complex */
#define IF_MONADIC_COMPLEX_VREG	423	/* monadic_complex_vreg */
#define IF_MONADIC_LOGICAL	185	/* monadic_logical */
#define IF_MONADIC_LOGICAL_ROI	186	/* monadic_logical_roi */
#define IF_MONADIC_LOGICAL_ROI_VREG	424	/* monadic_logical_roi_vreg */
#define IF_MONADIC_LOGICAL_VREG	425	/* monadic_logical_vreg */
#define IF_MORPHIC_PROCESSOR	187	/* morphic_processor */
#define IF_MORPHIC_PROCESSOR_ROI	188	/* morphic_processor_roi */
#define IF_MORPHIC_PROCESSOR_ROI_VREG	426	/* morphic_processor_roi_vreg */
#define IF_MORPHIC_PROCESSOR_VREG	427	/* morphic_processor_vreg */

Table 6 - IPI-PIKS function descriptors (continued)

Function Descriptor Name	Code	Function Name
#define IF MORPHOLOGY	189	/* morphology */
#define IF MORPHOLOGY_ROI	190	/* morphology_roi */
#define IF MORPHOLOGY_ROI_VREG	428	/* morphology_roi_vreg */
#define IF MORPHOLOGY_VREG	429	/* morphology_vreg */
#define IF_NBR_COUNT	191	/* neighbour_count */
#define IF_NBR_COUNT_VREG	430	/* neighbour_count_vreg */
#define IF_NOISE_COMBINATION	192	/* noise_combination */
#define IF_NOISE_COMBINATION_VREG	431	/* noise_combination_vreg */
#define IF_OPEN_CLOSE	193	/* open_close */
#define IF_OPEN_CLOSE_ROI	194	/* open_close_roi */
#define IF_OPEN_CLOSE_ROI_VREG	432	/* open_close_roi_vreg */
#define IF_OPEN_CLOSE_VREG	433	/* open_close_vreg */
#define IF_OPEN_PIKS	195	/* open_piks */
#define IF_OUTLIER_REMOVAL	196	/* outlier_removal */
#define IF_OUTLIER_REMOVAL_VREG	434	/* outlier_removal_vreg */
#define IF_OUTPUT_OBJECT	197	/* output_object */
#define IF_PAINT_PIXELS	198	/* paint_pixels */
#define IF_PAINT_PIXELS_VREG	435	/* paint_pixels_vreg */
#define IF_PERIMETER_CODE_GENERATOR	199	/* perimeter_code_generator */
#define IF_POLAR_TO_CARTESIAN	200	/* polar_to_cartesian */
#define IF_POLAR_TO_CARTESIAN_VREG	436	/* polar_to_cartesian_vreg */
#define IF_POWER_LAW_SCALING	201	/* power_law_scaling */
#define IF_POWER_LAW_SCALING_VREG	437	/* power_law_scaling_vreg */
#define IF_PSEUDOCOLOUR	202	/* pseudocolour */
#define IF_PSEUDOCOLOUR_VREG	438	/* pseudocolour_vreg */
#define IF_PUT_COLOUR_PIXEL	203	/* put_colour_pixel */
#define IF_PUT_COLOUR_PIXEL_VREG	439	/* put_colour_pixel_vreg */
#define IF_PUT_PIXEL	204	/* put_pixel */
#define IF_PUT_PIXEL_ROI	205	/* put_pixel_roi */
#define IF_PUT_PIXEL_ROI_VREG	440	/* put_pixel_roi_vreg */
#define IF_PUT_PIXEL_VREG	441	/* put_pixel_vreg */
#define IF_PUT_PIXEL_ARRAY	206	/* put_pixel_array */
#define IF_PUT_PIXEL_ARRAY_ROI	207	/* put_pixel_array_roi */
#define IF_PUT_PIXEL_ARRAY_ROI_VREG	442	/* put_pixel_array_roi_vreg */
#define IF_PUT_PIXEL_ARRAY_VREG	443	/* put_pixel_array_vreg */
#define IF_PUT_PIXEL_RECORD	208	/* put_pixel_record */
#define IF_PUT_PIXEL_RECORD_ROI	209	/* put_pixel_record_roi */
#define IF_PUT_PIXEL_RECORD_ROI_VREG	444	/* put_pixel_record_roi_vreg */
#define IF_PUT_PIXEL_RECORD_VREG	445	/* put_pixel_record_vreg */
#define IF_RESCALE	210	/* rescale */
#define IF_RESCALE_ROI	211	/* rescale_roi */
#define IF_RESCALE_ROI_VREG	446	/* rescale_roi_vreg */
#define IF_RESCALE_VREG	447	/* rescale_vreg */
#define IF_RESIZE	212	/* resize */
#define IF_RESIZE_ROI	213	/* resize_roi */
#define IF_RESIZE_ROI_VREG	448	/* resize_roi_vreg */
#define IF_RESIZE_VREG	449	/* resize_vreg */

Table 6 - IPI-PIKS function descriptors (continued)

Function Descriptor Name	Code	Function Name
#define IF_RETURN_REPOSITORY_ID	214	/* return_repository_id *
#define IF_RETURN_REPOSITORY_ID_VREG	450	/* return_repository_id_vreg *
#define IF_ROI_COORDINATE	215	/* roi_coordinate *
#define IF_ROI_COORDINATE_VREG	451	/* roi_coordinate_vreg *
#define IF_ROI_ELLIPTICAL	216	/* roi_elliptical *
#define IF_ROI_ELLIPTICAL_VREG	452	/* roi_elliptical_vreg *
#define IF_ROI_POLYGON	217	/* roi_polygon *
#define IF_ROI_POLYGON_VREG	453	/* roi_polygon_vreg *
#define IF_ROI_RECTANGULAR	218	/* roi_rectangular *
#define IF_ROI_RECTANGULAR_VREG	454	/* roi_rectangular_vreg *
#define IF_ROTATE	219	/* rotate *
#define IF_ROTATE_ROI	220	/* rotate_roi *
#define IF_ROTATE_ROI_VREG	455	/* rotate_roi_vreg *
#define IF_ROTATE_VREG	456	/* rotate_vreg *
#define IF_RUBBER_BAND_SCALING	221	/* rubber_band_scaling *
#define IF_RUBBER_BAND_SCALING_VREG	457	/* rubber_band_scaling_vreg *
#define IF_SEQ_AVERAGE	222	/* sequence_average *
#define IF_SEQ_AVERAGE_VREG	458	/* sequence_average_vreg *
#define IF_SEQ_KARHUNEN_LOEVE	223	/* sequence_karhunen_loeve *
#define IF_SEQ_KARHUNEN_LOEVE_VREG	459	/* sequence_karhunen_loeve_vreg *
#define IF_SEQ_RUNNING_MEASURES	224	/* sequence_running_measures *
#define IF_SEQ_RUNNING_MEASURES_VREG	460	/* sequence_running_measures_vreg *
#define IF_SET_ERROR_HANDLER	269	/* set_error_handler *
#define IF_SET_GLOBALS	225	/* set_globals *
#define IF_SET_GLOBALS_VREG	461	/* set_globals_vreg *
#define IF_SET_IMAGE_ATTRS	226	/* set_image_attributes *
#define IF_SET_IMAGE_ATTRS_VREG	462	/* set_image_attributes_vreg *
#define IF_SET_INDEX_ASSIGNMENT	227	/* set_index_assignment *
#define IF_SET_INDEX_ASSIGNMENT_VREG	463	/* set_index_assignment_vreg *
#define IF_SHAPE_METRICS	228	/* shape_metrics *
#define IF_SPATIAL_MOMENTS_INVARIANT	229	/* spatial_moments_invariant *
#define IF_SPATIAL_MOMENTS_SCALED	230	/* spatial_moments_scaled *
#define IF_SPLIT_IMAGE	231	/* split_image *
#define IF_SPLIT_IMAGE_VREG	464	/* split_image_vreg *
#define IF_SUBSAMPLE	232	/* subsample *
#define IF_SUBSAMPLE_ROI	233	/* subsample_roi *
#define IF_SUBSAMPLE_ROI_VREG	465	/* subsample_roi_vreg *
#define IF_SUBSAMPLE_VREG	466	/* subsample_vreg *
#define IF_SYNCH	234	/* synchronize *
#define IF_TEMPL_MATCH	235	/* template_match *
#define IF_TEMPL_MATCH_VREG	467	/* template_match_vreg *
#define IF_THRESHOLD	236	/* threshold *
#define IF_THRESHOLD_VREG	468	/* threshold_vreg *
#define IF_TRANSFORM_COSINE	237	/* transform_cosine *
#define IF_TRANSFORM_COSINE_VREG	469	/* transform_cosine_vreg *
#define IF_TRANSFORM_FOURIER	238	/* transform_fourier *
#define IF_TRANSFORM_FOURIER_VREG	470	/* transform_fourier_vreg *

Table 6 - IPI-PIKS function descriptors (continued)

Function Descriptor Name	Code	Function Name
#define IF_TRANSFORM_HADAMARD	239	/* transform_hadamard */
#define IF_TRANSFORM_HADAMARD_VREG	471	/* transform_hadamard_vreg */
#define IF_TRANSFORM_HARTLEY	240	/* transform_hartley */
#define IF_TRANSFORM_HARTLEY_VREG	472	/* transform_hartley_vreg */
#define IF_TRANSLATE	241	/* translate */
#define IF_TRANSLATE_ROI	242	/* translate_roi */
#define IF_TRANSLATE_ROI_VREG	473	/* translate_roi_vreg */
#define IF_TRANSLATE_VREG	474	/* translate_vreg */
#define IF_UNARY_INTEGER	243	/* unary_integer */
#define IF_UNARY_INTEGER_VREG	475	/* unary_integer_vreg */
#define IF_UNARY_REAL	244	/* unary_real */
#define IF_UNARY_REAL_VREG	476	/* unary_real_vreg */
#define IF_UNSHARP_MASK	245	/* unsharp_mask */
#define IF_UNSHARP_MASK_VREG	477	/* unsharp_mask_vreg */
#define IF_VALUE_BOUNDS	246	/* value_bounds */
#define IF_VALUE_BOUNDS_VREG	478	/* value_bounds_vreg */
#define IF_VREG_ALTER	247	/* vreg_alter */
#define IF_VREG_ALTER_VREG	479	/* vreg_alter_vreg */
#define IF_VREG_CLEAR	248	/* vreg_clear */
#define IF_VREG_CLEAR_VREG	480	/* vreg_clear_vreg */
#define IF_VREG_CONDITIONAL	249	/* vreg_conditional */
#define IF_VREG_CONDITIONAL_VREG	481	/* vreg_conditional_vreg */
#define IF_VREG_COPY	250	/* vreg_copy */
#define IF_VREG_COPY_VREG	482	/* vreg_copy_vreg */
#define IF_VREG_DELETE	251	/* vreg_delete */
#define IF_VREG_DELETE_VREG	483	/* vreg_delete_vreg */
#define IF_VREG_GET	252	/* vreg_get */
#define IF_VREG_SET	253	/* vreg_set */
#define IF_VREG_SET_VREG	484	/* vreg_set_vreg */
#define IF_VREG_WAIT	254	/* vreg_wait */
#define IF_VREG_WAIT_VREG	485	/* vreg_wait_vreg */
#define IF_WALLIS_STAT_DIFF	255	/* wallis_statistical_differencing */
#define IF_WALLIS_STAT_DIFF_VREG	486	/* wallis_statistical_differencing_vreg */
#define IF_WARP_CONTROL_POINT	256	/* warp_control_point */
#define IF_WARP_CONTROL_POINT_ROI	257	/* warp_control_point_roi */
#define IF_WARP_CONTROL_POINT_ROI_VREG	487	/* warp_control_point_roi_vreg */
#define IF_WARP_CONTROL_POINT_VREG	488	/* warp_control_point_vreg */
#define IF_WARP_LUT	258	/* warp_lut */
#define IF_WARP_LUT_ROI	259	/* warp_lut_roi */
#define IF_WARP_LUT_ROI_VREG	489	/* warp_lut_roi_vreg */
#define IF_WARP_LUT_VREG	490	/* warp_lut_vreg */
#define IF_WARP_POLYNOMIAL	260	/* warp_polynomial */
#define IF_WARP_POLYNOMIAL_ROI	261	/* warp_polynomial_roi */
#define IF_WARP_POLYNOMIAL_ROI_VREG	491	/* warp_polynomial_roi_vreg */
#define IF_WARP_POLYNOMIAL_VREG	492	/* warp_polynomial_vreg */
#define IF_WINDOWING_FUNCTION	262	/* windowing_function */
#define IF_WINDOWING_FUNCTION_VREG	493	/* windowing_function_vreg */

Table 6 - IPI-PIKS function descriptors (concluded)

Function Descriptor Name	Code	Function Name
#define IF_WINDOW_LEVEL	263	/* window_level *
#define IF_WINDOW_LEVEL_VREG	494	/* window_level_vreg *
#define IF_WINDOW_STATISTICS	264	/* window_statistics *
#define IF_WINDOW_STATISTICS_VREG	495	/* window_statistics_vreg *
#define IF_Z_MERGE	265	/* z_merge *
#define IF_Z_MERGE_VREG	496	/* z_merge_vreg *
#define IF_ZOOM	266	/* zoom *
#define IF_ZOOM_ROI	267	/* zoom_roi *
#define IF_ZOOM_ROI_VREG	497	/* zoom_roi_vreg *
#define IF_ZOOM_VREG	498	/* zoom_vreg *
#define IF_3D_SLICE	268	/* 3d_slice *
#define IF_3D_SLICE_VREG	499	/* 3d_slice_vreg *

6.3.2 IPI-PIKS convenience function descriptors

The IPI-PIKS convenience function descriptors are assigned values beginning at 1000.

The function descriptor numbers are presented in Table 7 in alphabetical order by defined function name.

Table 7 - IPI-PIKS convenience function descriptors

Function Descriptor Name	Code	Function Name
#define IC_CREATE_UNBND_IMAGE_COPY	1000	/* create unbounded image copy *
#define IC_GENERATE_2D_ROI_RECTANGULAR	1001	/* generate 2d roi rectangular *
#define IC_GENERATE_ROI_COORDINATE	1002	/* generate roi coordinate *
#define IC_GENERATE_ROI_ELLIPTICAL	1003	/* generate roi elliptical *
#define IC_GENERATE_ROI_POLYGON	1004	/* generate roi polygon *
#define IC_GENERATE_ROI_RECTANGULAR	1005	/* generate roi rectangular *
#define IC_GENERATE_ND_1_TUPLE	1006	/* generate nd 1 tuple *
#define IC_GENERATE_ND_3_TUPLE	1007	/* generate nd 3 tuple *
#define IC_GENERATE_ND_4_TUPLE	1008	/* generate nd 4 tuple *
#define IC_GENERATE_ND_5_TUPLE	1009	/* generate nd 5 tuple *
#define IC_GENERATE_RD_3_TUPLE	1010	/* generate rd 3 tuple *
#define IC_GENERATE_RD_4_TUPLE	1011	/* generate rd 4 tuple *
#define IC_GENERATE_RD_5_TUPLE	1012	/* generate rd 5 tuple *
#define IC_GENERATE_SD_1_TUPLE	1013	/* generate sd 1 tuple *
#define IC_GENERATE_SD_3_TUPLE	1014	/* generate sd 3 tuple *
#define IC_GENERATE_SD_4_TUPLE	1015	/* generate sd 4 tuple *
#define IC_GENERATE_SD_5_TUPLE	1016	/* generate sd 5 tuple *
#define IC_PREPARE_COLOUR_IMAGE	1017	/* prepare colour image *
#define IC_PREPARE_MONOCHROME_IMAGE	1018	/* prepare mon image *
#define IC_PREPARE_2D_ROI_RECTANGULAR	1019	/* prepare 2d roi rectangular *
#define IC_PREPARE_ROI	1020	/* prepare roi *
#define IC_TILED_IMAGE_EXPORT	1021	/* tiled image export *
#define IC_TILED_IMAGE_IMPORT	1022	/* tiled image import *

6.3.3 IPI-IIF function descriptors

The IPI-IIF function descriptor numbers correspond to those defined in the functional specification. The function which was added in this binding, *set_error_handler*, is assigned the next available sequential function number, 009, following the gateway control functionality elements presented in the functional specification.

The function descriptor numbers are presented in Table 8 in alphabetical order by defined function name.

Table 8 - IPI-IIF function descriptors

Function Descriptor Name	Code	Function Name
#define IF_ATTACH_ENTITY	404	/* attach iif_syntax_entity */
#define IF_CLOSE_GATEWAY	2	/* close_iif_gateway */
#define IF_CLOSE_PORT	102	/* close_port */
#define IF_COMPRESS_ENTITY	501	/* compress_iif_syntax_entity */
#define IF_COPY_ENTITY	402	/* copy_iif_syntax_entity */
#define IF_CREATE_ENTITY	401	/* create_iif_syntax_entity */
#define IF_DECOMPRESS_ENTITY	502	/* decompress_iif_syntax_entity */
#define IF_DELETE_ATOMIC_SEQ_ELEMENT	408	/* delete_iif_atomic_sequence_element */
#define IF_DELETE_ENTITY	403	/* delete_iif_syntax_entity */
#define IF_DETACH_ENTITY	405	/* detach_iif_syntax_entity */
#define IF_ERROR_HANDLER_IIF	6	/* error_handler */
#define IF_ERROR_LOGGER_IIF	7	/* error_logger */
#define IF_ERROR_TEST_IIF	8	/* error_test */
#define IF_EXPORT_IIF_DATA_TO_PIKS	105	/* export_iif_data_to_piks */
#define IF_GENERATE_AND_WRITE_IMAGE	202	/* generate_and_write_iif_image */
#define IF_GET_ENTITY_CMP	308	/* get_iif_syntax_entity_component */
#define IF_GET_ENTITY_VALUE	309	/* get_iif_syntax_entity_value */
#define IF_GET_IMAGE_SUBRANGE	601	/* get_image_subrange */
#define IF_IMPORT_IIF_DATA_FROM_PIKS	106	/* import_iif_data_from_piks */
#define IF_INQUIRE_GATEWAY_STATUS	4	/* inquire_iif_gateway_status */
#define IF_INQUIRE_CMP_TYPE_DEF	307	/* inquire_iif_syntax_component_type_definition */
#define IF_INQUIRE_ENTITY_CHOICE	301	/* inquire_iif_syntax_entity_choice */
#define IF_INQUIRE_ENTITY_NUMBER	304	/* inquire_iif_syntax_entity_number */
#define IF_INQUIRE_ENTITY_OPTIONAL	302	/* inquire_iif_syntax_entity_optional */
#define IF_INQUIRE_ENTITY_SEQ	303	/* inquire_iif_syntax_entity_sequence */
#define IF_INQUIRE_ENTITY_TYPE_DEF	306	/* inquire_iif_syntax_entity_type_definition */
#define IF_INQUIRE_ENTITY_VALIDITY	305	/* inquire_iif_syntax_entity_validity */
#define IF_INQUIRE_PORT	103	/* inquire_port */
#define IF_INQUIRE_FUNCTIONALITY	005	/* inquire_supported_functionality */
#define IF_INSERT_ATOMIC_SEQ_ELEMENT	407	/* insert_iif_atomic_sequence_element */
#define IF_OPEN_GATEWAY	1	/* open_iif_gateway */
#define IF_OPEN_PORT	101	/* open_port */
#define IF_PUT_ENTITY_VALUE	310	/* put_iif_syntax_entity_value */
#define IF_PUT_IMAGE_SUBRANGE	602	/* put_image_subrange */
#define IF_READ_AND_PARSE_IMAGE	201	/* read_and_parse_iif_image */
#define IF_RELEASE_ENTITY_ID	406	/* release_iif_syntax_entity_identifier */
#define IF_RESET_GATEWAY	3	/* reset_iif_gateway */
#define IF_RESET_PORT	104	/* reset_port */
#define IF_SET_ERROR_HANDLER	9	/* set_error_handler */

6.3.4 IPI-IIF convenience function descriptors

The IPI-IIF convenience function descriptors are assigned values beginning at 1000.

The function descriptor numbers are presented in Table 9 in alphabetical order by defined function name.

Table 9 - IPI-IIF convenience function descriptors

Function Descriptor Name	Code	Function Name
#define IC_ATTACH	1000	/* attach *
#define IC_ATTACH_SEQ	1001	/* attach_sequence *
#define IC_ATTACH_SEQ_END	1002	/* attach_sequence_end *
#define IC_CREATE_ID	1003	/* create_identifier *
#define IC_GET_CMP	1004	/* get_component *
#define IC_GET_SEQ_CMP	1005	/* get_sequence_component *
#define IC_GET_ENTITY_BOOL	1006	/* get_entity_boolean *
#define IC_GET_ENTITY_INTEGER	1007	/* get_entity_integer *
#define IC_GET_ENTITY_REAL	1008	/* get_entity_real *
#define IC_GET_ENTITY_STRING	1009	/* get_entity_string *
#define IC_GET_SEQ_BOOL	1010	/* get_sequence_boolean *
#define IC_GET_SEQ_END	1011	/* get_sequence_end *
#define IC_GET_SEQ_INTEGER	1012	/* get_sequence_integer *
#define IC_GET_SEQ_REAL	1013	/* get_sequence_real *
#define IC_GET_SEQ_STRING	1014	/* get_sequence_string *
#define IC_PUT_ENTITY_BOOL	1015	/* put_entity_boolean *
#define IC_PUT_ENTITY_INTEGER	1016	/* put_entity_integer *
#define IC_PUT_ENTITY_REAL	1017	/* put_entity_real *
#define IC_PUT_ENTITY_STRING	1018	/* put_entity_string *
#define IC_PUT_SEQ_BOOL	1019	/* put_sequence_boolean *
#define IC_PUT_SEQ_INTEGER	1020	/* put_sequence_integer *
#define IC_PUT_SEQ_REAL	1021	/* put_sequence_real *
#define IC_PUT_SEQ_STRING	1022	/* put_sequence_string *

6.4 Error descriptors

ISO/IEC 12087 defines two Application Program Interfaces (APIs). The error codes listed below support separate or combined implementation of the IPI-PIKS and IPI-IIF, i.e., the error codes are unique regardless of combined implementation.

The error functions require a unique mapping of the errors to a set of specified numbers.

The macro names of error descriptors are prefixed IE_.

6.4.1 IPI-PIKS errors

Table 10 presents the IPI-PIKS error descriptors. Most of the error codes and conditions correspond to those described in ISO/IEC 12087. A number of codes have been added in this part of ISO/IEC 12088 to represent error conditions not covered in ISO/IEC 12087. These additional error codes have been enumerated with code values near the collection of similar error conditions wherever possible. Codes added in this binding are 4, 5, 15, 16, 17, 18, 19, 31, 32, 40, 56 and 104.

Table 10 - IPI-PIKS error descriptors

Error Descriptor Name	Code	Error Condition
/*	< 000	Implementation-dependent errors. */
#define IE_NO_ERROR	0	/* No error. */
#define IE_PARAMETER_DATA_TYPE	1	/* Element input data parameter is incorrect data type. Abort operation and issue warning. */
#define IE_ENUM_PARAMETER_RANGE	2	/* Enumerated option parameter is out of range. Abort operation and issue warning. */
#define IE_OPTION	3	/* Signed integer option parameter is out of range. Abort operation and issue warning. */
#define IE_INCOMPATIBLE_ATTRS	4	/* Incompatible attributes specified will not be set. Warning will be issued and program will continue. */
#define IE_ATTEMPT_ATTR_CHANGE	5	/* Attempt to change existing attributes not accepted. Warning will be issued and program will continue. */
#define IE_IMAGE_ID	11	/* Image identifier is invalid. Abort operation and issue warning. */
#define IE_NON_ID	12	/* Non-image identifier is invalid. Abort operation and issue warning. */
#define IE_ID_NOT_IMAGE	13	/* Identifier is not to an image. Abort operation and issue warning. */
#define IE_ID_NOT_NON	14	/* Identifier is not to a non-image. Abort operation and issue warning. */
#define IE_SOURCE_IMAGE_EMPTY	15	/* Source image does not contain data. Abort operation and issue warning. */
#define IE_SOURCE_NON_EMPTY	16	/* Source non-image object does not contain data. Abort operation and issue warning. */
#define IE_SUB_IMAGE_REASSIGN	17	/* Attempt to change image identifier of existing sub image. Abort operation and issue warning. */
#define IE_SOURCE_DEST_IMAGE_SIZE	18	/* Mismatch between source and destination image dimension sizes. Abort operation and issue warning. */
#define IE_SOURCE_DEST_NON_SIZE	19	/* Mismatch between source and destination non image dimension sizes. Abort operation and issue warning. */

Table 10 - IPI-PIKS error descriptors (continued)

Error Descriptor Name	Code	Error Condition
#define IE_SOURCE_IMAGE_INPUT_STRUCTURE	21	/* Mismatch between source image and element input structure. Abort operation and issue warning. */
#define IE_SOURCE_NON_INPUT_STRUCTURE	22	/* Mismatch between source non-image and element input structure. Abort operation and issue warning. */
#define IE_DEST_IMAGE_OUTPUT_STRUCTURE	23	/* Mismatch between destination image and element output structure. Abort operation and issue warning. */
#define IE_DEST_NON_OUTPUT_STRUCTURE	24	/* Mismatch between destination non-image and element output structure. Abort operation and issue warning. */
#define IE_SOURCE_DEST_IMAGE_STRUCTURE	25	/* Mismatch between source image and destination image structure. Abort operation and issue warning. */
#define IE_SOURCE_DEST_NON_STRUCTURE	26	/* Mismatch between source non-image and destination non-image structures. Abort operation and issue warning. */
#define IE_SOURCE_IMAGE_STRUCTURES	27	/* Mismatch between multiple source image structures. Abort operation and issue warning. */
#define IE_SOURCE_NON_STRUCTURES	28	/* Mismatch between multiple source non-image structures. Abort operation and issue warning. */
#define IE_DEST_IMAGE_STRUCTURES	29	/* Mismatch between multiple destination image structures. Abort operation and issue warning. */
#define IE_DEST_NON_STRUCTURES	30	/* Mismatch between multiple destination non-image structures. Abort operation and issue warning. */
#define IE_SOURCE_IMAGE_DEST_NON	31	/* Mismatch between source image and destination non-image structure. Abort operation and issue warning. */
#define IE_SOURCE_NON_DEST_IMAGE	32	/* Mismatch between source non-image and destination image structure. Abort operation and issue warning. */
#define IE_SOURCE_IMAGE_DEST_NON_TYPE	40	/* Mismatch between source image and destination non-image data type. Abort operation and issue warning. */
#define IE_SOURCE_IMAGE_INPUT_TYPE	41	/* Mismatch between source image and element input data type. Abort operation and issue warning. */
#define IE_SOURCE_NON_INPUT_TYPE	42	/* Mismatch between source non-image and element input data type. Abort operation and issue warning. */
#define IE_DEST_IMAGE_OUTPUT_TYPE	43	/* Mismatch between destination image and element output data type. Abort operation and issue warning. */
#define IE_DEST_NON_OUTPUT_TYPE	44	/* Mismatch between destination non-image and element output data type. Abort operation and issue warning. */
#define IE_SOURCE_DEST_IMAGE_TYPE	45	/* Mismatch between source image and destination image data types. Abort operation and issue warning. */
#define IE_SOURCE_DEST_NON_TYPE	46	/* Mismatch between source non-image and destination non-image data types. Abort operation and issue warning. */
#define IE_SOURCE_IMAGE_TYPES	47	/* Mismatch between multiple source image data types. Abort operation and issue warning. */
#define IE_SOURCE_NON_TYPES	48	/* Mismatch between multiple source non-image data types. Abort operation and issue warning. */
#define IE_DEST_IMAGE_TYPES	49	/* Mismatch between multiple destination image data types. Abort operation and issue warning. */
#define IE_DEST_NON_TYPES	50	/* Mismatch between multiple destination non-image data types. Abort operation and issue warning. */
#define IE_DEST_IMAGE_NON_TYPES	51	/* Mismatch between destination image and destination non-image data types. Abort operation and issue warning. */

Table 10 - IPI-PIKS error descriptors (continued)

Error Descriptor Name	Code	Error Condition
#define IE_SOURCE_IMAGE_NON_TYPE	52	/* Mismatch between source image and source non-image data types. Abort operation and issue warning. */
#define IE_DEST_IMAGE_SOURCE_NON_TYPE	53	/* Mismatch between destination image and source non-image data type. Abort operation and issue warning. */
#define IE_SOURCE_IMAGE_INPUT_COLOUR	54	/* Mismatch between source image and element input colour space. Abort operation and issue warning. */
#define IE_DEST_IMAGE_OUTPUT_COLOUR	55	/* Mismatch between destination image and element output colour space. Abort operation and issue warning. */
#define IE_SOURCE_DEST_IMAGE_COLOUR	56	/* Mismatch between source image and destination image colour space. Abort operation and issue warning. */
#define IE_HETEROGENEOUS_BANDS	61	/* Element does not support heterogeneous bands. Abort operation and issue warning. */
#define IE_ELEMENT_NOT_CHAINABLE	62	/* Element is not chainable. Abort operation and issue warning. */
#define IE_INT_EXCESS	71	/* Specified integer size exceeds implementation integer size. Abort operation and issue warning. */
#define IE_REAL_EXCESS	72	/* Specified real arithmetic size exceeds implementation real arithmetic size. Abort operation and issue warning. */
#define IE_IMAGE_EXCESS	73	/* Specified image size exceeds implementation image size. Abort operation and issue warning. */
#define IE_ARRAY_EXCESS	74	/* Specified array size exceeds implementation array size. Abort operation and issue warning. */
#define IE_INPUT_PARAMETER_RANGE	101	/* Element input data parameter is out of range. Abort operation and issue warning. */
#define IE_UPPER_LOWER_PARAMETER	102	/* Upper parameter bound is smaller than lower parameter bound. Abort operation and issue warning. */
#define IE_SOURCE_ZERO_LESS	103	/* Source image contains pixel of zero or negative amplitude. Abort operation and issue warning. */
#define IE_SOURCE_NEGATIVE	104	/* Source image contains pixel of negative amplitude. Abort operation and issue warning. */
#define IE_ARRAY_ROW_SIZE	111	/* Array row dimension is not a positive integer. Abort operation and issue warning. */
#define IE_ARRAY_COLUMN_SIZE	112	/* Array column dimension is not a positive integer. Abort operation and issue warning. */
#define IE_ARRAY_ROW_NOT_ODD	113	/* Array row dimension is not a positive odd integer. Abort operation and issue warning. */
#define IE_ARRAY_COLUMN_NOT_ODD	114	/* Array column dimension is not a positive odd integer. Abort operation and issue warning. */
#define IE_ARRAY_ROW_NOT_EVEN	115	/* Array row dimension is not a positive even integer. Abort operation and issue warning. */
#define IE_ARRAY_COLUMN_NOT_EVEN	116	/* Array column dimension is not a positive even integer. Abort operation and issue warning. */
#define IE_TRUE_MASK_NOT_ODD	117	/* Number of TRUE state mask array elements is not a positive odd integer. Abort operation and issue warning. */
#define IE_LUT_SEGMENT_RANGE	121	/* Lookup table segment value is out of range. Abort operation and issue warning. */
#define IE_LUT_INDEX_RANGE	122	/* Lookup table index is out of range. Abort operation and issue warning. */

Table 10 - IPI-PIKS error descriptors (continued)

Error Descriptor Name	Code	Error Condition
#define IE_LUT_SIZE	123	/* Lookup table size is insufficient. Abort operation and issue warning. *
#define IE_SEED_NOT_FALSE	131	/* Seed pixel is not in FALSE state. Abort operation and issue warning. *
#define IE_BLOCK_SIZE_NOT_POWER_2	132	/* Block size dimensions not a power of two. Abort operation and issue warning. *
#define IE_SIZE_NOT_MULTIPLE	133	/* j-k pixel plane sizes are not integer multiples of the block size. Abort operation and issue warning. *
#define IE_COVARIANCE_NOT_SYMMETRIC	134	/* Covariance matrix is not symmetric. Abort operation and issue warning. *
#define IE_INTERNAL_TYPE_CONV	151	/* Illegal internal data type conversion. Abort operation and issue warning. *
#define IE_IMPORT_TYPE_CONV	152	/* Illegal import data type conversion. Abort operation and issue warning. *
#define IE_EXPORT_TYPE_CONV	153	/* Illegal export data type conversion. Abort operation and issue warning. *
#define IE_ID_ARRAYS_UNEQUAL	161	/* Composite image identifier arrays are of different length. Abort operation and issue warning. *
#define IE_ID_LISTS_UNEQUAL	162	/* Composite image identifier lists are of different length. Abort operation and issue warning. *
#define IE_ID_RECORD_UNEQUAL	163	/* Composite image identifier records are of different length. Abort operation and issue warning. *
#define IE_ID_LIST_EMPTY	164	/* Composite image identifier list is empty. Abort operation and issue warning. *
#define IE_ID_RECORD_EMPTY	165	/* Composite image identifier record is empty. Abort operation and issue warning. *
#define IE_ID_RECORD_NAMES	166	/* Composite image identifier record names are not unique. Abort operation and issue warning. *
#define IE_HIST_OVERFLOW	171	/* Histogram bin overflow. Abort operation and issue warning. *
#define IE_DIVISION_BY_ZERO	172	/* Attempted division by zero-valued constant. Abort operation and issue warning. *
#define IE_CHAIN_TYPE_INVALID	201	/* Chain type is invalid. Abort operation and issue warning. *
#define IE_CHAIN_ID_INVALID	202	/* Chain identifier is invalid. Abort operation and issue warning. *
#define IE_CHAIN_ID_NOT_IMAGE	203	/* Chain identifier is not to an image. Abort operation and issue warning. *
#define IE_CHAIN_RESOURCE_PARAMETER	204	/* Chain resource type parameter invalid. Abort operation and issue warning. *
#define IE_CHAIN_PARAMETER	205	/* Chain parameter is invalid. Abort operation and issue warning. *
#define IE_CHAIN_EXECUTING	206	/* Chain already executing. Abort operation and issue warning. *
#define IE_CHAIN_DELETED	207	/* Chain already deleted. Abort operation and issue warning. *

Table 10 - IPI-PIKS error descriptors (concluded)

Error Descriptor Name	Code	Error Condition
#define IE_CHAIN_NOT_EXECUTING	208	/* Chain is not executing. Abort operation and issue warning. */
#define IE_CHAIN_INCOMPLETE	209	/* Chain definition incomplete. Abort operation and issue warning. */
#define IE_CHAIN_DEFINED	210	/* Chain previously defined. Abort operation and issue warning. */
#define IE_CHAIN_NOT_BUILD_STATE	211	/* Not in CHAIN_BUILD state. Abort operation and issue warning. */
#define IE_CHAIN_EXECUTE	212	/* Attempt to begin a chain in CHAIN_EXECUTE state. Abort operation and issue warning. */
#define IE_CHAIN_SELECTOR	213	/* Selector less than one or greater than number of chain elements. Abort operation and issue warning. */
#define IE_CHAIN_MISMATCH	214	/* Next chainable PIKS element does not match selected element built into chain. Abort operation and issue warning. */
#define IE_CHAIN_ASYNC_EXECUTING	215	/* Chain is presently on the asynchronous queue or is in execution. Abort operation and issue warning. */
#define IE_VREG_SOURCE_DEST_TYPES	251	/* Mismatch between source and destination virtual register data types. Abort operation and issue warning. */
#define IE_VREG_NOT_SET	252	/* Virtual register is not in VREG_SET state. Abort operation and issue warning. */
#define IE_PIKS_OPEN	301	/* State is PIKS_OPEN when PIKS_CLOSED is expected. Log the error, abort operation, invoke <i>close_piks_emergency</i> . */
#define IE_PIKS_CLOSED	302	/* State is PIKS_CLOSED when PIKS_OPEN is expected. Return to the calling PIKS element to resume execution with the error flag parameter set to 'error' whether or not a prior error has been recorded. */
#define IE_PIKS_INITIALIZATION	303	/* PIKS initialization failure. Log the error, abort operation, invoke <i>close_piks_</i> <i>emergency</i> . */
#define IE_PIKS_CLOSURE	304	/* PIKS closure failure. Abort operation and issue warning. */
#define IE_NO_PARAMETER_DEFAULT	305	/* Unsupported default state for parameter. Log the error, abort operation, invoke <i>close_piks_emergency</i> . */
#define IE_DEFAULT	306	/* Invalid default state. Abort operation and issue warning. */
#define IE_ELEMENT_NON_PROFILE	307	/* Element is not in implementation conformance profile. Abort operation and issue warning. */

6.4.2 IPI-IIF errors

Table 11 presents the IPI-IIF error descriptors.

NOTE - The IE_IN_OUT error (code 807) applies to data content.

Table 11 - IPI-IIF error descriptors

Error Descriptor Name	Code	Error Condition
/*	< 000	Implementation-dependent errors.
#define IE_NO_ERROR	000	/* No error.
#define IE_IIF_IMPROPER_STATUS	801	/* IIF gateway is not in the proper status. Abort operation and issue warning.
#define IE_PORT_CANNOT_OPEN	802	/* Identified port cannot be opened. Abort operation and issue warning.
#define IE_READING_PORT_CLOSED	803	/* Identified port has not been opened for reading. Abort operation and issue warning.
#define IE_WRITING_PORT_CLOSED	804	/* Identified port has not been opened for writing. Abort operation and issue warning.
#define IE_IMPORT_FAILED	805	/* Import from IPI-PIKS failed. Abort operation and issue warning.
#define IE_EXPORT_FAILED	806	/* Export to IPI-PIKS failed. Abort operation and issue warning.
#define IE_IN_OUT	807	/* I/O error. Abort operation and issue warning.
#define IE_SYNTAX_DATA	808	/* Syntactical error within IIF data stream. Abort operation and issue warning.
#define IE_SEMANTICS_DATA	809	/* Semantical error within IIF data stream. Abort operation and issue warning.
#define IE_INVALID_SYNTAX_ID	810	/* IIF syntax entity identifier is invalid. Abort operation and issue warning.
#define IE_ENTITY_WRONG_TYPE	811	/* IIF syntax entity has a wrong type. Abort operation and issue warning.
#define IE_SUBENTITIES_MISSING	812	/* IIF syntax entity has missing subentities. Abort operation and issue warning.
#define IE_SEMANTIC_CONSTRAINT	813	/* Violation of semantical constraints within the IIF syntax entity. Abort operation and issue warning.
#define IE_ENTITY_NAME_WRONG	814	/* Component name does not exist for given IIF syntax entity type. Abort operation and issue warning.
#define IE_CMP_NAME_AMBIGUOUS	815	/* Component name is ambiguous at this level of IIF syntax entity. Abort operation and issue warning.
#define IE_NO_SUBENTITY	816	/* Specified component has no subentity at this index. Abort operation and issue warning.
#define IE_COMPRESSION_CONSTRAINTS	817	/* Violation of data compression constraints. Abort operation and issue warning.
#define IE_INDEX_SUBRANGE	818	/* Invalid index subrange. Abort operation and issue warning.
#define IE_SIZE_OR_START	819	/* Invalid size or starting coordinate. Abort operation and issue warning.
#define IE_COMPLEX_IMAGE_STRUCTURE	820	/* Input image structure too complex. Abort operation and issue warning.

Table 11 - IPI-IIF error descriptors (concluded)

Error Descriptor Name	Code	Error Condition
#define IE_SUBENTITY_ALREADY_ATTACHED	821	/* Subentity already attached to another syntax entity. Abort operation and issue warning. */
#define IE_ENTITY_ATTACHED_ELSEWHERE	822	/* IIF syntax entity still attached to another syntax entity. Abort operation and issue warning. */
#define IE_ENTITY_NO_PARENT	823	/* IIF syntax entity not attached to a parent. Abort operation and issue warning. */
#define IE_CMP_NOT_ATOMIC_TYPE	824	/* Component not atomic type. Abort operation and issue warning. */
#define IE_ENTITY_ALREADY_HAS_SUBENTITY	825	/* IIF syntax entity already has a subentity attached to the specified component, or (in case of a CHOICE construct) to another member of this CHOICE. Abort operation and issue warning. */
#define IE_SYNTAX_ENTITY_INVALID	826	/* IIF syntax entity is invalid. Abort operation and issue warning. */

6.4.3 System errors

Table 12 presents the system error descriptors.

Table 12 - System error descriptors

Error macro name	Code	Error condition
#define IE_POINTER	1001 /* Pointer out of range.	*
#define IE_ID	1002 /* Identifier not assigned.	*
#define IE_STORE	1003 /* Storage overflow.	*
#define IE_INPUT_OUTPUT	1004 /* Input/output failure.	*
#define IE_ALTERATION	1005 /* Attempt to alter a permanent object.	*
#define IE_DEALLOCATION	1006 /* Attempt to deallocate a permanent object.	*

6.4.4 Binding-specific errors

Table 13 presents the binding-specific error descriptors.

Table 13 - Binding-specific error descriptors

Error macro name	Code	Error condition
#define IE_START_POINT_INVALID	2201 /* Start index is out of range of data available.	*
#define IE_ENUM_TYPE_INVALID	2202 /* Enumerated type out of range.	*
#define IE_REPOSITORY_UNAVAILABLE	2203 /* Identifier points to non-existent repository object.	*

6.5 Implementation-specific data lengths

Some data collections employ array definitions within their structures, the sizes of which are left to the implementation to define by macro. Values in the following macros are to be revised as needed as a function of the memory allocated by the implementation. A temporary value applicable for minimum length for currently specified contents or possible representation is suggested by the macros in Table 14.

Table 14 - IPI-PIKS data lengths

	Macro Name	Size	Representation
#define	IL_PROFILE	15 /*	conformance profile label string length */
#define	IL_REAL	3 /*	implementation method of RD data type string length */

6.6 Repository object descriptors

Repository data objects defined in ISO/IEC 12087-2 are assigned descriptors which can be specified to the *return_repository_id* function to obtain an IPI-PIKS internal object identifier for access to the data object. Tables 15 through 17 present the three classes of repository data objects. The macro names contain the object names, some words abbreviated, following an "IR_" prefix. The tables are presented in alphabetical order of object names and the descriptors are the object numbers and names presented in ISO/IEC_12087-2.

6.6.1 Repository impulse response function arrays

Table 15 - Repository impulse response function array descriptors

Repository Object Descriptor Name	Code	Repository Object Name
#define IR_BOXCAR_5x5_HORIZONTAL	90	/* boxcar 5x5 horizontal */
#define IR_BOXCAR_5x5_VERTICAL	91	/* boxcar 5x5 vertical */
#define IR_CHEBYSHEV_1	69	/* Chebyshev 1 */
#define IR_CHEBYSHEV_2	70	/* Chebyshev 2 */
#define IR_CHEBYSHEV_3	71	/* Chebyshev 3 */
#define IR_CHEBYSHEV_4	72	/* Chebyshev 4 */
#define IR_CHEBYSHEV_5	73	/* Chebyshev 5 */
#define IR_CHEBYSHEV_6	74	/* Chebyshev 6 */
#define IR_CHEBYSHEV_7	75	/* Chebyshev 7 */
#define IR_CHEBYSHEV_8	76	/* Chebyshev 8 */
#define IR_CHEBYSHEV_9	77	/* Chebyshev 9 */
#define IR_CORNER_NBR_AVERAGE	7	/* corner neighbour average */
#define IR_DECORRELATE	65	/* decorrelate */
#define IR_DELTA_3x3	1	/* delta 3x3 */
#define IR_EIGHT_NBR_AVERAGE	5	/* eight neighbour average */
#define IR_FOUR_NBR_AVERAGE	6	/* four neighbour average */
#define IR_FREI_HORIZONTAL	21	/* Frei horizontal */
#define IR_FREI_VERTICAL	22	/* Frei vertical */
#define IR_HIGH_PASS_1_3x3	8	/* high pass 1 3x3 */
#define IR_HIGH_PASS_2_3x3	9	/* high pass 2 3x3 */
#define IR_HIGH_PASS_3_3x3	10	/* high pass 3 3x3 */
#define IR_KIRSCH_EAST	31	/* Kirsch east */
#define IR_KIRSCH_NORTH	33	/* Kirsch north */
#define IR_KIRSCH_NORTHEAST	32	/* Kirsch northeast */
#define IR_KIRSCH_NORTHWEST	34	/* Kirsch northwest */
#define IR_KIRSCH_SOUTH	37	/* Kirsch south */
#define IR_KIRSCH_SOUTHEAST	38	/* Kirsch southeast */
#define IR_KIRSCH_SOUTHWEST	36	/* Kirsch Southwest */
#define IR_KIRSCH_WEST	35	/* Kirsch west */
#define IR_LAPLACE_1	66	/* Laplacian 1 */
#define IR_LAPLACE_2	67	/* Laplacian 2 */
#define IR_LAPLACE_3	68	/* Laplacian 3 */
#define IR_LAWS_1	78	/* Laws 1 */
#define IR_LAWS_2	79	/* Laws 2 */
#define IR_LAWS_3	80	/* Laws 3 */
#define IR_LAWS_4	81	/* Laws 4 */
#define IR_LAWS_5	82	/* Laws 5 */

Table 15 - Repository impulse response function array descriptors (continued)

Repository Object Descriptor Name	Code	Repository Object Name
#define IR_LAWS_6	83	/* Laws 6 */
#define IR_LAWS_7	84	/* Laws 7 */
#define IR_LAWS_8	85	/* Laws 8 */
#define IR_LAWS_9	86	/* Laws 9 */
#define IR_LOW_PASS_3x3	4	/* Low pass 3x3 */
#define IR_NEVATIA_0	94	/* Nevatia 0 */
#define IR_NEVATIA_120	98	/* Nevatia 120 */
#define IR_NEVATIA_150	99	/* Nevatia 150 */
#define IR_NEVATIA_180	100	/* Nevatia 180 */
#define IR_NEVATIA_210	101	/* Nevatia 210 */
#define IR_NEVATIA_240	102	/* Nevatia 240 */
#define IR_NEVATIA_270	103	/* Nevatia 270 */
#define IR_NEVATIA_30	95	/* Nevatia 30 */
#define IR_NEVATIA_300	104	/* Nevatia 300 */
#define IR_NEVATIA_330	105	/* Nevatia 330 */
#define IR_NEVATIA_60	96	/* Nevatia 60 */
#define IR_NEVATIA_90	97	/* Nevatia 90 */
#define IR_PIXEL_DIFF_HORIZONTAL	11	/* pixel difference horizontal */
#define IR_PIXEL_DIFF_VERTICAL	12	/* pixel difference vertical */
#define IR_PIXEL_STACK	87	/* pixel stack */
#define IR_PREWITT_COMPASS_EAST	23	/* Prewitt compass east */
#define IR_PREWITT_COMPASS_NORTH	25	/* Prewitt compass north */
#define IR_PREWITT_COMPASS_NORTHEAST	24	/* Prewitt compass northeast */
#define IR_PREWITT_COMPASS_NORTHWEST	26	/* Prewitt compass northwest */
#define IR_PREWITT_COMPASS_SOUTH	29	/* Prewitt compass south */
#define IR_PREWITT_COMPASS_SOUTHEAST	30	/* Prewitt compass southeast */
#define IR_PREWITT_COMPASS_SOUTHWEST	28	/* Prewitt compass southwest */
#define IR_PREWITT_COMPASS_WEST	27	/* Prewitt compass west */
#define IR_PREWITT_HORIZONTAL	17	/* Prewitt horizontal */
#define IR_PREWITT_VERTICAL	18	/* Prewitt vertical */
#define IR_PYRAMID_3x3	3	/* pyramid 3x3 */
#define IR_PYRAMID_5x5	89	/* pyramid 5x5 */
#define IR_ROBERTS_3_LEVEL_EAST	39	/* Roberts 3 level east */
#define IR_ROBERTS_3_LEVEL_NORTH	41	/* Roberts 3 level north */
#define IR_ROBERTS_3_LEVEL_NORTHEAST	40	/* Roberts 3 level northeast */
#define IR_ROBERTS_3_LEVEL_NORTHWEST	42	/* Roberts 3 level northwest */
#define IR_ROBERTS_3_LEVEL_SOUTH	45	/* Roberts 3 level south */
#define IR_ROBERTS_3_LEVEL_SOUTHEAST	46	/* Roberts 3 level southeast */
#define IR_ROBERTS_3_LEVEL_SOUTHWEST	44	/* Roberts 3 level southwest */
#define IR_ROBERTS_3_LEVEL_WEST	43	/* Roberts 3 level west */
#define IR_ROBERTS_5_LEVEL_EAST	47	/* Roberts 5 level east */
#define IR_ROBERTS_5_LEVEL_NORTH	49	/* Roberts 5 level north */
#define IR_ROBERTS_5_LEVEL_NORTHEAST	48	/* Roberts 5 level northeast */
#define IR_ROBERTS_5_LEVEL_NORTHWEST	50	/* Roberts 5 level northwest */
#define IR_ROBERTS_5_LEVEL_SOUTH	53	/* Roberts 5 level south */
#define IR_ROBERTS_5_LEVEL_SOUTHEAST	54	/* Roberts 5 level southeast */
#define IR_ROBERTS_5_LEVEL_SOUTHWEST	52	/* Roberts 5 level southwest */

Table 15 - Repository impulse response function array descriptors (concluded)

Repository Object Descriptor Name	Code	Repository Object Name
#define IR_ROBERTS_5_LEVEL_WEST	51	/* Roberts 5 level west */
#define IR_ROBERTS_HORIZONTAL	15	/* Roberts horizontal */
#define IR_ROBERTS_VERTICAL	16	/* Roberts vertical */
#define IR_SEP_PIXEL_DIFF_HORIZONTAL	13	/* separated pixel difference horizontal */
#define IR_SEP_PIXEL_DIFF_VERTICAL	14	/* separated pixel difference vertical */
#define IR_SOBEL_HORIZONTAL	19	/* Sobel horizontal */
#define IR_SOBEL_VERTICAL	20	/* Sobel vertical */
#define IR_SPOT_4_CONNECTED	63	/* spot 4 connected */
#define IR_SPOT_8_CONNECTED	64	/* spot 8 connected */
#define IR_TRUNC_PYRAMID_5x5_HORIZONTAL	92	/* truncated pyramid 5x5 horizontal */
#define IR_TRUNC_PYRAMID_5x5_VERTICAL	93	/* truncated pyramid 5x5 vertical */
#define IR_UNIFORM_3x3	2	/* uniform 3x3 */
#define IR_UNIFORM_5x5	88	/* uniform 5x5 */
#define IR_UNWEIGHTED_LINE_1	55	/* unweighted line 1 */
#define IR_UNWEIGHTED_LINE_2	56	/* unweighted line 2 */
#define IR_UNWEIGHTED_LINE_3	57	/* unweighted line 3 */
#define IR_UNWEIGHTED_LINE_4	58	/* unweighted line 4 */
#define IR_WEIGHTED_LINE_1	59	/* weighted line 1 */
#define IR_WEIGHTED_LINE_2	60	/* weighted line 2 */
#define IR_WEIGHTED_LINE_3	61	/* weighted line 3 */
#define IR_WEIGHTED_LINE_4	62	/* weighted line 4 */

6.6.2 Repository dither arrays

Table 16 - Repository dither array indices

Repository Object Descriptor Name	Code	Repository Object Name
#define IR_DITHER_2x2	1	/* dither 2x2 */
#define IR_DITHER_4x4	2	/* dither 4x4 */
#define IR_DITHER_8x8	3	/* dither 8x8 */
#define IR_DITHER_16x16	4	/* dither 16x16 */

6.6.3 Repository colour conversion matrices

Table 17 - Repository colour conversion matrix indices

Repository Object Descriptor Name	Code	Repository Object Name
#define IR_A_LIN_CCIR_D65_CIE_E_RGB	73	/* appearance match - linear CCIR illuminant D65 RGB to linear CIE illuminant E RGB */
#define IR_A_LIN_CCIR_D65_EBU_C_RGB	74	/* appearance match - linear CCIR illuminant D65 RGB to linear EBU illuminant C RGB */
#define IR_A_LIN_CCIR_D65_NTSC_C_RGB	75	/* appearance match - linear CCIR illuminant D65 RGB to linear NTSC illuminant C RGB */
#define IR_A_LIN_CIE_E_CCIR_D65_RGB	76	/* appearance match - linear CIE illuminant E RGB to linear CCIR illuminant D65 RGB */

Table 17 - Repository colour conversion matrix indices (continued)

Repository Object Descriptor Name	Code	Repository Object Name
#define IR_A_LIN_CIE_E_EBU_C_RGB	77	/* appearance match - linear CIE illuminant E RGB to linear EBU illuminant C RGB *
#define IR_A_LIN_CIE_E_EBU_D65_RGB	78	/* appearance match - linear CIE illuminant E RGB to linear EBU illuminant D65 RGB *
#define IR_A_LIN_CIE_E_NTSC_C_RGB	79	/* appearance match - linear CIE illuminant E RGB to linear NTSC illuminant C RGB *
#define IR_A_LIN_CIE_E_NTSC_D65_RGB	80	/* appearance match - linear CIE illuminant E RGB to linear NTSC illuminant D65 RGB *
#define IR_A_LIN_CIE_E_SMPTE_D65_RGB	81	/* appearance match - linear CIE illuminant E RGB to linear SMPTE illuminant D65 RGB *
#define IR_A_LIN_EBU_C_CCIR_D65_RGB	82	/* appearance match - linear EBU illuminant C RGB to linear CCIR illuminant C65 RGB *
#define IR_A_LIN_EBU_C_CIE_E_RGB	83	/* appearance match - linear EBU illuminant C RGB to linear CIE illuminant E RGB *
#define IR_A_LIN_EBU_C_EBU_D65_RGB	84	/* appearance match - linear EBU illuminant C RGB to linear EBU illuminant D65 RGB *
#define IR_A_LIN_EBU_C_NTSC_D65_RGB	85	/* appearance match - linear EBU illuminant C RGB to linear NTSC illuminant D65 RGB *
#define IR_A_LIN_EBU_C_SMPTE_D65_RGB	86	/* appearance match - linear EBU illuminant C RGB to linear SMPTE illuminant D65 RGB *
#define IR_A_LIN_EBU_D65_CIE_E_RGB	87	/* appearance match - linear EBU illuminant D65 RGB to linear CIE illuminant E RGB *
#define IR_A_LIN_EBU_D65_EBU_C_RGB	88	/* appearance match - linear EBU illuminant D65 RGB to linear EBU illuminant C RGB *
#define IR_A_LIN_EBU_D65_NTSC_C_RGB	89	/* appearance match - linear EBU illuminant D65 RGB to linear NTSC illuminant C RGB *
#define IR_A_LIN_NTSC_C_CCIR_D65_RGB	90	/* appearance match - linear NTSC illuminant C RGB to linear CCIR illuminant D65 RGB *
#define IR_A_LIN_NTSC_C_CIE_E_RGB	91	/* appearance match - linear NTSC illuminant C RGB to linear CIE illuminant E RGB *
#define IR_A_LIN_NTSC_C_EBU_D65_RGB	92	/* appearance match - linear NTSC illuminant C RGB to linear EBU illuminant D65 RGB *
#define IR_A_LIN_NTSC_C_NTSC_D65_RGB	93	/* appearance match - linear NTSC illuminant C RGB to linear NTSC illuminant D65 RGB *
#define IR_A_LIN_NTSC_C_SMPTE_D65_RGB	94	/* appearance match - linear NTSC illuminant C RGB to linear SMPTE illuminant D65 RGB *
#define IR_A_LIN_NTSC_D65_CIE_E_RGB	95	/* appearance match - linear NTSC illuminant D65 RGB to linear CIE illuminant E RGB *
#define IR_A_LIN_NTSC_D65_EBU_C_RGB	96	/* appearance match - linear NTSC illuminant D65 RGB to linear EBU illuminant C RGB *
#define IR_A_LIN_NTSC_D65_NTSC_C_RGB	97	/* appearance match - linear NTSC illuminant D65 RGB to linear NTSC illuminant C RGB *
#define IR_A_LIN_SMPTE_D65_CIE_E_RGB	98	/* appearance match - linear SMPTE illuminant D65 RGB to linear CIE illuminant E RGB *
#define IR_A_LIN_SMPTE_D65_EBU_C_RGB	99	/* appearance match - linear SMPTE illuminant D65 RGB to linear EBU illuminant C RGB *

Table 17 - Repository colour conversion matrix indices (continued)

Repository Object Descriptor Name	Code	Repository Object Name
#define IR_A_LIN_SMPTE_D65_NTSC_C_RGB	100	/* appearance match - linear SMPTE illuminant D65 RGB to linear NTSC illuminant C RGB */
#define IR_C_LIN_CCIR_D65_CIE_E_RGB	1	/* colourmetric match - linear CCIR illuminant D65 RGB to linear CIE illuminant E RGB */
#define IR_C_LIN_CCIR_D65_EBU_C_RGB	2	/* colourmetric match - linear CCIR illuminant D65 RGB to linear EBU illuminant C RGB */
#define IR_C_LIN_CCIR_D65_EBU_D65_RGB	3	/* colourmetric match - linear CCIR illuminant D65 RGB to linear EBU illuminant D65 RGB */
#define IR_C_LIN_CCIR_D65_NTSC_C_RGB	4	/* colourmetric match - linear CCIR illuminant D65 RGB to linear NTSC illuminant C RGB */
#define IR_C_LIN_CCIR_D65_NTSC_D65_RGB	5	/* colourmetric match - linear CCIR illuminant D65 RGB to linear NTSC illuminant D65 RGB */
#define IR_C_LIN_CCIR_D65_SMPTE_D65_RGB	6	/* colourmetric match - linear CCIR illuminant D65 RGB to linear SMPTE illuminant D65 RGB */
#define IR_C_LIN_CCIR_D65_RGB_UVW	7	/* colourmetric match - linear CCIR illuminant D65 RGB to UVW */
#define IR_C_LIN_CCIR_D65_RGB_XYZ	8	/* colourmetric match - linear CCIR illuminant D65 RGB to XYZ */
#define IR_C_LIN_CIE_E_CCIR_D65_RGB	9	/* colourmetric match - linear CIE illuminant E RGB to linear CCIR illuminant D65 RGB */
#define IR_C_LIN_CIE_E_EBU_C_RGB	10	/* colourmetric match - linear CIE illuminant E RGB to linear EBU illuminant C RGB */
#define IR_C_LIN_CIE_E_EBU_D65_RGB	11	/* colourmetric match - linear CIE illuminant E RGB to linear EBU illuminant D65 RGB */
#define IR_C_LIN_CIE_E_NTSC_C_RGB	12	/* colourmetric match - linear CIE illuminant E RGB to linear NTSC illuminant C RGB */
#define IR_C_LIN_CIE_E_NTSC_D65_RGB	13	/* colourmetric match - linear CIE illuminant E RGB to linear NTSC illuminant D65 RGB */
#define IR_C_LIN_CIE_E_SMPTE_D65_RGB	14	/* colourmetric match - linear CIE illuminant E RGB to linear SMPTE illuminant D65 RGB */
#define IR_C_LIN_CIE_E_RGB_UVW	15	/* colourmetric match - linear CIE illuminant E RGB to UVW */
#define IR_C_LIN_CIE_E_RGB_XYZ	16	/* colourmetric match - linear CIE illuminant E RGB to XYZ */
#define IR_C_LIN_EBU_C_CCIR_D65_RGB	17	/* colourmetric match - linear EBU illuminant C RGB to linear CCIR illuminant D65 RGB */
#define IR_C_LIN_EBU_C_CIE_E_RGB	18	/* colourmetric match - linear EBU illuminant C RGB to linear CIE illuminant E RGB */
#define IR_C_LIN_EBU_C_EBU_D65_RGB	19	/* colourmetric match - linear EBU illuminant C RGB to linear EBU illuminant D65 RGB */
#define IR_C_LIN_EBU_C_NTSC_C_RGB	20	/* colourmetric match - linear EBU illuminant C RGB to linear NTSC illuminant C RGB */
#define IR_C_LIN_EBU_C_NTSC_D65_RGB	21	/* colourmetric match - linear EBU illuminant C RGB to linear NTSC illuminant D65 RGB */
#define IR_C_LIN_EBU_C_SMPTE_D65_RGB	22	/* colourmetric match - linear EBU illuminant C RGB to linear SMPTE illuminant D65 RGB */

Table 17 - Repository colour conversion matrix indices (continued)

Repository Object Descriptor Name	Code	Repository Object Name
#define IR_C_LIN_EBU_C_RGB_UVW	23	/* colourmetric match - linear EBU illuminant C RGB to UVW *
#define IR_C_LIN_EBU_C_RGB_XYZ	24	/* colourmetric match - linear EBU illuminant C RGB to XYZ *
#define IR_C_LIN_EBU_D65_CCIR_D65_RGB	25	/* colourmetric match - linear EBU illuminant D65 RGB to linear CCIR illuminant D65 RGB *
#define IR_C_LIN_EBU_D65_CIE_E_RGB	26	/* colourmetric match - linear EBU illuminant D65 RGB to linear CIE illuminant E RGB *
#define IR_C_LIN_EBU_D65_EBU_C_RGB	27	/* colourmetric match - linear EBU illuminant D65 RGB to linear EBU illuminant C RGB *
#define IR_C_LIN_EBU_D65_NTSC_C_RGB	28	/* colourmetric match - linear EBU illuminant D65 RGB to linear NTSC illuminant C RGB *
#define IR_C_LIN_EBU_D65_NTSC_D65_RGB	29	/* colourmetric match - linear EBU illuminant D65 RGB to linear NTSC illuminant D65 RGB *
#define IR_C_LIN_EBU_D65_SMPTE_D65_RGB	30	/* colourmetric match - linear EBU illuminant D65 RGB to linear SMPTE illuminant D65 RGB *
#define IR_C_LIN_EBU_D65_RGB_UVW	31	/* colourmetric match - linear EBU illuminant D65 RGB to UVW *
#define IR_C_LIN_EBU_D65_RGB_XYZ	32	/* colourmetric match - linear EBU illuminant D65 RGB to XYZ *
#define IR_C_LIN_NTSC_C_CCIR_D65_RGB	33	/* colourmetric match - linear NTSC illuminant C RGB to linear CCIR illuminant D65 RGB *
#define IR_C_LIN_NTSC_C_CIE_E_RGB	34	/* colourmetric match - linear NTSC illuminant C RGB to linear CIE illuminant E RGB *
#define IR_C_LIN_NTSC_C_EBU_C_RGB	35	/* colourmetric match - linear NTSC illuminant C RGB to linear EBU illuminant C RGB *
#define IR_C_LIN_NTSC_C_EBU_D65_RGB	36	/* colourmetric match - linear NTSC illuminant C RGB to linear EBU illuminant D65 RGB *
#define IR_C_LIN_NTSC_C_NTSC_D65_RGB	37	/* colourmetric match - linear NTSC illuminant C RGB to linear NTSC illuminant D65 RGB *
#define IR_C_LIN_NTSC_C_SMPTE_D65_RGB	38	/* colourmetric match - linear NTSC illuminant C RGB to linear SMPTE illuminant D65 RGB *
#define IR_C_LIN_NTSC_C_RGB_UVW	39	/* colourmetric match - linear NTSC illuminant C RGB to UVW *
#define IR_C_LIN_NTSC_C_RGB_XYZ	40	/* colourmetric match - linear NTSC illuminant C RGB to XYZ *
#define IR_C_LIN_NTSC_D65_CCIR_D65_RGB	41	/* colourmetric match - linear NTSC illuminant D65 RGB to linear CCIR illuminant D65 RGB *
#define IR_C_LIN_NTSC_D65_CIE_E_RGB	42	/* colourmetric match - linear NTSC illuminant D65 RGB to linear CIE illuminant E RGB *
#define IR_C_LIN_NTSC_D65_EBU_C_RGB	43	/* colourmetric match - linear NTSC illuminant D65 RGB to linear EBU illuminant C RGB *
#define IR_C_LIN_NTSC_D65_EBU_D65_RGB	44	/* colourmetric match - linear NTSC illuminant D65 RGB to linear EBU illuminant D65 RGB *
#define IR_C_LIN_NTSC_D65_NTSC_C_RGB	45	/* colourmetric match - linear NTSC illuminant D65 RGB to linear NTSC illuminant C RGB *

Table 17 - Repository colour conversion matrix indices (continued)

Repository Object Descriptor Name	Code	Repository Object Name
#define IR_C_LIN_NTSC_D65_SMPTE_D65_RGB	46	/* colourmetric match - linear NTSC illuminant D65 RGB to linear SMPTE illuminant D65 RGB */
#define IR_C_LIN_NTSC_D65_RGB_UVW	47	/* colourmetric match - linear NTSC illuminant D65 RGB to UVW */
#define IR_C_LIN_NTSC_D65_RGB_XYZ	48	/* colourmetric match - linear NTSC illuminant D65 RGB to XYZ */
#define IR_C_LIN_SMPTE_D65_CCIR_D65_RGB	49	/* colourmetric match - linear SMPTE illuminant D65 RGB to linear CCIR illuminant D65 RGB */
#define IR_C_LIN_SMPTE_D65_CIE_E_RGB	50	/* colourmetric match - linear SMPTE illuminant D65 RGB to linear CIE illuminant E RGB */
#define IR_C_LIN_SMPTE_D65_EBU_C_RGB	51	/* colourmetric match - linear SMPTE illuminant D65 RGB to linear EBU illuminant C RGB */
#define IR_C_LIN_SMPTE_D65_EBU_D65_RGB	52	/* colourmetric match - linear SMPTE illuminant D65 RGB to linear EBU illuminant D65 RGB */
#define IR_C_LIN_SMPTE_D65_NTSC_C_RGB	53	/* colourmetric match - linear SMPTE illuminant D65 RGB to linear NTSC illuminant C RGB */
#define IR_C_LIN_SMPTE_D65_NTSC_D65_RGB	54	/* colourmetric match - linear SMPTE illuminant D65 RGB to linear NTSC illuminant D65 RGB */
#define IR_C_LIN_SMPTE_D65_RGB_UVW	55	/* colourmetric match - linear SMPTE illuminant D65 RGB to UVW */
#define IR_C_LIN_SMPTE_D65_RGB_XYZ	56	/* colourmetric match - linear SMPTE illuminant D65 RGB to XYZ */
#define IR_C_UVW_LIN_CCIR_D65_RGB	57	/* colourmetric match - UVW to linear CCIR illuminant D65 RGB */
#define IR_C_UVW_LIN_CIE_E_RGB	58	/* colourmetric match - UVW to linear CIE illuminant E RGB */
#define IR_C_UVW_LIN_EBU_C_RGB	59	/* colourmetric match - UVW to linear EBU illuminant C RGB */
#define IR_C_UVW_LIN_EBU_D65_RGB	60	/* colourmetric match - UVW to linear EBU illuminant D65 RGB */
#define IR_C_UVW_LIN_NTSC_C_RGB	61	/* colourmetric match - UVW to linear NTSC illuminant C RGB */
#define IR_C_UVW_LIN_NTSC_D65_RGB	62	/* colourmetric match - UVW to linear NTSC illuminant D65 RGB */
#define IR_C_UVW_LIN_SMPTE_D65_RGB	63	/* colourmetric match - UVW to linear SMPTE illuminant D65 RGB */
#define IR_C_UVW_XYZ	64	/* colourmetric match - UVW to XYZ */
#define IR_C_XYZ_LIN_CCIR_D65_RGB	65	/* colourmetric match - XYZ to linear CCIR illuminant D65 RGB */
#define IR_C_XYZ_LIN_CIE_E_RGB	66	/* colourmetric match - XYZ to linear CIE illuminant E RGB */
#define IR_C_XYZ_LIN_EBU_C_RGB	67	/* colourmetric match - XYZ to linear EBU illuminant C RGB */
#define IR_C_XYZ_LIN_EBU_D65_RGB	68	/* colourmetric match - XYZ to linear EBU illuminant D65 RGB */

Table 17 - Repository colour conversion matrix indices (concluded)

Repository Object Descriptor Name	Code	Repository Object Name
#define IR_C_XYZ_LIN_NTSC_C_RGB	69	/* colourmetric match - XYZ to linear NTSC illuminant C RGB *
#define IR_C_XYZ_LIN_NTSC_D65_RGB	70	/* colourmetric match - XYZ to linear NTSC illuminant D65 RGB *
#define IR_C_XYZ_LIN_SMPTE_D65_RGB	71	/* colourmetric match - XYZ to linear SMPTE illuminant D65 RGB *
#define IR_C_XYZ_UVW	72	/* colourmetric match - XYZ to UVW *
#define IR_GAMMA_EBU_C_RGB_YUV	101	/* gamma EBU illuminant C RGB to EBU illuminant C YUV *
#define IR_GAMMA_EBU_D65_RGB_YUV	102	/* gamma EBU illuminant D65 RGB to EBU illuminant D65 YUV *
#define IR_GAMMA_NTSC_C_RGB_YIQ	103	/* gamma NTSC illuminant C RGB to NTSC illuminant C YIQ *
#define IR_GAMMA_NTSC_D65_RGB_YIQ	104	/* gamma NTSC illuminant D65 RGB to NTSC illuminant D65 YIQ *
#define IR_GAMMA_SMPTE_D65_RGB_YCBCR	105	/* gamma SMPTE illuminant D65 RGB to SMPTE illuminant D65 YCbCr *
#define IR_YUV_GAMMA_EBU_C_RGB	106	/* EBU illuminant C YUV to gamma EBU illuminant C RGB *
#define IR_YUV_GAMMA_EBU_D65_RGB	107	/* EBU illuminant D65 YUV to gamma EBU illuminant D65 RGB *
#define IR_YIQ_GAMMA_NTSC_C_RGB	108	/* NTSC illuminant C YIQ to gamma NTSC illuminant C RGB *
#define IR_YIQ_GAMMA_NTSC_D65_RGB	109	/* NTSC illuminant D65 YIQ to gamma NTSC illuminant D65 RGB *
#define IR_YCBCR_GAMMA_SMPTE_D65_RGB	110	/* SMPTE illuminant D65 YCbCr to gamma SMPTE illuminant D65 RGB *
#define IR_ILLUMINANT_C_D50	111	/* illuminant C XYZ to illuminant D50 XYZ *
#define IR_ILLUMINANT_C_D65	112	/* illuminant C XYZ to illuminant D65 XYZ *
#define IR_ILLUMINANT_C_E	113	/* illuminant C XYZ to illuminant E XYZ *
#define IR_ILLUMINANT_D50_C	114	/* illuminant D50 XYZ to illuminant C XYZ *
#define IR_ILLUMINANT_D50_D65	115	/* illuminant D50 XYZ to illuminant D65 XYZ *
#define IR_ILLUMINANT_D50_E	116	/* illuminant D50 XYZ to illuminant E XYZ *
#define IR_ILLUMINANT_D65_C	117	/* illuminant D65 XYZ to illuminant C XYZ *
#define IR_ILLUMINANT_D65_D50	118	/* illuminant D65 XYZ to illuminant D50 XYZ *
#define IR_ILLUMINANT_D65_E	119	/* illuminant D65 XYZ to illuminant E XYZ *
#define IR_ILLUMINANT_E_C	120	/* illuminant E XYZ to illuminant C XYZ *
#define IR_ILLUMINANT_E_D50	121	/* illuminant E XYZ to illuminant D50 XYZ *
#define IR_ILLUMINANT_E_D65	122	/* illuminant E XYZ to illuminant D65 XYZ *

7 C functional elements

7.1 Notational conventions

The binding of each function utilizes the following template:

Function Name

I(prefix1)return I(prefix2)Function (	/* return value explanation	*/
I_type0 prefixArg0,	/* argument 0 explanation	*/
I_type1 prefixArg1,	/* argument 1 explanation	*/
....	...	
I_typeN prefixArgN	/* last argument explanation	*/

);

"Function Name" is the name of the function as listed in the functional specification ISO/IEC 12087 or as added in this part of ISO/IEC 12088.

The "I(prefix1)return" return function parameter type is most often one of the identifier types defined in 5.2.3. Some exceptions include void, basic data type, pointer, structure or pointer to a structure. If a return parameter is an identifier, it is either a PIKS internal or application data identifier (parametric identifier), the delineation of which is denoted by the prefix. The return generally identifies internal or external destination data produced by the function. Exceptions will be noted by the comment which accompanies the line containing the function name in the function prototype.

The "I(prefix2)Function" label is the function name bound to the function with basic data type, array, identifier, pointer, structure and/or void identification included in the prefix. When a pointer or structure, additional prefix notation may identify the type of data indicated by the pointer or contained in the structure, if the latter type contents are homogeneous. The "I" is the sentinel of the function label.

The "prefixArg0", "prefixArg1", ..., "prefixArgN" are the arguments to the function and correspond to the parameters of the function as presented in the functional specification definition with type specified with the prefix "I_type". For each argument, its prefix identifies type and construct information such as array, basic data type, enumerated, structure and/or pointer.

To the right of each return value and argument declaration is a C comment field which contains a brief explanation of the return value itself or the argument itself. If the comment begins with "OUT", it means the implementation returns data to the application as directed by such argument. All non-void return values represent "OUT" data. Arguments without "OUT" are parameters input in the function call.

7.2 IPI-PIKS functional element prototypes

abort_asynchronous_execution

```
void IvAbortAsynchExecution (                /* no return value output */
    void                                /* no input arguments */
);
```

accumulator

```
Ipfloat *ItraAccumulator (                /* OUT sum of pixels data pointer */
    Idnimage                nSourceImage, /* source image identifier */
    Ipfloat                *traSum,        /* sum of pixels data pointer */
    Ipeprocess            eMode,          /* processing mode option */
    Ipint                iApplBufferSize, /* data transfer buffer size or option */
    Ipuint                uStartPoint,    /* index offset within internal data */
    Ipuint                *tuDataSize     /* OUT post transfer data size pointer */
);
```

adaptive_histogram_equalization

```
Idnimage InAdaptiveHistEqual (            /* OUT destination image identifier */
    Idnimage                nSourceImage, /* source image identifier */
    Idnimage                nDestImage,   /* destination image identifier */
    Ipuint                uPoints,        /* number of histogram reference points/index */
    Ipuint                uWindowSize,    /* window size */
    Ipuint                uOffset        /* number of lookup table source offset entries */
);
```

adaptive_histogram_equalization_vreg

```
Idnvreg InAdaptiveHistEqualVReg (        /* OUT destination image identifier */
    Idnvreg                nVReg,         /* output virtual register identifier or NULL */
    Ipsimage_vreg          snSourceImage, /* source image identifier */
    Ipsimage_vreg          snDestImage,   /* destination image identifier */
    Ipsuint_vreg_chc       sPoints,       /* number of histogram reference points/index */
    Ipsuint_vreg_chc       sWindowSize,   /* window size */
    Ipsuint_vreg_chc       sOffset        /* number of lookup table source offset entries */
);
```

allocate_chain

```
Idnchain InAllocateChain (                /* OUT chain identifier */
    void                                /* no input arguments */
);
```

allocate_chain_vreg

```
Idnvreg InAllocateChainVReg (          /* OUT chain identifier */
    Idnvreg                          nVReg /* output virtual register identifier */
);
```

allocate_composite_identifier_array

```
Idnid_array InAllocateIdArray (          /* OUT composite identifier array identifier */
    Ipsattrs_id_array                  sAttrs /* composite identifier array attributes */
);
```

allocate_composite_identifier_array_vreg

```
Idnvreg InAllocateIdArrayVReg (          /* OUT composite identifier array identifier */
    Idnvreg                          nVReg, /* output virtual register identifier */
    Ipsattrs_id_array_vreg_chc      sAttrs /* composite identifier array attributes */
);
```

allocate_composite_identifier_list

```
Idnid_list InAllocateIdList (          /* OUT composite identifier list identifier */
    void                             /* no input arguments */
);
```

allocate_composite_identifier_list_vreg

```
Idnvreg InAllocateIdListVReg (          /* OUT composite identifier list identifier */
    Idnvreg                          nVReg /* output virtual register identifier */
);
```

allocate_composite_identifier_record

```
Idnid_record InAllocateIdRecord (          /* OUT composite identifier record identifier */
    Ipsattrs_id_record                  sAttrs /* composite identifier record attributes */
);
```

allocate_composite_identifier_record_vreg

```
Idnvreg InAllocateIdRecordVReg (          /* OUT composite identifier record identifier */
    Idnvreg                          nVReg, /* output virtual register identifier */
    Ipsattrs_id_record_vreg_chc      sAttrs /* composite id record attributes */
);
```

allocate_histogram

```
Idnhist InAllocateHist (
    Ipsattrs_hist          sAttrs          /* OUT histogram identifier */
                                   /* histogram attributes */
);
```

allocate_histogram_vreg

```
Idnvreg InAllocateHistVReg (
    Idnvreg          nVReg,          /* OUT histogram identifier */
    Ipsattrs_hist_vreg_chc          /* output virtual register identifier */
                                   /* histogram attributes */
);
```

allocate_image

```
Idnimage InAllocateImage (
    Ipsattrs_image          sAttrs          /* OUT image identifier */
                                   /* image attributes */
);
```

allocate_image_vreg

```
Idnvreg InAllocateImageVReg (
    Idnvreg          nVReg,          /* OUT image identifier */
    Ipsattrs_image_vreg_chc          /* output virtual register identifier */
                                   /* image attributes */
);
```

allocate_lookup_table

```
Idnlut InAllocateLUT (
    Ipsattrs_lut          sAttrs          /* OUT lookup table identifier */
                                   /* lookup table attributes */
);
```

allocate_lookup_table_vreg

```
Idnvreg InAllocateLUTVReg (
    Idnvreg          nVReg,          /* OUT lookup table identifier */
    Ipsattrs_lut_vreg_chc          /* output virtual register identifier */
                                   /* lookup table attributes */
);
```

allocate_matrix

```
Idnmatrix InAllocateMatrix (
    Ipsattrs_matrix          sAttrs          /* OUT matrix identifier */
                                   /* matrix attributes */
);
```

allocate_matrix_vreg

```
Idnvreg InAllocateMatrixVReg (          /* OUT matrix identifier */
    Idnvreg          nVReg,              /* output virtual register identifier */
    Ipsattrs_matrix_vreg_chc  sAttrs     /* matrix attributes */
);
```

allocate_neighbourhood_array

```
Idnnbhood InAllocateNbhoodArray (          /* OUT neighbourhood array identifier */
    Ipsattrs_nbhood          sAttrs         /* neighbourhood array attributes */
);
```

allocate_neighbourhood_array_vreg

```
Idnvreg InAllocateNbhoodArrayVReg (          /* OUT neighbourhood array identifier */
    Idnvreg          nVReg,              /* output virtual register identifier */
    Ipsattrs_nbhood_vreg_chc  sAttrs     /* neighbourhood array attributes */
);
```

allocate_pixel_record

```
Idnpixel_record InAllocatePixelRecord (          /* OUT pixel record identifier */
    Ipsattrs_pixel_record          sAttrs         /* pixel record attributes */
);
```

allocate_pixel_record_vreg

```
Idnvreg InAllocatePixelRecordVReg (          /* OUT pixel record identifier */
    Idnvreg          nVReg,              /* output virtual register identifier */
    Ipsattrs_pixel_record_vreg_chc  sAttrs     /* pixel record attributes */
);
```

allocate_roi

```
Idnroi InAllocateROI (          /* OUT ROI identifier */
    Ipsattrs_roi          sAttrs         /* ROI attributes */
);
```

allocate_roi_vreg

```
Idnvreg InAllocateROIvReg (          /* OUT ROI identifier          */
    Idnvreg          nVReg,          /* output virtual register identifier */
    Ipsattrs_roi_vreg_chc  sAttrs    /* ROI attributes                */
);
```

allocate_static_array

```
Idnstatic InAllocateStaticArray (          /* OUT static array identifier */
    Ipsattrs_static          sAttrs    /* static array attributes      */
);
```

allocate_static_array_vreg

```
Idnvreg InAllocateStaticArrayVReg (          /* OUT static array identifier */
    Idnvreg          nVReg,          /* output virtual register identifier */
    Ipsattrs_static_vreg_chc  sAttrs    /* static array attributes        */
);
```

allocate_tuple

```
Idntuple InAllocateTuple (          /* OUT tuple identifier */
    Ipsattrs_tuple          sAttrs    /* tuple attributes       */
);
```

allocate_tuple_vreg

```
Idnvreg InAllocateTupleVReg (          /* OUT tuple identifier */
    Idnvreg          nVReg,          /* output virtual register identifier */
    Ipsattrs_tuple_vreg_chc  sAttrs    /* tuple attributes      */
);
```

allocate_value_bounds

```
Idnbounds InAllocateValueBounds (          /* OUT value bounds identifier */
    Ipsattrs_bounds          sAttrs    /* value bounds collection attributes */
);
```

allocate_value_bounds_vreg

```
Idnvreg InAllocateValueBoundsVReg (          /* OUT value bounds identifier */
    Idnvreg          nVReg,                  /* output virtual register identifier */
    Ipsattrs_bounds_vreg_chc    sAttrs      /* value bounds collection attributes */
);
```

allocate_vreg

```
Idnvreg InAllocateVReg (          /* OUT virtual register identifier */
    Ipsattrs_vreg          sAttrs      /* virtual register attributes */
);
```

allocate_vreg_vreg

```
Idnvreg InAllocateVRegVReg (      /* OUT virtual register identifier */
    Idnvreg          nId,          /* output virtual register identifier */
    Ipsattrs_vreg_vreg_chc    sAttrs      /* virtual register attributes */
);
```

alpha_blend_constant

```
Idnimage InAlphaBlendConstant (    /* OUT destination image identifier */
    Idnimage          nSourceImage1, /* first input source image identifier */
    Idnimage          nSourceImage2, /* second input source image identifier */
    Idnimage          nDestImage,    /* destination image identifier */
    Ipfloat           rAlphaFactor   /* alpha factor */
);
```

alpha_blend_constant_vreg

```
Idnvreg InAlphaBlendConstantVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage1, /* first input source image identifier */
    Ipsimage_vreg    snSourceImage2, /* second input source image identifier */
    Ipsimage_vreg    snDestImage,    /* destination image identifier */
    Ipsfloat_vreg_chc    sAlphaFactor /* alpha factor */
);
```

alpha_blend_variable

```
Idnimage InAlphaBlendVariable (                               /* OUT destination image identifier */
    Idnimage          nSourceImage1, /* first input source image identifier */
    Idnimage          nSourceImage2, /* second input source image identifier */
    Idnimage          nSourceImage3, /* third input source image identifier */
    Idnimage          nSourceImage4, /* fourth input source image identifier */
    Idnimage          nDestImage     /* destination image identifier */
);
```

alpha_blend_variable_vreg

```
Idnvreg InAlphaBlendVariableVReg (                               /* OUT destination image identifier */
    Idnvreg          nVReg,      /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage1, /* first input source image identifier */
    Ipsimage_vreg    snSourceImage2, /* second input source image identifier */
    Ipsimage_vreg    snSourceImage3, /* third input source image identifier */
    Ipsimage_vreg    snSourceImage4, /* fourth input source image identifier */
    Ipsimage_vreg    snDestImage   /* destination image identifier */
);
```

array_to_lut

```
Idnlut InArrayToLUT (                               /* OUT lookup table identifier */
    Idnlut          nLUT,      /* lookup table identifier */
    Ipsparameter_pixel stArray, /* lookup table value array pointer union */
    Ipuint          nEntryOffset, /* lookup table entry offset */
    Ipuint          uBandOffset,  /* lookup table band offset */
    Ipuint          uColumns,     /* number of array columns */
    Ipuint          uRows        /* number of array rows */
);
```

array_to_lut_vreg

```
Idnvreg InArrayToLUTVReg (                               /* OUT lookup table identifier */
    Idnvreg          nVReg,      /* output virtual register identifier or NULL */
    Ipslut_vreg      snLUT,      /* lookup table identifier */
    Ipsparameter_pixel_vreg_chc sArray, /* lookup table value array */
    Ipsuint_vreg_chc sEntryOffset, /* lookup table entry offset */
    Ipsuint_vreg_chc sBandOffset,  /* lookup table band offset */
    Ipsuint_vreg_chc sColumns,     /* number of array columns */
    Ipsuint_vreg_chc sRows        /* number of array rows */
);
```

bind_match_point

```
Idnimage InBindMatchPoint (
    Idnimage          nSourceImage, /* OUT source image identifier */
    Idntuple          nMatch         /* source image identifier      */
    /* match point RD 5-tuple identifier */
);
```

bind_match_point_vreg

```
Idnvreg InBindMatchPointVReg (
    Idnvreg          nVReg, /* OUT source image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipstuple_vreg    snMatch         /* source image identifier      */
    /* match point RD 5-tuple identifier */
);
```

bind_roi

```
Idnimage InBindROI (
    Idnimage          nSourceImage, /* OUT source image identifier */
    Idnroi            nROI,         /* source image identifier      */
    Idntuple          nOffset       /* ROI identifier              */
    /* offset SD 5-tuple identifier */
);
```

bind_roi_vreg

```
Idnvreg InBindROIIVReg (
    Idnvreg          nVReg, /* OUT source image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsroi_vreg      snROI,         /* source image identifier      */
    Ipstuple_vreg    snOffset       /* ROI identifier              */
    /* offset SD 5-tuple identifier */
);
```

bit_shift

```
Idnimage InBitShift (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier      */
    Ipuint            uBits,        /* destination image identifier  */
    Ipint             iOption       /* number of bits to be shifted */
    /* bit shift option            */
);
```

bit_shift_vreg

```
Idnvreg InBitShiftVReg (                                /* OUT destination image identifier */
    Idnvreg                                nVReg,        /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage,   /* destination image identifier */
    Ipsuint_vreg_chc sBits,      /* number of bits to be shifted */
    Ipsint_vreg_chc sOption      /* bit shift option */
);
```

cartesian_to_polar

```
Idnimage InCartesianToPolar (                                /* OUT destination image identifier */
    Idnimage                                nSourceImage, /* source image identifier */
    Idnimage                                nDestImage,   /* destination image identifier */
    Ipfloat                                rMinimumRho,    /* minimum value of rho */
    Ipfloat                                rMaximumRho,    /* maximum value of rho */
    Ipfloat                                rMinimumTheta,  /* minimum value of theta */
    Ipfloat                                rMaximumTheta /* maximum value of theta */
);
```

cartesian_to_polar_vreg

```
Idnvreg InCartesianToPolarVReg (                                /* OUT destination image identifier */
    Idnvreg                                nVReg,        /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage,   /* destination image identifier */
    Ipsfloat_vreg_chc sMinimumRho, /* minimum value of rho */
    Ipsfloat_vreg_chc sMaximumRho, /* maximum value of rho */
    Ipsfloat_vreg_chc sMinimumTheta, /* minimum value of theta */
    Ipsfloat_vreg_chc sMaximumTheta /* maximum value of theta */
);
```

chain_abort

```
Idnchain InChainAbort (                                /* OUT chain identifier */
    Idnchain                                nChain        /* chain identifier */
);
```

chain_abort_vreg

```
Idnvreg InChainAbortVReg (                                /* OUT chain identifier */
    Idnvreg                                nVReg,        /* output virtual register identifier or NULL */
    Ipschain_vreg snVReg                  /* chain identifier */
);
```

chain_begin

Idnchain InChainBegin (		/* OUT chain identifier	*/
Idnchain	nChain	/* chain identifier	*/
);			

chain_delete

Idnchain InChainDelete (		/* OUT chain identifier	*/
Idnchain	nChain	/* chain identifier	*/
);			

chain_delete_vreg

Idnvreg InChainDeleteVReg (		/* OUT chain identifier	*/
Idnvreg	nVReg,	/* output virtual register identifier or NULL	*/
Ipschain_vreg	snVReg	/* chain identifier	*/
);			

chain_end

Idnchain InChainEnd (		/* OUT chain identifier	*/
Idnchain	nChain	/* chain identifier	*/
);			

chain_execute

Idnchain InChainExecute (		/* OUT chain identifier	*/
Idnchain	nChain	/* chain identifier	*/
);			

chain_execute_vreg

Idnvreg InChainExecuteVReg (		/* OUT chain identifier	*/
Idnvreg	nVReg,	/* output virtual register identifier or NULL	*/
Ipschain_vreg	snVReg	/* chain identifier	*/
);			

chain_reload

```
Idnchain InChainReload (
    Idnchain          nChain,          /* OUT chain identifier */
    Ipoint            uElement,        /* chain identifier */
    Ipnarg            nArgs,           /* element selector */
                                /* argument parameter collection identifier */
);
```

chain_reload_vreg

```
Idnvreg InChainReloadVReg (
    Idnvreg          nVReg,          /* OUT chain identifier */
    Ipschain_vreg    snVReg,        /* output virtual register identifier or NULL */
    Ipsuint_vreg_chc sElement,      /* chain identifier */
    Ipsarg_vreg      snArgs,        /* element selector */
                                /* argument parameter collection identifier */
);
```

classifier_bayes

```
Idnimage InClassifierBayes (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Ipfloating         *traProbability, /* destination image identifier */
    Ipfloating         *traMeanVector, /* class a priori probability array pointer */
    Ipfloating         *traCovariance, /* class mean vector array pointer */
    Ipoint            uClasses,      /* class covariance matrix array pointer */
    Ipoint            iApplBufferSize, /* number of classes */
    Ipoint            uStartPoint,   /* data transfer buffer size or option of each array */
    Ipoint            *tuDataSize,   /* index offset within internal data of each array */
                                /* OUT post transfer data size pointer */
);
```

classifier_bayes_vreg

```
Idnvreg InClassifierBayesVReg (
    Idnvreg          nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier */
    Ipsafloat_vreg_chc sProbability, /* destination image identifier */
    Ipsafloat_vreg_chc sMeanVector, /* class a priori probability array */
    Ipsafloat_vreg_chc sCovariance, /* class mean vector array */
    Ipsuint_vreg_chc sClasses,      /* class covariance matrix array */
                                /* number of classes */
);
```

classifier_nearest_neighbour

```
Idnimage InClassifierNearestNbr (          /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    lpsfloat          *traTraining, /* training pattern set array pointer */
    lpuint            uClasses,      /* number of classes */
    lpuint            uPatterns,     /* number of patterns per class */
    lpint             iOption,       /* distance measure option */
    lpint             iApplBufferSize, /* data transfer buffer size or option */
    lpuint            uStartPoint,   /* index offset within internal data */
    lpuint            *tuDataSize    /* OUT post transfer data size pointer */
);
```

classifier_nearest_neighbour_vreg

```
Idnvreg InClassifierNearestNbrVReg (          /* OUT destination image identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    lpsimage_vreg    snSourceImage, /* source image identifier */
    lpsimage_vreg    snDestImage,   /* destination image identifier */
    lpsfloat_vreg_chc sTraining,     /* training pattern set array */
    lpsuint_vreg_chc sClasses,       /* number of classes */
    lpsuint_vreg_chc sPatterns,     /* number of patterns per class */
    lpsint_vreg_chc  sOption        /* distance measure option */
);
```

close_piks

```
void IvClosePIKS (          /* no return value output */
    void                /* no input arguments */
);
```

close_piks_emergency

```
void IvClosePIKSEmergency (          /* no return value output */
    void                /* no input arguments */
);
```

colour_conversion_linear

```
Idnimage InColourConvLin (          /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Idnmatrix         nColourMatrix /* colour conversion matrix identifier or NULL */
);
```

colour_conversion_linear_vreg

```
Idnvreg InColourConvLinVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage, /* destination image identifier */
    Ipsmatrix_vreg   snColourMatrix /* colour conversion matrix identifier or NULL */
);
```

colour_conversion_matrix

```
void IvColourConvMatrix ( /* no return value output */
    Idnmatrix      nXYZMatrixRGB, /* OUT XYZ-RGB conversion matrix identifier */
    Idnmatrix      nRGBMatrixXYZ, /* OUT RGB-XYZ conversion matrix identifier */
    Ipsfloat       rRedXChrom, /* red primary x chromaticity coordinate */
    Ipsfloat       rRedYChrom, /* red primary y chromaticity coordinate */
    Ipsfloat       rGreenXChrom, /* green primary x chromaticity coordinate */
    Ipsfloat       rGreenYChrom, /* green primary y chromaticity coordinate */
    Ipsfloat       rBlueXChrom, /* blue primary x chromaticity coordinate */
    Ipsfloat       rBlueYChrom, /* blue primary y chromaticity coordinate */
    Ipsfloat       rWhiteXChrom, /* white reference x chromaticity coordinate */
    Ipsfloat       rWhiteYChrom /* white reference y chromaticity coordinate */
);
```

colour_conversion_matrix_vreg

```
void IvColourConvMatrixVReg ( /* no return value output */
    Idnvreg      nXYZMatrixRGB, /* OUT XYZ-RGB conversion matrix identifier */
    Idnvreg      nRGBMatrixXYZ, /* OUT RGB-XYZ conversion matrix identifier */
    Ipsfloat_vreg_chc sRedXChrom, /* red primary x chromaticity coordinate */
    Ipsfloat_vreg_chc sRedYChrom, /* red primary y chromaticity coordinate */
    Ipsfloat_vreg_chc sGreenXChrom, /* green primary x chromaticity coordinate */
    Ipsfloat_vreg_chc sGreenYChrom, /* green primary y chromaticity coordinate */
    Ipsfloat_vreg_chc sBlueXChrom, /* blue primary x chromaticity coordinate */
    Ipsfloat_vreg_chc sBlueYChrom, /* blue primary y chromaticity coordinate */
    Ipsfloat_vreg_chc sWhiteXChrom, /* white reference x chromaticity coordinate */
    Ipsfloat_vreg_chc sWhiteYChrom /* white reference y chromaticity coordinate */
);
```

colour_conversion_nonlinear

```
Idnimage InColourConvNonlin (
    Idnimage      nSourceImage, /* OUT destination image identifier */
    Idnimage      nDestImage,   /* source image identifier */
    Idnimage      nDestImage,   /* destination image identifier */
    Ipfloating     rXWhite,      /* X tristimulus value white point or 0 */
    Ipfloating     rYWhite,      /* Y tristimulus value white point or 0 */
    Ipfloating     rZWhite,      /* Z tristimulus value white point or 0 */
    Ipin           iRGBOption,   /* RGB colour option */
    Ipin           iInputOption, /* colour conversion input option */
    Ipin           iOutputOption /* colour conversion output option */
);
```

colour_conversion_nonlinear_vreg

```
Idnvreg InColourConvNonlinVReg (
    Idnvreg      nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg snSourceImage, /* output virtual register identifier of NULL */
    Ipsimage_vreg snDestImage,   /* source image identifier */
    Ipsimage_vreg snDestImage,   /* destination image identifier */
    Ipsfloat_vreg_chc sXWhite,    /* X tristimulus value white point or 0 */
    Ipsfloat_vreg_chc sYWhite,    /* Y tristimulus value white point or 0 */
    Ipsfloat_vreg_chc sZWhite,    /* Z tristimulus value white point or 0 */
    Ipsint_vreg_chc sRGBOption,   /* RGB colour option */
    Ipsint_vreg_chc sInputOption, /* colour conversion input option */
    Ipsint_vreg_chc sOutputOption /* colour conversion output option */
);
```

colour_conversion_subtractive

```
Idnimage InColourConvSubtractive (
    Idnimage      nSourceImage, /* OUT destination image identifier */
    Idnimage      nDestImage,   /* source image identifier */
    Idnimage      nDestImage,   /* destination image identifier */
    Ipfloating     rBlackFactor, /* blackness factor */
    Ipin           uIntegerSize, /* maximum integer size */
    Ipsparameter_arith stMethod, /* method data pointer union */
    Ipin           iDirectionOption, /* colour conversion, direction option */
    Ipin           iMethodOption /* colour conversion, method option */
);
```

colour_conversion_subtractive_vreg

```
Idnvreg InColourConvSubtractiveVReg (          /* OUT destination image identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,  /* source image identifier */
    Ipsimage_vreg    snDestImage,    /* destination image identifier */
    Ipsfloat_vreg_chc sBlackFactor,   /* blackness factor */
    Ipsuint_vreg_chc sIntegerSize,    /* maximum integer size */
    Ipsparameter_arith_vreg_chc sMethod, /* method data */
    Ipsint_vreg_chc  sDirectionOption, /* colour conversion, direction option */
    Ipsint_vreg_chc  sMethodOption    /* colour conversion, method option */
);
```

colour_lookup_interpolated

```
Idnimage InColourLookupInterp (          /* OUT destination image identifier */
    Idnimage          nSourceImage,    /* source image identifier */
    Idnimage          nDestImage,      /* destination image identifier */
    Idnlut            nTable,          /* colour lookup table identifier */
    Idntuple          nCellCount,      /* number of cells ND B-tuple identifier */
    Idntuple          nCellSize,       /* size of cells RD B-tuple identifier */
    Idntuple          nDimensions      /* array dimension ND B-tuple identifier */
);
```

colour_lookup_interpolated_vreg

```
Idnvreg InColourLookupInterpVReg (      /* OUT destination image identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,    /* source image identifier */
    Ipsimage_vreg    snDestImage,      /* destination image identifier */
    Ipslut_vreg       snTable,         /* colour lookup table identifier */
    Ipstuple_vreg     snCellCount,      /* number of cells ND B-tuple identifier */
    Ipstuple_vreg     snCellSize,       /* size of cells RD B-tuple identifier */
    Ipstuple_vreg     snDimensions     /* array dimension ND B-tuple identifier */
);
```

complement

```
Idnimage InComplement (
    Idnimage          nSourceImage,    /* source image identifier */
    Idnimage          nDestImage      /* destination image identifier */
);
```

complement_vreg

```
Idnvreg InComplementVReg (                                /* OUT destination image identifier */
    Idnvreg          nVReg,                                /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,                       /* source image identifier */
    Ipsimage_vreg    snDestImage                          /* destination image identifier */
);
```

complement_roi

```
Idnroi InComplementROI (                                /* OUT destination ROI identifier */
    Idnroi          nSourceROI,                            /* source ROI identifier */
    Idnroi          nDestROI                             /* destination ROI identifier */
);
```

complement_roi_vreg

```
Idnvreg InComplementROIvReg (                          /* OUT destination ROI identifier */
    Idnvreg          nVReg,                                /* output virtual register identifier or NULL */
    Ipsroi_vreg      snSourceROI,                         /* source ROI identifier */
    Ipsroi_vreg      snDestROI                           /* destination ROI identifier */
);
```

complex_composition

```
Idnimage InComplexComposition (                          /* OUT destination image identifier */
    Idnimage          nSourceImage1,                     /* first source image identifier */
    Idnimage          nSourceImage2,                     /* second source image identifier */
    Idnimage          nDestImage,                       /* destination image identifier */
    Ipecomplex_conv    eOption                          /* complex conversion option */
);
```

complex_composition_vreg

```
Idnvreg InComplexCompositionVReg (                      /* OUT destination image identifier */
    Idnvreg          nVReg,                                /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage1,                     /* first source image identifier */
    Ipsimage_vreg    snSourceImage2,                     /* second source image identifier */
    Ipsimage_vreg    snDestImage,                       /* destination image identifier */
    Ipscomplex_conv_vreg_chc sOption                     /* complex conversion option */
);
```

complex_conjugate

```
Idnimage InComplexConjugate (          /* OUT destination image identifier */
    Idnimage          nSourceImage,    /* source image identifier          */
    Idnimage          nDestImage       /* destination image identifier     */
);
```

complex_conjugate_vreg

```
Idnvreg InComplexConjugateVReg (        /* OUT destination image identifier */
    Idnvreg          nVReg,            /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,    /* source image identifier          */
    Ipsimage_vreg    snDestImage       /* destination image identifier     */
);
```

complex_decomposition

```
void IvComplexDecomposition (           /* no return value output          */
    Idnimage          nSourceImage,    /* source image identifier          */
    Idnimage          nDestFirst,      /* OUT first destination image identifier */
    Idnimage          nDestSecond,     /* OUT second destination image identifier */
    Ipscomplex_conv   eOption          /* complex conversion option        */
);
```

complex_decomposition_vreg

```
void IvComplexDecompositionVReg (       /* no return value output          */
    Ipsimage_vreg     snSourceImage,    /* source image identifier          */
    Idnvreg           nDestFirst,      /* OUT first destination image identifier */
    Idnvreg           nDestSecond,     /* OUT second destination image identifier */
    Ipscomplex_conv_vreg_chc sOption    /* complex conversion option        */
);
```

complex_magnitude

```
Idnimage InComplexMagnitude (           /* OUT destination image identifier */
    Idnimage          nSourceImage,    /* source image identifier          */
    Idnimage          nDestImage,      /* destination image identifier     */
    Ipemagnitude      eOption          /* complex magnitude option        */
);
```

complex_magnitude_vreg

```
Idnvreg InComplexMagnitudeVReg (          /* OUT destination image identifier */
    Idnvreg          nVReg,                /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,        /* source image identifier */
    Ipsimage_vreg    snDestImage,         /* destination image identifier */
    Ipsmagnitude_vreg_chc    sOption        /* complex magnitude option */
);
```

composite_identifier_array_equal

```
Ipbool IbIdArrayEqual (          /* OUT equality flag */
    Idnid_array      nFirst,          /* first image identifier array identifier */
    Idnid_array      nSecond         /* second image identifier array identifier */
);
```

composite_identifier_array_equal_vreg

```
Idnvreg InIdArrayEqualVReg (      /* OUT equality flag virtual register identifier */
    Idnvreg          nVReg,          /* output virtual register identifier */
    Ipsid_array_vreg snFirst,        /* first image identifier array identifier */
    Ipsid_array_vreg snSecond       /* second image identifier array identifier */
);
```

composite_identifier_array_get

```
Idnimage InIdArrayGet (          /* OUT image identifier */
    Idnid_array      nIdArray,       /* composite image identifier array identifier */
    Ipuint           uIndex          /* array index value */
);
```

composite_identifier_array_get_vreg

```
Idnvreg InIdArrayGetVReg (       /* OUT image identifier */
    Idnyreg          nVReg,          /* output virtual register identifier */
    Ipsid_array_vreg snIdArray,     /* composite image identifier array identifier */
    Ipsuint_vreg_chc sIndex         /* array index value */
);
```

composite_identifier_array_put

```

Idnid_array InIdArrayPut (
    Idnimage          nImage,          /* OUT image identifier array identifier */
    Idnid_array       nIdArray,        /* source image identifier */
    Ipuint            uIndex,          /* composite image identifier array identifier */
    Ipuint            uIndex,          /* array index value */
    );

```

composite_identifier_array_put_vreg

```

Idnvreg InIdArrayPutVReg (
    Idnvreg          nVReg,          /* OUT image identifier array identifier */
    Ipsimage_vreg    snImage,        /* output virtual register identifier or NULL */
    Ipsid_array_vreg snIdArray,      /* source image identifier */
    Ipsuint_vreg_chc sIndex,         /* composite image identifier array identifier */
    Ipsuint_vreg_chc sIndex,         /* array index value */
    );

```

composite_identifier_list_empty

```

Ipbool IbIdListEmpty (
    Idnid_list        nList,          /* OUT empty flag */
    Idnid_list        nList,          /* composite image identifier list identifier */
    );

```

composite_identifier_list_empty_vreg

```

Idnvreg InIdListEmptyVReg (
    Idnvreg          nVReg,          /* OUT empty flag */
    Ipsid_list_vreg  snList,         /* output virtual register identifier */
    Ipsid_list_vreg  snList,         /* composite image identifier list identifier */
    );

```

composite_identifier_list_equal

```

Ipbool IbIdListEqual (
    Idnid_list        nFirst,         /* OUT equality flag */
    Idnid_list        nFirst,         /* first image identifier list identifier */
    Idnid_list        nSecond,        /* second image identifier list identifier */
    );

```

composite_identifier_list_equal_vreg

```

Idnvreg InIdListEqualVReg (
    Idnvreg          nVReg,          /* OUT equality flag */
    Ipsid_list_vreg  snFirst,        /* output virtual register identifier */
    Ipsid_list_vreg  snFirst,        /* first image identifier list identifier */
    Ipsid_list_vreg  snSecond,       /* second image identifier list identifier */
    );

```

composite_identifier_list_get

```
Idnimage InIdListGet (
    Idnid_list          nIdList,
    Ipint               iOption
);
```

/* OUT image identifier */
/* composite image identifier list identifier */
/* composite list pointer option */

composite_identifier_list_get_vreg

```
Idnvreg InIdListGetVReg (
    Idnvreg          nVReg,
    Ipsid_list_vreg snIdList,
    Ipsint_vreg_chc sOption
);
```

/* OUT image identifier */
/* output virtual register identifier */
/* composite image identifier list identifier */
/* composite identifier list pointer option */

composite_identifier_list_insert

```
Idnid_list InIdListInsert (
    Idnimage          nImage,
    Idnid_list        nIdList,
    Ipint             iOption
);
```

/* OUT composite identifier list identifier */
/* source image identifier */
/* list identifier */
/* list pointer option */

composite_identifier_list_insert_vreg

```
Idnvreg InIdListInsertVReg (
    Idnvreg          nVReg,
    Ipsimage_vreg    snImage,
    Ipsid_list_vreg snIdList,
    Ipsint_vreg_chc sOption
);
```

/* OUT composite identifier list identifier */
/* output virtual register identifier or NULL */
/* source image identifier */
/* list identifier */
/* list pointer option */

composite_identifier_list_remove

```
Idnid_list InIdListRemove (
    Idnid_list          nList
);
```

/* OUT composite image identifier list identifier */
/* composite image identifier list identifier */

composite_identifier_list_remove_vreg

```
Idnvreg InIdListRemoveVReg (
    Idnvreg          nVReg,
    Ipsid_list_vreg snList
);
```

/* OUT composite image identifier list identifier */
/* output virtual register identifier or NULL */
/* composite image identifier list identifier */

composite_identifier_record_equal

```

Ipbool IbIdRecordEqual (                                /* OUT equality flag */
    Idnid_record      nFirst,                            /* 1st image identifier record identifier */
    Idnid_record      nSecond,                           /* 2nd image identifier record identifier */
);

```

composite_identifier_record_equal_vreg

```

Idnvreg InIdRecordEqualVReg (                          /* OUT equality flag */
    Idnvreg          nVReg,                              /* output virtual register identifier */
    Ipsid_record_vreg snFirst,                           /* 1st image identifier record identifier */
    Ipsid_record_vreg snSecond,                          /* 2nd image identifier record identifier */
);

```

composite_identifier_record_get

```

Idnimage InIdRecordGet (                                /* OUT image identifier */
    Idnid_record      nIdRecord,                          /* composite image identifier record identifier */
    char              *tzName,                            /* attribute name pointer */
);

```

composite_identifier_record_get_vreg

```

Idnvreg InIdRecordGetVReg (                             /* OUT image identifier */
    Idnvreg          nVReg,                              /* output virtual register identifier */
    Ipsid_record_vreg snIdRecord,                        /* composite image identifier record identifier */
    Ipschar_vreg_chc sName,                             /* attribute name */
);

```

composite_identifier_record_put

```

Idnid_record InIdRecordPut (                            /* OUT image identifier record identifier */
    Idnimage          nImage,                            /* source image identifier */
    Idnid_record      nIdRecord,                         /* composite image identifier record identifier */
    char              *tzName,                           /* attribute name pointer */
);

```

composite_identifier_record_put_vreg

```
Idnvreg InIdRecordPutVReg (                                /* OUT image identifier record identifier */
    Idnvreg          nVReg,                                /* output virtual register identifier or NULL */
    Ipsimage_vreg    snImage,                              /* source image identifier */
    Ipsid_record_vreg snIdRecord,                          /* composite image identifier record identifier */
    Ipschar_vreg_chc sName                                /* attribute name */
);
```

constant_predicate

```
Ipbool IbConstantPredicate (                                /* OUT predicate flag */
    Idnimage          nSourceImage,                        /* source image identifier */
    Ipfloat           rPixelValue,                         /* pixel value */
    Ipfloat           rPixelTolerance                      /* pixel tolerance */
);
```

constant_predicate_vreg

```
Idnvreg InConstantPredicateVReg (                          /* OUT predicate flag */
    Idnvreg          nVReg,                                /* output virtual register identifier */
    Ipsimage_vreg    snSourceImage,                       /* source image identifier */
    Ipsfloat_vreg_chc sPixelValue,                        /* pixel value */
    Ipsfloat_vreg_chc sPixelTolerance                     /* pixel tolerance */
);
```

convert_array_to_image

```
Idnimage InConvertArrayToImage (                          /* OUT destination image identifier */
    Idnimage          nImage,                              /* destination image identifier */
    Ipsarray_id       snArray                             /* array identifier */
);
```

convert_array_to_image_vreg

```
Idnvreg InConvertArrayToImageVReg (                      /* OUT destination image identifier */
    Idnvreg          nVReg,                                /* output virtual register identifier or NULL */
    Ipsimage_vreg    snImage,                              /* destination image identifier */
    Ipsarray_id_vreg snArray                               /* array identifier */
);
```

convert_image_datatype

```
Idnimage InConvertImageDatatype (          /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage    /* destination image identifier */
);
```

NOTE - Any internal data type (BD, ND, SD, RD, CD) can be converted to any other internal data type.

convert_image_datatype_vreg

```
Idnvreg InConvertImageDatatypeVReg (          /* OUT destination image identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage    /* destination image identifier */
);
```

NOTE - Any internal data type (BD, ND, SD, RD, CD) can be converted to any other internal data type.

convert_image_to_array

```
Ipsarray_id IsnConvertImageToArray (          /* OUT array identifier */
    Idnimage          nImage,          /* source image identifier */
    Ipsarray_id       snArray          /* array identifier */
);
```

convert_image_to_array_vreg

```
Idnvreg InConvertImageToArrayVReg (          /* OUT array identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg    snImage,        /* source image identifier */
    Ipsarray_id_vreg snArray          /* array identifier */
);
```

convert_image_to_roi

```
Idnroi InConvertImageToROI (          /* OUT ROI identifier */
    Idnimage          nImage,          /* source image identifier */
    Idnroi            nROI             /* ROI identifier */
);
```

convert_image_to_roi_vreg

```
Idnvreg InConvertImageToROIvReg (          /* OUT ROI identifier */
    Idnvreg          nVReg,                /* output virtual register identifier or NULL */
    Ipsimage_vreg    snImage,              /* source image identifier */
    Ipsroi_vreg      snROI                 /* ROI identifier */
);
```

convert_roi_to_image

```
Idnimage InConvertROIToImage (          /* OUT destination image identifier */
    Idnimage          nImage,              /* destination image identifier */
    Idnroi            nROI                 /* ROI identifier */
);
```

convert_roi_to_image_vreg

```
Idnvreg InConvertROIToImageVReg (      /* OUT destination image identifier */
    Idnvreg          nVReg,                /* output virtual register identifier or NULL */
    Ipsimage_vreg    snImage,              /* destination image identifier */
    Ipsroi_vreg      snROI                 /* ROI identifier */
);
```

convolve_2d

```
Idnimage InConvolve2D (                /* OUT destination image identifier */
    Idnimage          nSourceImage,        /* source image identifier */
    Idnimage          nDestImage,          /* destination image identifier */
    Idnnbhood         nImpulse,            /* impulse response array identifier */
    Ipint             iOption              /* convolution option */
);
```

convolve_2d_vreg

```
Idnvreg InConvolve2DVReg (             /* OUT destination image identifier */
    Idnvreg          nVReg,                /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,        /* source image identifier */
    Ipsimage_vreg    snDestImage,          /* destination image identifier */
    Ipsnbhood_vreg   snImpulse,            /* impulse response array identifier */
    Ipsint_vreg_chc  sOption              /* convolution option */
);
```

convolve_5d

```
Idnimage InConvolve5D (                                /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage,   /* destination image identifier */
    Idnnbhood     nImpulse      /* impulse response array identifier */
);
```

convolve_5d_vreg

```
Idnvreg InConvolve5DVReg (                                /* OUT destination image identifier */
    Idnvreg      nVReg,         /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage,   /* destination image identifier */
    Ipsnbhood_vreg snImpulse     /* impulse response array identifier */
);
```

copy_window

```
Idnimage InCopyWindow (                                /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage,   /* destination image identifier */
    Idntuple      nSourceOffset, /* source index offset SD 5-tuple identifier */
    Idntuple      nDestOffset,   /* destination index offset SD 5-tuple identifier */
    Idntuple      nWindowSize   /* window size ND 5-tuple identifier */
);
```

copy_window_vreg

```
Idnvreg InCopyWindowVReg (                                /* OUT destination image identifier */
    Idnvreg      nVReg,         /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage,   /* destination image identifier */
    Ipstuple_vreg snSourceOffset, /* source index offset SD 5-tuple identifier */
    Ipstuple_vreg snDestOffset,   /* destination index offset SD 5-tuple identifier */
    Ipstuple_vreg snWindowSize   /* window size ND 5-tuple identifier */
);
```

copy_window_roi

```
Idnroi InCopyWindowROI (
    Idnroi          nSourceROI, /* OUT destination ROI identifier */
    Idnroi          nDestROI,   /* source ROI identifier */
    Idntuple        nSourceOffset, /* destination ROI identifier */
    Idntuple        nDestOffset, /* source index offset SD 5-tuple identifier */
    Idntuple        nDestOffset, /* destination index offset SD 5-tuple identifier */
    Idntuple        nWindowSize /* window size ND 5-tuple identifier */
);
```

copy_window_roi_vreg

```
Idnvreg InCopyWindowROIvReg (
    Idnvreg          nVReg, /* OUT destination ROI identifier */
    Ipsroi_vreg      snSourceROI, /* output virtual register identifier */
    Ipsroi_vreg      snDestROI,   /* source ROI identifier */
    Ipstuple_vreg     snSourceOffset, /* destination ROI identifier */
    Ipstuple_vreg     snDestOffset, /* source index offset SD 5-tuple identifier */
    Ipstuple_vreg     snDestOffset, /* destination index offset SD 5-tuple identifier */
    Ipstuple_vreg     snWindowSize /* window size ND 5-tuple identifier */
);
```

create_tuple

```
Idntuple InCreateTuple (
    Ipsparameter_basic stTupleData, /* OUT tuple data identifier */
    Idntuple           nTupleId,    /* tuple data pointer union */
    Ipsattrs_tuple     sTupleAttrs /* tuple identifier */
);
```

create_tuple_vreg

```
Idnvreg InCreateTupleVReg (
    Idnvreg          nVReg, /* OUT tuple data identifier */
    Ipsparameter_basic_vreg_chc stTupleData, /* output virtual register identifier or NULL */
    Ipstuple_vreg     snTupleId, /* tuple data */
    Ipsattrs_tuple_vreg_chc sTupleAttrs /* tuple identifier */
);
```

cross_correlation

```

Idnstatic InCrossCorrelation (                                /* OUT cross-correlation array identifier */
    Idnimage          nSourceImage1, /* first source image identifier */
    Idnimage          nSourceImage2, /* second source image identifier */
    Idnstatic         snCorrelation, /* cross-correlation RD array identifier */
    Ipoint            uIndex3,       /* third index value */
    Ipoint            uIndex4,       /* fourth index value */
    Ipoint            uIndex5,       /* fifth index value */
    Ipoint            uColumns,      /* number of array columns */
    Ipoint            uRows,         /* number of array rows */
);

```

cross_correlation_vreg

```

Idnvreg InCrossCorrelationVReg (                                /* OUT cross-correlation array identifier */
    Idnvreg          nVReg,         /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage1, /* first source image identifier */
    Ipsimage_vreg    snSourceImage2, /* second source image identifier */
    Ipsstatic_vreg    snCorrelation, /* cross-correlation array identifier */
    Ipsuint_vreg_chc  sIndex3,       /* third index value */
    Ipsuint_vreg_chc  sIndex4,       /* fourth index value */
    Ipsuint_vreg_chc  sIndex5,       /* fifth index value */
    Ipsuint_vreg_chc  sColumns,      /* number of array columns */
    Ipsuint_vreg_chc  sRows,         /* number of array rows */
);

```

deallocate_data_object

```

void IvDeallocateDataObject (                                /* no return value output */
    Ipsobject         snObject      /* data identifier */
);

```

deallocate_data_object_vreg

```

void IvDeallocateDataObjectVReg (                            /* no return value output */
    Ipsobject_vreg    snObject      /* data identifier */
);

```

define_sub_image

```
Idnimage InDefineSubImage (                                /* OUT destination sub image identifier */
    Idnimage          nSourceImage,                        /* source image identifier */
    Idnimage          nDestImage,                          /* destination sub image identifier */
    Idntuple          nOffset,                             /* index offset ND 5-tuple */
    Idntuple          nSubImageSize                       /* sub image size ND 5-tuple */
);
```

define_sub_image_vreg

```
Idnvreg InDefineSubImageVReg (                             /* OUT destination sub image identifier */
    Idnvreg          nVReg,                                /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,                       /* source image identifier */
    Ipsimage_vreg    snDestImage,                        /* destination sub image identifier */
    Ipstuple_vreg    snOffset,                            /* index offset ND 5-tuple */
    Ipstuple_vreg    snSubImageSize                      /* sub image size ND 5-tuple */
);
```

difference_measures

```
Ipfloat IrDiffMeasures (                                   /* OUT distance measure value */
    Idnimage          nSourceImage1,                      /* first source image identifier */
    Idnimage          nSourceImage2,                     /* second source image identifier */
    Ipint             iOption                             /* difference measures option */
);
```

difference_measures_vreg

```
Idnvreg InDiffMeasuresVReg (                               /* OUT distance measure value */
    Idnvreg          nVReg,                                /* output virtual register identifier */
    Ipsimage_vreg    snSourceImage1,                     /* first source image identifier */
    Ipsimage_vreg    snSourceImage2,                    /* second source image identifier */
    Ipsint_vreg_chc  sOption                             /* difference measures option */
);
```

diffuse

```
Idnimage InDiffuse (                                       /* OUT destination image identifier */
    Idnimage          nSourceImage,                      /* source image identifier */
    Idnimage          nDestImage,                        /* destination image identifier */
    Idntuple          nAmplitudeLevel                   /* destination amplitude ND B-tuple identifier */
);
```

diffuse_vreg

```
Idnvreg InDiffuseVReg (                                /* OUT destination image identifier */
    Idnvreg                                nVReg,        /* output virtual register identifier or NULL */
    Ipsimage_vreg                        snSourceImage,   /* source image identifier */
    Ipsimage_vreg                        snDestImage,     /* destination image identifier */
    Ipstuple_vreg                        snAmplitudeLevel /* destination amplitude ND B-tuple identifier */
);
```

dither

```
Idnimage InDither (                                    /* OUT destination image identifier */
    Idnimage                                nSourceImage, /* source image identifier */
    Idnimage                                nDestImage,   /* destination image identifier */
    Idnnbhood                             nDither,       /* dither array identifier */
    Idntuple                             nAmplitudeLevel /* destination amplitude ND B-tuple identifier */
);
```

dither_vreg

```
Idnvreg InDitherVReg (                                /* OUT destination image identifier */
    Idnvreg                                nVReg,        /* output virtual register identifier or NULL */
    Ipsimage_vreg                        snSourceImage,   /* source image identifier */
    Ipsimage_vreg                        snDestImage,     /* destination image identifier */
    Ipsnbhood_vreg                      snDither,        /* dither array identifier */
    Ipstuple_vreg                        snAmplitudeLevel /* destination amplitude ND B-tuple identifier */
);
```

draw_pixels

```
Idnimage InDrawPixels (                                /* OUT destination image identifier */
    Ipsdraw_pixels                        *tsaPixels,     /* pointer to array of pixel collection structures */
    Idnimage                                nDestImage,   /* destination image identifier */
    Ipuint                                uEntries        /* number of array entries */
);
```

draw_pixels_vreg

```
Idnvreg InDrawPixelsVReg (                            /* OUT destination image identifier */
    Idnvreg                                nVReg,        /* output virtual register identifier or NULL */
    Ipsdraw_pixels_vreg_chc              saPixels,       /* pixel collection structure array choice */
    Ipsimage_vreg                        snDestImage,     /* destination image identifier */
    Ipsuint_vreg_chc                     sEntries        /* number of array entries */
);
```

draw_pixels_roi

```
Idnroi InDrawPixelsROI (                                /* OUT destination ROI identifier */
    Ipsdraw_pixels_roi    *tsaPixels,                    /* pointer to array of data collection structures */
    Idnroi                nDestROI,                      /* destination ROI identifier */
    Ipoint                uEntries                       /* number of array entries */
);
```

draw_pixels_roi_vreg

```
Idnvreg InDrawPixelsROIvReg (                            /* OUT destination ROI identifier */
    Idnvreg              nVReg,                          /* output virtual register identifier or NULL */
    Ipsdraw_pixels_roi_vreg_chc saPixels,                /* data collection structure array choice */
    Ipsroi_vreg          snDestROI,                      /* destination ROI identifier */
    Ipsuint_vreg_chc     sEntries                       /* number of array entries */
);
```

dyadic_arithmetic

```
Idnimage InDyadicArith (                                /* OUT destination image identifier */
    Idnimage            nSourceImage1,                  /* first source image identifier */
    Idnimage            nSourceImage2,                  /* second source image identifier */
    Idnimage            nDestImage,                    /* destination image identifier */
    Ipoint              iOption                        /* dyadic arithmetic combination option */
);
```

dyadic_arithmetic_vreg

```
Idnvreg InDyadicArithVReg (                             /* OUT destination image identifier */
    Idnvreg            nVReg,                          /* output virtual register identifier or NULL */
    Ipsimage_vreg      snSourceImage1,                  /* first source image identifier */
    Ipsimage_vreg      snSourceImage2,                  /* second source image identifier */
    Ipsimage_vreg      snDestImage,                    /* destination image identifier */
    Ipsint_vreg_chc     sOption                        /* dyadic arithmetic combination option */
);
```

dyadic_complex

```
Idnimage InDyadicComplex (                              /* OUT destination image identifier */
    Idnimage            nSourceImage1,                  /* first source image identifier */
    Idnimage            nSourceImage2,                  /* second source image identifier */
    Idnimage            nDestImage,                    /* destination image identifier */
    Ipoint              iOption                        /* dyadic complex combination option */
);
```

dyadic_complex_vreg

```
Idnvreg InDyadicComplexVReg (          /* OUT destination image identifier */
    Idnvreg          nVReg,             /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage1,    /* first source image identifier */
    Ipsimage_vreg    snSourceImage2,    /* second source image identifier */
    Ipsimage_vreg    snDestImage,       /* destination image identifier */
    Ipsint_vreg_chc  sOption            /* dyadic complex combination option */
);
```

dyadic_logical

```
Idnimage InDyadicLogical (              /* OUT destination image identifier */
    Idnimage          nSourceImage1,    /* first source image identifier */
    Idnimage          nSourceImage2,    /* second source image identifier */
    Idnimage          nDestImage,       /* destination image identifier */
    Ipint             iOption            /* dyadic logical combination option */
);
```

dyadic_logical_vreg

```
Idnvreg InDyadicLogicalVReg (           /* OUT destination image identifier */
    Idnvreg          nVReg,             /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage1,    /* first source image identifier */
    Ipsimage_vreg    snSourceImage2,    /* second source image identifier */
    Ipsimage_vreg    snDestImage,       /* destination image identifier */
    Ipsint_vreg_chc  sOption            /* dyadic logical combination option */
);
```

dyadic_logical_roi

```
Idnroi InDyadicLogicalROI (             /* OUT destination ROI identifier */
    Idnroi           nSourceROI1,       /* first source ROI identifier */
    Idnroi           nSourceROI2,       /* second source ROI identifier */
    Idnroi           nDestROI,          /* destination ROI identifier */
    Ipint            iOption             /* dyadic logical combination option */
);
```

dyadic_logical_roi_vreg

```
Idnvreg InDyadicLogicalROIvReg ( /* OUT destination ROI identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsroi_vreg      snSourceROI1, /* first source ROI identifier */
    Ipsroi_vreg      snSourceROI2, /* second source ROI identifier */
    Ipsroi_vreg      snDestROI, /* destination ROI identifier */
    Ipsint_vreg_chc  sOption /* dyadic logical combination option */
);
```

dyadic_predicate

```
Idnimage InDyadicPredicate ( /* OUT destination image identifier */
    Idnimage      nSourceImage1, /* first source image identifier */
    Idnimage      nSourceImage2, /* second source image identifier */
    Idnimage      nDestImage, /* destination image identifier */
    Ipint         iOption /* dyadic predicate combination option */
);
```

dyadic_predicate_vreg

```
Idnvreg InDyadicPredicateVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage1, /* first source image identifier */
    Ipsimage_vreg    snSourceImage2, /* second source image identifier */
    Ipsimage_vreg    snDestImage, /* destination image identifier */
    Ipsint_vreg_chc  sOption /* dyadic predicate combination option */
);
```

edge_detection_orthogonal_gradient

```
Idnimage InEdgeDetectOrthGrad ( /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage, /* destination image identifier */
    Idnnbhood     nHorizontalGrad, /* horizontal gradient impulse array identifier */
    Idnnbhood     nVerticalGrad, /* vertical gradient impulse array identifier */
    Ipsparameter_arith stThreshold, /* threshold value pointer union */
    Ipsparameter_edge stBelowValue, /* below threshold output value pointer union */
    Ipsparameter_edge stAboveValue, /* above threshold output value pointer union */
    Ipint         iEdgeOption, /* edge detection, orthogonal option */
    Ipereflection   eHorizontalOption, /* horizontal impulse rotation option */
    Ipereflection   eVerticalOption, /* vertical impulse rotation option */
    Ipint         iTypeOption /* BD or RD destination data type option */
);
```

edge_detection_orthogonal_gradient_vreg

```
Idnvreg InEdgeDetectOrthGradVReg (                                /* OUT destination image identifier */
    Idnvreg          nVReg,                                       /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipsnbhood_vreg   snHorizontalGrad, /* horizontal gradient impulse array identifier */
    Ipsnbhood_vreg   snVerticalGrad, /* vertical gradient impulse array identifier */
    Ipsparameter_arith_vreg_chc sThreshold, /* threshold value */
    Ipsparameter_edge_vreg_chc sBelowValue, /* below threshold output value */
    Ipsparameter_edge_vreg_chc sAboveValue, /* above threshold output value */
    Ipsint_vreg_chc   sEdgeOption, /* edge detection, orthogonal option */
    Ipsreflection_vreg_chc sHorizontalOption, /* horizontal impulse rotation option */
    Ipsreflection_vreg_chc sVerticalOption, /* vertical impulse rotation option */
    Ipsint_vreg_chc   sTypeOption /* BD or RD destination data type option */
);
```

edge_detection_second_derivative

```
Idnimage InEdgeDetect2ndDeriv (                                /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,  /* destination image identifier */
    Idnnbhood         n2ndDeriv, /* second derivative impulse array identifier */
    Ipsparameter_edge stNoCrossing, /* no zero crossing output value pointer union */
    Ipsparameter_edge stZeroCrossing, /* zero crossing output value pointer union */
    Ipint             iDerivOption, /* edge detection, option */
    Ipint             iTypeOption /* BD or RD destination data type option */
);
```

edge_detection_second_derivative_vreg

```
Idnvreg InEdgeDetect2ndDerivVReg (                                /* OUT destination image identifier */
    Idnvreg          nVReg,                                       /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipsnbhood_vreg   sn2ndDeriv, /* second derivative impulse array identifier */
    Ipsparameter_edge_vreg_chc sNoCrossing, /* no zero crossing output value */
    Ipsparameter_edge_vreg_chc sZeroCrossing, /* zero crossing output value */
    Ipsint_vreg_chc   sDerivOption, /* edge detection, option */
    Ipsint_vreg_chc   sTypeOption /* BD or RD destination data type option */
);
```

edge_detection_template_gradient

```
Idnimage InEdgeDetectTemplGrad (          /* OUT destination image identifier */
    Idnimage          nSourceImage,        /* source image identifier */
    Idnimage          nDestImage,         /* destination image identifier */
    Idnnbhood          *tnaArrays,         /* template gradient array identifier set */
    Ipuint            uNumber,             /* number of template gradient arrays */
    Ipsparameter_arith stThreshold,         /* threshold value pointer union */
    Ipsparameter_edge  stBelowValue,       /* below threshold output value pointer union */
    Ipsparameter_edge  stAboveValue,      /* above threshold output value pointer union */
    Ipint              iGradOption,        /* edge detection option */
    Ipint              iTypeOption        /* BD or RD destination data type option */
);
```

edge_detection_template_gradient_vreg

```
Idnvreg InEdgeDetectTemplGradVReg (      /* OUT destination image identifier */
    Idnvreg          nVReg,               /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,       /* source image identifier */
    Ipsimage_vreg    snDestImage,        /* destination image identifier */
    Ipsanbhood_vreg_chc sArrays,          /* template gradient array identifier set */
    Ipsuint_vreg_chc  sNumber,            /* number of template gradient arrays */
    Ipsparameter_arith_vreg_chc sThreshold, /* threshold value */
    Ipsparameter_edge_vreg_chc sBelowValue, /* below threshold output value */
    Ipsparameter_edge_vreg_chc sAboveValue, /* above threshold output value */
    Ipsint_vreg_chc   sGradOption,        /* edge detection option */
    Ipsint_vreg_chc   sTypeOption        /* BD or RD destination data type option */
);
```

equal_predicate

```
Ipbool IbEqualPredicate (                /* OUT predicate flag */
    Idnimage          nSourceImage1,      /* first source image identifier */
    Idnimage          nSourceImage2,      /* second source image identifier */
    Ipfloat            rTolerance          /* pixel tolerance */
);
```

equal_predicate_vreg

```
Idnvreg InEqualPredicateVReg (           /* OUT predicate flag */
    Idnvreg          nVReg,               /* output virtual register identifier */
    Ipsimage_vreg    snSourceImage1,      /* first source image identifier */
    Ipsimage_vreg    snSourceImage2,      /* second source input image identifier */
    Ipsfloat_vreg_chc sTolerance          /* pixel tolerance */
);
```

erosion_dilation_boolean

```
Idnimage InErosionDilateBool ( /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage,   /* destination image identifier */
    Idnnbhood     nArray,       /* structuring element array identifier */
    Ipoint        uIterations,   /* number of iterations */
    Ipeerosion     eErosionOption, /* erosion/dilation operation option */
    Ipoint        iProcessOption /* processing option */
);
```

erosion_dilation_boolean_vreg

```
Idnvreg InErosionDilateBoolVReg ( /* OUT destination image identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage,   /* destination image identifier */
    Ipsnbhood_vreg snArray,       /* structuring element array identifier */
    Ipsuint_vreg_chc sIterations, /* number of iterations */
    Ipserosion_vreg_chc sErosionOption, /* erosion/dilation operation option */
    Ipsint_vreg_chc sProcessOption /* processing option */
);
```

erosion_dilation_boolean_roi

```
Idnroi InErosionDilateBoolROI ( /* OUT destination ROI identifier */
    Idnroi      nSourceROI, /* source ROI identifier */
    Idnroi      nDestROI,   /* destination ROI identifier */
    Idnnbhood     nArray,   /* structuring element array identifier */
    Ipoint        uIterations, /* number of iterations */
    Ipeerosion     eErosionOption, /* erosion/dilation operation option */
    Ipoint        iProcessOption /* processing option */
);
```

erosion_dilation_boolean_roi_vreg

```
Idnvreg InErosionDilateBoolROIvReg ( /* OUT destination ROI identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    Ipsroi_vreg snSourceROI, /* source ROI identifier */
    Ipsroi_vreg snDestROI,   /* destination ROI identifier */
    Ipsnbhood_vreg snArray, /* structuring element array identifier */
    Ipsuint_vreg_chc sIterations, /* number of iterations */
    Ipserosion_vreg_chc sErosionOption, /* erosion/dilation operation option */
    Ipsint_vreg_chc sProcessOption /* processing option */
);
```

erosion_dilation_grey

```
Idnimage InErosionDilateGrey (          /* OUT destination image identifier */
    Idnimage          nSourceImage,      /* source image identifier */
    Idnimage          nDestImage,        /* destination image identifier */
    Idnnbhood         nStructure,        /* structuring element array identifier */
    Idnnbhood         nMask,            /* mask array identifier */
    Ipuint            uIterations,       /* number of iterations */
    lpeerosion        eErosionOption,    /* erosion/dilation operation option */
    Ipint             iProcessOption     /* processing option */
);
```

erosion_dilation_grey_vreg

```
Idnvreg InErosionDilateGreyVReg (        /* OUT destination image identifier */
    Idnvreg          nVReg,             /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,     /* source image identifier */
    Ipsimage_vreg    snDestImage,       /* destination image identifier */
    Ipsnbhood_vreg   snStructure,       /* structuring element array identifier */
    Ipsnbhood_vreg   snMask,           /* mask array identifier */
    Ipsuint_vreg_chc slterations,       /* number of iterations */
    Ipserosion_vreg_chc sErosionOption, /* erosion/dilation operation option */
    Ipsint_vreg_chc  sProcessOption     /* processing option */
);
```

error_function_scaling

```
Idnimage InErrorFunctionScaling (        /* OUT destination image identifier */
    Idnimage          nSourceImage,     /* source image identifier */
    Idnimage          nDestImage,       /* destination image identifier */
    Ipfloat           rFactor            /* Gaussian spread factor */
);
```

error_function_scaling_vreg

```
Idnvreg InErrorFunctionScalingVReg (     /* OUT destination image identifier */
    Idnvreg          nVReg,             /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,     /* source image identifier */
    Ipsimage_vreg    snDestImage,       /* destination image identifier */
    Ipsfloat_vreg_chc sFactor           /* Gaussian spread factor */
);
```

error_handler

```
void IvErrorHandler (
    Ipoint      iCode,          /* no return value output */
    Ipoint      iElement,       /* error identifier code */
    Idnchain    nChain,         /* descriptor of element detecting error */
    Ipnerror    nError,         /* identifier of chain detecting error */
    Ipsynchronicity_state eSynchronicityState /* error file identifier */
);
```

error_logger

```
void IvErrorLogger (
    Ipoint      iCode,          /* no return value output */
    Ipoint      iElement,       /* error identifier code */
    Idnchain    nChain,         /* descriptor of function detecting error */
    Ipnerror    nError,         /* identifier of chain detecting error */
    Ipsynchronicity_state eSynchronicityState /* error file identifier */
);
```

error_test

```
Ipbool IbErrorTest (
    void          /* OUT error flag */
);
```

error_test_vreg

```
Idnvreg InErrorTestVReg (
    Idnvreg      nVReg          /* OUT error flag */
);
```

export_histogram

```
Ipoint *ItuaExportHist (
    Idnhist      nHist,         /* OUT external histogram data array pointer */
    Ipoint      *tuaData        /* histogram identifier */
);
```

export_image

```
Ipspiks_pixel_types IstExportImage ( /* OUT external image pixel data pointer union */
    Idnimage          nSourceImage, /* internal source image identifier */
    Ipspiks_pixel_types stDestImage, /* external image pixel data pointer union */
    Ipoint            uPrecision, /* number of bits of destination pixel */
    Ipoint            iType, /* destination data type option */
    Ipoint            iApplBufferSize, /* data transfer buffer size or option */
    Ipoint            uStartPoint, /* index offset within internal data */
    Ipoint            *tuDataSize /* OUT post transfer data size pointer */
);
```

export_lut

```
Ipsparameter_pixel IstExportLUT ( /* OUT external data array pointer union */
    Idnlut          nLUT, /* lookup table data identifier */
    Ipsparameter_pixel stData, /* external data array pointer union */
    Ipoint          iApplBufferSize, /* data transfer buffer size or option */
    Ipoint          uStartPoint, /* index offset within internal data */
    Ipoint          *tuDataSize /* OUT post transfer data size pointer */
);
```

export_matrix

```
Ipsparameter_numeric IstExportMatrix ( /* OUT external data array pointer union */
    Idnmatrix          nMatrix, /* matrix identifier */
    Ipsparameter_numeric stData /* external data array pointer union */
);
```

export_neighbourhood_array

```
Ipsparameter_pixel IstExportNbhoodArray ( /* OUT neighbourhood array pointer union */
    Idnnbhood          nNbhood, /* neighbourhood array identifier */
    Ipsparameter_pixel stData, /* neighbourhood array data pointer union */
    Ipoint            *tiScaleFactor /* OUT scale factor */
);
```

export_roi_array

```
Ipbool *ItbaExportROIArray ( /* OUT external data value array pointer */
    Idnroi          nROI, /* ROI array identifier */
    Ipbool          *tbaData, /* external data values array pointer */
    Ipoint          iApplBufferSize, /* data transfer buffer size or option */
    Ipoint          uStartPoint, /* index offset within internal data */
    Ipoint          *tuDataSize /* OUT post transfer data size pointer */
);
```

export_static_array

```

Ipsparameter_pixel IstExportStaticArray (          /* OUT external data array pointer union */
    Idnstatic          nArray,          /* static array identifier */
    Ipsparameter_pixel stData,          /* external array of data array pointer union */
    Ipoint             iApplBufferSize, /* data transfer buffer size or option */
    Ipoint             uStartPoint,     /* index offset within internal data */
    Ipoint             *tuDataSize      /* OUT post transfer data size pointer */
);

```

export_tuple

```

Ipsparameter_basic IstExportTuple (          /* OUT external data array pointer union */
    Idntuple          nTuple,          /* tuple identifier */
    Ipsparameter_basic stData          /* external data value array pointer union */
);

```

export_value_bounds

```

Ipsparameter_arith IstExportValueBounds (          /* OUT to external data array pointer union */
    Idnbounds          nValueBounds, /* value bound collection identifier */
    Ipsparameter_arith stData,          /* external data value array pointer union */
    Ipoint             iApplBufferSize, /* data transfer buffer size or option */
    Ipoint             uStartPoint,     /* index offset within internal data */
    Ipoint             *tuDataSize      /* OUT post transfer data size pointer */
);

```

extract_pixel_plane

```

Idnimage InExtractPixelPlane (          /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Ipoint            uDepth,        /* depth index value */
    Ipoint            uTemporal,     /* temporal index value */
    Ipoint            uBand          /* band index value */
);

```

extract_pixel_plane_vreg

```
Idnvreg InExtractPixelPlaneVReg (          /* OUT destination image identifier */
    Idnvreg          nVReg,                /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,        /* source image identifier */
    Ipsimage_vreg    snDestImage,         /* destination image identifier */
    Ipsuint_vreg_chc sDepth,               /* depth index value */
    Ipsuint_vreg_chc sTemporal,           /* temporal index value */
    Ipsuint_vreg_chc sBand                /* band index value */
);
```

extract_pixel_plane_roi

```
Idnroi InExtractPixelPlaneROI (          /* OUT destination ROI identifier */
    Idnroi          nSourceROI,           /* source ROI identifier */
    Idnroi          nDestROI,            /* destination ROI identifier */
    Ipuint          uDepth,              /* depth index value */
    Ipuint          uTemporal,           /* temporal index value */
    Ipuint          uBand                /* band index value */
);
```

extract_pixel_plane_roi_vreg

```
Idnvreg InExtractPixelPlaneROIvReg (     /* OUT destination ROI identifier */
    Idnvreg          nVReg,                /* output virtual register identifier or NULL */
    Ipsroi_vreg      snSourceROI,         /* source ROI identifier */
    Ipsroi_vreg      snDestROI,          /* destination ROI identifier */
    Ipsuint_vreg_chc sDepth,              /* depth index value */
    Ipsuint_vreg_chc sTemporal,           /* temporal index value */
    Ipsuint_vreg_chc sBand                /* band index value */
);
```

extrema

```
void IvExtrema (                          /* no return value output */
    Idnimage          nSourceImage,       /* source image identifier */
    Ipfloating         *traMinimum,        /* OUT minimum extrema pixel array pointers */
    Ipfloating         *traMaximum,        /* OUT maximum extrema pixel array pointers */
    Ipeprocess         eOption,            /* processing mode option */
    Ipuint            iApplBufferSize,     /* data transfer buffer size or option of each array */
    Ipuint            uStartPoint,         /* index offset within internal data of each array */
    Ipuint            *tuDataSize         /* OUT post transfer data size pointer */
);
```

false_colour

```
Idnimage InFalseColour (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Idnmatrix         nColour,      /* destination image identifier */
                                /* colour conversion matrix identifier */
);
```

false_colour_vreg

```
Idnvreg InFalseColourVReg (
    Idnvreg          nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier */
    Ipsmatrix_vreg   snColour,      /* destination image identifier */
                                /* colour conversion matrix identifier */
);
```

fill_region

```
Idnimage InFillRegion (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Ipoint            uSeed1,       /* destination image identifier */
    Ipoint            uSeed2,       /* first index value - seed pixel within contour */
    Ipoint            uIndex3,      /* second index value - seed pixel within contour */
    Ipoint            uIndex4,      /* third index value */
    Ipoint            uIndex5,      /* fourth index value */
                                /* fifth index value */
);
```

fill_region_vreg

```
Idnvreg InFillRegionVReg (
    Idnvreg          nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier */
    Ipsuint_vreg_chc sSeed1,        /* destination image identifier */
    Ipsuint_vreg_chc sSeed2,        /* first index value - seed pixel within contour */
    Ipsuint_vreg_chc sIndex3,       /* second index value - seed pixel within contour */
    Ipsuint_vreg_chc sIndex4,       /* third index value */
    Ipsuint_vreg_chc sIndex5,       /* fourth index value */
                                /* fifth index value */
);
```

fill_region_roi

```
Idnroi InFillRegionROI (
    Idnroi          nSourceROI, /* OUT destination ROI identifier */
    Idnroi          nDestROI,   /* source ROI identifier */
    Ipuint          uSeed1,     /* destination ROI identifier */
    Ipuint          uSeed2,     /* first index value - seed pixel within contour */
    Ipuint          uIndex3,    /* second index value - seed pixel within contour */
    Ipuint          uIndex4,    /* third index value */
    Ipuint          uIndex5,    /* fourth index value */
    Ipuint          uIndex5,    /* fifth index value */
);
```

fill_region_roi_vreg

```
Idnvreg InFillRegionROIvReg (
    Idnvreg          nVReg,     /* OUT destination ROI identifier */
    Ipsroi_vreg      snSourceROI, /* output virtual register identifier or NULL */
    Ipsroi_vreg      snDestROI, /* source ROI identifier */
    Ipsuint_vreg_chc sSeed1,    /* destination ROI identifier */
    Ipsuint_vreg_chc sSeed2,    /* first index value - seed pixel within contour */
    Ipsuint_vreg_chc sIndex3,   /* second index value - seed pixel within contour */
    Ipsuint_vreg_chc sIndex4,   /* third index value */
    Ipsuint_vreg_chc sIndex5,   /* fourth index value */
    Ipsuint_vreg_chc sIndex5,   /* fifth index value */
);
```

filter_butterworth

```
Idnstatic InFilterButterworth
    Idnstatic          nFilter, /* OUT filter transfer function array identifier */
    Idntuple           nSize,   /* filter transfer function array identifier */
    Ipfloat            rOrder,  /* array size ND 5-tuple */
    Ipfloat            rCutoff, /* filter order */
    Ipint             iOption,  /* cutoff frequency */
    Ipint             iOption, /* filter (Butterworth) type option */
);
```

filter_butterworth_vreg

```
Idnvreg InFilterButterworthVReg (
    Idnvreg          nVReg,     /* OUT filter transfer function array identifier */
    Ipsstatic_vreg   snFilter,  /* output virtual register identifier or NULL */
    Ipsstatic_vreg   snSize,    /* filter transfer function array identifier */
    Ipsstatic_vreg   snSize,    /* array size ND 5-tuple */
    Ipsfloat_vreg_chc sOrder,   /* filter order */
    Ipsfloat_vreg_chc sCutoff,  /* cutoff frequency */
    Ipsint_vreg_chc  sOption,   /* filter (Butterworth) type option */
);
```

filter_gaussian

```

Idnstatic InFilterGauss (                                /* OUT filter transfer function array identifier */
    Idnstatic      nFilter,                               /* filter transfer function array identifier */
    Idntuple       nSize,                                /* array size ND 5-tuple */
    Ipsfloat       rColumnSpread,                         /* column index spread factor */
    Ipsfloat       rRowSpread                             /* row index spread factor */
);

```

filter_gaussian_vreg

```

Idnvreg InFilterGaussVReg (                             /* OUT filter transfer function array identifier */
    Idnvreg      nVReg,                                  /* output virtual register identifier or NULL */
    Ipsstatic_vreg snFilter,                             /* filter transfer function array identifier */
    Ipsstuple_vreg snSize,                               /* array size ND 5-tuple */
    Ipsfloat_vreg_chc sColumnSpread,                     /* column index spread factor */
    Ipsfloat_vreg_chc sRowSpread                         /* row index spread factor */
);

```

filter_inverse

```

Idnstatic InFilterInverse (                             /* OUT filter transfer function array identifier */
    Idnstatic      nBlur,                                /* image blur transfer function array identifier */
    Idnstatic      nTransfer,                            /* filter transfer function array identifier */
    Idntuple       nSize,                                /* array size ND 5-tuple */
    Ipsuint        uColumnCutoff,                       /* column index cutoff frequency */
    Ipsuint        uRowCutoff                           /* row index cutoff frequency */
);

```

filter_inverse_vreg

```

Idnvreg InFilterInverseVReg (                           /* OUT filter transfer function array identifier */
    Idnvreg      nVReg,                                  /* output virtual register identifier or NULL */
    Ipsstatic_vreg snBlur,                               /* image blur transfer function array identifier */
    Ipsstatic_vreg snTransfer,                           /* filter transfer function array identifier */
    Ipsstuple_vreg snSize,                               /* array size ND 5-tuple */
    Ipsuint_vreg_chc sColumnCutoff,                     /* column index cutoff frequency */
    Ipsuint_vreg_chc sRowCutoff                         /* row index cutoff frequency */
);

```

filter_matched

```
Idnstatic InFilterMatched (
    Idnstatic      nFourier, /* OUT filter transfer function array identifier */
    Idnstatic      nNoise,   /* reference image Fourier transform array id */
    Idnstatic      nFilter,  /* noise power spectrum array identifier */
    Idntuple       nSize,    /* filter transfer function array identifier */
    Ipuint         uColumnOffset, /* array size ND 5-tuple */
    Ipuint         uRowOffset, /* column index offset */
    );                  /* row index offset */
```

filter_matched_vreg

```
Idnvreg InFilterMatchedVReg (
    Idnvreg      nVReg, /* OUT filter transfer function array identifier */
    Ipsstatic_vreg snFourier, /* output virtual register identifier or NULL */
    Ipsstatic_vreg snNoise,   /* reference image Fourier transform array id */
    Ipsstatic_vreg snFilter,  /* noise power spectrum array identifier */
    Ipsstatic_vreg snSize,    /* filter transfer function array identifier */
    Ipsuint_vreg_chc sColumnOffset, /* array size ND 5-tuple */
    Ipsuint_vreg_chc sRowOffset, /* column index offset */
    );                  /* row index offset */
```

filter_wiener

```
Idnstatic InFilterWiener (
    Idnstatic      nBlur, /* OUT filter transfer function array identifier */
    Idnstatic      nImagePower, /* image blur transfer function array identifier */
    Idnstatic      nNoisePower, /* image power spectrum array identifier */
    Idnstatic      nFilter, /* noise power spectrum array identifier */
    Idntuple       nSize, /* filter transfer function array identifier */
    );              /* array size ND 5-tuple */
```

filter_wiener_vreg

```
Idnvreg InFilterWienerVReg (
    Idnvreg      nVReg, /* OUT filter transfer function array identifier */
    Ipsstatic_vreg snBlur, /* output virtual register identifier or NULL */
    Ipsstatic_vreg snImagePower, /* image blur transfer function array identifier */
    Ipsstatic_vreg snNoisePower, /* image power spectrum array identifier */
    Ipsstatic_vreg snFilter, /* noise power spectrum array identifier */
    Ipsstatic_vreg snSize, /* filter transfer function array identifier */
    );              /* array size ND 5-tuple */
```

filter_zonal

```
Idnstatic InFilterZonal (                                /* OUT filter transfer function array identifier */
    Idnstatic      nFilter,                               /* filter transfer function array identifier */
    Idntuple       nSize,                                 /* array size ND 5-tuple */
    Ipuint         uColumnCutoff,                         /* column index cutoff frequency */
    Ipuint         uRowCutoff,                           /* row index cutoff frequency */
    Ipuint         iOption                               /* filter (zonal) type option */
);
```

filter_zonal_vreg

```
Idnvreg InFilterZonalVReg (                             /* OUT filter transfer function array identifier */
    Idnvreg      nVReg,                                  /* output virtual register identifier or NULL */
    Ipsstatic_vreg snFilter,                             /* filter transfer function array identifier */
    Ipstuple_vreg snSize,                                /* array size 5-tuple */
    Ipsuint_vreg_chc sColCutoff,                         /* column index cutoff frequency */
    Ipsuint_vreg_chc sRowCutoff,                         /* row index cutoff frequency */
    Ipsint_vreg_chc sOption                              /* filter (zonal) type option */
);
```

filtering_homomorphic

```
Idnimage InFilteringHomomorphic (                       /* OUT destination image identifier */
    Idnimage      nSourceImage,                          /* source image identifier */
    Idnimage      nDestImage,                            /* destination image identifier */
    Idnstatic     nWindow,                               /* windowing function array identifier */
    Idnstatic     nFilter                                /* filter transfer function array identifier */
);
```

filtering_homomorphic_vreg

```
Idnvreg InFilteringHomomorphicVReg (                   /* OUT destination image identifier */
    Idnvreg      nVReg,                                  /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage,                         /* source image identifier */
    Ipsimage_vreg snDestImage,                           /* destination image identifier */
    Ipsstatic_vreg snWindow,                             /* windowing function array identifier */
    Ipsstatic_vreg snFilter                              /* filter transfer function array identifier */
);
```

filtering_linear

```
Idnimage InFilteringLin (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier          */
    Idnstatic         nWindow,      /* destination image identifier     */
    Idnstatic         nFilter,      /* window function array identifier */
    Idnstatic         /* filter transfer function array identifier */
);
```

filtering_linear_vreg

```
Idnvreg InFilteringLinVReg (
    Idnvreg          nVReg, /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage, /* source image identifier          */
    Ipsstatic_vreg   snWindow, /* destination image identifier     */
    Ipsstatic_vreg   snFilter, /* window function array identifier */
    Ipsstatic_vreg   /* filter transfer function array identifier */
);
```

filtering_median

```
Idnimage InFilteringMedian (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier          */
    Idnnbhood         nMask,        /* destination image identifier     */
    Ipint             iOption,      /* mask array identifier            */
    Ipint             /* filtering process option         */
);
```

filtering_median_vreg

```
Idnvreg InFilteringMedianVReg (
    Idnvreg          nVReg, /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage, /* source image identifier          */
    Ipsnbhood_vreg   snMask,      /* destination image identifier     */
    Ipsint_vreg_chc  siOption, /* mask array identifier            */
    Ipsint_vreg_chc  /* filtering process option         */
);
```

filtering_pseudomedian

```
Idnimage InFilteringPseudomedian (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier          */
    Ipuint            uMaskWidth, /* destination image identifier     */
    Ipuint            uMaskHeight, /* mask width                      */
    Ipuint            /* mask height                     */
);
```

filtering_pseudomedian_vreg

```
Idnvreg InFilteringPseudomedianVReg (          /* OUT destination image identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,   /* destination image identifier */
    Ipsuint_vreg_chc sMaskWidth,    /* mask width */
    Ipsuint_vreg_chc sMaskHeight   /* mask height */
);
```

filtering_rank_order

```
Idnimage InFilteringRankOrder (          /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Idnnbhood         nMask,        /* mask array identifier */
    Ipuint            uRank,        /* rank value */
    Ipuint            iOption       /* filtering process option */
);
```

filtering_rank_order_vreg

```
Idnvreg InFilteringRankOrderVReg (          /* OUT destination image identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,   /* destination image identifier */
    Ipsnbhood_vreg   snMask,        /* mask array identifier */
    Ipsuint_vreg_chc sRank,        /* rank value */
    Ipsint_vreg_chc  sOption       /* filtering process option */
);
```

flip_spin_transpose

```
Idnimage InFlipSpinTranspose (          /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Ipeflip           eOption       /* flip, spin, transpose option */
);
```

flip_spin_transpose_vreg

```
Idnvreg InFlipSpinTransposeVReg (                /* OUT destination image identifier */
    Idnvreg          nVReg,                      /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,             /* source image identifier */
    Ipsimage_vreg    snDestImage,              /* destination image identifier */
    Ipsflip_vreg_chc sOption                    /* flip, spin, transpose option */
);
```

flip_spin_transpose_roi

```
Idnroi InFlipSpinTransposeROI (                 /* OUT destination ROI identifier */
    Idnroi          nSourceROI,                 /* source ROI identifier */
    Idnroi          nDestROI,                  /* destination ROI identifier */
    Ipeflip         eOption                    /* flip, spin, transpose option */
);
```

flip_spin_transpose_roi_vreg

```
Idnvreg InFlipSpinTransposeROIvReg (           /* OUT destination ROI identifier */
    Idnvreg          nVReg,                      /* output virtual register identifier or NULL */
    Ipsroi_vreg      snSourceROI,               /* source ROI identifier */
    Ipsroi_vreg      snDestROI,                /* destination ROI identifier */
    Ipsflip_vreg_chc sOption                    /* flip, spin, transpose option */
);
```

gamma_correction

```
Idnimage InGammaCorrection (                   /* OUT destination image identifier */
    Idnimage          nSourceImage,             /* source image identifier */
    Idnimage          nDestImage,              /* destination image identifier */
    Ipfloat           rFirst,                   /* first constant */
    Ipfloat           rSecond,                  /* second constant */
    Ipfloat           rThird,                   /* third constant */
    Ipfloat           rFourth,                  /* fourth constant */
    Ipfloat           rBreak,                   /* break point */
    Ipegamma          eOption                    /* gamma correction option */
);
```

gamma_correction_vreg

```
Idnvreg InGammaCorrectionVReg (                                /* OUT destination image identifier */
    Idnvreg                nVReg,                               /* output virtual register identifier or NULL */
    Ipsimage_vreg          snSourceImage,                       /* source image identifier */
    Ipsimage_vreg          snDestImage,                         /* destination image identifier */
    Ipsfloat_vreg_chc      sFirst,                             /* first constant */
    Ipsfloat_vreg_chc      sSecond,                            /* second constant */
    Ipsfloat_vreg_chc      sThird,                             /* third constant */
    Ipsfloat_vreg_chc      sFourth,                            /* fourth constant */
    Ipsfloat_vreg_chc      sBreak,                             /* break point */
    Ipsgamma_vreg_chc      sOption                             /* gamma correction option */
);
```

get_colour_pixel

```
void IvGetColourPixel (                                        /* no return value output */
    Idnimage              nSourceImage,                       /* source image identifier */
    Ipsparameter_colour   stBand0,                            /* OUT band 0 pixel value pointer union */
    Ipsparameter_colour   stBand1,                            /* OUT band 1 pixel value pointer union */
    Ipsparameter_colour   stBand2,                            /* OUT band 2 pixel value pointer union */
    Ipsparameter_colour   stBand3,                            /* OUT band 3 pixel value pointer union */
    Ipuint                uHorizontal,                        /* horizontal index value */
    Ipuint                uVertical,                          /* vertical index value */
    Ipuint                uDepth,                             /* depth index value */
    Ipuint                uTemporal                          /* temporal index value */
);
```

get_colour_pixel_vreg

```
void IvGetColourPixelVReg (                                  /* no return value output */
    Ipsimage_vreg          snSourceImage,                     /* source image identifier */
    Idnvreg                nBand0VReg,                        /* OUT band 0 pixel virtual register identifier */
    Idnvreg                nBand1VReg,                        /* OUT band 1 pixel virtual register identifier */
    Idnvreg                nBand2VReg,                        /* OUT band 2 pixel virtual register identifier */
    Idnvreg                nBand3VReg,                        /* OUT band 3 pixel virtual register identifier */
    Ipsuint_vreg_chc       sHorizontal,                       /* horizontal index value */
    Ipsuint_vreg_chc       sVertical,                         /* vertical index value */
    Ipsuint_vreg_chc       sDepth,                           /* depth index value */
    Ipsuint_vreg_chc       sTemporal                          /* temporal index value */
);
```

get_pixel

```
Ipsparameter_pixel IstGetPixel (
    Idnimage          nSourceImage, /* OUT pixel value pointer union */
    Ipsparameter_pixel stPixel,      /* source image identifier */
    Ipoint            uHorizontal,    /* pixel value pointer union */
    Ipoint            uVertical,      /* horizontal index value */
    Ipoint            uDepth,         /* vertical index value */
    Ipoint            uTemporal,      /* depth index value */
    Ipoint            uBand,          /* temporal index value */
    Ipoint            uBand,          /* band index value */
    );
```

get_pixel_vreg

```
Idnvreg InGetPixelVReg (
    Idnvreg          nVReg,          /* OUT pixel value */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier */
    Ipsuint_vreg_chc sHorizontal,    /* source image identifier */
    Ipsuint_vreg_chc sVertical,      /* horizontal index value */
    Ipsuint_vreg_chc sDepth,         /* vertical index value */
    Ipsuint_vreg_chc sTemporal,      /* depth index value */
    Ipsuint_vreg_chc sBand,          /* temporal index value */
    Ipsuint_vreg_chc sBand,          /* band index value */
    );
```

get_pixel_roi

```
Ipbool IbGetPixelROI (
    Idnroi           nSourceROI,     /* OUT ROI pixel value */
    Ipbool           *tbPixel,       /* source ROI identifier */
    Ipoint           uHorizontal,    /* pixel value pointer */
    Ipoint           uVertical,      /* horizontal index value */
    Ipoint           uDepth,         /* vertical index value */
    Ipoint           uTemporal,      /* depth index value */
    Ipoint           uBand,          /* temporal index value */
    Ipoint           uBand,          /* band index value */
    );
```

get_pixel_roi_vreg

```
Idnvreg InGetPixelROIIVReg (
    Idnvreg          nVReg,          /* OUT ROI pixel value */
    Ipsroi_vreg      snSourceROI,    /* output virtual register identifier */
    Ipsuint_vreg_chc sHorizontal,    /* source ROI identifier */
    Ipsuint_vreg_chc sVertical,      /* horizontal index value */
    Ipsuint_vreg_chc sDepth,         /* vertical index value */
    Ipsuint_vreg_chc sTemporal,      /* depth index value */
    Ipsuint_vreg_chc sBand,          /* temporal index value */
    Ipsuint_vreg_chc sBand,          /* band index value */
    );
```

get_pixel_array

```

Ipsparameter_pixel IstGetPixelArray (                /* OUT pixel array pointer union */
    Idnimage                                           /* source image identifier */
    Ipsparameter_pixel                                /* pixel values array pointer union */
    Ipoint                                             /* first index offset */
    Ipoint                                             /* second index offset */
    Ipoint                                             /* third index value */
    Ipoint                                             /* fourth index value */
    Ipoint                                             /* fifth index value */
    Ipoint                                             /* number of array columns */
    Ipoint                                             /* number of array rows */
);

```

get_pixel_array_vreg

```

Idnvreg InGetPixelArrayVReg (                        /* OUT pixel array */
    Ipsimage_vreg                                     /* source image identifier */
    Idnvreg                                           /* output array virtual register identifier */
    Ipsuint_vreg_chc                                 /* first index offset */
    Ipsuint_vreg_chc                                 /* second index offset */
    Ipsuint_vreg_chc                                 /* third index value */
    Ipsuint_vreg_chc                                 /* fourth index value */
    Ipsuint_vreg_chc                                 /* fifth index value */
    Ipsuint_vreg_chc                                 /* number of array columns */
    Ipsuint_vreg_chc                                 /* number of array rows */
);

```

get_pixel_array_roi

```

Ipbool *ItbaGetPixelArrayROI (                      /* OUT pixel array pointer */
    Idnroi                                             /* source ROI identifier */
    Ipbool                                             /* tbaPixels */
    Ipoint                                             /* first index offset */
    Ipoint                                             /* second index offset */
    Ipoint                                             /* third index value */
    Ipoint                                             /* fourth index value */
    Ipoint                                             /* fifth index value */
    Ipoint                                             /* number of array columns */
    Ipoint                                             /* number of array rows */
);

```

get_pixel_array_roi_vreg

```
Idnvreg InGetPixelArrayROIvReg (          /* OUT pixel array */
    Ipsroi_vreg          snSourceROI,      /* source ROI identifier */
    Idnvreg              nPixelsVReg,      /* output array virtual register identifier */
    Ipsuint_vreg_chc     sOffset1,         /* first index offset */
    Ipsuint_vreg_chc     sOffset2,         /* second index offset */
    Ipsuint_vreg_chc     sValue3,          /* third index value */
    Ipsuint_vreg_chc     sValue4,          /* fourth index value */
    Ipsuint_vreg_chc     sValue5,          /* fifth index value */
    Ipsuint_vreg_chc     sColumns,         /* number of array columns */
    Ipsuint_vreg_chc     sRows             /* number of array rows */
);
```

get_pixel_record

```
Ipsparameter_pixel IstGetPixelRecord (    /* OUT pixel record pointer union */
    Idnimage             nSourceImage,     /* source image identifier */
    Ipsparameter_pixel   stPixel,          /* pixel record pointer union */
    Idntuple             nType,            /* data type SD B-tuple */
    Ipuint               uHorizontal,       /* horizontal index value */
    Ipuint               uVertical,        /* vertical index value */
    Ipuint               uDepth,           /* depth index value */
    Ipuint               uTemporal         /* temporal index value */
);
```

get_pixel_record_vreg

```
Idnvreg InGetPixelRecordVReg (            /* OUT pixel record */
    Ipsimage_vreg        snSourceImage,    /* source image identifier */
    Idnvreg              nPixelVReg,       /* pixel record virtual register identifier */
    Ipstuple_vreg        snType,           /* data type B-tuple */
    Ipsuint_vreg_chc     sHorizontal,      /* horizontal index value */
    Ipsuint_vreg_chc     sVertical,        /* vertical index value */
    Ipsuint_vreg_chc     sDepth,           /* depth index value */
    Ipsuint_vreg_chc     sTemporal         /* temporal index value */
);
```

get_pixel_record_roi

```

Ipbool *ItbaGetPixelRecordROI (                /* OUT pixel record pointer */
    Idnroi                                     /* source ROI identifier */
    Ipbool                                     /* pixel record value pointer */
    Ipuint                                     /* horizontal index value */
    Ipuint                                     /* vertical index value */
    Ipuint                                     /* depth index value */
    Ipuint                                     /* temporal index value */
);

```

get_pixel_record_roi_vreg

```

Idnvreg InGetPixelRecordROIvReg (              /* OUT pixel record */
    Ipsroi_vreg                               /* source ROI identifier */
    Idnvreg                                   /* pixel record virtual register identifier */
    Ipsuint_vreg_chc                          /* horizontal index value */
    Ipsuint_vreg_chc                          /* vertical index value */
    Ipsuint_vreg_chc                          /* depth index value */
    Ipsuint_vreg_chc                          /* temporal index value */
);

```

histogram_modification

```

Idnimage InHistModification (                  /* OUT destination image identifier */
    Idnimage                                   /* source image identifier */
    Idnimage                                   /* destination image identifier */
    Ipsparameter_arith                         /* output histogram upper value */
    Ipsparameter_arith                         /* output histogram lower value */
    Ipfloa                                     /* exponential spread factor */
    Ipuint                                     /* number of lookup table source offset entries */
    Ipint                                       /* histogram modification option */
);

```

histogram_modification_vreg

```
Idnvreg InHistModificationVReg (
    Idnvreg          nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier */
    Ipsparameter_arith_vreg_chc sUpperValue, /* destination image identifier */
    Ipsparameter_arith_vreg_chc sLowerValue, /* output histogram upper value */
    Ipsfloat_vreg_chc sSpreadFactor, /* output histogram lower value */
    Ipsuint_vreg_chc sEntries,       /* exponential spread factor */
    Ipsint_vreg_chc  sOption,        /* number of lookup table source offset entries */
    );                             /* histogram option */
```

histogram_scaling

```
Idnimage InHistScaling (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Ipsparameter_arith stLowerValue, /* destination image identifier */
    Ipsparameter_arith stUpperValue, /* lower output value pointer union */
    Ipsfloat           rLowerClip,    /* upper output value pointer union */
    Ipsfloat           rUpperClip,    /* lower input clip fraction */
    Ipsuint            uBins,         /* upper input clip fraction */
    );                             /* number of histogram bins */
```

histogram_scaling_vreg

```
Idnvreg InHistScalingVReg (
    Idnvreg          nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier */
    Ipsparameter_arith_vreg_chc sLowerValue, /* destination image identifier */
    Ipsparameter_arith_vreg_chc sUpperValue, /* lower output value */
    Ipsfloat_vreg_chc sLowerClip,    /* upper output value */
    Ipsfloat_vreg_chc sUpperClip,    /* lower input clip fraction */
    Ipsuint_vreg_chc sBins,         /* upper input clip fraction */
    );                             /* number of histogram bins */
```

histogram_shape_1d

```
Ipsfloat *ItraHistShape1D (
    Idnhist          nHist,          /* OUT shape array identifier */
    Ipsfloat         *traShape,      /* histogram identifier */
    Ipsint           iApplBufferSize, /* histogram shape array pointer */
    Ipsuint          uStartPoint,    /* data transfer buffer size or option */
    Ipsuint          *tuDataSize,    /* index offset within internal data */
    );                             /* OUT post transfer data size pointer */
```

histogram_shape_2d

```

Ipfloor *ItraHistShape2D (                                /* OUT shape array identifier */
    Idnhist          nHist,                                /* histogram identifier */
    Ipfloor          *traShape,                            /* histogram shape array pointer */
    Ipoint           iApplBufferSize,                      /* data transfer buffer size or option */
    Ipoint           uStartPoint,                          /* index offset within internal data */
    Ipoint           *tuDataSize                           /* OUT post transfer data size pointer */
);

```

histogram_1d

```

Idnhist InHist1D (                                        /* OUT histogram identifier */
    Idnimage         nSourceImage,                        /* source image identifier */
    Idnhist          nHist,                                /* histogram identifier */
    Ipoint           uBins,                                /* number of histogram bins */
    Ipfloor          rLowerBound,                          /* lower amplitude bound */
    Ipfloor          rUpperBound,                          /* upper amplitude bound */
    Ipoint           iOption                               /* histogram computation option */
);

```

histogram_1d_vreg

```

Idnvreg InHist1DVReg (                                    /* OUT histogram identifier */
    Idnvreg          nVReg,                                /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,                       /* source image identifier */
    Ipshist_vreg     snHist,                                /* histogram identifier */
    Ipsuint_vreg_chc sBins,                                /* number of histogram bins */
    Ipsfloat_vreg_chc sLowerBound,                          /* lower amplitude bound */
    Ipsfloat_vreg_chc sUpperBound,                          /* upper amplitude bound */
    Ipsint_vreg_chc  sOption                               /* histogram computation option */
);

```

histogram_2d

```

Idnhist InHist2D (                                        /* OUT histogram identifier */
    Idnimage         nSourceImage,                        /* source image identifier */
    Idnhist          nHist,                                /* histogram identifier */
    Ipoint           uBins,                                /* number of histogram bins in each dimension */
    Ipfloor          rLowerBound,                          /* lower amplitude bound */
    Ipfloor          rUpperBound,                          /* upper amplitude bound */
    Ipoint           uHorizontal,                          /* horizontal offset */
    Ipoint           uVertical                             /* vertical offset */
);

```

histogram_2d_vreg

```
Idnvreg InHist2DVReg (                                /* OUT histogram identifier */
    Idnvreg                                /* output virtual register identifier or NULL */
    Ipsimage_vreg    nVReg,                /* source image identifier */
    Ipshist_vreg     snSourceImage,        /* source image identifier */
    Ipsuint_vreg_chc snHist,              /* histogram identifier */
    Ipsfloat_vreg_chc sBins,              /* number of histogram bins in each dimension */
    Ipsfloat_vreg_chc sLowerBound,        /* lower amplitude bound */
    Ipsfloat_vreg_chc sUpperBound,        /* upper amplitude bound */
    Ipsuint_vreg_chc sHorizontal,         /* horizontal offset */
    Ipsuint_vreg_chc sVertical            /* vertical offset */
);
```

hit_or_miss

```
Idnimage InHitMiss (                                /* OUT destination image identifier */
    Idnimage                                /* source image identifier */
    Idnimage                                /* destination image identifier */
    Ipint    iterations,                    /* number of iterations */
    Ipuint    *tuCycles,                  /* OUT number of cycles */
    Ipint    iOption                      /* hit or miss transformation option */
);
```

hit_or_miss_vreg

```
Idnvreg InHitMissVReg (                            /* OUT destination image identifier */
    Idnvreg                                /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,        /* source image identifier */
    Ipsimage_vreg    snDestImage,         /* destination image identifier */
    Ipsint_vreg_chc  sIterations,          /* number of iterations */
    Idnvreg          nCycles,              /* OUT cycles virtual register identifier */
    Ipsint_vreg_chc  sOption              /* hit or miss transformation option */
);
```

hit_or_miss_roi

```
Idnroi InHitMissROI (                                /* OUT destination ROI identifier */
    Idnroi                                /* source ROI identifier */
    Idnroi                                /* destination ROI identifier */
    Ipint    iterations,                    /* number of iterations */
    Ipuint    *tuCycles,                  /* OUT number of cycles */
    Ipint    iOption                      /* hit or miss transformation option */
);
```

hit_or_miss_roi_vreg

Idnvreg InHitMissROIvReg (		/* OUT destination ROI identifier	*/
Idnvreg	nVReg,	/* output virtual register identifier or NULL	*/
Ipsroi_vreg	snSourceROI,	/* source ROI identifier	*/
Ipsroi_vreg	snDestROI,	/* destination ROI identifier	*/
Ipsint_vreg_chc	sIterations,	/* number of iterations	*/
Idnvreg	nCycles,	/* OUT cycles virtual register identifier	*/
Ipsint_vreg_chc	sOption	/* hit or miss transformation option	*/

);

hough_transform

Idnstatic InHoughTransform (		/* OUT Hough array identifier	*/
Idnimage	nSourceImage,	/* source image identifier	*/
Idnstatic	nHough,	/* Hough array identifier	*/
Ipsint	uColumns,	/* number of array columns	*/
Ipsint	uRows	/* number of array rows	*/

);

hough_transform_vreg

Idnvreg InHoughTransformVReg (		/* OUT Hough array identifier	*/
Idnvreg	nVReg,	/* output virtual register identifier or NULL	*/
Ipsimage_vreg	snSourceImage,	/* source image identifier	*/
Ipsstatic_vreg	snHough,	/* Hough array identifier	*/
Ipsuint_vreg_chc	sColumns,	/* number of array columns	*/
Ipsuint_vreg_chc	sRows	/* number of array rows	*/

);

image_bar_chart

Idnimage InImageBarChart (		/* OUT destination image identifier	*/
Idnimage	nDestImage,	/* destination image identifier	*/
Idntuple	nSize,	/* image size ND 5-tuple identifier	*/
Ipsint	uMaximum,	/* maximum integer size	*/
Ipint	iStructure,	/* image structure option	*/
Ipint	iType	/* destination data type (internal) option	*/

);

image_bar_chart_vreg

```
Idnvreg InImageBarChartVReg (                                /* OUT destination image identifier */
    Idnvreg                                nVReg,             /* output virtual register identifier or NULL */
    Ipsimage_vreg                          snDestImage,       /* destination image identifier */
    Ipstuple_vreg                          snSize,            /* image size ND 5-tuple identifier */
    Ipsuint_vreg_chc                      sMaximum,          /* maximum integer size */
    Ipsint_vreg_chc                       sStructure,         /* image structure option */
    Ipsint_vreg_chc                       sType               /* destination data type (internal) option */
);
```

image_constant

```
Idnimage InImageConstant (                                    /* OUT destination image identifier */
    Idnimage                                nDestImage,       /* destination image identifier */
    Idntuple                              nSize,             /* image size ND 5-tuple identifier */
    Ipsparameter_pixel                    stValue,            /* value of constant pointer union */
    Ipint                                 iTyep               /* destination data type (internal) option */
);
```

image_constant_vreg

```
Idnvreg InImageConstantVReg (                                /* OUT destination image identifier */
    Idnvreg                                nVReg,             /* output virtual register identifier or NULL */
    Ipsimage_vreg                          snDestImage,       /* destination image identifier */
    Ipstuple_vreg                          snSize,            /* image size ND 5-tuple identifier */
    Ipsparameter_pixel_vreg_chc          sValue,              /* value of constant */
    Ipsint_vreg_chc                      sType               /* destination data type (internal) option */
);
```

image_gaussian

```
Idnimage InImageGauss (                                       /* OUT destination image identifier */
    Idnimage                                nDestImage,       /* destination image identifier */
    Idntuple                              nSize,             /* image size ND 5-tuple identifier */
    Ipfloat                               rHorizontalCentre, /* horizontal centre */
    Ipfloat                               rVerticalCentre,   /* vertical centre */
    Ipfloat                               rHorizontalSDev,    /* horizontal standard deviation */
    Ipfloat                               rVerticalSDev       /* vertical standard deviation */
);
```

image_gaussian_vreg

```
Idnvreg InImageGaussVReg (                                /* OUT destination image identifier */
    Idnvreg          nVReg,                                /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,                          /* destination image identifier */
    Ipstuple_vreg    snSize,                                /* image size ND 5-tuple identifier */
    Ipsfloat_vreg_chc sHorizontalCentre,                    /* horizontal centre */
    Ipsfloat_vreg_chc sVerticalCentre,                      /* vertical centre */
    Ipsfloat_vreg_chc sHorizontalSDev,                      /* horizontal standard deviation */
    Ipsfloat_vreg_chc sVerticalSDev                        /* vertical standard deviation */
);
```

image_grey_scale

```
Idnimage InImageGreyScale (                                /* OUT destination image identifier */
    Idnimage          nDestImage,                          /* destination image identifier */
    Idntuple          nSize,                                /* image size ND 5-tuple identifier */
    Ipuint            uMaximum,                            /* maximum positive integer size */
    Ipint             iGreyOption,                         /* grey scale orientation option */
    Ipint             iTypeOption                          /* destination data type option */
);
```

image_grey_scale_vreg

```
Idnvreg InImageGreyScaleVReg (                            /* OUT destination image identifier */
    Idnvreg          nVReg,                                /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,                          /* destination image identifier */
    Ipstuple_vreg    snSize,                                /* image size ND 5-tuple identifier */
    Ipsuint_vreg_chc sMaximum,                             /* maximum positive integer size */
    Ipsint_vreg_chc  sGreyOption,                          /* grey scale orientation option */
    Ipsint_vreg_chc  sTypeOption                           /* destination data type option */
);
```

image_random_number

```
Idnimage InImageRandomNumber (                            /* OUT destination image identifier */
    Idnimage          nDestImage,                          /* destination image identifier */
    Idntuple          nSize,                                /* image size ND 5-tuple identifier */
    Ipfloat            rMean,                               /* probability density mean */
    Ipfloat            rSDev,                               /* probability standard deviation */
    Ipfloat            rMaximum,                           /* pixel amplitude maximum value */
    Ipfloat            rMinimum,                           /* pixel amplitude minimum value */
    Ipint             iOption                              /* probability density option */
);
```

image_random_number_vreg

```
Idnvreg InImageRandomNumberVReg (          /* OUT destination image identifier */
    Idnvreg          nVReg,                /* output virtual register identifier or NULL */
    Ipsimage_vreg     snDestImage,         /* destination image identifier */
    Ipstuple_vreg     snSize,              /* image size 5-tuple identifier */
    Ipsfloat_vreg_chc sMean,               /* probability density mean */
    Ipsfloat_vreg_chc sSDev,              /* probability standard deviation */
    Ipsfloat_vreg_chc sMaximum,           /* pixel amplitude maximum value */
    Ipsfloat_vreg_chc sMinimum,          /* pixel amplitude minimum value */
    Ipsint_vreg_chc   sOption             /* probability density option */
);
```

import_histogram

```
Idnhist InImportHist (          /* OUT histogram object identifier */
    Ipuint          *tuaData,             /* external data value array pointer */
    Idnhist         nHist,               /* histogram object identifier */
    Ipsattrs_hist   sAttrs               /* histogram attributes */
);
```

import_histogram_vreg

```
Idnvreg InImportHistVReg (      /* OUT histogram object identifier */
    Idnvreg          nVReg,                /* output virtual register identifier or NULL */
    Ipsaint_vreg_chc sData,              /* external data value array */
    Ipshist_vreg     snHist,             /* histogram object identifier */
    Ipsattrs_hist_vreg_chc sAttrs        /* histogram attributes */
);
```

import_image

```
Idnimage InImportImage (        /* OUT destination image identifier */
    Ipspiks_pixel_types stSourceImage,    /* external source image data pointer union */
    Idnimage          nDestImage,        /* destination image identifier */
    Ipuint            uPrecision,        /* number of source bits per pixel */
    Ipin              iType,             /* source data type option */
    Ipin              iApplBufferSize,   /* data transfer buffer size or option */
    Ipuint            uStartPoint,       /* index offset within internal data */
    Ipuint            *tuDataSize        /* OUT post transfer data size pointer */
);
```

import_image_vreg

```
Idnvreg InImportImageVReg (
    Idnvreg          nVReg,          /* OUT destination image identifier */
    Ipspiks_pixel_types_vreg_chc sSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg     snDestImage,  /* external source image data */
    Ipsuint_vreg_chc  sPrecision,   /* destination image identifier */
    Ipsint_vreg_chc   sType         /* number of source bits per pixel */
    /* source data type option */
);
```

import_lut

```
Idnlut InImportLUT (
    Ipsparameter_pixel stData,      /* OUT destination lookup table identifier */
    Idnlut             nLUT,        /* external table of data values pointer union */
    Ipsattrs_lut       sAttrs,      /* lookup table identifier */
    Ipint              iApplBufferSize, /* lookup table attributes */
    Ipuint             uStartPoint,  /* data transfer buffer size or option */
    Ipuint             *tuDataSize,  /* index offset within internal data */
    /* OUT post transfer data size pointer */
);
```

import_lut_vreg

```
Idnvreg InImportLUTVReg (
    Idnvreg          nVReg,          /* OUT destination lookup table identifier */
    Ipsparameter_pixel_vreg_chc sData, /* output virtual register identifier or NULL */
    Ipslut_vreg       snLUT,        /* external table of data values */
    Ipsattrs_lut_vreg_chc sAttrs,   /* lookup table identifier */
    /* lookup table attributes */
);
```

import_matrix

```
Idnmatrix InImportMatrix (
    Ipsparameter_numeric stData,      /* OUT destination matrix identifier */
    Idnmatrix           nMatrix,      /* external table of data values pointer union */
    Ipsattrs_matrix     sAttrs        /* matrix identifier */
    /* matrix attributes */
);
```

import_matrix_vreg

```
Idnvreg InImportMatrixVReg (
    Idnvreg          nVReg,          /* OUT destination matrix identifier */
    Ipsparameter_numeric_vreg_chc sData, /* output virtual register identifier or NULL */
    Ipsmatrix_vreg     snMatrix,      /* external table of data values */
    Ipsattrs_matrix_vreg_chc sAttrs,  /* matrix identifier */
    /* matrix attributes */
);
```

import_neighbourhood_array

```
Idnnbhood InImportNbhoodArray (          /* OUT destination array identifier */
    Ipsparameter_pixel      stData,        /* neighbourhood array data pointer union */
    Idnnbhood               nNbhood,       /* neighbourhood array identifier */
    Ipsattrs_nbhood         sAttrs        /* neighbourhood array attributes */
);
```

import_neighbourhood_array_vreg

```
Idnvreg InImportNbhoodArrayVReg (        /* OUT destination array identifier */
    Idnvreg                 nVReg,         /* output virtual register identifier or NULL */
    Ipsparameter_pixel_vreg_chc sData,     /* neighbourhood array data */
    Ipsnbhood_vreg          snNbhood,      /* neighbourhood array data identifier */
    Ipsattrs_nbhood_vreg_chc sAttrs       /* neighbourhood array attributes */
);
```

import_roi_array

```
Idnroi InImportROIArray (                /* OUT destination ROI identifier */
    Ipbool                               *tbaData, /* external data value array pointer */
    Idnroi                               nROI,      /* ROI array identifier */
    Ipsattrs_roi                         sAttrs,    /* ROI attributes */
    Ipint                               iApplBufferSize, /* data transfer buffer size or option */
    Ipuint                              uStartPoint, /* index offset within internal data */
    Ipuint                              *tuDataSize /* OUT post transfer data size pointer */
);
```

import_roi_array_vreg

```
Idnvreg InImportROIArrayVReg (           /* OUT destination ROI identifier */
    Idnvreg                             nVReg,     /* output virtual register identifier or NULL */
    Ipsabool_vreg_chc                   sData,      /* external data value array */
    Ipsroi_vreg                         snROI,      /* ROI array identifier */
    Ipsattrs_roi_vreg_chc               sAttrs     /* ROI attributes */
);
```

import_static_array

```
Idnstatic InImportStaticArray (
    Ipsparameter_pixel    stData, /* OUT destination static array identifier */
    Idnstatic             nArray, /* external static array data pointer union */
    Ipsattrs_static       sAttrs, /* static array object identifier */
    Ipint                 iApplBufferSize, /* static array attributes */
    Ipuint                uStartPoint, /* data transfer buffer size or option */
    Ipuint                *tuDataSize, /* index offset within internal data */
    /* OUT post transfer data size pointer */
);
```

import_static_array_vreg

```
Idnvreg InImportStaticArrayVReg (
    Idnvreg             nVReg, /* OUT destination static array identifier */
    Ipsparameter_pixel_vreg_chc sData, /* output virtual register identifier or NULL */
    Ipsstatic_vreg       snArray, /* external static array data */
    Ipsattrs_static_vreg_chc sAttrs, /* static array object identifier */
    /* static array attributes */
);
```

import_tuple

```
Idntuple InImportTuple (
    Ipsparameter_basic    stTupleData, /* OUT tuple identifier */
    Idntuple              nTupleId, /* external tuple data pointer union */
    Ipsattrs_tuple        sTupleAttrs, /* tuple identifier */
    /* static array attributes */
);
```

import_tuple_vreg

```
Idnvreg InImportTupleVReg (
    Idnvreg             nVReg, /* OUT tuple identifier */
    Ipsparameter_basic_vreg_chc sTupleData, /* output virtual register identifier or NULL */
    Ipstuple_vreg       snTupleId, /* external tuple data */
    Ipsattrs_tuple_vreg_chc sTupleAttrs, /* tuple identifier */
    /* static array attributes */
);
```

import_value_bounds

```
Idnbounds InImportValueBounds (          /* OUT value bounds collection identifier */
    Ipsparameter_arith    stData,          /* external data value collection pointer union */
    Idnbounds             nValueBounds,    /* value bounds collection identifier */
    Ipsattrs_bounds       sAttrs,          /* value bounds collection attributes */
    Ipoint                iApplBufferSize, /* data transfer buffer size or option */
    Ipoint                uStartPoint,      /* index offset within internal data */
    Ipoint                *tuDataSize       /* OUT post transfer data size pointer */
);
```

import_value_bounds_vreg

```
Idnvreg InImportValueBoundsVReg (          /* OUT value bounds collection identifier */
    Idnvreg               nVReg,           /* output virtual register identifier or NULL */
    Ipsparameter_arith_vreg_chc sData,     /* external data value collection */
    Ipsbounds_vreg        snValue,         /* value bounds collection identifier */
    Ipsattrs_bounds_vreg_chc sAttrs        /* value bounds collection attributes */
);
```

impulse_boxcar

```
Idnnbhood InImpulseBoxcar (                /* OUT impulse response array identifier */
    Idnnbhood             nArray,          /* impulse response array identifier */
    Idntuple              nSize,           /* array size ND 5-tuple identifier */
    Idntuple              nKey,            /* key pixel SD 5-tuple identifier */
    Ipederiv              eOption          /* derivative option */
);
```

impulse_boxcar_vreg

```
Idnvreg InImpulseBoxcarVReg (              /* OUT impulse response array identifier */
    Idnvreg               nVReg,           /* output virtual register identifier or NULL */
    Ipsnbhood_vreg        snArray,         /* impulse response array identifier */
    Ipstuple_vreg         snSize,          /* array size ND 5-tuple identifier */
    Ipstuple_vreg         snKey,           /* key pixel SD 5-tuple identifier */
    Ipsderiv_vreg_chc     seOption         /* derivative option */
);
```

impulse_derivative_of_gaussian

```
Idnnbhood InImpulseDerivGauss ( /* OUT impulse response array identifier */
    Idnnbhood          nArray, /* impulse response array identifier */
    Idntuple           nSize, /* array size ND 5-tuple identifier */
    Idntuple           nKey, /* key pixel SD 5-tuple identifier */
    Ipfloat            rColumnSDev, /* column index standard deviation */
    Ipfloat            rRowSDev, /* row index standard deviation */
    Ipderiv            eOption /* derivative option */
);
```

impulse_derivative_of_gaussian_vreg

```
Idnvreg InImpulseDerivGaussVReg ( /* OUT impulse response array identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsnbhood_vreg   snArray, /* impulse response array identifier */
    Ipstuple_vreg    snSize, /* array size ND 5-tuple identifier */
    Ipstuple_vreg    snKey, /* key pixel SD 5-tuple identifier */
    Ipsfloat_vreg_chc sColumnSDev, /* column index standard deviation */
    Ipsfloat_vreg_chc sRowSDev, /* row index standard deviation */
    Ipsderiv_vreg_chc sOption /* derivative option */
);
```

impulse_difference_of_gaussians

```
Idnnbhood InImpulseDiffGauss ( /* OUT impulse response array identifier */
    Idnnbhood          nArray, /* impulse response array identifier */
    Idntuple           nSize, /* array size ND 5-tuple identifier */
    Idntuple           nKey, /* key pixel SD 5-tuple identifier */
    Ipfloat            rColumnWide, /* column index wide standard deviation */
    Ipfloat            rRowWide, /* row index wide standard deviation */
    Ipfloat            rColumnNarrow, /* column index narrow standard deviation */
    Ipfloat            rRowNarrow /* row index narrow standard deviation */
);
```

impulse_difference_of_gaussians_vreg

```
Idnvreg InImpulseDiffGaussVReg ( /* OUT impulse response array identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsnbhood_vreg   snArray, /* impulse response array identifier */
    Ipstuple_vreg    snSize, /* array size ND 5-tuple identifier */
    Ipstuple_vreg    snKey, /* key pixel SD 5-tuple identifier */
    Ipsfloat_vreg_chc sColumnWide, /* column index wide standard deviation */
    Ipsfloat_vreg_chc sRowWide, /* row index wide standard deviation */
    Ipsfloat_vreg_chc sColumnNarrow, /* column index narrow standard deviation */
    Ipsfloat_vreg_chc sRowNarrow /* row index narrow standard deviation */
);
```

impulse_elliptical

Idnnbhood InImpulseElliptical (		/* OUT impulse response array identifier	*/
Idnnbhood	nArray,	/* impulse response array identifier	*/
Idntuple	nSize,	/* array size ND 5-tuple identifier	*/
Idntuple	nKey	/* key pixel SD 5-tuple identifier	*/

);

impulse_elliptical_vreg

Idnvreg InImpulseEllipticalVReg (		/* OUT impulse response array identifier	*/
Idnvreg	nVReg,	/* output virtual register identifier or NULL	*/
Ipsnbhood_vreg	snArray,	/* impulse response array identifier	*/
Ipstuple_vreg	snSize,	/* array size ND 5-tuple identifier	*/
Ipstuple_vreg	snKey	/* key pixel SD 5-tuple identifier	*/

);

impulse_gaussian

Idnnbhood InImpulseGauss (		/* OUT impulse response array identifier	*/
Idnnbhood	nArray,	/* impulse response array identifier	*/
Idntuple	nSize,	/* array size ND 5-tuple identifier	*/
Idntuple	nKey,	/* key pixel SD 5-tuple identifier	*/
Ipfloat	rHorizontalSDev,	/* horizontal standard deviation	*/
Ipfloat	rVerticalSDev	/* vertical standard deviation	*/

);

impulse_gaussian_vreg

Idnvreg InImpulseGaussVReg (		/* OUT impulse response array identifier	*/
Idnvreg	nVReg,	/* output virtual register identifier or NULL	*/
Ipsnbhood_vreg	snArray,	/* impulse response array identifier	*/
Ipstuple_vreg	snSize,	/* array size ND 5-tuple identifier	*/
Ipstuple_vreg	snKey,	/* key pixel SD 5-tuple identifier	*/
Ipsfloat_vreg_chc	sHorizaontalSDev,	/* horizontal standard deviation	*/
Ipsfloat_vreg_chc	sVerticalSDev	/* vertical standard deviation	*/

);

impulse_laplacian_of_gaussian

Idnnbhood InImpulseLaplaceGauss (		/* OUT impulse response array identifier	*/
Idnnbhood	nArray,	/* impulse response array identifier	*/
Idntuple	nSize,	/* array size ND 5-tuple identifier	*/
Idntuple	nKey,	/* key pixel SD 5-tuple identifier	*/
Ipfloat	rSDev	/* standard deviation	*/

);

impulse_laplacian_of_gaussian_vreg

```
Idnvreg InImpulseLaplaceGaussVReg (          /* OUT impulse response array identifier */
    Idnvreg          nVReg,                  /* output virtual register identifier or NULL */
    Ipsnbhood_vreg   snArray,                /* impulse response array identifier */
    Ipstuple_vreg     snSize,                /* array size ND 5-tuple identifier */
    Ipstuple_vreg     snKey,                 /* key pixel SD 5-tuple identifier */
    Ipsfloat_vreg_chc sSDev                  /* standard deviation */
);
```

impulse_pyramid

```
Idnnbhood InImpulsePyramid (                /* OUT impulse response array identifier */
    Idnnbhood        nArray,                /* impulse response array identifier */
    Idntuple          nSize,                /* array size ND 5-tuple identifier */
    Idntuple          nKey,                 /* key pixel SD 5-tuple identifier */
    Ipint             iOption              /* impulse, pyramid shape option */
);
```

impulse_pyramid_vreg

```
Idnvreg InImpulsePyramidVReg (              /* OUT impulse response array identifier */
    Idnvreg          nVReg,                /* output virtual register identifier or NULL */
    Ipsnbhood_vreg   snArray,              /* impulse response array identifier */
    Ipstuple_vreg     snSize,              /* array size ND 5-tuple identifier */
    Ipstuple_vreg     snKey,               /* key pixel SD 5-tuple identifier */
    Ipsint_vreg_chc  sOption              /* impulse, pyramid shape option */
);
```

impulse_rectangular

```
Idnnbhood InImpulseRectangular (            /* OUT impulse response array identifier */
    Idnnbhood        nArray,              /* impulse response array identifier */
    Idntuple          nSize,              /* array size ND 5-tuple identifier */
    Idntuple          nKey                /* key pixel SD 5-tuple identifier */
);
```

impulse_rectangular_vreg

```
Idnvreg InImpulseRectangularVReg (          /* OUT impulse response array identifier */
    Idnvreg          nVReg,                /* output virtual register identifier or NULL */
    Ipsnbhood_vreg   snArray,              /* impulse response array identifier */
    Ipstuple_vreg     snSize,              /* array size ND 5-tuple identifier */
    Ipstuple_vreg     snKey                /* key pixel SD 5-tuple identifier */
);
```

impulse_sinc

```
Idnnbhood InImpulseSinc (
    Idnnbhood      nArray,      /* OUT impulse response array identifier */
    Idntuple       nSize,       /* impulse response array identifier */
    Idntuple       nKey,        /* array size ND 5-tuple identifier */
    Ipfloat        rHorizontalScale, /* key pixel SD 5-tuple identifier */
    Ipfloat        rVerticalScale, /* horizontal scale factor */
    Ipfloat        rVerticalScale /* vertical scale factor */
);
```

impulse_sinc_vreg

```
Idnvreg InImpulseSincVReg (
    Idnvreg      nVReg,      /* OUT impulse response array identifier */
    Ipsnbhood_vreg snArray,  /* output virtual register identifier or NULL */
    Ipstuple_vreg snSize,    /* impulse response array identifier */
    Ipstuple_vreg snKey,     /* array size ND 5-tuple identifier */
    Ipsfloat_vreg_chc sHorizontalScale, /* key pixel SD 5-tuple identifier */
    Ipsfloat_vreg_chc sVerticalScale,   /* horizontal scale factor */
    Ipsfloat_vreg_chc sVerticalScale    /* vertical scale factor */
);
```

input_object

```
Ipsobject IsnInputObject (
    Ipsport      nExternal, /* OUT PIKS identifier */
    Ipsobject    snInternal, /* external identifier */
    Ipint        iTransferOption, /* PIKS identifier */
    Ipint        iSyntaxOption, /* source data transfer option */
    Ipint        iApplBufferSize, /* source data transfer syntax option */
    Ipuint       uStartPoint, /* data transfer buffer size or option */
    Ipuint       *tuDataSize, /* index offset within internal data */
    Ipuint       /* OUT post transfer data size pointer */
);
```

input_object_vreg

```
Idnvreg InInputObjectVReg (
    Idnvreg      nVReg,      /* OUT PIKS identifier */
    Ipsport_vreg snExternal, /* output virtual register identifier or NULL */
    Ipsobject_vreg snInternal, /* external identifier */
    Ipsint_vreg_chc sTransfer, /* PIKS identifier */
    Ipsint_vreg_chc sSyntax,   /* source data transfer option */
    Ipsint_vreg_chc sSyntax    /* source data transfer syntax option */
);
```

inquire_chain_environment

```
void IvInquireChainEnvironment (           /* no return value output */
    char          *tzInformation,          /* OUT chaining attribute list */
    Ipevalidity   *teValid,               /* OUT data validity indicator */
    Ipint         iApplBufferSize,        /* data transfer buffer size or option */
    Ipoint        uStartPoint,            /* index offset within internal data */
    Ipoint        *tuDataSize             /* OUT post transfer data size pointer */
);
```

inquire_chain_status

```
void IvInquireChainStatus (               /* no return value output */
    Idnchain      nChain,                 /* chain identifier */
    Ipoint        *uStatus,               /* OUT chain status */
    Ipevalidity   *teValid                /* OUT data validity indicator */
);
```

inquire_elements

```
void IvInquireElements (                 /* no return value output */
    Ipint         *tiaInput,              /* array of element registration numbers pointer */
    Ipbool        *tbaOutput,             /* OUT array of registration responses pointer */
    Ipoint        uSize,                  /* array size */
    char          *tzLabel,               /* OUT private label information */
    Ipevalidity   *teValid                /* OUT data validity indicator */
);
```

inquire_image

```
void IvInquireImage (                   /* no return value output */
    Idnimage      nSourceImage,           /* source image identifier */
    Ipsattrs_image *tsAttrs,               /* OUT image attributes */
    Ipsbind_image *tsBind,                /* OUT image match point and ROI bindings */
    Ipevalidity   *teValid                /* OUT data validity indicator */
);
```

inquire_index_assignment

```
void IvInquireIndexAssignment (
    char          *tcFirst,      /* no return value output */
    char          *tc2nd,       /* OUT first operator index assignment */
    char          *tcThird,     /* OUT second operator index assignment */
    char          *tcFourth,    /* OUT third operator index assignment */
    char          *tcFifth,     /* OUT fourth operator index assignment */
    Ipevalidity   *teValid,     /* OUT fifth operator index assignment */
    Ipevalidity   *teValid,     /* OUT data validity indicator */
);
```

inquire_non_image_object

```
void IvInquireNonImageObject (
    Ipsnon_image_object snObject, /* no return value output */
    Ipsattrs_non_image stData,    /* object identifier */
    Ipint               iType,     /* OUT non-image attributes */
    Ipevalidity         *teValid,  /* object type option */
    Ipevalidity         *teValid,  /* OUT data validity indicator */
);
```

inquire_piks_implementation

```
void IvInquirePIKSImpl (
    Ipspiks_impl *tsData, /* no return value output */
    Ipevalidity  *teValid, /* OUT implementation information */
    Ipint        iApplBufferSize, /* OUT data validity indicator */
    Ipuint       uStartPoint, /* data transfer buffer size or option */
    Ipuint       *tuDataSize, /* index offset within internal data */
    Ipuint       *tuDataSize, /* OUT post transfer data size pointer */
);
```

inquire_piks_status

```
void IvInquirePIKSStatus (
    Ipspiks_states *tsStates, /* no return value output */
    Ipspiks_modes  *tsModes, /* OUT inquire PIKS states pointer */
    Ipevalidity     *teValid, /* OUT inquire PIKS modes pointer */
    Ipevalidity     *teValid, /* OUT data validity indicator */
);
```

inquire_repository

```
void IvInquireRepository (
    Ipint          /* no return value output */
    Ipbool         /* repository registration numbers array pointer */
    Ipoint         /* OUT registration responses array pointer */
    char           /* array size */
    Ipint          /* OUT private label information */
    Ipevalidity    /* repository class option */
    *tiaInput,     /* OUT data validity indicator */
    *tbaOutput,
    uSize,
    *tzLabel,
    iClassOption,
    *teValid
);
```

inquire_resampling

```
void IvInquireResampling (
    Ipint          /* no return value output */
    Ipbool         /* resampling registration numbers array pointer */
    Ipoint         /* OUT resampling responses array pointer */
    char           /* array size */
    Ipint          /* OUT private label information */
    Ipevalidity    /* OUT data validity indicator */
    *tiaInput,
    *tbaOutput,
    uSize,
    *tzLabel,
    *teValid
);
```

insert_pixel_plane

```
Idnimage InInsertPixelPlane (
    Idnimage       /* OUT destination image identifier */
    Idnimage       /* source image identifier */
    Idnimage       /* destination image identifier */
    Ipoint         /* depth index value */
    Ipoint         /* temporal index value */
    Ipoint         /* band index value */
    nSourceImage,
    nDestImage,
    uDepth,
    uTemporal,
    uBand
);
```

insert_pixel_plane_vreg

```
Idnvreg InInsertPixelPlaneVReg (
    Idnvreg        /* OUT destination image identifier */
    Idnvreg        /* output virtual register identifier or NULL */
    Ipsimage_vreg  /* source image identifier */
    Ipsimage_vreg  /* destination image identifier */
    Ipsuint_vreg_chc /* depth index value */
    Ipsuint_vreg_chc /* temporal index value */
    Ipsuint_vreg_chc /* band index value */
    nVReg,
    snSourceImage,
    snDestImage,
    sDepth,
    sTemporal,
    sBand
);
```

insert_pixel_plane_roi

```
Idnroi InInsertPixelPlaneROI (
    Idnroi          nSourceROI, /* OUT destination ROI identifier */
    Idnroi          nDestROI,   /* source ROI identifier */
    Ipuint          uDepth,     /* destination ROI identifier */
    Ipuint          uTemporal,  /* depth index value */
    Ipuint          uBand,      /* temporal index value */
    Ipuint          /* band index value */
);
```

insert_pixel_plane_roi_vreg

```
Idnvreg InInsertPixelPlaneROIvReg (
    Idnvreg          nVReg,      /* OUT destination ROI identifier */
    Ipsroi_vreg      snSourceROI, /* output virtual register identifier or NULL */
    Ipsroi_vreg      snDestROI,  /* source ROI identifier */
    Ipsuint_vreg_chc sDepth,     /* destination ROI identifier */
    Ipsuint_vreg_chc sTemporal,  /* depth index value */
    Ipsuint_vreg_chc sBand,      /* temporal index value */
    Ipsuint_vreg_chc /* band index value */
);
```

label_objects

```
Idnimage InLabelObjects (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Ipuint            uIndex3,      /* destination image identifier */
    Ipuint            uIndex4,      /* third index value */
    Ipuint            uIndex5,      /* fourth index value */
    Ipuint            uLimit,       /* fifth index value */
    Ipuint            *tuObjects,   /* object number limit */
    Ipuint            /* OUT number of objects labeled */
);
```

label_objects_vreg

```
Idnvreg InLabelObjectsVReg (
    Idnvreg          nVReg,      /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,  /* source image identifier */
    Ipsuint_vreg_chc sIndex3,      /* destination image identifier */
    Ipsuint_vreg_chc sIndex4,      /* third index value */
    Ipsuint_vreg_chc sIndex5,      /* fourth index value */
    Ipsuint_vreg_chc sLimit,       /* fifth index value */
    Idnvreg          nObjectsVReg, /* number limit */
    Idnvreg          /* OUT number of objects labeled */
);
```

laws_texture_features

```
Idnimage InLawsTextureFeatures (                /* OUT destination image identifier */
    Idnimage          nSourceImage,             /* source image identifier */
    Idnimage          nDestImage,               /* destination image identifier */
    Idnnbhood         nFilter,                  /* low pass filter response array identifier */
    Idnnbhood         nMask                     /* feature mask response array identifier */
);
```

laws_texture_features_vreg

```
Idnvreg InLawsTextureFeaturesVReg (              /* OUT destination image identifier */
    Idnvreg          nVReg,                     /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,             /* source image identifier */
    Ipsimage_vreg    snDestImage,              /* destination image identifier */
    Ipsnbhood_vreg   snFilter,                  /* low pass filter response array identifier */
    Ipsnbhood_vreg   snMask                     /* feature mask response array identifier */
);
```

level_slice

```
Idnimage InLevelSlice (                          /* OUT destination image identifier */
    Idnimage          nSourceImage,             /* source image identifier */
    Idnimage          nDestImage,              /* destination image identifier */
    Ipsparameter_arith stUpper,                 /* upper slice amplitude pointer union */
    Ipsparameter_arith stLower,                /* lower slice amplitude pointer union */
    Ipsparameter_arith stSlice,                /* slice output value pointer union */
    lpebackground     eOption                   /* level slice background option */
);
```

level_slice_vreg

```
Idnvreg InLevelSliceVReg (                       /* OUT destination image identifier */
    Idnvreg          nVReg,                     /* output virtual register identifier or NULL */
    Ipsimage_vreg    nSourceImage,             /* source image identifier */
    Ipsimage_vreg    nDestImage,              /* destination image identifier */
    Ipsparameter_arith_vreg_chc sUpper,        /* upper slice amplitude */
    Ipsparameter_arith_vreg_chc sLower,        /* lower slice amplitude */
    Ipsparameter_arith_vreg_chc sSlice,        /* slice output value */
    lpsbackground_vreg_chc sOption              /* level slice background option */
);
```

line_profile

```

Ipfloat *ItraLineProfile (                                /* OUT array of pixels along line pointer */
    Idnimage          nSourceImage, /* source image identifier */
    Ipfloat          *traDestArray, /* array of pixels along line objects pointer */
    Idntuple         nStart, /* start point ND 5-tuple */
    Idntuple         nEnd, /* end point ND 5-tuple */
    Ipoint           uSamples, /* number of samples along line */
    Ipoint           iApplBufferSize, /* data transfer buffer size or option */
    Ipoint           uStartPoint, /* index offset within internal data */
    Ipoint           *tuDataSize /* OUT post transfer data size pointer */
);

```

line_profile_vreg

```

Idnvreg InLineProfileVReg (                                /* OUT array of pixels along line pointer */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Idnvreg         nDestVReg, /* pixels along line virtual register identifier */
    Ipstuple_vreg    snStart, /* start point ND 5-tuple */
    Ipstuple_vreg    snEnd, /* end point ND 5-tuple */
    Ipsuint_vreg_chc sSamples /* number of samples along line */
);

```

lookup

```

Idnimage InLookup (                                /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage, /* destination image identifier */
    Idnlut           nLUT, /* lookup table identifier */
    Ipoint           uOffset, /* source offset entry */
    Ipoint           iOption /* lookup table mode option */
);

```

lookup_vreg

```

Idnvreg InLookupVReg (                                /* OUT destination image identifier */
    Idnvreg         nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage, /* destination image identifier */
    Ipslut_vreg     snLUT, /* lookup table identifier */
    Ipsuint_vreg_chc sOffset, /* source offset entry */
    Ipsint_vreg_chc sOption /* lookup table mode option */
);

```

lookup_interpolated

```

Idnimage InLookupInterp (
    Idnimage          /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Idnlut            nLUT,         /* lookup table identifier */
    Ipuint            uOffset       /* source offset bits */
);

```

lookup_interpolated_vreg

```

Idnvreg InLookupInterpVReg (
    Idnvreg          /* OUT destination image identifier */
    Ipsimage_vreg    nVReg,        /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,   /* destination image identifier */
    Ipslut_vreg      snLUT,         /* lookup table identifier */
    Ipsuint_vreg_chc sOffset       /* source offset bits */
);

```

luminance_generation

```

Idnimage InLumGeneration (
    Idnimage          /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Ipint             iOption       /* luminance RGB colour option */
);

```

luminance_generation_vreg

```

Idnvreg InLumGenerationVReg (
    Idnvreg          /* OUT destination image identifier */
    Ipsimage_vreg    nVReg,        /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,   /* destination image identifier */
    Ipsint_vreg_chc  sOption       /* luminance RGB colour option */
);

```

markov_spectrum

```

Idnstatic InMarkovSpectrum (
    Idnstatic          /* OUT power spectrum static array identifier */
    Idntuple           nFilter,    /* power spectrum static array identifier */
    Ipfloating         nSize,      /* array size ND 5-tuple identifier */
    Ipfloating         rColumn,    /* column index correlation factor */
    Ipfloating         rRow,       /* row index correlation factor */
    Ipint              iOption     /* Markov spectrum symmetry option */
);

```

markov_spectrum_vreg

```
Idnvreg InMarkovSpectrumVReg (
    Idnvreg          nVReg,          /* OUT power spectrum static array identifier */
    Ipsstatic_vreg   snFilter,       /* output virtual register identifier or NULL */
    Ipstuple_vreg     snSize,        /* power spectrum static array identifier */
    Ipsfloat_vreg_chc sColumn,       /* array size ND 5-tuple identifier */
    Ipsfloat_vreg_chc sRow,          /* column index correlation factor */
    Ipsint_vreg_chc   sOption        /* row index correlation factor */
);                                /* Markov spectrum symmetry option */
```

moments

```
void IvMoments (
    Idnimage          nSourceImage, /* no return value output */
    Ipsfloat           *trMean,      /* source image identifier */
    Ipsfloat           *trSDev,      /* OUT mean value */
    Ipsfloat           /* OUT standard deviation value */
);
```

moments_vreg

```
void IvMomentsVReg (
    Ipsimage_vreg     snSourceImage, /* no return value output */
    Idnvreg           nMean,          /* source image identifier */
    Idnvreg           nSDev,          /* OUT mean value virtual register identifier */
    Idnvreg           /* OUT deviation value virtual register identifier */
);
```

monadic_arithmetic

```
Idnimage InMonadicArith (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Ipsparameter_arith stConstant,   /* destination image identifier */
    Ipsint             iOption       /* value of constant pointer union */
);                                /* monadic, arithmetic combination option */
```

monadic_arithmetic_vreg

```
Idnvreg InMonadicArithVReg (
    Idnvreg          nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg     snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg     snDestImage,   /* source image identifier */
    Ipsparameter_arith_vreg_chc sConstant, /* destination image identifier */
    Ipsint_vreg_chc   sOption        /* value of constant */
);                                /* monadic, arithmetic combination option */
```

monadic_complex

```
Idnimage InMonadicComplex (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier          */
    Ipscomplex        sxConstant,   /* destination image identifier     */
    Ipint             iOption,      /* value of constant               */
    );                  /* monadic, complex combination option */
```

monadic_complex_vreg

```
Idnvreg InMonadicComplexVReg (
    Idnvreg          nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier          */
    Ipscomplex_vreg_chc sConstant, /* destination image identifier     */
    Ipsint_vreg_chc   sOption,      /* value of constant               */
    );                /* monadic, complex combination option */
```

monadic_logical

```
Idnimage InMonadicLogical (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier          */
    Ipsparameter_logical stConstant, /* destination image identifier     */
    Ipint             iOption,      /* value of constant pointer union  */
    );                /* monadic, logical combination option */
```

monadic_logical_vreg

```
Idnvreg InMonadicLogicalVReg (
    Idnvreg          nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier          */
    Ipsparameter_logical_vreg_chc sConstant, /* destination image identifier     */
    Ipsint_vreg_chc   sOption,      /* value of constant               */
    );                /* monadic, logical combination option */
```

monadic_logical_roi

```
Idnroi InMonadicLogicalROI (
    Idnroi          nSourceROI, /* OUT destination ROI identifier */
    Idnroi          nDestROI,   /* source ROI identifier          */
    Ipsparameter_logical stConstant, /* destination ROI identifier     */
    Ipint           iOption,     /* value of constant pointer union */
    );                /* monadic, logical combination option */
```

monadic_logical_roi_vreg

```
Idnvreg InMonadicLogicalROIvReg (          /* OUT destination ROI identifier */
    Idnvreg          nVReg,                /* output virtual register identifier or NULL */
    Ipsroi_vreg      snSourceROI,          /* source ROI identifier */
    Ipsroi_vreg      snDestROI,            /* destination ROI identifier */
    Ipsparameter_logical_vreg_chc sConstant, /* value of constant */
    Ipsint_vreg_chc   sOption              /* monadic, logical combination option */
);
```

morphic_processor

```
Idnimage InMorphicProcessor (          /* OUT destination image identifier */
    Idnimage          nSourceImage,        /* source image identifier */
    Idnimage          nDestImage,          /* destination image identifier */
    Idnlut            nLUT,                /* hit or miss lookup table identifier */
    Ipint             iIterations,          /* number of iterations */
    Ipuint            *tuCycles             /* OUT number of cycles */
);
```

morphic_processor_vreg

```
Idnvreg InMorphicProcessorVReg (        /* OUT destination image identifier */
    Idnvreg          nVReg,                /* output virtual register identifier or NULL */
    Ipsimage_vreg     snSourceImage,        /* source image identifier */
    Ipsimage_vreg     snDestImage,          /* destination image identifier */
    Ipslut_vreg       snLUT,                /* hit or miss lookup table identifier */
    Ipsint_vreg_chc   sIterations,          /* number of iterations */
    Idnvreg           nCycles              /* OUT cycles virtual register identifier */
);
```

morphic_processor_roi

```
Idnroi InMorphicProcessorROI (          /* OUT destination ROI identifier */
    Idnroi            nSourceROI,          /* source ROI identifier */
    Idnroi            nDestROI,            /* destination ROI identifier */
    Idnlut            nLUT,                /* hit or miss lookup table identifier */
    Ipint             iIterations,          /* number of iterations */
    Ipuint            *tuCycles            /* OUT number of cycles */
);
```

morphic_processor_roi_vreg

```
Idnvreg InMorphicProcessorROIvReg (
    Idnvreg          nVReg,          /* OUT destination ROI identifier */
    Ipsroi_vreg      snSourceROI,    /* output virtual register identifier or NULL */
    Ipsroi_vreg      snDestROI,     /* source ROI identifier */
    Ipslut_vreg      snLUT,         /* destination ROI identifier */
    Ipsint_vreg_chc  sIterations,    /* hit or miss lookup table identifier */
    Idnvreg          nCycles,       /* number of iterations */
    Idnvreg          /* OUT cycles virtual register identifier */
);
```

morphology

```
Idnimage InMorphology (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Ipint             iIterations,  /* destination image identifier */
    Ipuint            *tuCycles,    /* number of iterations */
    Ipint             iOption,      /* OUT number of cycles */
    Ipint             /* morphology option */
);
```

morphology_vreg

```
Idnvreg InMorphologyVReg (
    Idnvreg          nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier */
    Ipsint_vreg_chc  sIterations,    /* destination image identifier */
    Idnvreg          nCycles,       /* number of iterations */
    Ipsint_vreg_chc  sOption,      /* OUT cycles virtual register identifier */
    Idnvreg          /* morphology option */
);
```

morphology_roi

```
Idnroi InMorphologyROI (
    Idnroi          nSourceROI,     /* OUT destination ROI identifier */
    Idnroi          nDestROI,       /* source ROI identifier */
    Ipint           iIterations,    /* destination ROI identifier */
    Ipuint          *tuCycles,     /* number of iterations */
    Ipint           iOption,       /* OUT number of cycles */
    Ipint           /* morphology option */
);
```

morphology_roi_vreg

Idnvreg InMorphologyROIvReg (		/* OUT destination ROI identifier	*/
Idnvreg	nVReg,	/* output virtual register identifier or NULL	*/
Ipsroi_vreg	snSourceROI,	/* source ROI identifier	*/
Ipsroi_vreg	snDestROI,	/* destination ROI identifier	*/
Ipsint_vreg_chc	sIterations,	/* number of iterations	*/
Idnvreg	nCycles,	/* OUT cycles virtual register identifier	*/
Ipsint_vreg_chc	sOption	/* morphological option	*/

);

neighbour_count

Idnimage InNbrCount (		/* OUT destination image identifier	*/
Idnimage	nSourceImage,	/* source image identifier	*/
Idnimage	nDestImage,	/* destination image identifier	*/
Idnnbhood	nArray	/* structuring element array identifier	*/

);

neighbour_count_vreg

Idnvreg InNbrCountVReg (		/* OUT destination image identifier	*/
Idnvreg	nVReg,	/* output virtual register identifier or NULL	*/
Ipsimage_vreg	snSourceImage,	/* source image identifier	*/
Ipsimage_vreg	snDestImage,	/* destination image identifier	*/
Ipsnbhood_vreg	snArray	/* structuring element array identifier	*/

);

noise_combination

Idnimage InNoiseCombination (		/* OUT destination image identifier	*/
Idnimage	nSourceImage,	/* source image identifier	*/
Idnimage	nDestImage,	/* destination image identifier	*/
Ipfloor	rRatio,	/* signal-to-noise ratio	*/
Ipsint	iOption	/* probability density option	*/

);

noise_combination_vreg

Idnvreg InNoiseCombinationVReg (		/* OUT destination image identifier	*/
Idnvreg	nVReg,	/* output virtual register identifier or NULL	*/
Ipsimage_vreg	snSourceImage,	/* source image identifier	*/
Ipsimage_vreg	snDestImage,	/* destination image identifier	*/
Ipsfloat_vreg_chc	sRatio,	/* signal-to-noise ratio	*/
Ipsint_vreg_chc	sOption	/* probability density option	*/

);

open_close

```
Idnimage InOpenClose (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier          */
    Idnnbhood         nArray,       /* destination image identifier     */
    Ipeopen_close     eOperationOption, /* structuring element array identifier */
    Iperotation       eErosionOption, /* open close operation option      */
    Iperotation       eDilateOption, /* erosion structuring element rotation option */
    Iperotation       eDilateOption, /* dilation structuring element rotation option */
);
```

open_close_vreg

```
Idnvreg InOpenCloseVReg (
    Idnvreg          nVReg, /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier          */
    Ipsnbhood_vreg   snArray,       /* destination image identifier     */
    Ipsopen_close_vreg_chc sOperationOption, /* structuring element array identifier */
    Ipsrotation_vreg_chc sErosionOption, /* open close operation option      */
    Ipsrotation_vreg_chc sDilateOption, /* erosion structuring element rotation option */
    Ipsrotation_vreg_chc sDilateOption, /* dilation structuring element rotation option */
);
```

open_close_roi

```
Idnroi InOpenCloseROI (
    Idnroi          nSourceROI, /* OUT destination ROI identifier */
    Idnroi          nDestROI,   /* source ROI identifier          */
    Idnnbhood       nArray,     /* destination ROI identifier     */
    Ipeopen_close   eOperationOption, /* structuring element array identifier */
    Iperotation     eErosionOption, /* open close operation option      */
    Iperotation     eDilateOption, /* erosion structuring element rotation option */
    Iperotation     eDilateOption, /* dilation structuring element rotation option */
);
```

open_close_roi_vreg

```
Idnvreg InOpenCloseROIvReg (
    Idnvreg          nVReg, /* OUT destination ROI identifier */
    Ipsroi_vreg      snSourceROI, /* output virtual register identifier or NULL */
    Ipsroi_vreg      snDestROI,   /* source ROI identifier          */
    Ipsnbhood_vreg   snArray,     /* destination ROI identifier     */
    Ipsopen_close_vreg_chc sOperationOption, /* structuring element array identifier */
    Ipsrotation_vreg_chc sErosionOption, /* open close operation option      */
    Ipsrotation_vreg_chc sDilateOption, /* erosion structuring element rotation option */
    Ipsrotation_vreg_chc sDilateOption, /* dilation structuring element rotation option */
);
```

open_piks

```
void IvOpenPIKS (
    Iperror          nFile          /* no return value output          */
                                /* error file identifier          */
);
```

outlier_removal

```
Idnimage InOutlierRemoval (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier          */
    Idnnbhood         nImpulse,     /* destination image identifier     */
    Ipfloat           rThreshold,    /* impulse response array identifier */
                                /* threshold value                  */
);
```

outlier_removal_vreg

```
Idnvreg InOutlierRemovalVReg (
    Idnvreg          nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier          */
    Ipsnbhood_vreg   snImpulse,     /* destination image identifier     */
    Ipsfloat_vreg_chc sThreshold,    /* impulse response array identifier */
                                /* threshold value                  */
);
```

output_object

```
Ipnpport InOutputObject (
    Ipsobject          nInternal,    /* OUT external identifier          */
    Ipnpport          nExternal,    /* PIKS identifier                  */
    Ipint             iTransferOption, /* external identifier              */
    Ipint             iSyntaxOption, /* data destination transfer option */
    Ipint             iApplBufferSize, /* data destination transfer syntax option */
    Ipuint            uStartPoint, /* data transfer buffer size or option */
    Ipuint            *tuDataSize, /* index offset within internal data */
                                /* OUT post transfer data size pointer */
);
```

paint_pixels

```

Idnimage InPaintPixels (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Idnnbhood         nBrush,       /* destination image identifier */
    Ipoint            *tuaControl,   /* brush array identifier */
    Ipoint            uIndex3,       /* control point array pointer */
    Ipoint            uIndex4,       /* third index value */
    Ipoint            uIndex5,       /* fourth index value */
    Ipoint            uPoints,       /* fifth index value */
    Ipoint            iPaintingOption, /* number of control points */
    Ipoint            iBrushOption, /* painting option */
    Ipoint            /* brush option */
);

```

paint_pixels_vreg

```

Idnvreg InPaintPixelsVReg (
    Idnvreg          nVReg, /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier */
    Ipsnbhood_vreg   snBrush,       /* destination image identifier */
    Ipsaint_vreg_chc sControl,       /* brush array identifier */
    Ipsuint_vreg_chc sIndex3,        /* control point array */
    Ipsuint_vreg_chc sIndex4,        /* third index value */
    Ipsuint_vreg_chc sIndex5,        /* fourth index value */
    Ipsuint_vreg_chc sPoints,        /* fifth index value */
    Ipsint_vreg_chc  sPaintingOption, /* number of control points */
    Ipsint_vreg_chc  sBrushOption,   /* painting option */
    Ipsint_vreg_chc /* brush option */
);

```

perimeter_code_generator

```

Ipoint *ItuaPerimeterCodeGenerator (
    Idnimage          nSourceImage, /* OUT crack codes array pointer */
    Ipoint            *tuaCodes,     /* source image identifier */
    Ipoint            uIndex3,       /* crack codes array pointer */
    Ipoint            uIndex4,       /* third index value */
    Ipoint            uIndex5,       /* fourth index value */
    Ipoint            uLimit,        /* fifth index value */
    Ipoint            iApplBufferSize, /* object number limit */
    Ipoint            uStartPoint,   /* data transfer buffer size or option */
    Ipoint            *tuDataSize,   /* index offset within internal data */
    Ipoint            /* OUT post transfer data size pointer */
);

```

polar_to_cartesian

```
Idnimage InPolarToCartesian (                                /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Ipfloating         rMinimumRho, /* minimum value of rho */
    Ipfloating         rMaximumRho, /* maximum value of rho */
    Ipfloating         rMinimumTheta, /* minimum value of theta */
    Ipfloating         rMaximumTheta /* maximum value of theta */
);
```

polar_to_cartesian_vreg

```
Idnvreg InPolarToCartesianVReg (                                /* OUT destination image identifier */
    Idnvreg          nVReg,      /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipsfloat_vreg_chc sMinimumRho, /* minimum value of rho */
    Ipsfloat_vreg_chc sMaximumRho, /* maximum value of rho */
    Ipsfloat_vreg_chc sMinimumTheta, /* minimum value of theta */
    Ipsfloat_vreg_chc sMaximumTheta /* maximum value of theta */
);
```

power_law_scaling

```
Idnimage InPowerLawScaling (                                /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Ipfloating         rExponent     /* power law exponent */
);
```

power_law_scaling_vreg

```
Idnvreg InPowerLawScalingVReg (                                /* OUT destination image identifier */
    Idnvreg          nVReg,      /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipsfloat_vreg_chc sExponent     /* power law exponent */
);
```

pseudocolour

```
Idnimage InPseudocolour (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Idnlut            nLUT,         /* destination image identifier */
    Ipuint            uOffset,      /* lookup table identifier */
    );                  /* number of lookup table source offset entries */
```

pseudocolour_vreg

```
Idnvreg InPseudocolourVReg (
    Idnvreg          nVReg,         /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier */
    Ipslut_vreg      snLUT,         /* destination image identifier */
    Ipsuint_vreg_chc sOffset,      /* lookup table identifier */
    );                  /* number of lookup table source offset entries */
```

put_colour_pixel

```
Idnimage InPutColourPixel (
    Idnimage          nDestImage,   /* OUT destination image identifier */
    Ipsparameter_colour stBand0,    /* destination image identifier */
    Ipsparameter_colour stBand1,    /* band 0 pixel value pointer union */
    Ipsparameter_colour stBand2,    /* band 1 pixel value pointer union */
    Ipsparameter_colour stBand3,    /* band 2 pixel value pointer union */
    Ipuint            uHorizontal,  /* band 3 pixel value pointer union */
    Ipuint            uVertical,    /* horizontal index value */
    Ipuint            uDepth,       /* vertical index value */
    Ipuint            uTemporal,    /* depth index value */
    );                  /* temporal index value */
```

put_colour_pixel_vreg

```
Idnvreg InPutColourPixelVReg (
    Idnvreg          nVReg,         /* OUT destination image identifier */
    Ipsimage_vreg    snDestImage,   /* output virtual register identifier or NULL */
    Ipsparameter_colour_vreg_chc sBand0, /* destination image identifier */
    Ipsparameter_colour_vreg_chc sBand1, /* band 0 pixel value virtual register identifier */
    Ipsparameter_colour_vreg_chc sBand2, /* band 1 pixel value virtual register identifier */
    Ipsparameter_colour_vreg_chc sBand3, /* band 2 pixel value virtual register identifier */
    Ipsuint_vreg_chc sHorizontal,    /* band 3 pixel value virtual register identifier */
    Ipsuint_vreg_chc sVertical,      /* horizontal index value */
    Ipsuint_vreg_chc sDepth,         /* vertical index value */
    Ipsuint_vreg_chc sTemporal,      /* depth index value */
    );                  /* temporal index value */
```

put_pixel

```
Idnimage InPutPixel (
    Idnimage          nDestImage,      /* OUT destination image identifier */
    Ipsparameter_pixel stPixel,        /* destination image identifier */
    Ipoint            uHorizontal,      /* pixel value pointer union */
    Ipoint            uVertical,        /* horizontal index value */
    Ipoint            uDepth,          /* vertical index value */
    Ipoint            uTemporal,       /* depth index value */
    Ipoint            uBand,           /* temporal index value */
    Ipoint            /* band index value */
);
```

put_pixel_vreg

```
Idnvreg InPutPixelVReg (
    Idnvreg          nVReg,            /* OUT destination image identifier */
    Ipsimage_vreg    snDestImage,     /* output virtual register identifier or NULL */
    Ipsparameter_pixel_vreg_chc sPixel, /* destination image identifier */
    Ipsuint_vreg_chc sHorizontal,      /* pixel value */
    Ipsuint_vreg_chc sVertical,        /* horizontal index value */
    Ipsuint_vreg_chc sDepth,          /* vertical index value */
    Ipsuint_vreg_chc sTemporal,       /* depth index value */
    Ipsuint_vreg_chc sBand,           /* temporal index value */
    Ipsuint_vreg_chc /* band index value */
);
```

put_pixel_roi

```
Idnroi InPutPixelROI (
    Idnroi          nDestROI,         /* OUT destination ROI identifier */
    Ipbool          bPixel,          /* destination ROI identifier */
    Ipoint          uHorizontal,      /* pixel value */
    Ipoint          uVertical,        /* horizontal index value */
    Ipoint          uDepth,          /* vertical index value */
    Ipoint          uTemporal,       /* depth index value */
    Ipoint          uBand,           /* temporal index value */
    Ipoint          /* band index value */
);
```

put_pixel_roi_vreg

```
Idnvreg InPutPixelROIvReg (
    Idnvreg          nVReg,          /* OUT destination ROI identifier */
    Ipsroi_vreg      snDestROI,     /* output virtual register identifier or NULL */
    Ipsbool_vreg_chc sPixel,        /* destination ROI identifier */
    Ipsuint_vreg_chc sHorizontal,    /* pixel value */
    Ipsuint_vreg_chc sVertical,      /* horizontal index value */
    Ipsuint_vreg_chc sDepth,         /* vertical index value */
    Ipsuint_vreg_chc sTemporal,     /* depth index value */
    Ipsuint_vreg_chc sBand,         /* temporal index value */
    Ipsuint_vreg_chc          /* band index value */
);
```

put_pixel_array

```
Idnimage InPutPixelArray (
    Idnimage          nDestImage,    /* OUT destination image identifier */
    Ipsparameter_pixel stPixels,     /* destination image identifier */
    Ipuint            uOffset1,      /* pixel values array pointer union */
    Ipuint            uOffset2,     /* first index offset */
    Ipuint            uValue3,      /* second index offset */
    Ipuint            uValue4,     /* third index value */
    Ipuint            uValue5,     /* fourth index value */
    Ipuint            uColumns,     /* fifth index value */
    Ipuint            uRows,        /* number of array columns */
    Ipuint            /* number of array rows */
);
```

put_pixel_array_vreg

```
Idnvreg InPutPixelArrayVReg (
    Idnvreg          nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg    snDestImage,   /* output virtual register identifier or NULL */
    Ipsparameter_pixel_vreg_chc sPixels, /* destination image identifier */
    Ipsuint_vreg_chc sOffset1,      /* pixel values array */
    Ipsuint_vreg_chc sOffset2,     /* first index offset */
    Ipsuint_vreg_chc sValue3,      /* second index offset */
    Ipsuint_vreg_chc sValue4,     /* third index value */
    Ipsuint_vreg_chc sValue5,     /* fourth index value */
    Ipsuint_vreg_chc sColumns,     /* fifth index value */
    Ipsuint_vreg_chc sRows,        /* number of array columns */
    Ipsuint_vreg_chc          /* number of array rows */
);
```

put_pixel_array_roi

```
Idnroi InPutPixelArrayROI (
    Idnroi          nDestROI,          /* OUT destination ROI identifier */
    Ipbool          *tbaPixels,        /* destination ROI identifier */
    Ipuint          uOffset1,          /* pixel values array pointer */
    Ipuint          uOffset2,          /* first index offset */
    Ipuint          uValue3,           /* second index offset */
    Ipuint          uValue4,           /* third index value */
    Ipuint          uValue5,           /* fourth index value */
    Ipuint          uColumns,          /* fifth index value */
    Ipuint          uRows,             /* number of array columns */
    );                                /* number of array rows */
```

put_pixel_array_roi_vreg

```
Idnvreg InPutPixelArrayROIvReg (
    Idnvreg          nVReg,            /* OUT destination ROI identifier */
    Ipsroi_vreg      snDestROI,        /* output virtual register identifier or NULL */
    Ipsabool_vreg_chc sPixels,         /* destination ROI identifier */
    Ipsuint_vreg_chc sOffset1,         /* pixel values array */
    Ipsuint_vreg_chc sOffset2,         /* first index offset */
    Ipsuint_vreg_chc sValue3,          /* second index offset */
    Ipsuint_vreg_chc sValue4,          /* third index value */
    Ipsuint_vreg_chc sValue5,          /* fourth index value */
    Ipsuint_vreg_chc sColumns,         /* fifth index value */
    Ipsuint_vreg_chc sRows,            /* number of array columns */
    );                                /* number of array rows */
```

put_pixel_record

```
Idnimage InPutPixelRecord (
    Idnimage          nDestImage,      /* OUT destination image identifier */
    Ipsparameter_pixel stPixels,        /* destination image identifier */
    Idntuple          nType,            /* pixel values array pointer union */
    Ipuint            uHorizontal,      /* data type SP B-tuple */
    Ipuint            uVertical,        /* horizontal index value */
    Ipuint            uDepth,          /* vertical index value */
    Ipuint            uTemporal,       /* depth index value */
    );                                /* temporal index value */
```

put_pixel_record_vreg

```
Idnvreg InPutPixelRecordVReg (
    Idnvreg          nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg    snDestImage,   /* output virtual register identifier or NULL */
    Ipsparameter_pixel_vreg_chc sPixels, /* destination image identifier */
    Ipstuple_vreg    snType,        /* pixel values array */
    Ipsuint_vreg_chc sHorizontal,    /* data type SD B-tuple */
    Ipsuint_vreg_chc sVertical,      /* horizontal index value */
    Ipsuint_vreg_chc sDepth,        /* vertical index value */
    Ipsuint_vreg_chc sTemporal,     /* depth index value */
    Ipsuint_vreg_chc          /* temporal index value */
);
```

put_pixel_record_roi

```
Idnroi InPutPixelRecordROI (
    Idnroi          nDestROI,      /* OUT destination ROI identifier */
    Ipbool          *tbaPixels,    /* destination ROI identifier */
    Ipuint          uHorizontal,   /* pixel values array pointer union */
    Ipuint          uVertical,     /* horizontal index value */
    Ipuint          uDepth,       /* vertical index value */
    Ipuint          uTemporal,    /* depth index value */
    Ipuint          /* temporal index value */
);
```

put_pixel_record_roi_vreg

```
Idnvreg InPutPixelRecordROIvReg (
    Idnvreg          nVReg,          /* OUT destination ROI identifier */
    Ipsroi_vreg      snDestROI,     /* output virtual register identifier or NULL */
    Ipsabool_vreg_chc sPixels,      /* destination ROI identifier */
    Ipsuint_vreg_chc sHorizontal,   /* pixel values array */
    Ipsuint_vreg_chc sVertical,     /* horizontal index value */
    Ipsuint_vreg_chc sDepth,       /* vertical index value */
    Ipsuint_vreg_chc sTemporal,    /* depth index value */
    Ipsuint_vreg_chc          /* temporal index value */
);
```

rescale

```
Idnimage InRescale (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Idntuple          nScale,       /* destination image identifier */
    Idntuple          nUpper,      /* scale factor RD 5-tuple identifier */
    Idntuple          nLower,     /* upper value RD 5-tuple identifier */
    Idntuple          /* lower value RD 5-tuple identifier */
);
```

rescale_vreg

```
Idnvreg InRescaleVReg (                                /* OUT destination image identifier */
    Idnvreg                                nVReg,        /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,    /* source image identifier */
    Ipsimage_vreg    snDestImage,    /* destination image identifier */
    Ipstuple_vreg    snScale,        /* scale factor RD 5-tuple identifier */
    Ipstuple_vreg    snUpper,        /* upper value RD 5-tuple identifier */
    Ipstuple_vreg    snLower        /* lower value RD 5-tuple identifier */
);
```

rescale_roi

```
Idnroi InRescaleROI (                                /* OUT destination ROI identifier */
    Idnroi                                nSourceROI,    /* source ROI identifier */
    Idnroi                                nDestROI,    /* destination ROI identifier */
    Idntuple    nScale,        /* scale factor RD 5-tuple identifier */
    Idntuple    nUpper,        /* upper value RD 5-tuple identifier */
    Idntuple    nLower        /* lower value RD 5-tuple identifier */
);
```

rescale_roi_vreg

```
Idnvreg InRescaleROIvReg (                                /* OUT destination ROI identifier */
    Idnvreg                                nVReg,        /* output virtual register identifier or NULL */
    Ipsroi_vreg    snSourceROI,    /* source ROI identifier */
    Ipsroi_vreg    snDestROI,    /* destination ROI identifier */
    Ipstuple_vreg    snScale,        /* scale factor RD 5-tuple identifier */
    Ipstuple_vreg    snUpper,        /* upper value RD 5-tuple identifier */
    Ipstuple_vreg    snLower        /* lower value RD 5-tuple identifier */
);
```

resize

```
Idnimage InResize (                                /* OUT destination image identifier */
    Idnimage                                nSourceImage,    /* source image identifier */
    Idnimage                                nDestImage,    /* destination image identifier */
    Idntuple    nDestSize        /* destination size ND 5-tuple identifier */
);
```

resize_vreg

```
Idnvreg InResizeVReg (                                /* OUT destination image identifier */
    Idnvreg                                nVReg,      /* output virtual register identifier or NULL */
    Ipsimage_vreg                        snSourceImage, /* source image identifier */
    Ipsimage_vreg                        snDestImage,   /* destination image identifier */
    Ipstuple_vreg                        snDestSize     /* destination size ND 5-tuple identifier */
);
```

resize_roi

```
Idnroi InResizeROI (                                /* OUT destination ROI identifier */
    Idnroi                                nSourceROI,   /* source ROI identifier */
    Idnroi                                nDestROI,     /* destination ROI identifier */
    Idntuple                              nDestSize     /* destination size ND 5-tuple identifier */
);
```

resize_roi_vreg

```
Idnvreg InResizeROIvReg (                            /* OUT destination ROI identifier */
    Idnvreg                                nVReg,      /* output virtual register identifier or NULL */
    Ipsroi_vreg                        snSourceROI,    /* source ROI identifier */
    Ipsroi_vreg                        snDestROI,      /* destination ROI identifier */
    Ipstuple_vreg                        snDestSize     /* destination size ND 5-tuple identifier */
);
```

return_repository_id

```
Idnrepository InReturnRepositoryId (                  /* OUT repository identifier */
    Ipint                                iDescriptor,  /* repository descriptor within class */
    Ipint                                iClassOption /* repository class option */
);
```

return_repository_id_vreg

```
Idnvreg InReturnRepositoryIdVReg (                    /* OUT repository identifier */
    Idnvreg                                nVReg,      /* output virtual register identifier */
    Ipsint_vreg_chc                        sIndex,     /* repository descriptor within class */
    Ipsint_vreg_chc                        sClassOption /* repository class option */
);
```

roi_coordinate

```
Idnroi InROICoordinate (
    Ipuint          /* OUT destination ROI identifier */
    *tuaArray,      /* 5-dimensional coordinate array pointer */
    Idnroi          /* destination ROI identifier */
    Idntuple        /* ROI size NP 5-tuple */
    Ipuint          /* number of array entries */
    Ipepolarity     /* ROI polarity option */
    ePolarityOption
);
```

roi_coordinate_vreg

```
Idnvreg InROICoordinateVReg (
    Idnvreg        /* OUT destination ROI identifier */
    nVReg,         /* output virtual register identifier or NULL */
    Ipsauint_vreg_chc /* 5-dimensional coordinate array */
    sArray,        /* destination ROI identifier */
    Ipsroi_vreg     /* ROI size NP 5-tuple */
    snDestROI,     /* number of array entries */
    Ipstuple_vreg   /* ROI polarity option */
    sEntries,
    Ipspolarity_vreg_chc /* ROI polarity option */
    sPolarityOption
);
```

roi_elliptical

```
Idnroi InROIElliptical (
    Idnroi          /* OUT destination ROI identifier */
    nDestROI,       /* destination ROI identifier */
    Idntuple        /* ROI size ND 5-tuple */
    nSize,          /* ROI size ND 5-tuple */
    Idntuple        /* ellipse centre RD 5-tuple */
    nCentre,        /* ellipse axis length RD 5-tuple */
    Idntuple        /* ellipse axis length RD 5-tuple */
    nLength,        /* index manipulation CS 5-tuple identifier */
    Idntuple        /* ellipse dimension option */
    nIndexManip,    /* ellipse dimension option */
    Ipedimension    /* ROI polarity option */
    eDimensionOption,
    Ipepolarity     /* ROI polarity option */
    ePolarityOption
);
```

roi_elliptical_vreg

```
Idnvreg InROIEllipticalVReg (
    Idnvreg        /* OUT destination ROI identifier */
    nVReg,         /* output virtual register identifier or NULL */
    Ipsroi_vreg     /* destination ROI identifier */
    snDestROI,     /* ROI size ND 5-tuple */
    Ipstuple_vreg   /* ROI size ND 5-tuple */
    snSize,        /* ellipse centre RD 5-tuple */
    Ipstuple_vreg   /* ellipse axis length RD 5-tuple */
    snCentre,      /* ellipse axis length RD 5-tuple */
    Ipstuple_vreg   /* index manipulation CS 5-tuple identifier */
    snLength,      /* ellipse dimension option */
    Ipsdimension_vreg_chc /* ellipse dimension option */
    sDimensionOption,
    Ipspolarity_vreg_chc /* ROI polarity option */
    sPolarityOption
);
```

roi_polygon

```
Idnroi InROIPolygon (                                /* OUT destination ROI identifier */
    Ipuint                                           /* 2-dimensional coordinate array pointer */
    Idnroi                                           /* destination ROI identifier */
    Idntuple                                         /* ROI size ND 5-tuple */
    Ipuint                                           /* number of array entries */
    Idntuple                                         /* index manipulation CS 5-tuple identifier */
    Ipepolarity                                     /* ROI polarity option */
);
```

roi_polygon_vreg

```
Idnvreg InROIPolygonVReg (                            /* OUT destination ROI identifier */
    Idnvreg                                           /* output virtual register identifier or NULL */
    Ipsauint_vreg_chc                               /* 2-dimensional coordinate array */
    Ipsroi_vreg                                       /* destination ROI identifier */
    Ipstuple_vreg                                     /* ROI size ND 5-tuple */
    Ipsuint_vreg_chc                                 /* number of array entries */
    Ipstuple_vreg                                     /* index manipulation CS 5-tuple identifier */
    Ipspolarity_vreg_chc                             /* ROI polarity option */
);
```

roi_rectangular

```
Idnroi InROIRectangular (                            /* OUT destination ROI identifier */
    Idnroi                                           /* destination ROI identifier */
    Idntuple                                         /* ROI size ND 5-tuple */
    Idntuple                                         /* start position ND 5-tuple */
    Idntuple                                         /* end position ND 5-tuple */
    Idntuple                                         /* index manipulation CS 5-tuple */
    Ipedimension                                     /* rectangle dimension option */
    Ipepolarity                                     /* ROI polarity option */
);
```

roi_rectangular_vreg

```
Idnvreg InROIRectangularVReg (                      /* OUT destination ROI identifier */
    Idnvreg                                           /* output virtual register identifier or NULL */
    Ipsroi_vreg                                       /* destination ROI identifier */
    Ipstuple_vreg                                     /* ROI size 5-tuple */
    Ipstuple_vreg                                     /* start position 5-tuple */
    Ipstuple_vreg                                     /* end position 5-tuple */
    Ipstuple_vreg                                     /* index manipulation CS 5-tuple */
    Ipsdimension_vreg_chc                           /* rectangle dimension option */
    Ipspolarity_vreg_chc                             /* ROI polarity option */
);
```

rotate

```
Idnimage InRotate (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier          */
    Ipfloat           rCentre1,     /* destination image identifier     */
    Ipfloat           rCentre2,     /* first index source rotation centre */
    Ipfloat           rCentre3,     /* second index source rotation centre */
    Ipfloat           rAngleJK,     /* third index source rotation centre */
    Ipfloat           rAngleJL,     /* j-k plane rotation angle        */
    Ipfloat           rAngleKL,     /* j-l plane rotation angle        */
    Ipfloat           rAngleKL,     /* k-l plane rotation angle        */
    Ipint             iOption       /* rotation option                  */
);
```

rotate_vreg

```
Idnvreg InRotateVReg (
    Idnvreg          nVReg,         /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier          */
    Ipsfloat_vreg_chc sCentre1,     /* destination image identifier     */
    Ipsfloat_vreg_chc sCentre2,     /* first index source rotation centre */
    Ipsfloat_vreg_chc sCentre3,     /* second index source rotation centre */
    Ipsfloat_vreg_chc sAngleJK,     /* third index source rotation centre */
    Ipsfloat_vreg_chc sAngleJL,     /* j-k plane rotation angle        */
    Ipsfloat_vreg_chc sAngleJL,     /* j-l plane rotation angle        */
    Ipsfloat_vreg_chc sAngleKL,     /* k-l plane rotation angle        */
    Ipsint_vreg_chc  sOption       /* rotation option                  */
);
```

rotate_roi

```
Idnroi InRotateROI (
    Idnroi          nSourceROI,     /* OUT destination ROI identifier */
    Idnroi          nDestROI,       /* source ROI identifier          */
    Ipfloat         rCentre1,       /* destination ROI identifier     */
    Ipfloat         rCentre2,       /* first index source rotation centre */
    Ipfloat         rCentre3,       /* second index source rotation centre */
    Ipfloat         rAngleJK,       /* third index source rotation centre */
    Ipfloat         rAngleJL,       /* j-k plane rotation angle        */
    Ipfloat         rAngleJL,       /* j-l plane rotation angle        */
    Ipfloat         rAngleKL,       /* k-l plane rotation angle        */
    Ipint           iOption         /* rotation option                  */
);
```

rotate_roi_vreg

```
Idnvreg InRotateROIvReg (                                /* OUT destination ROI identifier */
    Idnvreg                                nVReg,          /* output virtual register identifier or NULL */
    Ipsroi_vreg                            snSourceROI,    /* source ROI identifier */
    Ipsroi_vreg                            snDestROI,       /* destination ROI identifier */
    Ipsfloat_vreg_chc                      sCentre1,       /* first index source rotation centre */
    Ipsfloat_vreg_chc                      sCentre2,       /* second index source rotation centre */
    Ipsfloat_vreg_chc                      sCentre3,       /* third index source rotation centre */
    Ipsfloat_vreg_chc                      sAngleJK,       /* j-k plane rotation angle */
    Ipsfloat_vreg_chc                      sAngleJL,       /* j-l plane rotation angle */
    Ipsfloat_vreg_chc                      sAngleKL,       /* k-l plane rotation angle */
    Ipsint_vreg_chc                        sOption          /* rotation option */
);
```

rubber_band_scaling

```
Idnimage InRubberBandScaling (                            /* OUT destination image identifier */
    Idnimage                                nSourceImage,   /* source image identifier */
    Idnimage                                nDestImage,     /* destination image identifier */
    Ipsparameter_arith                      stLowClip,       /* low input clip value pointer union */
    Ipsparameter_arith                      stLowOutput,     /* low output value pointer union */
    Ipsparameter_arith                      stHighClip,      /* high input clip value pointer union */
    Ipsparameter_arith                      stHighOutput,    /* high output value pointer union */
    Ipsparameter_arith                      stInflectionIn,   /* inflection input value pointer union */
    Ipsparameter_arith                      stInflectionOut  /* inflection output value pointer union */
);
```

rubber_band_scaling_vreg

```
Idnvreg InRubberBandScalingVReg (                        /* OUT destination image identifier */
    Idnvreg                                nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg                          snSourceImage,  /* source image identifier */
    Ipsimage_vreg                          snDestImage,    /* destination image identifier */
    Ipsparameter_arith_vreg_chc            sLowClip,       /* low input clip value */
    Ipsparameter_arith_vreg_chc            sLowOutput,     /* low output value */
    Ipsparameter_arith_vreg_chc            sHighClip,      /* high input clip value */
    Ipsparameter_arith_vreg_chc            sHighOutput,    /* high output value */
    Ipsparameter_arith_vreg_chc            sInflectionIn,   /* inflection input value */
    Ipsparameter_arith_vreg_chc            sInflectionOut  /* inflection output value */
);
```

sequence_average

```
Idnimage InSeqAverage (                                /* OUT destination image identifier */
    Idnimage                                           nSourceImage, /* source image identifier */
    Idnimage                                           nDestImage,   /* destination image identifier */
    Ipint                                             iOption       /* dimension option */
);
```

sequence_average_vreg

```
Idnvreg InSeqAverageVReg (                            /* OUT destination image identifier */
    Idnvreg                                           nVReg,        /* output virtual register identifier or NULL */
    Ipsimage_vreg                                   snSourceImage, /* source image identifier */
    Ipsimage_vreg                                   snDestImage,   /* destination image identifier */
    Ipsint_vreg_chc                                sOption      /* dimension option */
);
```

sequence_karhunen_loeve

```
Idnimage InSeqKarhunenLoeve (                        /* OUT destination image identifier */
    Idnimage                                           nSourceImage, /* source image identifier */
    Idnimage                                           nDestImage,   /* destination image identifier */
    Idnmatrix                                          nMatrix,      /* sequence covariance matrix identifier */
    Ipuint                                             uIndex4,      /* fourth index value */
    Ipuint                                             uIndex5,      /* fifth index value */
    Ipetransform                                     eOption       /* transform direction option */
);
```

sequence_karhunen_loeve_vreg

```
Idnvreg InSeqKarhunenLoeveVReg (                    /* OUT destination image identifier */
    Idnvreg                                           nVReg,        /* output virtual register identifier or NULL */
    Ipsimage_vreg                                   snSourceImage, /* source image identifier */
    Ipsimage_vreg                                   snDestImage,   /* destination image identifier */
    Ipsmatrix_vreg                                   snMatrix,      /* sequence covariance matrix identifier */
    Ipsuint_vreg_chc                                sIndex4,      /* fourth index value */
    Ipsuint_vreg_chc                                sIndex5,      /* fifth index value */
    Ipstransform_vreg_chc                           sOption       /* transform direction option */
);
```

sequence_running_measures

```
Idnimage InSeqRunningMeasures ( /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage,   /* destination image identifier */
    Ipint         iDimensionOption, /* dimension option */
    Ipint         iMeasureOption /* sequence running measure option */
);
```

sequence_running_measures_vreg

```
Idnvreg InSeqRunningMeasuresVReg ( /* OUT destination image identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage,   /* destination image identifier */
    Ipsint_vreg_chc sDimensionOption, /* dimension option */
    Ipsint_vreg_chc sMeasureOption /* sequence running measure option */
);
```

set_error_handler

```
Iptferror_function ItfSetErrorHandler ( /* OUT old error handling function pointer */
    Iptferror_function tfNewErrorHandler /* new error handler function pointer */
);
```

NOTE - This function has been added in this binding as defined in clause 3.8.1.

set_globals

```
void IvSetGlobals ( /* no return value output */
    Ipspiks_modes sModes /* mode settings */
);
```

set_globals_vreg

```
void IvSetGlobalsVReg ( /* no return value output */
    Ipspiks_modes_vreg_chc sModes /* mode settings */
);
```

set_image_attributes

```
Idnimage InSetImageAttrs ( /* OUT destination image identifier */
    Idnimage      nDestImage, /* destination image identifier */
    Ipsattrs_image sAttrs     /* image attributes */
);
```

set_image_attributes_vreg

```
Idnvreg InSetImageAttrsVReg (          /* OUT destination image identifier */
    Idnvreg          nVReg,              /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,       /* destination image identifier */
    Ipsattrs_image_vreg_chc sAttrs      /* image attributes */
);
```

set_index_assignment

```
void IvSetIndexAssignment (          /* no return value output */
    Ipeindex          eJAssign,          /* j index assignment selection */
    Ipeindex          eKAssign,          /* k index assignment selection */
    Ipeindex          eLAssign,          /* l index assignment selection */
    Ipeindex          eMAssign,         /* m index assignment selection */
    Ipeindex          eNAssign,         /* n index assignment selection */
);
```

set_index_assignment_vreg

```
void IvSetIndexAssignmentVReg (      /* no return value output */
    Ipsindex_vreg_chc sJAssign,         /* j index assignment selection */
    Ipsindex_vreg_chc sKAssign,         /* k index assignment selection */
    Ipsindex_vreg_chc sLAssign,         /* l index assignment selection */
    Ipsindex_vreg_chc sMAssign,        /* m index assignment selection */
    Ipsindex_vreg_chc sNAssign,        /* n index assignment selection */
);
```

shape_metrics

```
Ipfloating *ItraShapeMetrics (        /* OUT shape metric array pointer */
    Idnimage          nSourceImage,     /* source image identifier */
    Ipfloating        *traShape,        /* shape metric array pointer */
    Ipuint            uIndex3,          /* third index value */
    Ipuint            uIndex4,          /* fourth index value */
    Ipuint            uIndex5,          /* fifth index value */
    Ipuint            uLimit,           /* object number limit */
    Ipuint            iApplBufferSize, /* data transfer buffer size or option */
    Ipuint            uStartPoint,      /* index offset within internal data */
    Ipuint            *tuDataSize       /* OUT post transfer data size pointer */
);
```

spatial_moments_invariant

```

Ipfloor *ItraSpatialMomentsInvariant (          /* OUT spatial moments array pointer */
    Idnimage          nSourceImage,             /* source image identifier */
    Ipfloor           *traMoments,              /* spatial moments array pointer */
    Ipuint             uIndex3,                  /* third index value */
    Ipuint             uIndex4,                  /* fourth index value */
    Ipuint             uIndex5,                  /* fifth index value */
    );

```

spatial_moments_scaled

```

Ipfloor *ItraSpatialMomentsScaled (          /* OUT spatial moments array pointer */
    Idnimage          nSourceImage,             /* source image identifier */
    Ipfloor           *traMoments,              /* spatial moments array pointer */
    Ipuint             uIndex3,                  /* third index value */
    Ipuint             uIndex4,                  /* fourth index value */
    Ipuint             uIndex5,                  /* fifth index value */
    );

```

split_image

```

Idnimage InSplitImage (                      /* OUT destination image identifier */
    Idnimage          nSourceImage1,           /* first source image identifier */
    Idnimage          nSourceImage2,           /* second source image identifier */
    Idnimage          nDestImage,              /* destination image identifier */
    Ipuint             uSplitPosition,          /* split position */
    Ipuint             iOption,                 /* split image option */
    );

```

split_image_vreg

```

Idnvreg InSplitImageVReg (                   /* OUT destination image identifier */
    Idnvreg           nVReg,                    /* output virtual register identifier or NULL */
    Ipsimage_vreg     snSourceImage1,           /* first source image identifier */
    Ipsimage_vreg     snSourceImage2,           /* second source image identifier */
    Ipsimage_vreg     snDestImage,              /* destination image identifier */
    Ipsuint_vreg_chc  sSplitPosition,           /* split position */
    Ipsint_vreg_chc   sOption,                  /* split image option */
    );

```

subsample

```
Idnimage InSubsample (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Ipoint            uFactor1,     /* destination image identifier */
    Ipoint            uFactor2,     /* source first index subsampling factor */
    Ipoint            uSourceOffset1, /* source second index subsampling factor */
    Ipoint            uSourceOffset2, /* source first index offset */
    Ipoint            uDestOffset1,  /* source second index offset */
    Ipoint            uDestOffset2,  /* destination first index offset */
    Ipoint            uWindow1,     /* destination second index offset */
    Ipoint            uWindow2,     /* source first index window width */
    Ipoint            /* source second index window height */
);
```

subsample_vreg

```
Idnvreg InSubsampleVReg (
    Idnvreg          nVReg, /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier */
    Ipsuint_vreg_chc sFactor1,     /* destination image identifier */
    Ipsuint_vreg_chc sFactor2,     /* source first index subsampling factor */
    Ipsuint_vreg_chc sSourceOffset1, /* source second index subsampling factor */
    Ipsuint_vreg_chc sSourceOffset2, /* source first index offset */
    Ipsuint_vreg_chc sDestOffset1,  /* source second index offset */
    Ipsuint_vreg_chc sDestOffset2,  /* destination first index offset */
    Ipsuint_vreg_chc sWindow1,     /* destination second index offset */
    Ipsuint_vreg_chc sWindow2,     /* source first index window width */
    Ipsuint_vreg_chc /* source second index window height */
);
```

subsample_roi

```
Idnroi InSubsampleROI (
    Idnroi          nSourceROI, /* OUT destination ROI identifier */
    Idnroi          nDestROI,   /* source ROI identifier */
    Ipoint          uFactor1,   /* destination ROI identifier */
    Ipoint          uFactor2,   /* source first index subsampling factor */
    Ipoint          uSourceOffset1, /* source second index subsampling factor */
    Ipoint          uSourceOffset2, /* source first index offset */
    Ipoint          uDestOffset1,  /* source second index offset */
    Ipoint          uDestOffset2,  /* destination first index offset */
    Ipoint          uWindow1,     /* destination second index offset */
    Ipoint          uWindow2,     /* source first index window width */
    Ipoint          /* source second index window height */
);
```

subsample_roi_vreg

```
Idnvreg InSubsampleROIvReg (
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsroi_vreg      snSourceROI,    /* source ROI identifier */
    Ipsroi_vreg      snDestROI,      /* destination ROI identifier */
    Ipsuint_vreg_chc sFactor1,       /* source first index subsampling factor */
    Ipsuint_vreg_chc sFactor2,       /* source second index subsampling factor */
    Ipsuint_vreg_chc sSourceOffset1, /* source first index offset */
    Ipsuint_vreg_chc sSourceOffset2, /* source second index offset */
    Ipsuint_vreg_chc sDestOffset1,   /* destination first index offset */
    Ipsuint_vreg_chc sDestOffset2,   /* destination second index offset */
    Ipsuint_vreg_chc sWindow1,       /* source first index window width */
    Ipsuint_vreg_chc sWindow2        /* source second index window height */
);
```

synchronize

```
void IvSynch (
    void          /* no return value output */
);               /* no input arguments */
```

template_match

```
Idnstatic InTemplMatch (
    Idnimage      nSourceImage1,    /* first source image identifier */
    Idnimage      nSourceImage2,    /* second source image identifier */
    Idnstatic     nCrossCorrelation, /* cross-correlation array identifier */
    Ipuint        uIndex3,          /* third index value */
    Ipuint        uIndex4,          /* fourth index value */
    Ipuint        uIndex5,          /* fifth index value */
    Ipuint        uFirstOffsetImage1, /* first image first index offset */
    Ipuint        u2ndOffsetImage1,  /* first image second index offset */
    Ipuint        uFirstOffsetImage2, /* second image first index offset */
    Ipuint        u2ndOffsetImage2,  /* second image second index offset */
    Ipuint        uWindowColumns,    /* number of columns of search window */
    Ipuint        uWindowRows,       /* number of rows of search window */
    Ipuint        uTemplColumns,     /* number of columns of template */
    Ipuint        uTemplRows         /* number of rows of template */
);
```

template_match_vreg

```
Idnvreg InTemplMatchVReg (                                /* OUT cross-correlation array identifier */
    Idnvreg          nVReg,                                /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage1,                       /* first source image identifier */
    Ipsimage_vreg    snSourceImage2,                       /* second source image identifier */
    Ipsstatic_vreg   snCrossCorrelation,                   /* cross-correlation array identifier */
    Ipsuint_vreg_chc sIndex3,                              /* third index value */
    Ipsuint_vreg_chc sIndex4,                              /* fourth index value */
    Ipsuint_vreg_chc sIndex5,                              /* fifth index value */
    Ipsuint_vreg_chc sFirstOffsetImage1,                   /* first image first index offset */
    Ipsuint_vreg_chc s2ndOffsetImage1,                     /* first image second index offset */
    Ipsuint_vreg_chc sFirstOffsetImage2,                   /* second image first index offset */
    Ipsuint_vreg_chc s2ndOffsetImage2,                     /* second image second index offset */
    Ipsuint_vreg_chc sWindowColumns,                       /* number of columns of search window */
    Ipsuint_vreg_chc sWindowRows,                          /* number of rows of search window */
    Ipsuint_vreg_chc sTemplColumns,                        /* number of columns of template */
    Ipsuint_vreg_chc sTemplRows                            /* number of rows of template */
);
```

threshold

```
Idnimage InThreshold (                                    /* OUT destination image identifier */
    Idnimage          nSourceImage,                        /* source image identifier */
    Idnimage          nDestImage,                          /* destination image identifier */
    Ipsparameter_pixel stAbove,                            /* above threshold clip value pointer union */
    Ipsparameter_pixel stBelow,                            /* below threshold clip value pointer union */
    Ipsparameter_arith stThreshold                         /* threshold value pointer union */
);
```

threshold_vreg

```
Idnvreg InThresholdVReg (                                /* OUT destination image identifier */
    Idnvreg          nVReg,                                /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,                       /* source image identifier */
    Ipsimage_vreg    snDestImage,                         /* destination image identifier */
    Ipsparameter_pixel_vreg_chc sAbove,                   /* above threshold clip value */
    Ipsparameter_pixel_vreg_chc sBelow,                   /* below threshold clip value */
    Ipsparameter_arith_vreg_chc sThreshold                /* threshold value */
);
```

transform_cosine

```
Idnimage InTransformCosine (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Ipuint            uHorizontalSize, /* destination image identifier */
    Ipuint            uVerticalSize, /* horizontal block size */
    Ipetransform      eOption       /* vertical block size */
);                               /* transform direction option */
```

transform_cosine_vreg

```
Idnvreg InTransformCosineVReg (
    Idnvreg          nVReg, /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier */
    Ipsuint_vreg_chc sHorizontalSize, /* destination image identifier */
    Ipsuint_vreg_chc sVerticalSize, /* horizontal block size */
    Ipstransform_vreg_chc sOption    /* vertical block size */
);                               /* transform direction option */
```

transform_fourier

```
Idnimage InTransformFourier (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Ipedimension      eOrderOption, /* destination image identifier */
    Ipetransform      eDirectionOption, /* order option */
    Ipefrequency_shift eShiftOption, /* transform direction option */
);                               /* frequency shift option */
```

transform_fourier_vreg

```
Idnvreg InTransformFourierVReg (
    Idnvreg          nVReg, /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier */
    Ipsdimension_vreg_chc sOrderOption, /* destination image identifier */
    Ipstransform_vreg_chc sDirectionOption, /* order option */
    Ipsfrequency_shift_vreg_chc sShiftOption, /* transform direction option */
);                               /* frequency shift option */
```

transform_hadamard

```
Idnimage InTransformHadamard (          /* OUT destination image identifier */
    Idnimage          nSourceImage,      /* source image identifier          */
    Idnimage          nDestImage,        /* destination image identifier     */
    Ipuint            uBlockSize         /* block size                       */
);
```

transform_hadamard_vreg

```
Idnvreg InTransformHadamardVReg (        /* OUT destination image identifier */
    Idnvreg          nVReg,              /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,      /* source image identifier          */
    Ipsimage_vreg    snDestImage,        /* destination image identifier     */
    Ipsuint_vreg_chc sBlockSize         /* block size                       */
);
```

transform_hartley

```
Idnimage InTransformHartley (            /* OUT destination image identifier */
    Idnimage          nSourceImage,      /* source image identifier          */
    Idnimage          nDestImage,        /* destination image identifier     */
    Ipedimension      eOption            /* order option                     */
);
```

transform_hartley_vreg

```
Idnvreg InTransformHartleyVReg (         /* OUT destination image identifier */
    Idnvreg          nVReg,              /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,      /* source image identifier          */
    Ipsimage_vreg    snDestImage,        /* destination image identifier     */
    Ipsdimension_vreg_chc sOption        /* order option                     */
);
```

translate

```
Idnimage InTranslate (                  /* OUT destination image identifier */
    Idnimage          nSourceImage,      /* source image identifier          */
    Idnimage          nDestImage,        /* destination image identifier     */
    Idntuple          nOffset            /* index offset RD 5-tuple identifier */
);
```

translate_vreg

```
Idnvreg InTranslateVReg (
    Idnvreg          nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier */
    Ipstuple_vreg    snOffset,      /* destination image identifier */
    );                          /* index offset 5-tuple identifier */
```

translate_roi

```
Idnroi InTranslateROI (
    Idnroi          nSourceROI,      /* OUT destination ROI identifier */
    Idnroi          nDestROI,       /* source ROI identifier */
    Idntuple        nOffset,        /* destination ROI identifier */
    );                          /* index offset RD 5-tuple identifier */
```

translate_roi_vreg

```
Idnvreg InTranslateROIvReg (
    Idnvreg          nVReg,          /* OUT destination ROI identifier */
    Ipsroi_vreg      snSourceROI,   /* output virtual register identifier or NULL */
    Ipsroi_vreg      snDestROI,    /* source ROI identifier */
    Ipstuple_vreg    snOffset,      /* destination ROI identifier */
    );                          /* index offset RD 5-tuple identifier */
```

unary_integer

```
Idnimage InUnaryInteger (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Ipint             iOption,      /* destination image identifier */
    );                          /* unary integer conversion option */
```

unary_integer_vreg

```
Idnvreg InUnaryIntegerVReg (
    Idnvreg          nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier */
    Ipsint_vreg_chc  sOption,      /* destination image identifier */
    );                          /* unary integer conversion option */
```

unary_real

```
Idnimage InUnaryReal (                                /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage,  /* destination image identifier */
    Ipint         iOption      /* unary real conversion option */
);
```

unary_real_vreg

```
Idnvreg InUnaryRealVReg (                                /* OUT destination image identifier */
    Idnvreg      nVReg,        /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage,  /* destination image identifier */
    Ipsint_vreg_chc sOption     /* unary real conversion option */
);
```

unsharp_mask

```
Idnimage InUnsharpMask (                                /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage,  /* destination image identifier */
    Idnnbhood     nFilter,      /* low pass filter impulse response array identifier */
    Ipfloat       rWeight1,     /* first weight factor */
    Ipfloat       rWeight2,     /* second weight factor */
    Ipfloat       rBias        /* bias factor */
);
```

unsharp_mask_vreg

```
Idnvreg InUnsharpMaskVReg (                                /* OUT destination image identifier */
    Idnvreg      nVReg,        /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage,  /* destination image identifier */
    Ipsnbhood_vreg snFilter,     /* low pass filter impulse response array identifier */
    Ipsfloat_vreg_chc sWeight1, /* first weight factor */
    Ipsfloat_vreg_chc sWeight2, /* second weight factor */
    Ipsfloat_vreg_chc sBias     /* bias factor */
);
```

value_bounds

```
Idnbounds InValueBounds (
    Idnimage          nSourceImage, /* OUT value bounds collection identifier */
    Idnbounds         nValueBounds, /* source image identifier */
    Ipsparameter_arith stLowerBound, /* value bounds collection identifier */
    Ipsparameter_arith stUpperBound /* lower amplitude bound pointer union */
);
```

value_bounds_vreg

```
Idnvreg InValueBoundsVReg (
    Idnvreg          nVReg, /* OUT value bounds collection identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsbounds_vreg    snValueBounds, /* source image identifier */
    Ipsparameter_arith_vreg_chc sLowerBound, /* value bounds collection identifier */
    Ipsparameter_arith_vreg_chc sUpperBound /* lower amplitude bound */
);
```

vreg_alter

```
Idnvreg InVRegAlter (
    Idnvreg          nVReg, /* OUT virtual register identifier */
    Ipsparameter_pixel stDelta, /* virtual register identifier */
    Ipbool           bMode /* value to add to virtual register pointer union */
);
```

vreg_alter_vreg

```
Idnvreg InVRegAlterVReg (
    Idnvreg          nVReg, /* OUT virtual register identifier */
    Ipsparameter_pixel_vreg_chc sDelta, /* virtual register identifier */
    Ipsbool_vreg_chc sMode /* value to add to virtual register */
);
```

vreg_clear

```
Idnvreg InVRegClear (
    Idnvreg          nVReg, /* OUT virtual register identifier */
    Ipbool           bMode /* virtual register identifier */
);
```

vreg_clear_vreg

```
Idnvreg InVRegClearVReg (          /* OUT virtual register identifier */
    Idnvreg          nVReg,          /* virtual register identifier */
    Ipsbool_vreg_chc sMode          /* immediate mode flag */
);
```

vreg_conditional

```
void IvVRegConditional (          /* no return value output */
    Idnvreg          nVReg,          /* virtual register identifier */
    Ipsparameter_basic stLower,      /* lower test value pointer union */
    Ipsparameter_basic stUpper,      /* upper test value pointer union */
    Ipsbool          bChoice,        /* logical flag variable for conditional control */
    Idnchain         nChain,         /* chain identifier for branch execution */
    Ipsbool          bMode,          /* immediate mode flag */
    Ipsint           iReturnOption    /* virtual register conditional return option */
);
```

NOTE - The virtual register conditional return option is added to cover the return conditions described but not covered by parameter in the functional specification.

vreg_conditional_vreg

```
void IvVRegConditionalVReg (      /* no return value output */
    Idnvreg          nVReg,          /* virtual register identifier */
    Ipsparameter_basic_vreg_chc sLower, /* lower value to test register against */
    Ipsparameter_basic_vreg_chc sUpper, /* upper value to test register against */
    Ipsbool_vreg_chc sFlag,          /* logical flag variable for conditional control */
    Ipschain_vreg    snChain,         /* chain to branch for execution */
    Ipsbool_vreg_chc sMode,          /* immediate mode flag */
    Ipsint_vreg_chc  sReturnOption    /* virtual register conditional return option */
);
```

NOTE - The virtual register conditional return option is added to cover the return conditions described but not covered by parameter in the functional specification.

vreg_copy

```
Idnvreg InVRegCopy (          /* OUT destination virtual register identifier */
    Idnvreg          nSourceVReg,    /* source virtual register identifier */
    Idnvreg          nDestVReg,      /* destination virtual register identifier */
    Ipsbool          bMode           /* immediate mode flag */
);
```

vreg_copy_vreg

```
Idnvreg InVRegCopyVReg (
    Idnvreg          nSourceVReg, /* OUT destination virtual register identifier */
    Idnvreg          nDestVReg,   /* source virtual register identifier */
    Ipsbool_vreg_chc sMode,      /* destination virtual register identifier */
    );                          /* immediate mode flag */
```

vreg_delete

```
Idnvreg InVRegDelete (
    Idnvreg          nVReg, /* OUT virtual register identifier */
    Ipsbool          bMode, /* virtual register identifier */
    );                /* immediate mode flag */
```

vreg_delete_vreg

```
Idnvreg InVRegDeleteVReg (
    Idnvreg          nVReg, /* OUT virtual register identifier */
    Ipsbool_vreg_chc sMode, /* virtual register identifier */
    );                /* immediate mode flag */
```

vreg_get

```
void IvVRegGet (
    Idnvreg          nVReg, /* no return value output */
    Ipsbool          bMode, /* virtual register identifier */
    Ipsparameter_vreg sValue, /* immediate mode flag */
    Ipevreg_state    *teState, /* OUT value of virtual register pointer */
    Ipint            *tiType, /* OUT virtual register state pointer */
    );                /* OUT virtual register data type pointer */
```

vreg_set

```
Idnvreg InVRegSet (
    Idnvreg          nVReg, /* OUT virtual register identifier */
    Ipsparameter_vreg sValue, /* virtual register identifier */
    Ipsbool          bMode, /* value to be loaded */
    Ipint            iTyep, /* immediate mode flag */
    );                /* virtual register data type option */
```

vreg_set_vreg

```
Idnvreg InVRegSetVReg (
    Idnvreg          nVReg,          /* OUT virtual register identifier */
    Ipsparameter_vreg_vreg_chc sValue, /* virtual register identifier */
    Ipsbool_vreg_chc sMode,          /* value to be loaded */
    Ipsint_vreg_chc  sType,          /* immediate mode flag */
    /* virtual register data type option */
);
```

vreg_wait

```
Idnvreg InVRegWait (
    Idnvreg          nVReg,          /* OUT virtual register identifier */
    Ipsparameter_vreg sValue,        /* virtual register identifier */
    Ipsbool          bMode,          /* value to wait for */
    /* immediate mode flag */
);
```

vreg_wait_vreg

```
Idnvreg InVRegWaitVReg (
    Idnvreg          nVReg,          /* OUT virtual register identifier */
    Ipsparameter_vreg_vreg_chc sValue, /* virtual register identifier */
    Ipsbool_vreg_chc sMode,          /* value to wait for */
    /* immediate mode flag */
);
```

wallis_statistical_differencing

```
Idnimage InWallisStatDiff (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Idnnbhood         nFilter,      /* destination image identifier */
    Ipsfloat          rGain,        /* low pass filter impulse response array identifier */
    Ipsfloat          rAverage,     /* gain factor */
    Ipsfloat          rDeviation,   /* average factor */
    Ipsfloat          rProportion,  /* deviation factor */
    /* proportionality factor */
);
```

wallis_statistical_differencing_vreg

```
Idnvreg InWallisStatDiffVReg (                                /* OUT destination image identifier */
    Idnvreg                                nVReg,             /* output virtual register identifier or NULL */
    Ipsimage_vreg                          snSourceImage,     /* source image identifier */
    Ipsimage_vreg                          snDestImage,       /* destination image identifier */
    Ipsnbhood_vreg                         snFilter,          /* low pass filter impulse response array identifier */
    Ipsfloat_vreg_chc                      sGain,             /* gain factor */
    Ipsfloat_vreg_chc                      sAverage,          /* average factor */
    Ipsfloat_vreg_chc                      sDeviation,        /* deviation factor */
    Ipsfloat_vreg_chc                      sProportion        /* proportionality factor */
);
```

warp_control_point

```
Idnimage InWarpControlPoint (                                /* OUT destination image identifier */
    Idnimage                                nSourceImage,     /* source image identifier */
    Idnimage                                nDestImage,       /* destination image identifier */
    Ipuint                                  *tuaSourcePoints,  /* source image control point array pointer */
    Ipuint                                  *tuaDestPoints,    /* destination image control point array pointer */
    Ipuint                                  uPoints,           /* number of control points */
    Ipoint                                  iApplBufferSize,   /* data transfer buffer size or option of each array */
    Ipuint                                  uStartPoint,       /* index offset within internal data of each array */
    Ipuint                                  *tuDataSize        /* OUT post transfer data size pointer */
);
```

warp_control_point_vreg

```
Idnvreg InWarpControlPointVReg (                            /* OUT destination image identifier */
    Idnvreg                                nVReg,             /* output virtual register identifier or NULL */
    Ipsimage_vreg                          snSourceImage,     /* source image identifier */
    Ipsimage_vreg                          snDestImage,       /* destination image identifier */
    Ipsaint_vreg_chc                       sSourcePoints,     /* source image control point array */
    Ipsaint_vreg_chc                       sDestPoints,       /* destination image control point array */
    Ipsuint_vreg_chc                       sPoints            /* number of control points */
);
```

warp_control_point_roi

```
Idnroi InWarpControlPointROI (                /* OUT destination ROI identifier */
    Idnroi          nSourceROI,                /* source ROI identifier */
    Idnroi          nDestROI,                 /* destination ROI identifier */
    Ipuint          *tuaSourcePoints,          /* source image control point array pointer */
    Ipuint          *tuaDestPoints,           /* destination image control point array pointer */
    Ipuint          uPoints,                  /* number of control points */
    Ipoint          iApplBufferSize,          /* data transfer buffer size or option of each array */
    Ipuint          uStartPoint,              /* index offset within internal data of each array */
    Ipuint          *tuDataSize               /* OUT post transfer data size pointer */
);
```

warp_control_point_roi_vreg

```
Idnvreg InWarpControlPointROIvReg (          /* OUT destination ROI identifier */
    Idnvreg         nVReg,                    /* output virtual register identifier or NULL */
    Ipsroi_vreg     snSourceROI,              /* source ROI identifier */
    Ipsroi_vreg     snDestROI,               /* destination ROI identifier */
    Ipsaint_vreg_chc sSourcePoints,          /* source image control point array */
    Ipsaint_vreg_chc sDestPoints,            /* destination image control point array */
    Ipsuint_vreg_chc sPoints                 /* number of control points */
);
```

warp_lut

```
Idnimage InWarpLUT (                        /* OUT destination image identifier */
    Idnimage        nSourceImage,            /* source image identifier */
    Idnimage        nDestImage,             /* destination image identifier */
    Idnlut          nRowAddress,             /* row address lookup table identifier */
    Idnlut          nColumnAddress,          /* column address lookup table identifier */
    Ipuint          uEntries                 /* number of lookup table source offset entries */
);
```

warp_lut_vreg

```
Idnvreg InWarpLUTvReg (                    /* OUT destination image identifier */
    Idnvreg         nVReg,                    /* output virtual register identifier or NULL */
    Ipsimage_vreg   snSourceImage,           /* source image identifier */
    Ipsimage_vreg   snDestImage,            /* destination image identifier */
    Ipslut_vreg     snRowAddress,            /* row address lookup table identifier */
    Ipslut_vreg     snColumnAddress,         /* column address lookup table identifier */
    Ipsuint_vreg_chc sEntries                /* number of lookup table source offset entries */
);
```

warp_lut_roi

```
Idnroi InWarpLUTROI (
    Idnroi          nSourceROI, /* OUT destination ROI identifier */
    Idnroi          nDestROI,   /* source ROI identifier */
    Idnlut          nRowAddress, /* destination ROI identifier */
    Idnlut          nColumnAddress, /* row address lookup table identifier */
    Ipoint          uEntries,     /* column address lookup table identifier */
    /* number of lookup table source offset entries */
);
```

warp_lut_roi_vreg

```
Idnvreg InWarpLUTROIvReg (
    Idnvreg          nVReg,      /* OUT destination ROI identifier */
    Ipsroi_vreg      snSourceROI, /* output virtual register identifier or NULL */
    Ipsroi_vreg      snDestROI,  /* source ROI identifier */
    Ipslut_vreg      snRowAddress, /* destination ROI identifier */
    Ipslut_vreg      snColumnAddress, /* row address lookup table identifier */
    Ipsuint_vreg_chc sEntries,    /* column address lookup table identifier */
    /* number of lookup table source offset entries */
);
```

warp_polynomial

```
Idnimage InWarpPolynomial (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Ipsfloat          *traRow,      /* destination image identifier */
    Ipsfloat          *traColumn,   /* row polynomial coefficient array pointer */
    /* column polynomial coefficient array pointer */
);
```

warp_polynomial_vreg

```
Idnvreg InWarpPolynomialVReg (
    Idnvreg          nVReg,      /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,  /* source image identifier */
    Ipsafloat_vreg_chc sRow,      /* destination image identifier */
    Ipsafloat_vreg_chc sColumn,   /* row polynomial coefficient array */
    /* column polynomial coefficient array */
);
```

warp_polynomial_roi

```
Idnroi InWarpPolynomialROI (          /* OUT destination ROI identifier */
    Idnroi          nSourceROI,        /* source ROI identifier */
    Idnroi          nDestROI,          /* destination ROI identifier */
    lpfloor         *traRow,           /* row polynomial coefficient array pointer */
    lpfloor         *traColumn         /* column polynomial coefficient array pointer */
);
```

warp_polynomial_roi_vreg

```
Idnvreg InWarpPolynomialROIvReg (      /* OUT destination ROI identifier */
    Idnvreg         nVReg,             /* output virtual register identifier or NULL */
    lpsroi_vreg     snSourceROI,       /* source ROI identifier */
    lpsroi_vreg     snDestROI,         /* destination ROI identifier */
    lpsafloat_vreg_chc sRow,           /* row polynomial coefficient array */
    lpsafloat_vreg_chc sColumn         /* column polynomial coefficient array */
);
```

windowing_function

```
Idnstatic InWindowingFunction (        /* OUT windowing function array identifier */
    Idnstatic       nFunction,          /* windowing function array identifier */
    Idntuple        nSize,              /* array size ND 5-tuple */
    lpoint          uColumnGuard,       /* column guard band */
    lpoint          uRowGuard,          /* row guard band */
    lpoint          iOption             /* windowing option */
);
```

windowing_function_vreg

```
Idnvreg InWindowingFunctionVReg (      /* OUT windowing function array identifier */
    Idnvreg         nVReg,             /* output virtual register identifier or NULL */
    lpsstatic_vreg  snFunction,        /* windowing function array identifier */
    lpstuple_vreg   snSize,            /* array size ND 5-tuple */
    lpsuint_vreg_chc sColumnGuard,     /* column guard band */
    lpsuint_vreg_chc sRowGuard,        /* row guard band */
    lpsint_vreg_chc sOption            /* windowing option */
);
```

window_level

```
Idnimage InWindowLevel (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Ipsparameter_arith stAbove,      /* destination image identifier */
    Ipsparameter_arith stBelow,     /* above window clip value pointer union */
    Ipsparameter_arith stWidth,     /* below window clip value pointer union */
    Ipsparameter_arith stLevel,     /* window width pointer union */
    Ipsparameter_arith /* window level pointer union */
);
```

window_level_vreg

```
Idnvreg InWindowLevelVReg (
    Idnvreg          nVReg, /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier */
    Ipsparameter_arith_vreg_chc sAbove, /* destination image identifier */
    Ipsparameter_arith_vreg_chc sBelow, /* above window clip value */
    Ipsparameter_arith_vreg_chc sWidth, /* below window clip value */
    Ipsparameter_arith_vreg_chc sLevel, /* window width */
    Ipsparameter_arith_vreg_chc /* window level */
);
```

window_statistics

```
Idnimage InWindowStatistics (
    Idnimage          nSourceImage, /* OUT window statistics image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Ipuint            uIndex3,      /* destination image identifier */
    Ipuint            uIndex4,     /* third index value */
    Ipuint            uIndex5,     /* fourth index value */
    Ipuint            uWindowSize, /* fifth index value */
    Ipuint            iOption,     /* window size */
    Ipuint            /* window statistic option */
);
```

window_statistics_vreg

```
Idnvreg InWindowStatisticsVReg (
    Idnvreg          nVReg, /* OUT window statistics image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier */
    Ipsuint_vreg_chc sIndex3,      /* destination image identifier */
    Ipsuint_vreg_chc sIndex4,     /* third index value */
    Ipsuint_vreg_chc sIndex5,     /* fourth index value */
    Ipsuint_vreg_chc sWindowSize, /* fifth index value */
    Ipsint_vreg_chc  sOption,     /* window size */
    Ipsint_vreg_chc  /* window statistic option */
);
```

z_merge

```
Idnimage InZMerge (
    Idnimage          nSourceImage1, /* OUT destination image identifier */
    Idnimage          nSourceImage2, /* first source image (data) identifier */
    Idnimage          nSourceImage3, /* second source image (control) identifier */
    Idnimage          nSourceImage3, /* third source image (control) identifier */
    Idnimage          nDestImage     /* destination image (data) identifier */
);
```

z_merge_vreg

```
Idnvreg InZMergeVReg (
    Idnvreg          nVReg, /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage1, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage2, /* first source image (data) identifier */
    Ipsimage_vreg    snSourceImage3, /* second source image (control) identifier */
    Ipsimage_vreg    snSourceImage3, /* third source image (control) identifier */
    Ipsimage_vreg    snDestImage     /* destination image (data) identifier */
);
```

zoom

```
Idnimage InZoom (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Idntuple          nDestImage,   /* destination image identifier */
    Idntuple          nScale        /* zoom scale factor ND 5-tuple */
);
```

zoom_vreg

```
Idnvreg InZoomVReg (
    Idnvreg          nVReg, /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier */
    Ipsimage_vreg    snDestImage,   /* destination image identifier */
    Ipstuple_vreg     snScale        /* zoom scale factor ND 5-tuple */
);
```

zoom_roi

```
Idnroi InZoomROI (
    Idnroi          nSourceROI, /* OUT destination ROI identifier */
    Idnroi          nDestROI,   /* source ROI identifier */
    Idntuple        nDestROI,   /* destination ROI identifier */
    Idntuple        nScale      /* zoom scale factor ND 5-tuple */
);
```

zoom_roi_vreg

Idnvreg InZoomROIVReg (		/* OUT destination ROI identifier	*/
Idnvreg	nVReg,	/* output virtual register identifier or NULL	*/
Ipsroi_vreg	snSourceROI,	/* source ROI identifier	*/
Ipsroi_vreg	snDestROI,	/* destination ROI identifier	*/
Ipstuple_vreg	snScale	/* zoom scale factor ND 5-tuple	*/

);

3d_slice

Idnimage In3DSlice (		/* OUT destination image identifier	*/
Idnimage	nSourceImage,	/* source image identifier	*/
Idnimage	nDestImage,	/* destination image identifier	*/
Ipfloat	rSourceXOffset,	/* source x coordinate offset	*/
Ipfloat	rSourceYOffset,	/* source y coordinate offset	*/
Ipfloat	rSourceUOffset,	/* source third coordinate offset	*/
Ipfloat	rDestXOffset,	/* destination x coordinate offset	*/
Ipfloat	rDestYOffset,	/* destination y coordinate offset	*/
Ipfloat	rXYAngle,	/* x-y plane rotation angle	*/
Ipfloat	rXUAngle,	/* x-u plane rotation angle	*/
Ipfloat	rYUAngle	/* y-u plane rotation angle	*/

);

3d_slice_vreg

Idnvreg In3DSliceVReg (		/* OUT destination image identifier	*/
Idnvreg	nVReg,	/* output virtual register identifier or NULL	*/
Ipsimage_vreg	snSourceImage,	/* source image identifier	*/
Ipsimage_vreg	snDestImage,	/* destination image identifier	*/
Ipsfloat_vreg_chc	sSourceXOffset,	/* source x coordinate offset	*/
Ipsfloat_vreg_chc	sSourceYOffset,	/* source y coordinate offset	*/
Ipsfloat_vreg_chc	sSourceUOffset,	/* source third coordinate offset	*/
Ipsfloat_vreg_chc	sDestXOffset,	/* destination x coordinate offset	*/
Ipsfloat_vreg_chc	sDestYOffset,	/* destination y coordinate offset	*/
Ipsfloat_vreg_chc	sXYAngle,	/* x-y plane rotation angle	*/
Ipsfloat_vreg_chc	sXUAngle,	/* x-u plane rotation angle	*/
Ipsfloat_vreg_chc	sYUAngle	/* y-u plane rotation angle	*/

);

7.3 IPI-IIF functional element prototypes

attach_iif_syntax_entity

```

Ipntity InAttachEntity (                                /* OUT combined entities identifier */
    Ipntity      nDestEntity,                            /* destination IIF syntax entity identifier */
    Ipntity      nSubentity,                             /* source subentity identifier */
    Ipint        iCmp,                                   /* component number */
    Ipoint       uIndex                                 /* component index */
);

```

close_iif_gateway

```

void IvCloseGateway (                                /* no return value output */
    void                                /* no input arguments */
);

```

close_port

```

void IvClosePort (                                /* no return value output */
    Ipport      nPort                            /* port identifier */
);

```

compress_iif_syntax_entity

```

Ipndata InCompressEntity (                            /* OUT compressed data unit entity identifier */
    Ipsiif_pixel_types stInput,                        /* uncompressed pixel field pointer */
    Ipntity      nEntity,                             /* image structure entity identifier */
    Ipint        iOption,                             /* compression algorithm option */
    Ipscompress  sCompressData,                       /* compression parameters */
    Ipint        iApplBufferSize,                    /* data transfer buffer size or option */
    Ipoint       uStartPoint,                        /* index offset within internal data */
    Ipoint       *tuDataSize                         /* OUT post transfer data size pointer */
);

```

copy_iif_syntax_entity

```

Ipntity InCopyEntity (                                /* OUT new IIF syntax entity identifier */
    Ipntity      nInput                            /* input IIF syntax entity identifier */
);

```

create_iif_syntax_entity

```

Ipntity InCreateEntity (                            /* OUT IIF syntax entity identifier */
    Ipint        iNumber                            /* IIF syntax entity number */
);

```

decompress_iif_syntax_entity

```
Ipsiif_pixel_types IstDecompressEntity (      /* OUT uncompressed pixel field pointer */
    Ipnentity      nEntity,                    /* compressed data unit entity identifier */
    Ipnentity      nStructure,                 /* image structure entity identifier */
    Ipoint         iApplBufferSize,           /* data transfer buffer size or option */
    Ipoint         uStartPoint,               /* index offset within internal data */
    Ipoint         *tuDataSize                /* OUT post transfer data size pointer */
);
```

delete_iif_atomic_sequence_element

```
Ipnentity InDeleteAtomicSeqElement (          /* OUT IIF syntax entity identifier */
    Ipnentity      nEntity,                    /* IIF syntax entity identifier */
    Ipoint         iNumber,                    /* component number */
    Ipoint         uIndex                      /* component index */
);
```

delete_iif_syntax_entity

```
void IvDeleteEntity (                          /* no return value output */
    Ipnentity      nEntity                    /* IIF syntax entity identifier */
);
```

detach_iif_syntax_entity

```
Ipnentity InDetachEntity (                    /* OUT detached IIF syntax entity identifier */
    Ipnentity      nEntity,                    /* IIF syntax entity identifier */
    Ipoint         iNumber,                    /* component number */
    Ipoint         uIndex                      /* component index */
);
```

error_handler

```
void IvErrorHandler (                          /* no return value output */
    Ipoint         iCode,                      /* error number */
    Ipnentity      nEntity,                    /* IIF syntax entity identifier */
    Ipoint         iElement,                   /* error causing function descriptor */
    Ipnerror       nError                      /* error file identifier */
);
```

error_logger

```
void IvErrorLogger (
    Ipint      iCode,          /* no return output          */
    Ipntity    nEntity,       /* error number              */
    Ipint      iElement,      /* IIF syntax entity identifier */
    Ipnterror  nError,        /* error causing function descriptor */
    );                      /* error file identifier      */
```

error_test

```
Ipbool IbErrorTest (
    void          /* OUT error flag          */
    );           /* no input arguments      */
```

export_iif_data_to_piks

```
Ipport InExportIIFDataToPIKS (
    Ipport      nPort          /* OUT port identifier      */
    );           /* port identifier          */
```

generate_and_write_iif_image

```
Ipport InGenerateAndWriteImage (
    Ipntity    nEntity,       /* OUT port identifier      */
    Ipport      nPort,        /* IIF syntax entity identifier */
    char       *tzProfile,    /* port identifier          */
    );           /* profile string           */
```

get_iif_syntax_entity_component

```
Ipntity InGetEntityCmp (
    Ipntity    nEntity,       /* OUT subentity identifier  */
    Ipint      iNumber,       /* IIF syntax entity identifier */
    Ipuint     uIndex,        /* component number          */
    );           /* component index           */
```

get_iif_syntax_entity_value

```

Ipsentity_value IstGetEntityValue (                /* OUT value */
    Ipsentity      nEntity,                          /* IIF syntax entity identifier */
    Ipoint         iNumber,                          /* component number */
    Ipoint         uIndex,                          /* component index */
    Ipoint         iApplBufferSize,                 /* data transfer buffer size or option */
    Ipoint         uStartPoint,                     /* index offset within internal data */
    Ipoint         *tuDataSize                      /* OUT post transfer data size pointer */
);

```

get_image_subrange

```

Ipsentity InGetImageSubrange (                    /* OUT IIF syntax subrange entity identifier */
    Ipsentity      nEntity,                          /* IIF syntax input entity identifier */
    Ipoint         uHorizontalMin,                   /* horizontal minimum */
    Ipoint         uHorizontalMax,                  /* horizontal maximum */
    Ipoint         uVerticalMin,                    /* vertical minimum */
    Ipoint         uVerticalMax,                    /* vertical maximum */
    Ipoint         uDepthMin,                      /* depth minimum */
    Ipoint         uDepthMax,                      /* depth maximum */
    Ipoint         uTemporalMin,                   /* time minimum */
    Ipoint         uTemporalMax,                   /* time maximum */
    Ipoint         uSpectralMin,                   /* band minimum */
    Ipoint         uSpectralMax                    /* band maximum */
);

```

import_iif_data_from_piks

```

Ipsport InImportIIFDataFromPIKS (                /* OUT port identifier */
    Ipsport       nPort                             /* port identifier */
);

```

inquire_iif_gateway_status

```

void IvInquireGatewayStatus (                    /* no return value output */
    Ipoint         *tiStatus,                      /* OUT status of IIF gateway */
    Ipevalidity    *teValid                       /* OUT data validity indicator */
);

```

Note - A data validity parameter has been added for consistency with all other IPI inquire functions.

inquire_iif_syntax_component_type_definition

```
void IvInquireCmpTypeDef (                                /* no return value output */
    Ipint          iEntityNumber,                        /* IIF syntax entity type number */
    Ipoint         uCmpIndex,                            /* component index */
    Ipint          *tiCmpNumber,                        /* OUT component number */
    Ipbool         *tbOptional,                        /* OUT optional flag */
    Ipbool         *tbSequence,                        /* OUT sequence flag */
    Ipbool         *tbImplicit,                        /* OUT implicit flag */
    Ipbool         *tbChcFlag,                        /* OUT choice flag */
    Ipint          *tiChcNumber,                        /* OUT choice numeric identifier */
    Ipoint         *tuTag,                            /* OUT component tag value */
    Ipevalidity    *teValid                            /* OUT data validity indicator */
);
```

inquire_iif_syntax_entity_choice

```
void IvInquireEntityChc (                                /* no return value output */
    Ipntity        nEntity,                            /* IIF syntax entity identifier */
    Ipoint         uChcNumber,                        /* CHOICE number */
    Ipint          *tiCmpNumber,                        /* OUT IIF syntax entity component number */
    Ipevalidity    *teValid                            /* OUT data validity indicator */
);
```

inquire_iif_syntax_entity_number

```
void IvInquireEntityNumber (                            /* no return value output */
    Ipntity        nEntity,                            /* IIF syntax entity identifier */
    Ipint          *tiEntityNumber,                    /* OUT IIF syntax entity type number */
    Ipevalidity    *teValid                            /* OUT data validity indicator */
);
```

inquire_iif_syntax_entity_optional

```
void IvInquireEntityOptional (                          /* no return value output */
    Ipntity        nEntity,                            /* IIF syntax entity identifier */
    Ipint          iCmpNumber,                        /* component index */
    Ipbool         *tbOptional,                        /* OUT IIF syntax entity optional indicator */
    Ipevalidity    *teValid                            /* OUT data validity indicator */
);
```

inquire_iif_syntax_entity_sequence

```
void IvInquireEntitySeq (                                /* no return value output */
    Ipntity          nEntity,                            /* IIF syntax entity identifier */
    Ipint            iCmpNumber,                         /* component number */
    Ipoint           *tuLength,                          /* number of subentities */
    Ipevalidity      *teValid                            /* OUT data validity indicator */
);
```

inquire_iif_syntax_entity_type_definition

```
void IvInquireEntityTypeDef (                          /* no return value output */
    Ipint            nEntityNumber,                     /* IIF syntax entity type number */
    Ipint            *tiClass,                          /* OUT IIF syntax entity class */
    Ipoint           *tuNumber,                         /* OUT number of components */
    Ipevalidity      *teValid                          /* OUT data validity indicator */
);
```

inquire_iif_syntax_entity_validity

```
void IvInquireEntityValidity (                          /* no return value output */
    Ipntity          nEntity,                            /* IIF syntax entity identifier */
    char             *tzProfile,                        /* profile syntax entity */
    Ipbool           *tbValid,                         /* OUT syntax entity identifier valid indicator */
    Ipoint           *tuMissingCount,                  /* OUT number of missing components */
    Ipint            *tiMissing,                       /* OUT list of missing components */
    Ipbool           *tbConstraint,                    /* OUT constraint error indicator */
    Ipevalidity      *teValid                          /* OUT data validity indicator */
);
```

Note - The tuMissingCount parameter was added to indicate the number of integers returned in the tiMissing list.

inquire_port

```
void IvInquirePort (                                  /* no return value output */
    Iport            nPort,                             /* port identifier */
    Ipint            *tiDirection,                     /* OUT input/output direction */
    Ipint            *tiAccess,                        /* OUT input/output access type */
    Ipint            *tiDomain,                       /* OUT input/output domain */
    Ipint            *tiStatus,                       /* OUT input/output status */
    Ipevalidity      *teValid                          /* OUT data validity indicator */
);
```

inquire_supported_functionality

```
void IvInquireFunctionality (                /* no return value output */
    Ipint      iFunction,                    /* function descriptor */
    Ipbool     *tbResponse,                 /* OUT registration response */
    Ipevalidity *teValid                    /* OUT data validity indicator */
);
```

insert_iif_atomic_sequence_element

```
Ipntity InInsertAtomicSeqElement (          /* OUT IIF syntax entity identifier */
    Ipntity     nEntity,                    /* IIF syntax entity identifier */
    Ipint       iCmpNumber,                 /* component number */
    Ipoint      uIndex                      /* component index */
);
```

open_iif_gateway

```
void IvOpenGateway (                        /* OUT error file identifier */
    Ipnterror   nError                     /* error file identifier */
);
```

open_port

```
Ipport InOpenPort (                        /* OUT port identifier */
    Ipport_name nPort,                     /* port name */
    Ipint       iDirection,                /* input/output direction option */
    Ipint       iAccess,                   /* input/output access type option */
    Ipint       iDomain                     /* input/output domain option */
);
```

put_iif_syntax_entity_value

```
Ipntity InPutEntityValue (                 /* OUT IIF syntax entity identifier */
    Ipntity     nEntity,                    /* IIF syntax entity identifier */
    Ipsentity_value stValue,                 /* entity value */
    Ipint       iCmpNumber,                 /* component number */
    Ipoint      uIndex,                     /* component index */
    Ipint       iApplBufferSize,            /* data transfer buffer size or option */
    Ipoint      uStartPoint,                /* index offset within internal data */
    Ipoint      *tuDataSize                 /* OUT post transfer data size pointer */
);
```

put_image_subrange

```

Ipnentity InPutImageSubrange (                /* OUT subimage IIF syntax entity identifier */
    Ipnentity      nEntity,                    /* IIF syntax entity image identifier */
    Ipnentity      nSubEntity,                 /* IIF syntax entity subimage identifier */
    Ipoint         uHorizontal,                /* horizontal start */
    Ipoint         uVertical,                 /* vertical start */
    Ipoint         uDepth,                   /* depth start */
    Ipoint         uTemporal,                 /* temporal start */
    Ipoint         uSpectral,                 /* band start */
);

```

read_and_parse_iif_image

```

Ipnentity InReadAndParseImage (                /* OUT IIF syntax entity identifier */
    Ipnport        nPort,                    /* port identifier */
    Ipbbool        bImageSkip,               /* image data skipping option */
    Ipbbool        bRelatedSkip,             /* image-related data skipping option */
    Ipbbool        bAttrSkip,                /* image attribute skipping option */
    Ipbbool        bAnnotationSkip,          /* image annotation skipping option */
    char           *tzProfile,                /* profile string */
);

```

release_iif_syntax_entity_identifier

```

void IvReleaseEntityId (                      /* no return value output */
    Ipnentity      nEntity,                  /* IIF syntax entity identifier */
);

```

reset_iif_gateway

```

void IvResetGateway (                        /* no return value output */
    void,                                       /* no input arguments */
);

```

reset_port

```

void IvResetPort (                          /* no return value output */
    Ipnport        nPort,                   /* port identifier */
);

```

set_error_handler

```
Iptferror_function ItfSetErrorHandler (          /* OUT old error handling function pointer */
    Iptferror_function    tfNewErrorHandler      /* new error function pointer */
);
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 12088-4:1995

8 Convenience functions

Some frequently utilized IPI operations require the application of multiple functions or constant portions of function argumentation to achieve the desired result. In this clause, a number of functions are specified for the convenience of the application programmer in consolidating and simplifying several of these compound or simplified operations.

These convenience functions are normative to this part of ISO/IEC 12088 in order to assure portability of application software.

8.1 IPI-PIKS convenience functions

The creation of basic objects requires the execution of several functions. For example, to prepare an image object prior to transferring any image data into the object, the user must create three tuples and allocate the image desired. This image preparation may be accomplished by executing no less than seven functions: three *allocate_tuple* functions, three *create_tuple* functions and one *allocate_image* function.

For creating or preparing frequently specified objects with attributes alone or with attributes and data including images, ROI objects and tuples, a number of convenience functions are presented. IPI-PIKS convenience functions are provided for monochrome and colour image preparations, variously structured ROI creations, commonly used tuple generations and tiled image import and export.

8.1.1 Image preparation functions

Three image preparation convenience functions are provided:

- *prepare_colour_image* will return the identifier of a data-empty colour image object containing attributes, or NULL if error occurs, given the width, height, band count (3 or 4), pixel type and precision, tristimulus values and the color space option as input arguments.
- *prepare_monochrome_image* will return the identifier of a data-empty monochrome image object containing attributes, or NULL if error occurs, given the width and height, pixel type and precision as input arguments.
- *create_unbound_image_copy* will return the identifier of an image object containing attributes and data copied from the source image, specified as input argument, copied with ROI and match point control disabled. NULL is returned if error occurs.

These three functions are applicable within all PIKS profiles.

Convenience functions for tuple creation from 8.1.3 are utilized. All tuples required for the development of these image objects are deallocated within the function after use.

The functions require include of applicable system and PIKS header files, the latter sufficient to represent the PIKS foundation profile, as presented in Annex B.

8.1.1.1 prepare_colour_image

```

Idnimage InPrepareColourImage (                /* OUT 4 or less band image identifier */
    Ipuint          uHorizontal,                /* image width */
    Ipuint          uVertical,                  /* image height */
    Ipuint          uBands,                     /* image bands */
    Ipuint          iType,                      /* image pixel data type */
    Ipuint          uPrecision,                 /* image pixel precision */
    Ippfloat        rXWhite,                    /* X tristimulus value white point */
    Ippfloat        rYWhite,                    /* Y tristimulus value white point */
    Ippfloat        rZWhite,                    /* Z tristimulus value white point */
    Ipuint          iColourOption               /* image colour space code */
)

{
/* data declarations */
    Idnimage        nImage;
    Idntuple        nSize, nType = NULL, nPrecision = NULL;
    Ipsobject       snObject;
    Ipsattrs_image  slmageAttrs;

/* program */

/* generate temporary tuples */
    if (nSize = InGenerateND5Tuple(uHorizontal, uVertical, 1, 1, uBands)) {
        switch(uBands) {
            case 3:
                if (nPrecision = InGenerateND3Tuple(uPrecision, uPrecision, uPrecision))
                    nType = InGenerateSD3Tuple(iType, iType, iType);
                break;

            case 4:
                if (nPrecision = InGenerateND4Tuple(uPrecision, uPrecision, uPrecision,
                    uPrecision))
                    nType = InGenerateSD4Tuple(iType, iType, iType, iType);
                break;

            default:
                nPrecision = nType = NULL;
                break;
        }
        if (nType) {

```

```

/* set image attributes and allocate image */
    sImageAttrs.nImageSize = nSize;
    sImageAttrs.nBandType = nType;
    sImageAttrs.nBandPrecision = nPrecision;
    sImageAttrs.rXWhite = rXWhite;
    sImageAttrs.rYWhite = rYWhite;
    sImageAttrs.rZWhite = rZWhite;
    sImageAttrs.iImageStructureOption = IIMAGE_STRUCTURE_COLR;
    sImageAttrs.iColourOption = iColourOption;
    nImage = InAllocateImage(sImageAttrs);
}
}

/* deallocate temporary tuples */
if (snObject.nTuple = nSize)
    IvDeallocateDataObject(snObject);
if (snObject.nTuple = nType)
    IvDeallocateDataObject(snObject);
if (snObject.nTuple = nPrecision)
    IvDeallocateDataObject(snObject);

/* exit */
return nImage;
}

```

8.1.1.2 prepare_monochrome_image

```

Idnimage InPrepareMonochromeImage (      /* OUT monochrome image identifier */
    Ipuint      uHorizontal,              /* image width */
    Ipuint      uVertical,                /* image height */
    Ipuint      iType,                    /* image pixel data type */
    Ipuint      uPrecision                /* image pixel precision */
)

{
    /* data declarations */
    Idnimage      nImage;
    Idntuple      nSize, nType = NULL, nPrecision = NULL;
    Ipsobject      snObject;
    Ipsattrs_image sImageAttrs;

    /* program */

    /* generate temporary tuples */
    if (nSize = InGenerateND5Tuple(uHorizontal, uVertical, 1, 1, 1)) {
        if (nPrecision = InGenerateND1Tuple(uPrecision)) {
            if (nType = InGenerateSD1Tuple(iType)) {

```

```

/* set image attributes and allocate image */
    sImageAttrs.nImageSize = nSize;
    sImageAttrs.nBandType = nType;
    sImageAttrs.nBandPrecision = nPrecision;
    sImageAttrs.rXWhite = 0;
    sImageAttrs.rYWhite = 0;
    sImageAttrs.rZWhite = 0;
    sImageAttrs.iImageStructureOption = IIMAGE_STRUCTURE_MON;
    sImageAttrs.iColourOption = ICOLOUR_NONE;
    nImage = InAllocateImage(sImageAttrs);
}
}

/* deallocate temporary tuples */
    if (snObject.nTuple = nSize)
        IvDeallocateDataObject(snObject);
    if (snObject.nTuple = nType)
        IvDeallocateDataObject(snObject);
    if (snObject.nTuple = nPrecision)
        IvDeallocateDataObject(snObject);

/* exit */
    return nImage;
}

```

8.1.1.3 create_unbound_image_copy

```

Idnimage InCreateUnbndImageCopy (      /* OUT destination image identifier */
    Idnimage      nSourceImage          /* source image identifier */
)
{
    /* data declarations */
    Idnimage      nDestImage, nTemplImage;
    Idntuple      nOffset = NULL;
    Ipsobject      snObject;
    Ipsattrs_image sAttrs, *tsAttrs = &sAttrs;
    Ipspiks_modes sModes, *tsModes = &sModes;
    Ipspiks_modes sTempModes = {IROI_CONTROL_OFF, IROI_PROCESS_OFF,
                                IMATCH_POINT_OFF, IINDEX_ASSIGNMENT_OFF, 0, 0, 0, 0};
    Ipevalidity    eValid, *teValid = &eValid;

    /* program */

    /* acquire source image attributes */
    IvInquireImage(nSourceImage, tsAttrs, NULL, teValid);
    if (eValid == IVALIDITY_INVALID)
        return (Idnimage)NULL;
}

```

```
/* acquire existing modes */
    IvInquirePIKSStatus(NULL, tsModes, teValid);
    if (eValid == IVALIDITY_INVALID)
        return (Idnimage)NULL;

/* allocate image with duplicate attributes */
    if (nTempImage = InAllocateImage(sAttrs)) {

/* set temporary modes */
        sTempModes.eSynchronicityMode = sModes.eSynchronicityMode;
        sTempModes.eErrorReportMode = sModes.eErrorReportMode;
        sTempModes.iImageResampleMode = sModes.iImageResampleMode;
        sTempModes.iROIResampleMode = sModes.iROIResampleMode;
        IvSetGlobals(sTempModes);

/* copy unbound image */
        if (nOffset = InGenerateSD5Tuple(0, 0, 0, 0, 0)) {
            nDestImage = InCopyWindow(nSourceImage, nTempImage, nOffset,
                                      nOffset, sAttrs.nImageSize);
        }
        else {
            snObject.nImage = nTempImage;
            IvDeallocateDataObject(snObject);
        }
    }

/* restore modes */
    IvSetGlobals(sModes);

/* deallocate tuples */
    if (snObject.nTuple = sAttrs.nImageSize)
        IvDeallocateDataObject(snObject);
    if (snObject.nTuple = sAttrs.nBandType)
        IvDeallocateDataObject(snObject);
    if (snObject.nTuple = sAttrs.nBandPrecision)
        IvDeallocateDataObject(snObject);
    if (snObject.nTuple = nOffset)
        IvDeallocateDataObject(snObject);

/* exit */
    return nDestImage;
}
```


8.1.2 ROI creation functions

Generate ROI functions are provided to implement a ROI with attributes and internal rectangle. Prepare ROI functions are provided to create basic objects with attributes only. In total, seven functions are provided:

- *generate_2d_roi_rectangular* will return the identifier of a two-dimensional rectangular ROI containing attributes and an internal rectangle, or NULL if error occurs, given the width and height of the ROI, the horizontal and vertical, start and end coordinates of the rectangle and the polarity option to be applied to the ROI.
- *generate_roi_coordinate* will return the identifier of a ROI containing attributes and an internal structure defined by a coordinate array, or NULL if error occurs, given the pointer to the coordinate array, the five dimensions of the ROI, the count of array coordinate sets and the polarity option to be applied to the ROI.
- *generate_roi_elliptical* will return the identifier of a ROI containing attributes and an internal ellipse, or NULL if error occurs, given the five dimensions of the ROI, the five centre positions and axial lengths of the ellipse, the relative assignment of input and output indices, the dimensions of the ellipse and the polarity option to be applied to the ROI.
- *generate_roi_polygon* will return the identifier of a ROI containing attributes and an internal two-dimensional polygon, or NULL if error occurs, given the pointer to the coordinate array, the five dimensions of the ROI, the relative assignment of input and output indices, the count of array coordinate pairs and the polarity option to be applied to the ROI.
- *generate_roi_rectangular* will return the identifier of a rectangular ROI containing attributes and an internal multi-dimensional rectangular structure, or NULL if error occurs, given the five dimensions of the ROI, the five dimensional, start and end coordinates of the rectangular structure, the relative assignment of input and output indices, the dimensions of the rectangular structure and the polarity option to be applied to the ROI.
- *prepare_2d_roi_rectangular* will return the identifier of an empty two-dimensional ROI containing attributes including rectangular ROI structure option, or NULL if error occurs, given the width and height of the ROI and the polarity option to be applied to the ROI.
- *prepare_roi* will return the identifier of an empty ROI containing attributes, or NULL if error occurs, given the five dimensions of the ROI, the structure option and the polarity option to be applied to the ROI.

The *generate_2d_roi_rectangular* and *prepare_2d_roi_rectangular* functions are applicable within all PIKS profiles. The others are intended for all PIKS profiles except the foundation profile.

Convenience functions for tuple creation from 8.1.3 are utilized. All tuples required for the development of these ROI objects are deallocated with the function after use.

The functions require include of applicable system and PIKS header files, the latter sufficient to represent the PIKS foundation for *generate_2d_roi_rectangular* and *prepare_2d_roi_rectangular* functions and technical profiles for all other functions in this collection, as presented in Annex B.

8.1.2.1 generate_2d_roi_rectangular

```

Idnroi InGenerate2DROIRectangular (          /* OUT ROI identifier          */
    Ipuint      uHorizontal,                  /* ROI width                      */
    Ipuint      uVertical,                    /* ROI height                      */
    Ipuint      uStartHorizontal,             /* rectangle horizontal start      */
    Ipuint      uStartVertical,               /* rectangle vertical start        */
    Ipuint      uEndHorizontal,               /* rectangle horizontal end        */
    Ipuint      uEndVertical,                 /* rectangle vertical end          */
    Ipepolarity ePolarity                     /* polarity of ROI content        */
)
{
    /* data declarations */
    Idnroi      nROI, nTempROI;
    Idntuple     nSize, nStart = NULL, nEnd = NULL;
    Ipsobject     snObject;
    Ipsattrs_roi sROIAttrs;

    /* program */

    /* generate temporary tuples */
    if (nSize = InGenerateND5Tuple(uHorizontal, uVertical, 1, 1, 1)) {
        if (nStart = InGenerateND5Tuple(uStartHorizontal, uStartVertical, 0, 0, 0)) {
            if (nEnd = InGenerateND5Tuple(uEndHorizontal, uEndVertical, 0, 0, 0)) {

                /* allocate ROI */

                sROIAttrs.nROISize = nSize;
                sROIAttrs.iStructureOption = IROI_RECTANGULAR;
                sROIAttrs.ePolarityOption = ePolarity;
                if (nTempROI = InAllocateROI(sROIAttrs)) {

                    /* define ROI rectangular */

                    if (!(nROI = InROIRectangular(nTempROI, nSize, nStart,
                                                    nEnd, NULL, IDIMENSION_2D, ePolarity))) {
                        snObject.nROI = nTempROI;
                        IvDeallocateDataObject(snObject);
                    }
                }
            }
        }
    }

    /* deallocate temporary tuples */
    if (snObject.nTuple = nSize)
        IvDeallocateDataObject(snObject);
    if (snObject.nTuple = nStart)
        IvDeallocateDataObject(snObject);
    if (snObject.nTuple = nEnd)
        IvDeallocateDataObject(snObject);

    /* exit */
    return nROI;
}

```

8.1.2.2 generate_roi_coordinate

```

Idnroi InGenerateROICoordinate (                                /* OUT ROI identifier */
    Ipoint      *tuaCoordinateArray, /* pointer to coordinates array */
    Ipoint      uHorizontal,          /* ROI width */
    Ipoint      uVertical,            /* ROI height */
    Ipoint      uDepth,               /* ROI depth */
    Ipoint      uTemporal,            /* ROI temporal size */
    Ipoint      uSpectral,             /* ROI spectral size */
    Ipoint      uCoordinateCount,     /* number of coordinate sets */
    Ipepolarity ePolarity             /* polarity of ROI content */
)

{
    /* data declarations */
    Idnroi      nROI, nTempROI;
    Idntuple    nSize;
    Ipsobject    snObject;
    Ipsattrs_roi sROIAttrs;

    /* program */

    /* generate temporary tuple */
    if (nSize = InGenerateND5Tuple(uHorizontal, uVertical, uDepth, uTemporal, uSpectral)) {

        /* allocate ROI */
        sROIAttrs.nROISize = nSize;
        sROIAttrs.iStructureOption = IROI_COORDINATE;
        sROIAttrs.ePolarityOption = ePolarity;
        if (nTempROI = InAllocateROI(sROIAttrs)) {

            /* define ROI coordinate array */
            if (!(nROI = InROICoordinate(tuaCoordinateArray, nTempROI, nSize,
                                         uCoordinateCount, ePolarity))) {
                snObject.nROI = nTempROI;
                IvDeallocateDataObject(snObject);
            }
        }

        /* deallocate temporary tuple */
        if (snObject.nTuple = nSize)
            IvDeallocateDataObject(snObject);

        /* exit */
        return nROI;
    }
}

```

8.1.2.3 generate_roi_elliptical

```

Idnroi InGenerateROIElliptical (                                /* OUT ROI identifier */
    Ipuint          uHorizontal,                                /* ROI width */
    Ipuint          uVertical,                                  /* ROI height */
    Ipuint          uDepth,                                    /* ROI depth */
    Ipuint          uTemporal,                                /* ROI temporal size */
    Ipuint          uSpectral,                                  /* ROI spectral size */
    Ipfloat         rCentreHorizontal,                         /* ellipse centre horizontal */
    Ipfloat         rCentreVertical,                           /* ellipse centre vertical */
    Ipfloat         rCentreDepth,                              /* ellipse centre depth */
    Ipfloat         rCentreTemporal,                           /* ellipse centre temporal dimension */
    Ipfloat         rCentreSpectral,                           /* ellipse centre spectral dimension */
    Ipfloat         rLengthHorizontal,                         /* ellipse length horizontal */
    Ipfloat         rLengthVertical,                           /* ellipse length vertical */
    Ipfloat         rLengthDepth,                              /* ellipse length depth */
    Ipfloat         rLengthTemporal,                           /* ellipse length temporal dimension */
    Ipfloat         rLengthSpectral,                           /* ellipse length spectral dimension */
    Ipeindex        eJDimension,                              /* j dimension assignment */
    Ipeindex        eKDimension,                              /* k dimension assignment */
    Ipeindex        eLDimension,                              /* l dimension assignment */
    Ipeindex        eMDimension,                              /* m dimension assignment */
    Ipeindex        eNDimension,                              /* n dimension assignment */
    Ipedimension    eDimension,                               /* ellipse dimension option */
    Ipepolarity     ePolarity                                /* polarity of ROI content */
)

{
/* data declarations */
    Idnroi          nROI, nTempROI;
    Idntuple        nSize, nCentre = NULL, nLength = NULL, nIndex = NULL;
    Ipsobject       snObject;
    Ipsattrs_roi    sROIAttrs;

/* program */

/* generate temporary tuples */
    if (nSize = InGenerateND5Tuple(uHorizontal, uVertical, uDepth, uTemporal, uSpectral)) {
        if (nCentre = InGenerateRD5Tuple(rCentreHorizontal, rCentreVertical,
            rCentreDepth, rCentreTemporal, rCentreSpectral)) {
            if (nLength = InGenerateRD5Tuple(rLengthHorizontal, rLengthVertical,
                rLengthDepth, rLengthTemporal, rLengthSpectral)) {
                if (nIndex = InGenerateND5Tuple(eJDimension, eKDimension,
                    eLDimension, eMDimension, eNDimension)) {

/* allocate ROI */

                    sROIAttrs.nROISize = nSize;
                    sROIAttrs.iStructureOption = IROI_ELLIPTICAL;
                    sROIAttrs.ePolarityOption = ePolarity;
                    if (nTempROI = InAllocateROI(sROIAttrs)) {

```

```

/* define ROI elliptical */

if (!(nROI = InROIElliptical(nTempROI, nSize,
                             nCentre, nLength, nIndex, eDimension,
                             ePolarity))) {
    snObject.nROI = nTempROI;
    IvDeallocateDataObject(snObject);
}

}

}

}

}

/* deallocate temporary tuples */
if (snObject.nTuple = nSize)
    IvDeallocateDataObject(snObject);
if (snObject.nTuple = nCentre)
    IvDeallocateDataObject(snObject);
if (snObject.nTuple = nLength)
    IvDeallocateDataObject(snObject);
if (snObject.nTuple = nIndex)
    IvDeallocateDataObject(snObject);

/* exit */
return nROI;
}

```

8.1.2.4 generate_roi_polygon

```

Idnroi InGenerateROIPolygon ( /* OUT ROI identifier */
    Ipoint *tuaCoordinateArray, /* pointer to 2D coordinate array */
    Ipoint uHorizontal, /* ROI width */
    Ipoint uVertical, /* ROI height */
    Ipoint uDepth, /* ROI depth */
    Ipoint uTemporal, /* ROI temporal size */
    Ipoint uSpectral, /* ROI spectral size */
    Ipeindex eJDimension, /* j dimension assignment */
    Ipeindex eKDimension, /* k dimension assignment */
    Ipeindex eLDimension, /* l dimension assignment */
    Ipeindex eMDimension, /* m dimension assignment */
    Ipeindex eNDimension, /* n dimension assignment */
    Ipoint uCoordinateCount, /* number of coordinates sets */
    Ipepolarity ePolarity /* polarity of ROI content */
)

{
/* data declarations */
    Idnroi nROI, nTempROI;
    Idntuple nSize, nIndex = NULL;
    Ipsobject snObject;
    Ipsattr_s_roi sROIAttrs;
}

```

```

/* program */

/* generate temporary tuples */
    if (nSize = InGenerateNDSTuple(uHorizontal, uVertical, uDepth, uTemporal, uSpectral)) {
        if (nIndex = InGenerateNDSTuple(eJDimension, eKDimension, eLDimension,
            eMDimension, eNDimension)) {

/* allocate ROI */
            sROIAttrs.nROISize = nSize;
            sROIAttrs.iStructureOption = IROI_POLYGON;
            sROIAttrs.ePolarityOption = ePolarity;
            if (nTempROI = InAllocateROI(sROIAttrs)) {

/* define ROI polygon */
                if (!(nROI = InROIIPolygon(tuaCoordinateArray, nTempROI, nSize,
                    uCoordinateCount, nIndex, ePolarity))) {
                    snObject.nROI = nTempROI;
                    IvDeallocateDataObject(snObject);
                }
            }
        }
    }

/* deallocate temporary tuples */
    if (snObject.nTuple = nSize)
        IvDeallocateDataObject(snObject);
    if (snObject.nTuple = nIndex)
        IvDeallocateDataObject(snObject);

/* exit */
    return nROI;
}

```

8.1.2.5 generate_roi_rectangular

```

Idnroi InGenerateROIRectangular (                /* OUT ROI identifier */
    Ipoint    uHorizontal,                        /* ROI width */
    Ipoint    uVertical,                          /* ROI height */
    Ipoint    uDepth,                            /* ROI depth */
    Ipoint    uTemporal,                         /* ROI temporal size */
    Ipoint    uSpectral,                         /* ROI spectral size */
    Ipoint    uStartHorizontal,                  /* rectangle start horizontal */
    Ipoint    uStartVertical,                    /* rectangle start vertical */
    Ipoint    uStartDepth,                      /* rectangle start depth */
    Ipoint    uStartTemporal,                   /* rectangle start temporal dimension */
    Ipoint    uStartSpectral,                    /* rectangle start spectral dimension */
    Ipoint    uEndHorizontal,                    /* rectangle end horizontal */
    Ipoint    uEndVertical,                      /* rectangle end vertical */
    Ipoint    uEndDepth,                        /* rectangle end depth */
    Ipoint    uEndTemporal,                     /* rectangle end temporal dimension */
    Ipoint    uEndSpectral,                     /* rectangle end spectral dimension */
    Ipeindex  eJDimension,                       /* j dimension assignment */
    Ipeindex  eKDimension,                       /* k dimension assignment */
    Ipeindex  eLDimension,                       /* l dimension assignment */
    Ipeindex  eMDimension,                       /* m dimension assignment */
    Ipeindex  eNDimension,                       /* n dimension assignment */
    Ipedimension eDimension,                     /* rectangle dimension option */
    Ipepolarity ePolarity                       /* polarity of ROI content */
)

{
/* data declarations */
    Idnroi    nROI, nTempROI;
    Idntuple  nSize, nStart = NULL, nEnd = NULL, nIndex = NULL;
    Ipsobject snObject;
    Ipsattrs_roi sROIAttrs;

/* program */

/* generate temporary tuples */
    if (nSize = InGenerateND5Tuple(uHorizontal, uVertical, uDepth, uTemporal, uSpectral)) {
        if (nStart = InGenerateND5Tuple(uStartHorizontal, uStartVertical,
            uStartDepth, uStartTemporal, uStartSpectral)) {
            if (nEnd = InGenerateND5Tuple(uEndHorizontal, uEndVertical,
                uEndDepth, uEndTemporal, uEndSpectral)) {
                if (nIndex = InGenerateND5Tuple(eJDimension, eKDimension,
                    eLDimension, eMDimension, eNDimension)) {

/* allocate ROI */

                    sROIAttrs.nROISize = nSize;
                    sROIAttrs.iStructureOption = IROI_RECTANGULAR;
                    sROIAttrs.ePolarityOption = ePolarity;
                    if (nTempROI = InAllocateROI(sROIAttrs)) {

```

```

/* define ROI elliptical */
    if (! (nROI = InROIRectangular(nTempROI, nSize,
                                   nStart, nEnd, nIndex, eDimension,
                                   ePolarity))) {
        snObject.nROI = nTempROI;
        IvDeallocateDataObject(snObject);
    }
}
}
}
}

/* deallocate temporary tuples */
    if (snObject.nTuple = nSize)
        IvDeallocateDataObject(snObject);
    if (snObject.nTuple = nStart)
        IvDeallocateDataObject(snObject);
    if (snObject.nTuple = nEnd)
        IvDeallocateDataObject(snObject);
    if (snObject.nTuple = nIndex)
        IvDeallocateDataObject(snObject);

/* exit */
    return nROI;
}

```

8.1.2.6 prepare_2d_roi_rectangular

```

Idnroi InPrepare2DROIRectangular( /* OUT ROI identifier */
    Ipuint      uHorizontal, /* ROI width */
    Ipuint      uVertical,   /* ROI height */
    Ipepolarity  ePolarityOption /* ROI polarity option */
)
{
    /* data declarations */
    Idnroi nROI;
    Idntuple nSize;
    Ipsobject snObject;
    Ipsattrs_roi sROIAttrs;

    /* program */

    /* generate temporary size tuple */
    if (nSize = InGenerateND5Tuple(uHorizontal, uVertical, 1, 1, 1)) {

        /* set ROI attributes and allocate ROI */
        sROIAttrs.nROISize = nSize;
        sROIAttrs.iStructureOption = IROI_RECTANGULAR;
        sROIAttrs.ePolarityOption = ePolarityOption;
        nROI = InAllocateROI(sROIAttrs);
    }
}

```



```
/* deallocate temporary size tuple */
    if (snObject.nTuple = nSize)
        IvDeallocateDataObject(snObject);

/* exit */
    return nROI;
}
```

8.1.2.7 prepare_roi

```
Idnroi InPrepareROI ( /* OUT ROI identifier */
    Ipoint uHorizontal, /* ROI width */
    Ipoint uVertical, /* ROI height */
    Ipoint uDepth, /* ROI depth */
    Ipoint uTemporal, /* ROI temporal dimension */
    Ipoint uSpectral, /* ROI spectral dimension */
    Ipoint iStructureOption, /* ROI structure option */
    Ipoint ePolarityOption /* ROI polarity option */
)

{
/* data declarations */
    Idnroi nROI;
    Idntuple nSize;
    Ipsobject snObject;
    Ipsattrs_roi sROIAttrs;

/* program */

/* generate temporary size tuple */
    if (nSize = InGenerateND5Tuple(uHorizontal, uVertical, 1, 1, 1)) {

/* set ROI attributes and allocate ROI */
        sROIAttrs.nROISize = nSize;
        sROIAttrs.iStructureOption = IROI_RECTANGULAR;
        sROIAttrs.ePolarityOption = ePolarityOption;
        nROI = InAllocateROI(sROIAttrs);
    }

/* deallocate temporary size tuple */
    if (snObject.nTuple = nSize)
        IvDeallocateDataObject(snObject);

/* exit */
    return nROI;
}
```

8.1.3 Tuple generation functions

Eleven generate tuple functions are provided to expedite the creation of most commonly used tuple configurations and types:

- *generate_ND_1_tuple* will return the identifier of a 1-tuple of data type ND, or NULL if error occurs, given the single value to be installed in the tuple.
- *generate_ND_3_tuple* will return the identifier of a 3-tuple of data type ND, or NULL if error occurs, given the three values to be installed in the tuple.
- *generate_ND_4_tuple* will return the identifier of a 4-tuple of data type ND, or NULL if error occurs, given the four values to be installed in the tuple.
- *generate_ND_5_tuple* will return the identifier of a 5-tuple of data type ND, or NULL if error occurs, given the five values to be installed in the tuple.
- *generate_RD_3_tuple* will return the identifier of a 3-tuple of data type float, or NULL if error occurs, given the three values to be installed in the tuple.
- *generate_RD_4_tuple* will return the identifier of a 4-tuple of data type float, or NULL if error occurs, given the four values to be installed in the tuple.
- *generate_RD_5_tuple* will return the identifier of a 5-tuple of data type float, or NULL if error occurs, given the five values to be installed in the tuple.
- *generate_SD_1_tuple* will return the identifier of a 1-tuple of data type SD, or NULL if error occurs, given the single value to be installed in the tuple.
- *generate_SD_3_tuple* will return the identifier of a 3-tuple of data type SD, or NULL if error occurs, given the three values to be installed in the tuple.
- *generate_SD_4_tuple* will return the identifier of a 4-tuple of data type SD, or NULL if error occurs, given the four values to be installed in the tuple.
- *generate_SD_5_tuple* will return the identifier of a 5-tuple of data type SD, or NULL if error occurs, given the five values to be installed in the tuple.

These eleven functions are applicable within all PIKS profiles.

The functions require include of applicable system and PIKS header files, the latter sufficient to represent the PIKS foundation profile, as presented in Annex B.

8.1.3.1 generate_nd_1_tuple

```

Idntuple InGenerateND1Tuple (                                /* OUT one parameter ND type tuple identifier */
    Ipoint            uValue1                                /* first value, e.g., X */
)

{
/* data declarations */
    Idntuple          nIdAllocate, nIdCreate = NULL;
    Ipsattrs_tuple    sTupleAttrs;
    Ipoint            uData;
    Ipsparameter_basic stData;
    Ipsobject          snObject;

/* program */

/* initialize tuple attributes */
    sTupleAttrs.uTupleSize = 1;
    sTupleAttrs.iTypeOption = IDATA_TYPE_INTERNAL_ND;

/* allocate ND 1-tuple */
    if ((nIdAllocate = InAllocateTuple(sTupleAttrs)) != NULL) {

/* parameter array values */
        uData = uValue1;
        stData.tuParameter = &uData;

/* establish value in tuple */
        if ((nIdCreate = InCreateTuple(stData, nIdAllocate, sTupleAttrs)) == NULL) {
            snObject.nTuple = nIdAllocate;
            IvDeallocateDataObject(snObject);
        }
    }

/* exit */
    return nIdCreate;
}

```

8.1.3.2 generate_nd_3_tuple

```

Idntuple InGenerateND3Tuple (                                /* OUT three parameter ND type tuple identifier */
    Ipuint            uValue1,                                /* first value, e.g., X */
    Ipuint            uValue2,                                /* second value, e.g., Y */
    Ipuint            uValue3                                /* third value, e.g., Z */
)

{
    /* data declarations */
    Idntuple          nIdAllocate, nIdCreate = NULL;
    Ipsattrs_tuple    sTupleAttrs;
    Ipuint            uaData[3];
    Ipsparameter_basic stData;
    Ipsobject          snObject;

    /* program */

    /* initialize tuple attributes */
    sTupleAttrs.uTupleSize = 3;
    sTupleAttrs.iTypeOption = IDATA_TYPE_INTERNAL_ND;

    /* allocate ND 3-tuple */
    if ((nIdAllocate = InAllocateTuple(sTupleAttrs)) != NULL) {

        /* parameter array values */
        uaData[0] = uValue1;
        uaData[1] = uValue2;
        uaData[2] = uValue3;
        stData.tuParameter = &uaData[0];

        /* establish values in tuple */
        if ((nIdCreate = InCreateTuple(stData, nIdAllocate, sTupleAttrs)) == NULL) {
            snObject.nTuple = nIdAllocate;
            IvDeallocateDataObject(snObject);
        }
    }

    /* exit */
    return nIdCreate;
}

```

8.1.3.3 generate_nd_4_tuple

```

Idntuple InGenerateND4Tuple (                                /* OUT three parameter ND type tuple identifier */
    Ipuint          uValue1,                                /* first value, e.g., Band 1 precision */
    Ipuint          uValue2,                                /* second value, e.g., Band 2 precision */
    Ipuint          uValue3,                                /* third value, e.g., Band 3 precision */
    Ipuint          uValue4                                /* fourth value, e.g., Band 4 precision */
)

{
/* data declarations */
    Idntuple          nIdAllocate, nIdCreate = NULL;
    Ipsattrs_tuple    sTupleAttrs;
    Ipuint            uaData[4];
    Ipsparameter_basic stData;
    Ipsobject          snObject;

/* program */

/* initialize tuple attributes */
    sTupleAttrs.uTupleSize = 4;
    sTupleAttrs.iTypeOption = IDATA_TYPE_INTERNAL_ND;

/* allocate ND 4-tuple */
    if ((nIdAllocate = InAllocateTuple(sTupleAttrs)) != NULL) {

/* parameter array values */
        uaData[0] = uValue1;
        uaData[1] = uValue2;
        uaData[2] = uValue3;
        uaData[3] = uValue4;
        stData.tuParameter = &uaData[0];

/* establish values in tuple */
        if ((nIdCreate = InCreateTuple(stData, nIdAllocate, sTupleAttrs)) == NULL) {
            snObject.nTuple = nIdAllocate;
            IvDeallocateDataObject(snObject);
        }
    }

/* exit */
    return nIdCreate;
}

```

8.1.3.4 generate_nd_5_tuple

```

Idntuple InGenerateND5Tuple (                                /* OUT five parameter ND type tuple identifier */
    Ipoint            uValue1,                                /* first value, e.g., X */
    Ipoint            uValue2,                                /* second value, e.g., Y */
    Ipoint            uValue3,                                /* third value, e.g., Z */
    Ipoint            uValue4,                                /* fourth value, e.g., T */
    Ipoint            uValue5,                                /* fifth value, e.g., B */
)

{
/* data declarations */
    Idntuple          nIdAllocate, nIdCreate = NULL;
    Ipsattrs_tuple    sTupleAttrs;
    Ipoint            uaData[5];
    Ipsparameter_basic stData;
    Ipsobject          snObject;

/* program */

/* initialize tuple attributes */
    sTupleAttrs.uTupleSize = 5;
    sTupleAttrs.iTypeOption = IDATA_TYPE_INTERNAL_ND;

/* allocate ND 5-tuple */
    if ((nIdAllocate = InAllocateTuple(sTupleAttrs)) != NULL) {

/* parameter array values */
        uaData[0] = uValue1;
        uaData[1] = uValue2;
        uaData[2] = uValue3;
        uaData[3] = uValue4;
        uaData[4] = uValue5;
        stData.tuParameter = &uaData[0];

/* establish values in tuple */
        if ((nIdCreate = InCreateTuple(stData, nIdAllocate, sTupleAttrs)) == NULL) {
            snObject.nTuple = nIdAllocate;
            IvDeallocateDataObject(snObject);
        }
    }

/* exit */
    return nIdCreate;
}

```

8.1.3.5 generate_rd_3_tuple

```

Idntuple InGenerateRD3Tuple (                                /* OUT three parameter RD type tuple identifier */
    Ipfloat          rValue1,                                /* first value, e.g., X offset */
    Ipfloat          rValue2,                                /* second value, e.g., Y offset */
    Ipfloat          rValue3                                /* third value, e.g., Z offset */
)

{
    /* data declarations */
    Idntuple          nIdAllocate, nIdCreate = NULL;
    Ipsattrs_tuple    sTupleAttrs;
    Ipfloat           raData[3];
    Ipsparameter_basic stData;
    Ipsobject          snObject;

    /* program */

    /* initialize tuple attributes */
    sTupleAttrs.uTupleSize = 3;
    sTupleAttrs.iTypeOption = IDATA_TYPE_INTERNAL_RD;

    /* allocate RD 3-tuple */
    if ((nIdAllocate = InAllocateTuple(sTupleAttrs)) != NULL) {

        /* parameter array values */
        raData[0] = rValue1;
        raData[1] = rValue2;
        raData[2] = rValue3;
        stData.trParameterFloat = &raData[0];

        /* establish values in tuple */
        if ((nIdCreate = InCreateTuple(stData, nIdAllocate, sTupleAttrs)) == NULL) {
            snObject.nTuple = nIdAllocate;
            IvDeallocateDataObject(snObject);
        }
    }

    /* exit */
    return nIdCreate;
}

```

8.1.3.6 generate_rd_4_tuple

```

Idntuple InGenerateRD4Tuple (                                /* OUT four parameter RD type tuple identifier */
    Ipfloat          rValue1,                                /* first value, e.g., X offset */
    Ipfloat          rValue2,                                /* second value, e.g., Y offset */
    Ipfloat          rValue3,                                /* third value, e.g., Z offset */
    Ipfloat          rValue4,                                /* fourth value, e.g., T offset */
)

{
/* data declarations */
    Idntuple          nIdAllocate, nIdCreate = NULL;
    Ipsattrs_tuple    sTupleAttrs;
    Ipfloat           raData[4];
    Ipsparameter_basic stData;
    Ipsobject          snObject;

/* program */

/* initialize tuple attributes */
    sTupleAttrs.uTupleSize = 4;
    sTupleAttrs.iTypeOption = IDATA_TYPE_INTERNAL_RD;

/* allocate RD 4-tuple */
    if ((nIdAllocate = InAllocateTuple(sTupleAttrs)) != NULL) {

/* parameter array values */
        raData[0] = rValue1;
        raData[1] = rValue2;
        raData[2] = rValue3;
        raData[3] = rValue4;
        stData.trParameterFloat = &raData[0];

/* establish values in tuple */
        if ((nIdCreate = InCreateTuple(stData, nIdAllocate, sTupleAttrs)) == NULL) {
            snObject.nTuple = nIdAllocate;
            IvDeallocateDataObject(snObject);
        }
    }

/* exit */
    return nIdCreate;
}

```


8.1.3.7 generate_rd_5_tuple

```

Idntuple InGenerateRD5Tuple (                                /* OUT five parameter RD type tuple identifier */
    Ipfloat          rValue1,                                /* first value, e.g., X offset */
    Ipfloat          rValue2,                                /* second value, e.g., Y offset */
    Ipfloat          rValue3,                                /* third value, e.g., Z offset */
    Ipfloat          rValue4,                                /* fourth value, e.g., T offset */
    Ipfloat          rValue5,                                /* fifth value, e.g., B offset */
)

{
    /* data declarations */
    Idntuple          nIdAllocate, nIdCreate = NULL;
    Ipsattrs_tuple    sTupleAttrs;
    Ipfloat           raData[5];                                /* tuple data */
    Ipsparameter_basic stData;
    Ipsobject          snObject;

    /* program */

    /* initialize tuple attributes */
    sTupleAttrs.uTupleSize = 5;
    sTupleAttrs.iTypeOption = IDATA_TYPE_INTERNAL_RD;

    /* allocate RD 5-tuple */
    if ((nIdAllocate = InAllocateTuple(sTupleAttrs)) != NULL) {

        /* parameter array values */
        raData[0] = rValue1;
        raData[1] = rValue2;
        raData[2] = rValue3;
        raData[3] = rValue4;
        raData[4] = rValue5;
        stData.trParameterFloat = &raData[0];

        /* establish values in tuple */
        if ((nIdCreate = InCreateTuple(stData, nIdAllocate, sTupleAttrs)) == NULL) {
            snObject.nTuple = nIdAllocate;
            IvDeallocateDataObject(snObject);
        }
    }

    /* exit */
    return nIdCreate;
}

```

8.1.3.8 generate_sd_1_tuple

```

Idntuple InGenerateSD1Tuple (                                /* OUT one parameter SD type tuple identifier */
    Ipint                iValue1                             /* single value, e.g., X */
)

{
    /* data declarations */
    Idntuple            nIdAllocate, nIdCreate = NULL;
    Ipsattrs_tuple      sTupleAttrs;
    Ipint               iData;
    Ipsparameter_basic  stData;
    Ipsobject            snObject;

    /* program */

    /* initialize tuple attributes */
    sTupleAttrs.uTupleSize = 1;
    sTupleAttrs.iTypeOption = IDATA_TYPE_INTERNAL_SD;

    /* allocate SD 1-tuple */
    if ((nIdAllocate = InAllocateTuple(sTupleAttrs)) != NULL) {

        /* parameter array values */
        iData = iValue1;
        stData.tiParameter = &iData;

        /* establish value in tuple */
        if ((nIdCreate = InCreateTuple(stData, nIdAllocate, sTupleAttrs)) == NULL) {
            snObject.nTuple = nIdAllocate;
            IvDeallocateDataObject(snObject);
        }
    }

    /* exit */
    return nIdCreate;
}

```

8.1.3.9 generate_sd_3_tuple

```

Idntuple InGenerateSD3Tuple (                                /* OUT three parameter SD type tuple identifier */
    Ipoint            iValue1,                                /* first value, e.g., X */
    Ipoint            iValue2,                                /* second value, e.g., Y */
    Ipoint            iValue3                                /* third value, e.g., Z */
)

{
    /* data declarations */
    Idntuple          nIdAllocate, nIdCreate = NULL;
    Ipsattrs_tuple    sTupleAttrs;
    Ipoint            iaData[3];
    Ipsparameter_basic stData;
    Ipsobject          snObject;

    /* program */

    /* initialize tuple attributes */
    sTupleAttrs.uTupleSize = 3;
    sTupleAttrs.iTypeOption = IDATA_TYPE_INTERNAL_SD;

    /* allocate SD 3-tuple */
    if ((nIdAllocate = InAllocateTuple(sTupleAttrs)) != NULL) {

        /* parameter array values */
        iaData[0] = iValue1;
        iaData[1] = iValue2;
        iaData[2] = iValue3;
        stData.tiParameter = &iaData[0];

        /* establish values in tuple */
        if ((nIdCreate = InCreateTuple(stData, nIdAllocate, sTupleAttrs)) == NULL) {
            snObject.nTuple = nIdAllocate;
            IvDeallocateDataObject(snObject);
        }
    }

    /* exit */
    return nIdCreate;
}

```

8.1.3.10 generate_sd_4_tuple

```

Idntuple InGenerateSD4Tuple (                                /* OUT four parameter SD type tuple identifier */
    Ipoint            iValue1,                                /* first value, e.g., Band Type 1 */
    Ipoint            iValue2,                                /* second value, e.g., Band Type 2 */
    Ipoint            iValue3,                                /* third value, e.g., Band Type 3 */
    Ipoint            iValue4                                /* fourth value, e.g., Band Type 4 */
)

{
/* data declarations */
    Idntuple          nIdAllocate, nIdCreate = NULL;
    Ipsattrs_tuple    sTupleAttrs;
    Ipoint            iaData[4];
    Ipsparameter_basic stData;
    Ipsobject          snObject;

/* program */

/* initialize tuple attributes */
    sTupleAttrs.uTupleSize = 4;
    sTupleAttrs.iTypeOption = IDATA_TYPE_INTERNAL_SD;

/* allocate SD 4-tuple */
    if ((nIdAllocate = InAllocateTuple(sTupleAttrs)) != NULL) {

/* parameter array values */
        iaData[0] = iValue1;
        iaData[1] = iValue2;
        iaData[2] = iValue3;
        iaData[3] = iValue4;
        stData.tiParameter = &iaData[0];

/* establish values in tuple */
        if ((nIdCreate = InCreateTuple(stData, nIdAllocate, sTupleAttrs)) == NULL) {
            snObject.nTuple = nIdAllocate;
            IvDeallocateDataObject(snObject);
        }
    }

/* exit */
    return nIdCreate;
}

```

8.1.3.11 generate_sd_5_tuple

```

Idntuple InGenerateSD5Tuple (                                /* OUT five parameter SD type tuple identifier */
    Ipoint            iValue1,                                /* first value, e.g., X */
    Ipoint            iValue2,                                /* second value, e.g., Y */
    Ipoint            iValue3,                                /* third value, e.g., Z */
    Ipoint            iValue4,                                /* fourth value, e.g., T */
    Ipoint            iValue5                                /* fifth value, e.g., B */
)

{
    /* data declarations */
    Idntuple          nIdAllocate, nIdCreate = NULL;
    Ipsattrs_tuple    sTupleAttrs;
    Ipoint            iaData[5];
    Ipsparameter_basic stData;
    Ipsobject          snObject;

    /* program */

    /* initialize tuple attributes */
    sTupleAttrs.uTupleSize = 5;
    sTupleAttrs.iTypeOption = IDATA_TYPE_INTERNAL_SD;

    /* allocate SD 5-tuple */
    if ((nIdAllocate = InAllocateTuple(sTupleAttrs)) != NULL) {

        /* parameter array values */
        iaData[0] = iValue1;
        iaData[1] = iValue2;
        iaData[2] = iValue3;
        iaData[3] = iValue4;
        iaData[4] = iValue5;
        stData.tiParameter = &iaData[0];

        /* establish values in tuple */
        if ((nIdCreate = InCreateTuple(stData, nIdAllocate, sTupleAttrs)) == NULL) {
            snObject.nTuple = nIdAllocate;
            IvDeallocateDataObject(snObject);
        }
    }

    /* exit */
    return nIdCreate;
}

```

8.1.4 Tiled image import and export

Memory management has been introduced into this part of ISO/IEC 12088 to buffer image data in strip form. These two functions are provided to support the tiled image form:

- *tiled_image_export* will return the destination image pointer, or null if error occurs, given the source image identifier, the destination image pointer, a tile-sized work image identifier, the ND 5-tuples containing the tile size and tile offset within the source image, the external pixel precision and the external pixel data type as input arguments.
- *tiled_image_import* will return the destination image identifier, or null if error occurs, given the source image pointer, the destination image identifier, a tile-sized work image identifier, the ND t-tuples containing the tile size and tile offset within the destination image, the external pixel precision and the external pixel data type as input arguments.

These two functions are applicable within all PIKS profiles.

While the tile-sized work image could have been allocated and deallocated internal to these functions, it is left to the application to allocate and deallocate this image for import/export performance. Allocation should precede the first tile transfer and deallocation should follow the last tile transfer.

The functions require include of applicable system and PIKS header files, the latter sufficient to represent the PIKS foundation profile, as presented in Annex B.

8.1.4.1 tiled_image_export

```

Ipspiks_pixel_types IstTiledImageExport (      /* OUT destination image data pointer */
    Idnimage          nSourceImage,           /* source image identifier */
    Ipspiks_pixel_types stDestImage,          /* tile destination image data pointer */
    Idnimage          nTileImage,             /* tile-sized image identifier */
    Idntuple          nTileSize,              /* ND 5-tuple tile size */
    Idntuple          nSourceOffset,          /* ND 5-tuple offset of tile in source image */
    Ipuint            uPrecision,             /* number of destination bits per pixel */
    Ipint             iExternalType           /* data type (external pixel) option */
)
{
    /* program */

    /* acquire sub image */
    if (InDefineSubImage(nSourceImage, nTileImage, nSourceOffset, nTileSize)) {

        /* export tile */
        stDestImage.tuPixel = IstExportImage(nTileImage, stDestImage, uPrecision,
                                              iExternalType, -1, 0, NULL).tuPixel;
    }

    /* exit */
    return stDestImage;
}

```

8.1.4.2 tiled_image_import

```

Idnimage InTiledImageImport (                                /* OUT destination image identifier */
    Ipspiks_pixel_types stSourceImage,                       /* tile source image data pointer */
    Idnimage nDestImage,                                     /* destination image identifier */
    Idnimage nTileImage,                                    /* tile-sized image identifier */
    Idntuple nTileSize,                                     /* ND 5-tuple tile size */
    Idntuple nDestOffset,                                   /* ND 5-tuple offset of tile in destination image */
    Ipoint uPrecision,                                     /* number of source bits per pixel */
    Ipoint iExternalType,                                  /* data type (external pixel) option */
)

{
/* program */

/* initialize sub image */
    if (InDefineSubImage(nDestImage, nTileImage, nDestOffset, nTileSize)) {

/* import tile */
        if (!(InImportImage(stSourceImage, nTileImage, uPrecision,
                            iExternalType, -1, 0, NULL)))
            nDestImage = (Idnimage)NULL;
    }

/* exit */
    return nDestImage;
}

```

8.2 IPI-IIF convenience functions

Convenience functions are provided for shortening or simplifying frequent IPI-IIF operations in common applications.

8.2.1 Attach entity functions

Three attach entity convenience functions are provided:

- *attach* will return the entity identifier to which components that are not SEQUENCE OF types have been attached, given the entity identifier, the subordinate entity identifier, and the component number as input arguments.
- *attach_sequence* will return the entity identifier to which components that are SEQUENCE OF types have been attached, given the entity identifier, the subordinate entity identifier and the component index as input arguments.
- *attach_sequence_end* will return the entity identifier to which a subordinate entity has been attached to the end of a SEQUENCE OF component, given the entity identifier, the subordinate entity identifier and the component number as input arguments.

These functions are specializations of the *attach_iif_syntax_entity* function requiring fewer input arguments.

The functions require include of applicable system and IIF header files, the latter sufficient to represent the IIF foundation profile, as presented in Annex B.

8.2.1.1 *attach*

```

Ipenentity InAttach (                                /* OUT IIF syntax entity identifier */
    Ipenentity          nEntity,                      /* IIF syntax entity identifier */
    Ipenentity          nChild,                       /* subentity identifier */
    Ipoint              iNumber                       /* component number */
)
{
/* program */
    return InAttachEntity(nEntity, nChild, iNumber, 0);
}

```

8.2.1.2 *attach_sequence*

```

Ipenentity InAttachSeq (                             /* OUT IIF syntax entity identifier */
    Ipenentity          nEntity,                      /* IIF syntax entity identifier */
    Ipenentity          nChild,                       /* subentity identifier */
    Ipoint              uIndex                       /* component index */
)
{
/* program */
    return InAttachEntity(nEntity, nChild, 0, uIndex);
}

```


8.2.1.3 attach_sequence_end

```

Ipntity InAttachSeqEnd (                                /* OUT identifier */
    Ipntity      nEntity,                                /* IIF syntax entity identifier */
    Ipntity      nChild,                                /* subentity identifier */
    Iuint        iNumber                                /* component number */
)
{
/* data declarations */
    Ipevalidity   eValid;
    Iuint         uLength;

/* program */
    IvInquireEntitySeq(nEntity, iNumber, &uLength, &eValid);
    if (eValid == IVALIDITY_VALID && uLength > 0)
        return InAttachEntity(nEntity, nChild, iNumber, uLength);
    else
        return (Ipntity)NULL;
}

```

8.2.2 Create identifier function

A create identifier convenience function is provided:

- *create_identifier* returns an IIF syntax entity identifier whose value is specified by the character string parameter given as pointer input argument.

The function requires include of applicable system and IIF header files, the latter sufficient to represent the IIF foundation profile, as presented in Annex B.

```

Ipntity InCreateId(                                /* OUT new identifier */
    char      *tzString                                /* label of identifier */
)
{
/* data declaration */
    Ipntity      nId;

/* program */
    nId = InCreateEntity(ISE_ID);

    return InPutEntityString(nId, tzString, 0);
}

```

8.2.3 Get entity component functions

Two get entity component convenience functions are provided:

- *get_component* will return the entity identifier in which the specified component that is not SEQUENCE OF type has been located, given entity identifier and the component number as input arguments.
- *get_sequence_component* will return the entity identifier in which the specified component that is SEQUENCE OF type has been located, given the entity identifier and the component index as input arguments.

These functions are specializations of the *get_iif_syntax_entity_component* function requiring fewer input arguments.

The functions require include of applicable system and IIF header files, the latter sufficient to represent the IIF foundation profile, as presented in Annex B.

8.2.3.1 get_component

```

Ipntity InGetCmp (                                /* OUT IIF syntax entity identifier */
    Ipntity      nEntity,                          /* IIF syntax entity identifier */
    Ipoint       iNumber                           /* component number */
)
{
/* program */
    return InGetEntityCmp(nEntity, iNumber, 0);
}

```

8.2.3.2 get_sequence_component

```

Ipntity InGetSeqCmp (                            /* OUT IIF syntax entity identifier */
    Ipntity      nEntity,                          /* IIF syntax entity identifier */
    Ipoint       uIndex                           /* component index */
)
{
/* program */
    return InGetEntityCmp(nEntity, 0, uIndex);
}

```

8.2.4 Get entity value functions

Nine get entity value convenience functions are provided:

- *get_entity_boolean* will return the elementary Boolean entity or component value that is not SEQUENCE OF type, given the entity identifier and component number as input arguments.
- *get_entity_integer* will return the elementary integer entity or component value that is not SEQUENCE OF type, given the entity identifier and component number as input arguments.
- *get_entity_real* will return the elementary real entity or component value that is not SEQUENCE OF type, given the entity identifier and component number as input arguments.
- *get_entity_string* will return the elementary string entity or component that is not SEQUENCE OF type, given the entity identifier, and component number as input arguments.
- *get_sequence_boolean* will return the elementary Boolean entity or component value that is SEQUENCE OF type, given the entity identifier, component number and component index as input arguments.
- *get_sequence_integer* will return the elementary integer entity or component value that is SEQUENCE OF type, given the entity identifier, component number and component index as input arguments.
- *get_sequence_real* will return the elementary real entity or component value that is SEQUENCE OF type, given the entity identifier, component number and component index as input arguments.
- *get_sequence_string* will return the elementary string entity or component that is SEQUENCE OF type, given the entity identifier, component number and component index as input arguments.
- *get_sequence_end* will return the end subentity identifier, given the entity identifier and component number as input arguments.

These functions are specializations of the *get_iif_syntax_entity_value* function requiring limiting the input arguments and return value types.

The functions require include of applicable system and IIF header files, the latter sufficient to represent the IIF foundation profile, as presented in Annex B.

8.2.4.1 *get_entity_boolean*

```

Ipbbool IbGetEntityBool (                                /* OUT Boolean value */
    Ipntity          nEntity,                             /* IIF syntax entity identifier */
    Ipint            iNumber,                             /* component number */
)
{
/* program */
    return *IstGetEntityValue(nEntity, iNumber, 0, -1, 0, NULL).tbParameter;
}

```

8.2.4.2 get_entity_integer

```

Ipoint IiGetEntityInteger (                                /* OUT integer value */
    Ipointentity      nEntity,                             /* IIF syntax entity identifier */
    Ipoint            iNumber                              /* component number */
)
{
/* program */
    return *IstGetEntityValue(nEntity, iNumber, 0, -1, 0, NULL).tiParameter;
}

```

8.2.4.3 get_entity_real

```

Ipointdouble IrGetEntityReal (                            /* OUT real value */
    Ipointentity      nEntity,                             /* IIF syntax entity identifier */
    Ipoint            iNumber                              /* component number */
)
{
/* program */
    return *IstGetEntityValue(nEntity, iNumber, 0, -1, 0, NULL).trParameterDouble;
}

```

8.2.4.4 get_entity_string

```

char *ItzGetEntityString (                                /* OUT string pointer */
    Ipointentity      nEntity,                             /* IIF syntax entity identifier */
    Ipoint            iNumber                              /* component number */
)
{
/* program */
    return IstGetEntityValue(nEntity, iNumber, 0, -1, 0, NULL).tzParameter;
}

```

8.2.4.5 get_sequence_boolean

```

Ipointbool IbGetSeqBool (                                /* OUT Boolean value */
    Ipointentity      nEntity,                             /* IIF syntax entity identifier */
    Ipoint            iNumber,                             /* component number */
    Ipoint            uIndex                              /* component index */
)
{
/* program */
    return *IstGetEntityValue(nEntity, iNumber, uIndex, -1, 0, NULL).tbParameter;
}

```

8.2.4.6 get_sequence_integer

```
Ipint liGetSeqInteger (                                /* OUT integer value */
    Ipntity      nEntity,                             /* IIF syntax entity identifier */
    Ipint        iNumber,                             /* component number */
    Ipuint       uIndex                              /* component index */
)
{
/* program */
    return *IstGetEntityValue(nEntity, iNumber, uIndex, -1, 0, NULL).tiParameter;
}
```

8.2.4.7 get_sequence_real

```
Ipdouble IrGetSeqReal (                                /* OUT real value */
    Ipntity      nEntity,                             /* IIF syntax entity identifier */
    Ipint        iNumber,                             /* component number */
    Ipuint       uIndex                              /* component index */
)
{
/* program */
    return *IstGetEntityValue(nEntity, iNumber, uIndex, -1, 0, NULL).trParameterDouble;
}
```

8.2.4.8 get_sequence_string

```
char *ItzGetSeqString (                                /* OUT string pointer */
    Ipntity      nEntity,                             /* IIF syntax entity identifier */
    Ipint        iNumber,                             /* component number */
    Ipuint       uIndex                              /* component index */
)
{
/* program */
    return IstGetEntityValue(nEntity, iNumber, uIndex, -1, 0, NULL).tzParameter;
}
```

8.2.4.9 get_sequence_end

```
Ipntity InGetSeqEnd (                                /* OUT IIF syntax subentity identifier */
    Ipntity      nEntity,                             /* IIF syntax entity identifier */
    Iuint        iNumber                             /* component number */
)
{
    /* data declarations */
    Ipevalidity   eValid;
    Iuint         uLength;

    /* program */
    IvInquireEntitySeq(nEntity, iNumber, &uLength, &eValid);
    if (eValid == IVALIDITY_VALID && uLength > 0)
        return InGetEntityCmp(nEntity, iNumber, (uLength - 1));
    else
        return (Ipntity) NULL;
}
```

8.2.5 Put entity value functions

Eight put entity value convenience functions are provided:

- *put_entity_boolean* returns the entity identifier and sets the Boolean value of an elementary component, given the entity identifier, the Boolean value and the component number as input arguments.
- *put_entity_integer* returns the entity identifier and sets the integer value of an elementary component, given the entity identifier, the integer value and the component number as input arguments.
- *put_entity_real* returns the entity identifier and sets the real value of an elementary component, given the entity identifier, the real value and the component number as input arguments.
- *put_entity_string* returns the entity identifier and sets the string value of an elementary IA5String component, given the entity identifier, the string pointer and the component number as input arguments.
- *put_sequence_boolean* returns the entity identifier and sets the Boolean value of an elementary SEQUENCE OF BOOLEAN component, given the entity identifier, the Boolean value, the component number and the component index as input arguments.
- *put_sequence_integer* returns the entity identifier and sets the integer value of an elementary SEQUENCE OF INTEGER component, given the entity identifier, the integer value, the component number and the component index as input arguments.
- *put_sequence_real* returns the entity identifier and sets the real value of an elementary SEQUENCE OF REAL component, given the entity identifier, the real value, the component number and the component index as input arguments.
- *put_sequence_string* returns the entity identifier and sets the string of an elementary SEQUENCE OF IA5String component, given the entity identifier, the string pointer, the component number and the component index as input arguments.

These functions are specializations of the *put_iif_syntax_entity_value* function requiring fewer input arguments.

The functions require include of applicable system and IIF header files, the latter sufficient to represent the IIF foundation profile, as presented in Annex B.

8.2.5.1 put_entity_boolean

```

Ipntity InPutEntityBool (                                /* OUT IIF syntax entity identifier */
    Ipntity      nEntity,                                /* IIF syntax entity identifier */
    Ipbool       bValue,                                /* Boolean value */
    Iputint      uComponent                              /* component number */
)
{
/* data declaration */
    Ipsentity_value    stVal;

/* program */
    stVal.tbParameter = &bValue;
    return InPutEntityValue(nEntity, stVal, uComponent, 0, -1, 0, NULL);
}

```

8.2.5.2 put_entity_integer

```

Ipntity InPutEntityInteger (                            /* OUT IIF syntax entity identifier */
    Ipntity      nEntity,                                /* IIF syntax entity identifier */
    Iputint      iValue,                                /* integer value */
    Iputint      uComponent                              /* component number */
)
{
/* data declaration */
    Ipsentity_value    stVal;

/* program */
    stVal.tiParameter = &iValue;
    return InPutEntityValue(nEntity, stVal, uComponent, 0, -1, 0, NULL);
}

```

8.2.5.3 put_entity_real

```

Ipntity InPutEntityReal (                                /* OUT IIF syntax entity identifier */
    Ipntity      nEntity,                                /* IIF syntax entity identifier */
    Ipdouble     rValue,                                /* real value */
    Iputint      uComponent                              /* component number */
)
{
/* data declaration */
    Ipsentity_value    stVal;

/* program */
    stVal.trParameterDouble = &rValue;
    return InPutEntityValue(nEntity, stVal, uComponent, 0, -1, 0, NULL);
}

```


8.2.5.4 put_entity_string

```

Ipnentity InPutEntityString (                                /* OUT IIF syntax entity identifier */
    Ipnentity          nEntity,                               /* IIF syntax entity identifier */
    char               *tzString,                             /* string pointer */
    Ipoint             uComponent                             /* component number */
)
{
/* data declaration */
    Ipsentity_value    stVal;

/* program */
    stVal.tzParameter = tzString;
    return InPutEntityValue(nEntity, stVal, uComponent, 0, -1, 0, NULL);
}

```

8.2.5.5 put_sequence_boolean

```

Ipnentity InPutSeqBool (                                    /* OUT IIF syntax entity identifier */
    Ipnentity          nEntity,                               /* IIF syntax entity identifier */
    Ipbbool            bValue,                                /* Boolean value */
    Ipoint             uComponent,                             /* component number */
    Ipoint             uIndex                                 /* component index in sequence */
)
{
/* data declaration */
    Ipsentity_value    stVal;

/* program */
    stVal.tbParameter = &bValue;
    return InPutEntityValue(nEntity, stVal, uComponent, uIndex, -1, 0, NULL);
}

```

8.2.5.6 put_sequence_integer

```

Ipnentity InPutSeqInteger (                                /* OUT IIF syntax entity identifier */
    Ipnentity          nEntity,                               /* IIF syntax entity identifier */
    Ipoint             iValue,                                /* integer value */
    Ipoint             uComponent,                             /* component number */
    Ipoint             uIndex                                 /* component index in sequence */
)
{
/* data declaration */
    Ipsentity_value    stVal;

/* program */
    stVal.tiParameter = &iValue;
    return InPutEntityValue(nEntity, stVal, uComponent, uIndex, -1, 0, NULL);
}

```

8.2.5.7 put_sequence_real

```

Ipntity InPutSeqReal (                                /* OUT IIF syntax entity identifier */
    Ipntity      nEntity,                             /* IIF syntax entity identifier */
    Ipdouble     rValue,                              /* real value */
    Ipuint       uComponent,                          /* component number */
    Ipuint       uIndex,                             /* component index in sequence */
)
{
/* data declaration */
    Ipsentity_value stVal;

/* program */
    stVal.trParameterDouble = &rValue;
    return InPutEntityValue(nEntity, stVal, uComponent, uIndex, -1, 0, NULL);
}

```

8.2.5.8 put_sequence_string

```

Ipntity InPutSeqString (                             /* OUT IIF syntax entity identifier */
    Ipntity      nEntity,                             /* IIF syntax entity identifier */
    char         *tzString,                          /* string pointer */
    Ipuint       uComponent,                          /* component number */
    Ipuint       uIndex,                             /* component index in sequence */
)
{
/* data declaration */
    Ipsentity_value stVal;

/* program */
    stVal.tzParameter = tzString;
    return InPutEntityValue(nEntity, stVal, uComponent, uIndex, -1, 0, NULL);
}

```

Annex A **(informative)** **Memory management**

This annex describes the memory management provided for and accessed by several functions bound to the C language in this part of ISO/IEC 12088.

A.1 Introduction

The amount of data contained in some of the objects specified ISO/IEC 12087 might exceed the size of local memory which the application can allocate for transferring the object data between the IPI implementation and storage or communication port in a single operation. Furthermore, some functions exist in IPI which return data whose length is not known a priori by the application. Memory management is specified to assist in these conditions. The memory management policies for these functions are introduced in 3.7. This annex discusses sample code and presents diagrams to help illustrate the memory management mechanism.

The transfer of data imported to or exported from the IPI implementation is described by three parameters: `iApplBufferSize`, `uStartPoint` and `uDataSize` represented in the appropriate input/output function arguments. The `iApplBufferSize` and `uStartPoint` argument values are specified by the application to indicate the transfer size and offset of the data within the implementation domain, respectively. The `uDataSize` parameter, which represents the amount of total or post transfer data available in the object, is a value returned from the implementation to the application through the pointer argument `tuDataSize`.

The data transferred between application and implementation is described as a one-dimensional array which represent a multi-dimensional object. The multi-dimensioning can extend to five dimensions for IPI objects such as images which are described with horizontal, vertical, depth, temporal and spectral (band) dimensions with sizes expressed as X, Y, Z, T and B, respectively. Additionally, images may be represented with planar or interleaved pixel formats, that is, with the band dimension as the fifth or first dimension, respectively. In the representation of five-dimensional images the sequence of one-dimensional array elements are as follows:

Planar Representation

Data(0, 0, 0, 0, 0),
Data(1, 0, 0, 0, 0),
...,
Data(X, 0, 0, 0, 0),
...,
Data(0, 1, 0, 0, 0),
...,
Data(X, Y, 0, 0, 0),
...,
Data(0, 0, 1, 0, 0),
...,
Data(X, Y, Z, 0, 0),
...,
Data(0, 0, 0, 1, 0),
...,
Data(X, Y, Z, T, 0),
...,
Data(0, 0, 0, 0, 1),
...,
Data(X, Y, Z, T, B).

Interleaved Representation

Data(0, 0, 0, 0, 0),
Data(1, 0, 0, 0, 0),
...,
Data(B, 0, 0, 0, 0),
...,
Data(0, 1, 0, 0, 0),
...,
Data(B, X, 0, 0, 0),
...,
Data(0, 0, 1, 0, 0),
...,
Data(B, X, Y, 0, 0),
...,
Data(0, 0, 0, 1, 0),
...,
Data(B, X, Y, Z, 0),
...,
Data(0, 0, 0, 0, 1),
...,
Data(B, X, Y, Z, T).

Memory management is applied in two cases. The first is for functions which import the application data to the implementation internal memory; this case is discussed in clause A.2. The other case is for functions which export the implementation's internal data to the application; this case is discussed in clause A.3.

A.2 Functions that import the application data to the implementation memory

The application can not directly access implementation internal memory. To import application data to the implementation's internal memory, IPI provides some functions (e.g., *import_image*). The auxiliary parameters *iApplBufferSize*, *uStartPoint* and *tuDataSize* are provided for these functions. These parameters allow the application to transfer the contents of the application data in buffered amounts smaller than the total data required by the object.

Annex C (see C.1.1) presents a sample program which illustrates the use of these parameters to transfer buffered portions of the application data. This mechanism imports a portion of the image which is taken from the input source, e.g., storage or port, is temporarily stored in the application buffer and is transferred to the implementation image object by repeated execution of the *InImportImage* function.

The application knows the data size of the application image buffer (*IMAGE_BUFFER_SIZE* in the sample program) and the amount of image data expected by the object in terms of the sizes of the image dimensions and the pixel data type ($X * Y * Z * T * B * \text{sizeof}(\text{pixel_data_type})$). To import the image data, the following steps are repeated until the *uDataSize* value is returned as zero:

- 1) The application copies a buffer-sized sequential portion of the image data from the external source, e.g., storage or port, into the application buffer.
- 2) The *InImportImage* function is called by the application with the memory management parameters *iApplBufferSize* set to the buffer size value and *uStartPoint* incremented from zero by the buffer size value as shown in Figure 1.
- 3) The implementation copies corresponding image data from the application buffer in an amount equal to or less than specified by *iApplBufferSize*, as limited by the image dimensions, to a portion of the implementation's prepared image object beginning at the index offset *uStartPoint*.

The operation of *InImportImage* is therefore dependent upon the three variables; *iApplBufferSize*, the size of application buffer, *uStartPoint*, the index offset within the implementation image object, and *uDataSize*, the post transfer data size which remains to be transferred into the implementation image object. The *iApplBufferSize* can remain constant regardless that the last portion transferred may be less than this amount. The *uStartPoint* is incremented by the *iApplBufferSize* value after each portion of the image is transferred to the application buffer. The process of buffering the image data continues until the *uDataSize* returned from the export is zero.

There are three cases of *iApplBufferSize* value interpretation:

a) *iApplBufferSize* = 0

The application receives the image size, in pixels, into *uDataSize* through the pointer *tuDataSize* to confirm the adequacy of the implementation buffer to receive the image data. No image data transfer takes place under this condition.

b) *iApplBufferSize* < 0

The implementation expects to receive the full amount of object data in this *InImportImage* execution, regardless of size. The *uDataSize* parameter will be zero and its pointer may be unspecified, i.e., given as NULL argument.

b) *iApplBufferSize* > 0

Beginning at the internal index offset indicated by the *uStartPoint* parameter, the implementation copies the *iApplBufferSize*-specified amount of data in the application buffer to the implementation image object. If the amount of implementation image data remaining beyond the offset is less than the *iApplBufferSize* value, only the remaining amount of data is transferred. Following each data transfer, the implementation sets the post transfer data size value returned through the *tuDataSize* pointer to the application, as shown in Figure 1.

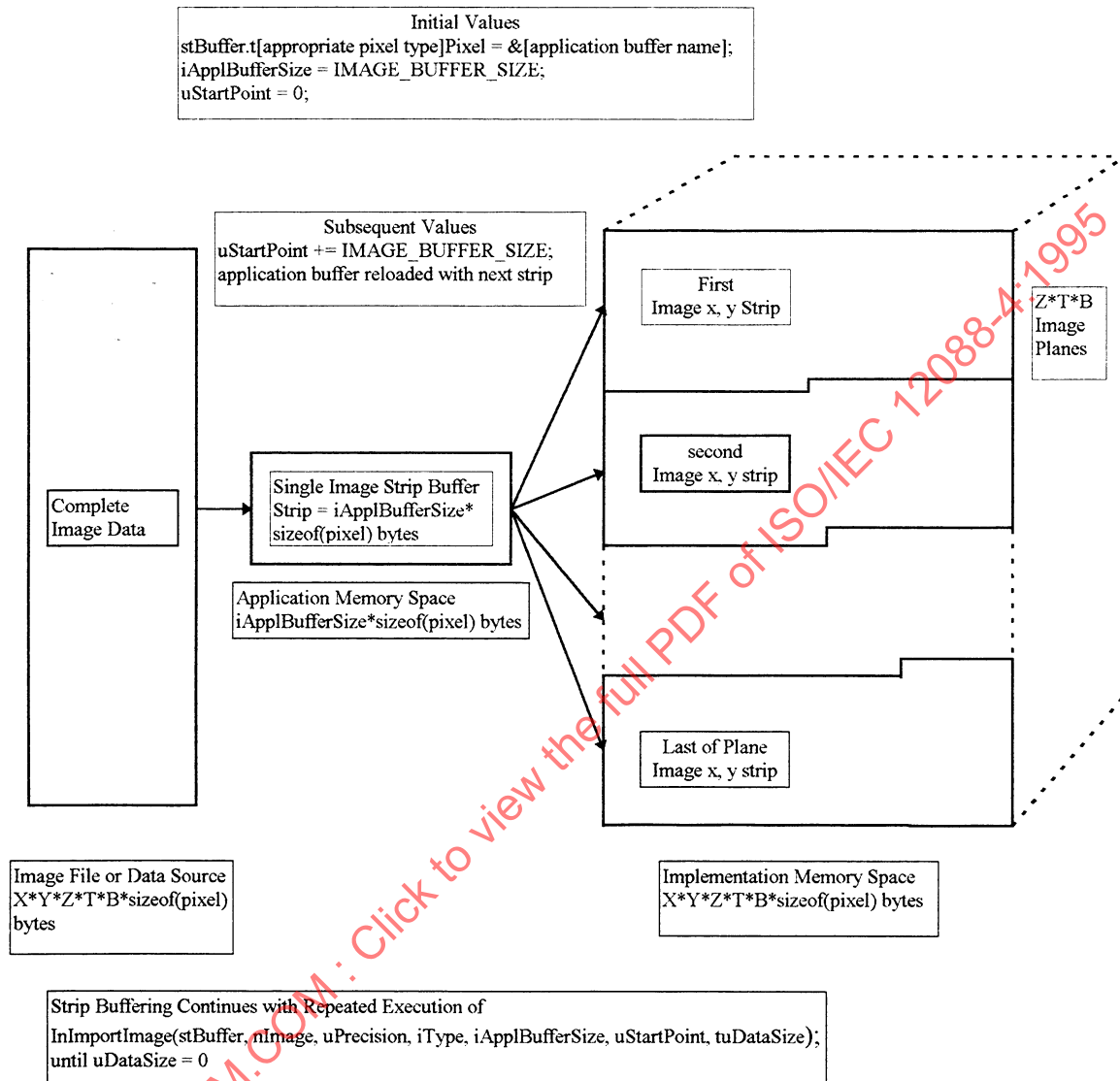


Figure 1 - Buffered memory management of `InImportImage(...)`

A.3 Functions that export the implementation data to the application memory

The application can not directly access implementation internal memory. To export application data from the implementation's internal memory, IPI provides some functions (e.g., *export_image*). The auxiliary parameters *iApplBufferSize*, *uStartPoint* and *tuDataSize* are provided for these functions. These parameters allow the application to transfer the contents of the application data in buffered amounts smaller than the total data available from the object.

Annex C (see C.1.1) presents a sample program which illustrates the use of these parameters to transfer buffered portions of the application data. This mechanism exports a portion of the image which is taken from the implementation image object by repeated execution of the *InExportImage* function, is temporarily stored in the application buffer and is transferred to the output destination, e.g., storage or port.

The application knows the data size of the application image buffer (*IMAGE_BUFFER_SIZE* in the sample program) and the amount of image data available from the object in terms of the sizes of the image dimensions and the pixel data type ($X * Y * Z * T * B * \text{sizeof}(\text{pixel_data_type})$). To export the image data, the following steps are repeated until the *uDataSize* value is returned as zero:

- 1) The *IstExportImage* is called by the application with the memory management parameters *iApplBufferSize* set to the buffer size value and *uStartPoint* incremented from zero by the buffer size value as shown in Figure 2.
- 2) The implementation copies corresponding image data into the application buffer in an amount equal to or less than specified by *iApplBufferSize*, as limited by the image dimensions, from a portion of the implementation's prepared image object beginning at the index offset *uStartPoint*.
- 3) The application copies the buffer-sized sequential portion of the image data, adjusted for smaller final transfer size, from the application buffer to the external destination, e.g., storage or port.

The operation of *IstExportImage* is therefore dependent upon the three variables; *iApplBufferSize*, the size of application buffer, *uStartPoint*, the index offset within the implementation image object and *uDataSize*, the post transfer data size which remains to be transferred from the implementation image object. The *iApplBufferSize* can remain constant regardless that the last portion transferred may be less than this amount. The *uStartPoint* is incremented by the *iApplBufferSize* value after each portion of the image is transferred to the application buffer. The process of buffering the image data continues until the *uDataSize* returned from the export is zero.

There are three cases of *iApplBufferSize* value interpretation:

- a) *iApplBufferSize* = 0
The application receives the image size, in pixels, into *uDataSize* through the pointer *tuDataSize*.
No image data transfer takes place under this condition.
- b) *iApplBufferSize* < 0
The implementation expects to send the full amount of the object data in this *InExportImage* execution, regardless of size. The *uDataSize* parameter will be zero and its pointer may be unspecified, i.e., given as NULL argument.
- b) *iApplBufferSize* > 0
Beginning at the internal index offset indicated by the *uStartPoint* parameter, the implementation copies the *iApplBufferSize*-specified amount of data into the application buffer from the implementation object. If the amount of implementation image data remaining beyond the offset is less than the *iApplBufferSize* value, only the remaining amount of data is transferred. Following each data transfer, the implementation sets the post transfer data size value returned through the *tuDataSize* pointer to the application, as shown in Figure 2.

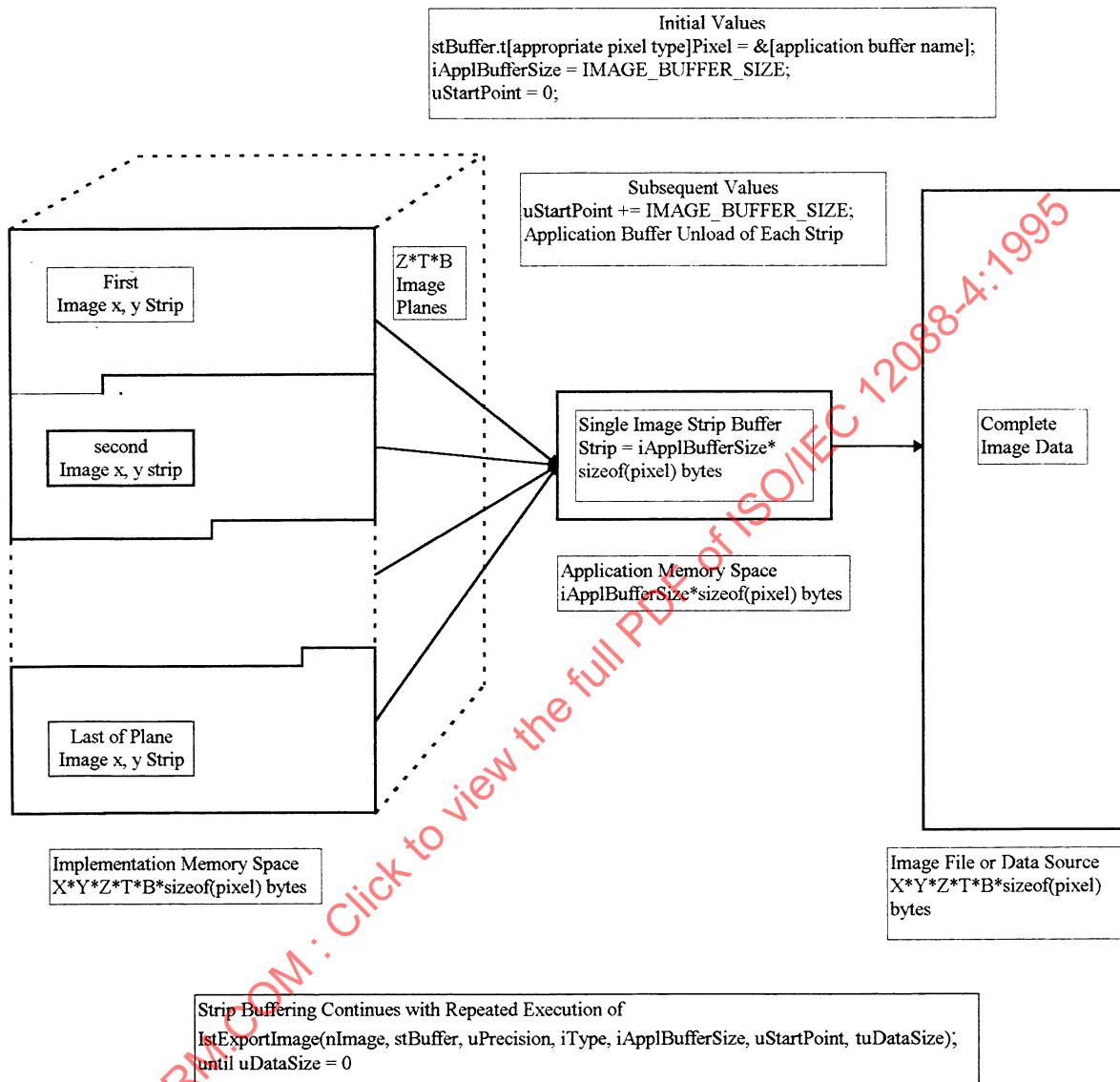


Figure 2 - Buffered memory management of InExportImage(...)

Annex B (informative)

Macros, data types in compilation order and external functions

This Annex lists macros, data types and external declarations for each function in an order suitable for compilation.

B.1 IPI-PIKS macros, types and functions

B.1.1 Macro definitions

```

/*                                OPTION MACROS                                */

/*                                bit shift option                                */
#define IBIT_SHIFT_LEFT_OVERFLOW    1 /* left overflow shift                    */
#define IBIT_SHIFT_RIGHT_OVERFLOW   2 /* right overflow shift                   */
#define IBIT_SHIFT_LEFT_BARREL     3 /* left barrel shift                     */
#define IBIT_SHIFT_RIGHT_BARREL    4 /* right barrel shift                    */
#define IBIT_SHIFT_LEFT_ARITH      5 /* left arithmetic shift                 */
#define IBIT_SHIFT_RIGHT_ARITH     6 /* right arithmetic shift                */

/*                                Boolean values                                */
#define IBOOL_FALSE                0 /* false Boolean value                  */
#define IBOOL_TRUE                 1 /* true Boolean value                   */

/*                                chain status codes                            */
#define ICHAIN_NONEXISTENT          0 /* no chain exists for specified identifier */
#define ICHAIN_UNSTARTED           1 /* execution has not been requested      */
#define ICHAIN_SCHEDULED           2 /* execution requested, not begun       */
#define ICHAIN_RUNNING             3 /* execution has begun and is active    */
#define ICHAIN_IDLING              4 /* execution begun, elements inactive   */
#define ICHAIN_ABORTED             5 /* execution was explicitly aborted      */
#define ICHAIN_FAILURE             6 /* execution terminated with failure     */
#define ICHAIN_COMPLETE            7 /* execution completed naturally         */

/*                                colour conversion, direction options          */
#define ICOLOUR_DIRECTION_RGB_CMY   1 /* RGB to CMY option                   */
#define ICOLOUR_DIRECTION_RGB_CMYK  2 /* RGB to CMYK option                  */
#define ICOLOUR_DIRECTION_CMY_RGB   3 /* CMY to RGB option                   */
#define ICOLOUR_DIRECTION_CMYK_RGB  4 /* CMYK to RGB option                  */

/*                                colour conversion, function input and output options */
#define ICOLOUR_CONV_RGB            1 /* RGB                                  */
#define ICOLOUR_CONV_IHS            2 /* IHS                                  */
#define ICOLOUR_CONV_YXY            3 /* Yxy                                  */
#define ICOLOUR_CONV_YUV            4 /* Yuv                                  */
#define ICOLOUR_CONV_LAB            5 /* L*a*b*                              */
#define ICOLOUR_CONV_LUV            6 /* L*u*v*                              */

/*                                colour conversion, method options              */
#define ICOLOUR_METHOD_SIMPLE        1 /* Simple conversion                    */
#define ICOLOUR_METHOD_IMPL_DEPENDENT 2 /* Implementation dependent conversion */

```

```

/*          colour conversion, RGB colour options          */
#define ICOLOUR_RGB_UNCALIBRATED      1 /* uncalibrated      */
#define ICOLOUR_RGB_CIE_SPECTRAL      2 /* CIE spectral      */
#define ICOLOUR_RGB_LIN_NTSC          3 /* linear NTSC      */
#define ICOLOUR_RGB_LIN_EBU           4 /* linear EBU       */
#define ICOLOUR_RGB_LIN_SMPTE         5 /* linear SMPTE     */
#define ICOLOUR_RGB_LIN_CCIR          6 /* linear CCIR      */

/*          colour space options          */
#define ICOLOUR_NONE                   1 /* unspecified      */
#define ICOLOUR_NON_STANDARD_RGB      2 /* non-standard RGB */
#define ICOLOUR_LIN_CCIR_D65_RGB      3 /* linear CCIR illuminant D65 RGB */
#define ICOLOUR_LIN_CIE_E_RGB         4 /* linear CIE illuminant E RGB */
#define ICOLOUR_LIN_EBU_C_RGB         5 /* linear EBU illuminant C RGB */
#define ICOLOUR_LIN_EBU_D65_RGB       6 /* linear EBU illuminant D65 RGB */
#define ICOLOUR_LIN_NTSC_C_RGB        7 /* linear NTSC illuminant C RGB */
#define ICOLOUR_LIN_NTSC_D65_RGB      8 /* linear NTSC illuminant D65 RGB */
#define ICOLOUR_LIN_SMPTE_D65_RGB     9 /* linear SMPTE illuminant D65 RGB */
#define ICOLOUR_GAMMA_CCIR_D65_RGB    10 /* gamma CCIR illuminant D65 RGB */
#define ICOLOUR_GAMMA_EBU_C_RGB       11 /* gamma EBU illuminant C RGB */
#define ICOLOUR_GAMMA_EBU_D65_RGB     12 /* gamma EBU illuminant D65 RGB */
#define ICOLOUR_GAMMA_NTSC_C_RGB      13 /* gamma NTSC illuminant C RGB */
#define ICOLOUR_GAMMA_NTSC_D65_RGB    14 /* gamma NTSC illuminant D65 RGB */
#define ICOLOUR_GAMMA_SMPTE_D65_RGB   15 /* gamma SMPTE illuminant D65 RGB */
#define ICOLOUR_LUM_CHR_EBU_C_YUV     16 /* EBU illuminant C YUV */
#define ICOLOUR_LUM_CHR_EBU_D65_YUV   17 /* EBU illuminant D65 YUV */
#define ICOLOUR_LUM_CHR_NTSC_C_YIQ    18 /* NTSC illuminant C YIQ */
#define ICOLOUR_LUM_CHR_NTSC_D65_YIQ  19 /* NTSC illuminant D65 YIQ */
#define ICOLOUR_LUM_CHR_SMPTE_D65_YCBCR 20 /* SMPTE illuminant D65 YCbCr */
#define ICOLOUR_CIE_XYZ               21 /* CIE XYZ          */
#define ICOLOUR_CIE_UVW               22 /* CIE UVW          */
#define ICOLOUR_CIE_YXY               23 /* CIE Yxy          */
#define ICOLOUR_CIE_YUV               24 /* CIE Yuv          */
#define ICOLOUR_CIE_LAB               25 /* CIE L*a*b*       */
#define ICOLOUR_CIE_LUV               26 /* CIE L*u*v*       */
#define ICOLOUR_IHS                   27 /* IHS              */
#define ICOLOUR_CMY                   28 /* CMY              */
#define ICOLOUR_CMYK                   29 /* CMYK             */

/*          composite image identifier list get pointer options          */
#define IID_LIST_CURRENT               1 /* current position */
#define IID_LIST_PREDECESSOR           2 /* predecessor position */
#define IID_LIST_SUCCESSION           3 /* successor position */
#define IID_LIST_HEAD                  4 /* head position    */
#define IID_LIST_TAIL                  5 /* tail position    */

/*          composite identifier list insert pointer options          */
#define IID_LIST_BEFORE                1 /* insert before current pointer position */
#define IID_LIST_AFTER                 2 /* insert after current pointer position */

```

```

/*                                convolve 2d options                                */
#define ICONVOLVE_UPPER_LEFT      1 /* upper left corner justified                */
#define ICONVOLVE_ENCLOSED        2 /* enclosed array                        */
#define ICONVOLVE_KEY_ZERO        3 /* key pixel, zero exterior              */
#define ICONVOLVE_KEY_REFLECTED    4 /* key pixel, reflected exterior          */

/*                                data transfer options                                */
#define ITRANSFER_IIF             1 /* IIF as source                        */
#define ITRANSFER_APPL            2 /* application as source                 */

/*                                data transfer syntax options                        */
#define ISYNTAX_ASN1              1 /* ASN.1 notation                      */
#define ISYNTAX_PRIVATE           2 /* private notation                     */

/*                                data type (external pixel) options                  */
#define IPIXEL_BI_MSB_PLANAR      1 /* Boolean planar data from msb         */
#define IPIXEL_BI_MSB_INTERLEAVED 2 /* Boolean interleaved data from msb     */
#define IPIXEL_BI_LSB_PLANAR      3 /* Boolean planar data from lsb         */
#define IPIXEL_BI_LSB_INTERLEAVED 4 /* Boolean interleaved data from lsb     */
#define IPIXEL_NI_PLANAR          5 /* non-negative integer planar data     */
#define IPIXEL_NI_INTERLEAVED     6 /* non-negative integer interleaved data */
#define IPIXEL_SI_PLANAR          7 /* signed integer planar data           */
#define IPIXEL_SI_INTERLEAVED     8 /* signed integer interleaved data       */
#define IPIXEL_TI_PLANAR          9 /* fixed point integer planar data       */
#define IPIXEL_TI_INTERLEAVED    10 /* fixed point integer interleaved data   */
#define IPIXEL_RF_PLANAR         11 /* arithmetic real planar data           */
#define IPIXEL_RF_INTERLEAVED    12 /* arithmetic real interleaved data       */
#define IPIXEL_CF                 13 /* complex data                          */

/*                                data type (internal) options                        */
#define IDATA_TYPE_INTERNAL_BD     1 /* Boolean                               */
#define IDATA_TYPE_INTERNAL_ND     2 /* unsigned integer                      */
#define IDATA_TYPE_INTERNAL_SD     3 /* signed integer                        */
#define IDATA_TYPE_INTERNAL_RD     4 /* real arithmetic                       */
#define IDATA_TYPE_INTERNAL_CD     5 /* complex arithmetic                    */
#define IDATA_TYPE_INTERNAL_CS     6 /* character                             */

/*                                data type (parameter) options                      */
#define IDATA_TYPE_PARAMETER_BP    1 /* Boolean                               */
#define IDATA_TYPE_PARAMETER_NP    2 /* unsigned integer                      */
#define IDATA_TYPE_PARAMETER_SP    3 /* signed integer                        */
#define IDATA_TYPE_PARAMETER_RP    4 /* real arithmetic                       */
#define IDATA_TYPE_PARAMETER_CP    5 /* complex arithmetic                    */
#define IDATA_TYPE_PARAMETER_CS    6 /* character                             */

```

```

/*                                data type (virtual register) options                                */
#define IDATA_TYPE_VREG_ID          1 /* internal identifier                                */
#define IDATA_TYPE_VREG_IP          2 /* external identifier                                */
#define IDATA_TYPE_VREG_BP          3 /* Boolean                                              */
#define IDATA_TYPE_VREG_NP          4 /* unsigned integer                                    */
#define IDATA_TYPE_VREG_SP          5 /* signed integer                                       */
#define IDATA_TYPE_VREG_RP          6 /* real arithmetic                                      */
#define IDATA_TYPE_VREG_CP          7 /* complex arithmetic                                   */
#define IDATA_TYPE_VREG_CS          8 /* character                                           */

/*                                difference measures options                                */
#define IDIFF_MEASURE_SQUARED        1 /* squares difference                                  */
#define IDIFF_MEASURE_NORMALIZED    2 /* normalized squares difference                      */

/*                                dimension options                                        */
#define IDIMENSION_VOL              1 /* volume image dimensions                            */
#define IDIMENSION_TEM              2 /* temporal image dimensions                          */
#define IDIMENSION_SPE              3 /* spectral image dimensions                          */

/*                                distance measure options                                */
#define IDISTANCE_MEASURE_EUCLIDEAN  1 /* Euclidean                                          */
#define IDISTANCE_MEASURE_CITY_BLOCK 2 /* City block                                         */

/*                                dyadic, arithmetic combination options                    */
#define IDYADIC_ABSOLUTE_DIFF        1 /* absolute value difference                          */
#define IDYADIC_ADDITION              2 /* addition                                           */
#define IDYADIC_ADDITION_SCALED      3 /* addition, scaled                                   */
#define IDYADIC_ARCTANGENT           4 /* arctangent ratio                                   */
#define IDYADIC_DIVISION              5 /* division                                           */
#define IDYADIC_MAXIMUM              6 /* maximum                                            */
#define IDYADIC_MINIMUM              7 /* minimum                                            */
#define IDYADIC_MULTIPLICATION        8 /* multiplication                                     */
#define IDYADIC_SUBTRACTION           9 /* subtraction                                        */
#define IDYADIC_SUBTRACTION_SCALED   10 /* subtraction, scaled                               */

/*                                dyadic, complex combination options                    */
#define IDYADIC_COMPLEX_ADDITION     1 /* addition                                           */
#define IDYADIC_COMPLEX_DIVISION     2 /* division                                           */
#define IDYADIC_COMPLEX_MULTIPLICATION 3 /* multiplication                                     */
#define IDYADIC_COMPLEX_SUBTRACTION  4 /* subtraction                                        */

/*                                dyadic, logical combination options                    */
#define IDYADIC_LOGICAL_AND           1 /* bitwise AND                                        */
#define IDYADIC_LOGICAL_NAND          2 /* bitwise NAND                                       */
#define IDYADIC_LOGICAL_NOR          3 /* bitwise NOR                                        */
#define IDYADIC_LOGICAL_OR           4 /* bitwise OR                                         */
#define IDYADIC_LOGICAL_XOR          5 /* bitwise XOR                                        */
#define IDYADIC_LOGICAL_INTERSECTION 6 /* Boolean intersection                               */
#define IDYADIC_LOGICAL_UNION         7 /* Boolean union                                      */

```

```

/*                                dyadic, predicate combination options                                */
#define IDYADIC_PREDICATE_GREATER          1 /* greater than                                */
#define IDYADIC_PREDICATE_GREATER_EQUAL    2 /* greater than or equal to                        */
#define IDYADIC_PREDICATE_LESS            3 /* less than                                    */
#define IDYADIC_PREDICATE_LESS_EQUAL       4 /* less than or equal to                          */
#define IDYADIC_PREDICATE_EQUAL           5 /* equal to                                       */
#define IDYADIC_PREDICATE_NOT_EQUAL        6 /* not equal to                                   */

/*                                edge detection options                                */
#define IEDGE_DETECT_GRAD                  1 /* edge gradient                                */
#define IEDGE_DETECT_THRESHOLDED          2 /* thresholded edge gradient                    */
#define IEDGE_DETECT_QUANTIZED            3 /* quantized edge angle                        */

/*                                edge detection, orthogonal options                                */
#define IEDGE_ORTH_EDGE_GRADIENT          1 /* edge gradient                                */
#define IEDGE_ORTH_THRESHOLDED            2 /* thresholded edge gradient                    */
#define IEDGE_ORTH_EDGE_ANGLE             3 /* edge angle                                  */
#define IEDGE_ORTH_ANGLE_QUANTIZED        4 /* quantized edge angle                        */

/*                                erosion & dilation processing options                                */
#define IEROSION_ARRAY_NON_ROTATE          1 /* enclosed array, non-rotated element          */
#define IEROSION_ARRAY_ROTATE              2 /* enclosed array, rotated element              */
#define IEROSION_KEY_NON_ROTATE            3 /* key pixel, non-rotated element                */
#define IEROSION_KEY_ROTATE                4 /* key pixel, rotated structuring element        */

/*                                filter (Butterworth) type options                                */
#define IFILTER_BUTTERWORTH_LOW_PASS       1 /* low-pass                                    */
#define IFILTER_BUTTERWORTH_HIGH_PASS      2 /* high-pass                                   */

/*                                filter (zonal) type options                                */
#define IFILTER_ZONAL_LOW_PASS              1 /* low-pass filter                              */
#define IFILTER_ZONAL_HIGH_PASS_TERM        2 /* high-pass filter with constant term          */
#define IFILTER_ZONAL_HIGH_PASS_NO_TERM    3 /* high-pass filter without constant term        */

/*                                filtering process options                                */
#define IFILTERING_ENCLOSED_ARRAY          1 /* enclosed array                              */
#define IFILTERING_KEY_PIXEL               2 /* key pixel                                   */

/*                                histogram computation option                                */
#define IHIST_LIMIT_SPECIFIED              1 /* specified limits                             */
#define IHIST_LIMIT_EXTREMA                2 /* image extrema limits                         */

/*                                histogram modification options                                */
#define IHIST_MODIFICATION_CUBE_ROOT       1 /* cube root                                   */
#define IHIST_MODIFICATION_EXPONENTIAL     2 /* exponential                                 */
#define IHIST_MODIFICATION_LOGARITHMIC     3 /* logarithmic                                 */
#define IHIST_MODIFICATION_RAYLEIGH        4 /* Rayleigh                                   */
#define IHIST_MODIFICATION_UNIFORM         5 /* uniform                                    */

```



```

/*                                hit or miss transformation options                                */
#define IHIT_MISS_BRANCH_DETECT      1 /* branch detect                                */
#define IHIT_MISS_BRIDGE              2 /* bridge                                    */
#define IHIT_MISS_CONVEX_HULL_FILL    3 /* convex hull fill                        */
#define IHIT_MISS_DIAGONAL_FILL      4 /* diagonal fill                          */
#define IHIT_MISS_DILATE_8_NEIGHBOUR 5 /* dilate, eight-neighbour                */
#define IHIT_MISS_END_DETECT          6 /* end detect                              */
#define IHIT_MISS_END_DETECT_EAST     7 /* end detect, east                       */
#define IHIT_MISS_END_DETECT_NORTH    8 /* end detect, north                      */
#define IHIT_MISS_END_DETECT_NORTHEAST 9 /* end detect, northeast                  */
#define IHIT_MISS_END_DETECT_NORTHWEST 10 /* end detect, northwest                 */
#define IHIT_MISS_END_DETECT_SOUTH    11 /* end detect, south                     */
#define IHIT_MISS_END_DETECT_SOUTHEAST 12 /* end detect, southeast                 */
#define IHIT_MISS_END_DETECT_SOUTHWEST 13 /* end detect, southwest                 */
#define IHIT_MISS_END_DETECT_WEST     14 /* end detect, west                      */
#define IHIT_MISS_ERODE_8_NEIGHBOUR 15 /* erode, eight-neighbour                */
#define IHIT_MISS_FATTEN              16 /* fatten                                */
#define IHIT_MISS_H_BREAK             17 /* H break                               */
#define IHIT_MISS_HOLE_DETECT         18 /* hole detect                           */
#define IHIT_MISS_INTERIOR_PIXEL_FILL 19 /* interior pixel fill                    */
#define IHIT_MISS_INTERIOR_PIXEL_REMOVE 20 /* interior pixel remove                  */
#define IHIT_MISS_ISOLATED_PIXEL_DETECT 21 /* isolated pixel detect                  */
#define IHIT_MISS_ISOLATED_PIXEL_REMOVE 22 /* isolated pixel remove                  */
#define IHIT_MISS_MAJORITY_BLACK      23 /* majority black                        */
#define IHIT_MISS_SPUR_REMOVE         24 /* spur remove                           */

/*                                image, grey scale orientation options                                */
#define IGREY_LEFT_RIGHT              1 /* left-to-right                         */
#define IGREY_RIGHT_LEFT              2 /* right-to-left                         */
#define IGREY_TOP_BOTTOM              3 /* top-to-bottom                         */
#define IGREY_BOTTOM_TOP              4 /* bottom-to-top                         */

/*                                image structure options                                */
#define IIMAGE_STRUCTURE_MON          1 /* monochrome image                      */
#define IIMAGE_STRUCTURE_VOL          2 /* volume image                          */
#define IIMAGE_STRUCTURE_TEM          3 /* temporal image                        */
#define IIMAGE_STRUCTURE_COLR         4 /* colour image                          */
#define IIMAGE_STRUCTURE_SPE          5 /* spectral image                        */
#define IIMAGE_STRUCTURE_VT           6 /* volume-temporal image                 */
#define IIMAGE_STRUCTURE_VC           7 /* volume-colour image                   */
#define IIMAGE_STRUCTURE_VS           8 /* volume-spectral image                 */
#define IIMAGE_STRUCTURE_TC           9 /* temporal-colour image                 */
#define IIMAGE_STRUCTURE_TS           10 /* temporal-spectral image               */
#define IIMAGE_STRUCTURE_VTC          11 /* volume-temporal-colour image          */
#define IIMAGE_STRUCTURE_VTS          12 /* volume-temporal-spectral image        */
#define IIMAGE_STRUCTURE_GEN          13 /* generic image                         */

/*                                impulse, pyramid shape options                                */
#define IPYRAMID_SMOOTHING            1 /* smoothing                             */
#define IPYRAMID_HORIZONTAL           2 /* horizontal derivative                  */
#define IPYRAMID_VERTICAL             3 /* vertical derivative                    */

```

```

/*                                     impulse rotation options                                     */
#define IIMPULSE_NO_REFLECTION          1 /* no reflection                                     */
#define IIMPULSE_REFLECTION            2 /* reflection                                       */

/*                                     lookup table modes                                       */
#define ILOOKUP_ID                      1 /* one dimension                                   */
#define ILOOKUP_2D                     2 /* two dimensions                                  */

/*                                     luminance RGB colour options                             */
#define ILUM_LIN_CCIR_D65_RGB          1 /* linear CCIR illuminant D65 RGB                 */
#define ILUM_LIN_CIE_E_RGB             2 /* linear CIE illuminant E RGB                   */
#define ILUM_LIN_EBU_C_RGB             3 /* linear EBU illuminant C RGB                   */
#define ILUM_LIN_EBU_D65_RGB           4 /* linear EBU illuminant D65 RGB                 */
#define ILUM_LIN_NTSC_C_RGB            5 /* linear NTSC illuminant C RGB                  */
#define ILUM_LIN_NTSC_D65_RGB          6 /* linear NTSC illuminant D65 RGB                 */
#define ILUM_LIN_SMPTE_D65_RGB         7 /* linear SMPTE illuminant D65 RGB                */
#define ILUM_GAMMA_EBU_C_RGB           8 /* gamma EBU illuminant C RGB                   */
#define ILUM_GAMMA_EBU_D65_RGB         9 /* gamma EBU illuminant D65 RGB                 */
#define ILUM_GAMMA_NTSC_C_RGB          10 /* gamma NTSC illuminant C RGB                  */
#define ILUM_GAMMA_NTSC_D65_RGB        11 /* gamma NTSC illuminant D65 RGB                 */
#define ILUM_GAMMA_SMPTE_D65_RGB       12 /* gamma SMPTE illuminant D65 RGB                */

/*                                     Markov spectrum symmetry options                         */
#define IMARKOV_ELLIPTICAL              1 /* elliptical                                     */
#define IMARKOV_SEPARABLE               2 /* separable                                      */

/*                                     monadic, arithmetic options                             */
#define IMONADIC_ADDITION_BY           1 /* addition by constant                          */
#define IMONADIC_ADDITION_BY_SCALED    2 /* addition by constant, scaled                  */
#define IMONADIC_DIVISION_BY           3 /* division by constant                          */
#define IMONADIC_DIVISION_OF           4 /* division of constant                          */
#define IMONADIC_MAXIMUM               5 /* maximum with constant                        */
#define IMONADIC_MINIMUM               6 /* minimum with constant                        */
#define IMONADIC_MULTIPLICATION        7 /* multiplication by constant                   */
#define IMONADIC_SUBTRACTION_BY        8 /* subtraction by constant                     */
#define IMONADIC_SUBTRACTION_BY_SCALED 9 /* subtraction by constant, scaled              */
#define IMONADIC_SUBTRACTION_OF        10 /* subtraction of constant                     */
#define IMONADIC_SUBTRACTION_OF_SCALED 11 /* subtraction of constant, scaled              */

/*                                     monadic, complex options                             */
#define IMONADIC_COMPLEX_ADDITION_BY   1 /* addition by constant                          */
#define IMONADIC_COMPLEX_DIVISION_BY   2 /* division by constant                         */
#define IMONADIC_COMPLEX_DIVISION_OF   3 /* division of constant                         */
#define IMONADIC_COMPLEX_MULTIPLICATION 4 /* multiplication by constant                  */
#define IMONADIC_COMPLEX_SUBTRACTION_BY 5 /* subtraction by constant                     */
#define IMONADIC_COMPLEX_SUBTRACTION_OF 6 /* subtraction of constant                     */

```

```

/*                                monadic, logical options                                */
#define IMONADIC_LOGICAL_AND      1 /* bitwise AND                                */
#define IMONADIC_LOGICAL_NAND     2 /* bitwise NAND                             */
#define IMONADIC_LOGICAL_NOR     3 /* bitwise NOR                              */
#define IMONADIC_LOGICAL_OR      4 /* bitwise OR                               */
#define IMONADIC_LOGICAL_XOR     5 /* bitwise XOR                              */
#define IMONADIC_LOGICAL_INTERSECTION 6 /* Boolean intersection                     */
#define IMONADIC_LOGICAL_UNION    7 /* Boolean union                            */

/*                                morphological options                                */
#define IMORPHOLOGICAL_GROW       1 /* end grow                                */
#define IMORPHOLOGICAL_GROW_EAST  2 /* end grow, east                          */
#define IMORPHOLOGICAL_GROW_NORTH 3 /* end grow, north                         */
#define IMORPHOLOGICAL_GROW_NORTHEAST 4 /* end grow, northeast                     */
#define IMORPHOLOGICAL_GROW_NORTHWEST 5 /* end grow, northwest                     */
#define IMORPHOLOGICAL_GROW_SOUTH 6 /* end grow, south                        */
#define IMORPHOLOGICAL_GROW_SOUTHEAST 7 /* end grow, southeast                     */
#define IMORPHOLOGICAL_GROW_SOUTHWEST 8 /* end grow, southwest                     */
#define IMORPHOLOGICAL_GROW_WEST  9 /* end grow, west                         */
#define IMORPHOLOGICAL_EXOSKELETON 10 /* exoskeleton                            */
#define IMORPHOLOGICAL_EXOTHIN    11 /* exothin                                */
#define IMORPHOLOGICAL_SHRINK     12 /* shrink                                 */
#define IMORPHOLOGICAL_SKELETON   13 /* skeleton                               */
#define IMORPHOLOGICAL_THICKEN    14 /* thicken                                */
#define IMORPHOLOGICAL_THIN       15 /* thin                                   */

/*                                neighbourhood array semantic label options            */
#define INBHOOD_LABEL_GENERIC     1 /* generic                                */
#define INBHOOD_LABEL_DITHER     2 /* dither                                 */
#define INBHOOD_LABEL_IMPULSE     3 /* impulse response                       */
#define INBHOOD_LABEL_MASK        4 /* mask                                   */
#define INBHOOD_LABEL_STRUCTURE   5 /* structuring element                    */

/*                                object type options                                */
#define IOBJECT_IMAGE             1 /* image                                  */
#define IOBJECT_HIST              2 /* histogram                             */
#define IOBJECT_ID_ARRAY          3 /* identifier array                       */
#define IOBJECT_ID_LIST           4 /* identifier list                        */
#define IOBJECT_ID_RECORD         5 /* identifier record                      */
#define IOBJECT_LUT               6 /* lookup table                           */
#define IOBJECT_MATRIX            7 /* matrix                                 */
#define IOBJECT_NBHOOD            8 /* neighbourhood array                    */
#define IOBJECT_PIXEL_RECORD      9 /* pixel record                           */
#define IOBJECT_ROI_ARRAY         10 /* ROI, array type                       */
#define IOBJECT_ROI_COORDINATE    11 /* ROI, coordinate array type            */
#define IOBJECT_ROI_ELLIPTICAL    12 /* ROI, elliptical type                  */
#define IOBJECT_ROI_GENERIC       13 /* ROI, generic type                     */
#define IOBJECT_ROI_POLYGON       14 /* ROI, polygon type                     */
#define IOBJECT_ROI_RECTANGULAR   15 /* ROI, rectangular type                 */
#define IOBJECT_STATIC            16 /* static array                           */
#define IOBJECT_TUPLE             17 /* tuple                                  */
#define IOBJECT_VALUE_BOUNDS      18 /* value bounds collection                */
#define IOBJECT_VREG              19 /* virtual register                       */

```



```

/*                                paint brush options                                */
#define IBRUSH_HARD                1 /* hard brush                                */
#define IBRUSH_SOFT                2 /* soft brush                                */

/*                                painting options                                */
#define IPAIN_T_POINT              1 /* point painting                            */
#define IPAIN_T_LINE              2 /* straight line painting                    */

/*                                probability density options                      */
#define IPROBABILITY_UNIFORM       1 /* uniform                                  */
#define IPROBABILITY_GAUSS        2 /* Gaussian                                */

/*                                repository class options                          */
#define IREPOSITORY_IMPULSE        1 /* impulse response array                    */
#define IREPOSITORY_DITHER        2 /* dither array                              */
#define IREPOSITORY_COLOUR        3 /* colour conversion matrix                  */

/*                                resampling options                              */
#define IRESAMPLE_NEAREST_NEIGHBOUR 1 /* nearest neighbour                        */
#define IRESAMPLE_2_BILINEAR       2 /* support 2 bilinear                      */
#define IRESAMPLE_2_LUT            3 /* support 2 lookup table                  */
#define IRESAMPLE_4_TRUNC_SINC     4 /* support 4 truncated sinc                */
#define IRESAMPLE_4_CUBIC_B_SPLINE 5 /* support 4 cubic B-spline                */
#define IRESAMPLE_4_CUBIC_CONVOLUTION 6 /* support 4 cubic convolution              */
#define IRESAMPLE_4_LAGRANGIAN     7 /* support 4 Lagrangian                    */
#define IRESAMPLE_4_LUT            8 /* support 4 lookup table                  */
#define IRESAMPLE_8_TRUNC_SINC     9 /* support 8 truncated sinc                */
#define IRESAMPLE_8_LUT           10 /* support 8 lookup table                  */

/*                                ROI structure options                            */
#define IROI_ARRAY                 1 /* array ROI structure                      */
#define IROI_COORDINATE            2 /* coordinate array ROI structure           */
#define IROI_ELLIPTICAL            3 /* elliptical ROI structure                 */
#define IROI_GENERIC               4 /* generic ROI structure                    */
#define IROI_POLYGON               5 /* polygon ROI structure                    */
#define IROI_RECTANGULAR           6 /* rectangular ROI structure                */

/*                                rotation options                                */
#define IROTATION_SPACE_2D         1 /* 2D rotation                             */
#define IROTATION_SPACE_3D         2 /* 3D rotation                             */

/*                                sequence running measures options                */
#define ISEQ_MEASURE_DIFF          1 /* difference                               */
#define ISEQ_MEASURE_RATIO         2 /* ratio                                   */
#define ISEQ_MEASURE_LOGARITHM_RATIO 3 /* logarithm ratio                         */

/*                                split image options                              */
#define ISPLIT_LEFT_LEFT           1 /* split left 1 - left 2                   */
#define ISPLIT_RIGHT_RIGHT         2 /* split right 1 - right 2                  */
#define ISPLIT_TOP_TOP             3 /* split top 1 - top 2                     */
#define ISPLIT_BOTTOM_BOTTOM       4 /* split bottom 1 - bottom 2                */
#define ISPLIT_LEFT_RIGHT          5 /* split left 1 - right 2                   */
#define ISPLIT_TOP_BOTTOM          6 /* split top 1 - bottom 2                   */

```

```

/*          static array semantic label options          */
#define ISTATIC_GENERIC          1 /* generic          */
#define ISTATIC_POWER           2 /* power spectrum */
#define ISTATIC_TRANSFER         3 /* transfer function */
#define ISTATIC_WINDOWING       4 /* windowing function */

/*          unary integer conversion options          */
#define IUNARY_INTEGER_ABSOLUTE  1 /* absolute value */
#define IUNARY_INTEGER_CUBE      2 /* cube          */
#define IUNARY_INTEGER_NEGATIVE  3 /* negative      */
#define IUNARY_INTEGER_SQUARE    4 /* square        */

/*          unary real conversion options          */
#define IUNARY_REAL_ABSOLUTE      1 /* absolute value */
#define IUNARY_REAL_ARC_COSINE    2 /* arc cosine    */
#define IUNARY_REAL_ARC_SINE      3 /* arc sine      */
#define IUNARY_REAL_ARC_TANGENT   4 /* arc tangent   */
#define IUNARY_REAL_COSECANT      5 /* cosecant      */
#define IUNARY_REAL_COSINE        6 /* cosine        */
#define IUNARY_REAL_COTANGENT     7 /* cotangent     */
#define IUNARY_REAL_CUBE          8 /* cube          */
#define IUNARY_REAL_CUBE_ROOT     9 /* cube root     */
#define IUNARY_REAL_EXPONENTIAL  10 /* exponential   */
#define IUNARY_REAL_LOGARITHM_E   11 /* logarithm, base e */
#define IUNARY_REAL_LOGARITHM_10  12 /* logarithm, base 10 */
#define IUNARY_REAL_LOGARITHM_2   13 /* logarithm, base 2 */
#define IUNARY_REAL_NEGATIVE      14 /* negative      */
#define IUNARY_REAL_RECIPROCAL    15 /* reciprocal    */
#define IUNARY_REAL_SECANT        16 /* secant        */
#define IUNARY_REAL_SINE          17 /* sine          */
#define IUNARY_REAL_SQUARE        18 /* square        */
#define IUNARY_REAL_SQUARE_ROOT   19 /* square root   */
#define IUNARY_REAL_TANGENT       20 /* tangent       */

/*          virtual register conditional return options          */
#define IVREG_CONDITIONAL_TERMINATE 1 /* terminate chain */
#define IVREG_CONDITIONAL_PROCEED   2 /* proceed to next chain operator */

/*          window statistic options          */
#define IWINDOW_MEAN              1 /* mean statistic option */
#define IWINDOW_SDEV              2 /* standard deviation option */
#define IWINDOW_MEDIAN            3 /* median option */
#define IWINDOW_MODE              4 /* mode option */

/*          windowing options          */
#define IWINDOWING_BARTLETT        1 /* Bartlett */
#define IWINDOWING_BLACKMAN        2 /* Blackman */
#define IWINDOWING_HAMMING         3 /* Hamming */
#define IWINDOWING_HANNING         4 /* Hanning */
#define IWINDOWING_RECTANGULAR     5 /* rectangular */

```

/*	FUNCTION DESCRIPTORS	*/
#define IF_ABORT_ASYNC_EXECUTION	1 /* abort_asynchronous_execution	*/
#define IF_ACCUMULATOR	2 /* accumulator	*/
#define IF_ADAPTIVE_HIST_EQUAL	3 /* adaptive_histogram_equalization	*/
#define IF_ADAPTIVE_HIST_EQUAL_VREG	270 /* adaptive_histogram_equalization_vreg	*/
#define IF_ALLOCATE_CHAIN	4 /* allocate_chain	*/
#define IF_ALLOCATE_CHAIN_VREG	271 /* allocate_chain_vreg	*/
#define IF_ALLOCATE_ID_ARRAY	5 /* allocate_composite_identifier_array	*/
#define IF_ALLOCATE_ID_ARRAY_VREG	272 /* allocate_composite_identifier_array_vreg	*/
#define IF_ALLOCATE_ID_LIST	6 /* allocate_composite_identifier_list	*/
#define IF_ALLOCATE_ID_LIST_VREG	273 /* allocate_composite_identifier_list_vreg	*/
#define IF_ALLOCATE_ID_RECORD	7 /* allocate_composite_identifier_record	*/
#define IF_ALLOCATE_ID_RECORD_VREG	274 /* allocate_composite_identifier_record_vreg	*/
#define IF_ALLOCATE_HIST	8 /* allocate_histogram	*/
#define IF_ALLOCATE_HIST_VREG	275 /* allocate_histogram_vreg	*/
#define IF_ALLOCATE_IMAGE	9 /* allocate_image	*/
#define IF_ALLOCATE_IMAGE_VREG	276 /* allocate_image_vreg	*/
#define IF_ALLOCATE_LUT	10 /* allocate_lookup_table	*/
#define IF_ALLOCATE_LUT_VREG	277 /* allocate_lookup_table_vreg	*/
#define IF_ALLOCATE_MATRIX	11 /* allocate_matrix	*/
#define IF_ALLOCATE_MATRIX_VREG	278 /* allocate_matrix_vreg	*/
#define IF_ALLOCATE_NBHOOD_ARRAY	12 /* allocate_neighbourhood_array	*/
#define IF_ALLOCATE_NBHOOD_ARRAY_VREG	279 /* allocate_neighbourhood_array_vreg	*/
#define IF_ALLOCATE_PIXEL_RECORD	13 /* allocate_pixel_record	*/
#define IF_ALLOCATE_PIXEL_RECORD_VREG	280 /* allocate_pixel_record_vreg	*/
#define IF_ALLOCATE_ROI	14 /* allocate_roi	*/
#define IF_ALLOCATE_ROI_VREG	281 /* allocate_roi_vreg	*/
#define IF_ALLOCATE_STATIC_ARRAY	15 /* allocate_static_array	*/
#define IF_ALLOCATE_STATIC_ARRAY_VREG	282 /* allocate_static_array_vreg	*/
#define IF_ALLOCATE_TUPLE	16 /* allocate_tuple	*/
#define IF_ALLOCATE_TUPLE_VREG	283 /* allocate_tuple_vreg	*/
#define IF_ALLOCATE_VALUE_BOUNDS	17 /* allocate_value_bounds	*/
#define IF_ALLOCATE_VALUE_BOUNDS_VREG	284 /* allocate_value_bounds_vreg	*/
#define IF_ALLOCATE_VREG	18 /* allocate_vreg	*/
#define IF_ALLOCATE_VREG_VREG	285 /* allocate_vreg_vreg	*/
#define IF_ALPHA_BLEND_CONSTANT	19 /* alpha_blend_constant	*/
#define IF_ALPHA_BLEND_CONSTANT_VREG	286 /* alpha_blend_constant_vreg	*/
#define IF_ALPHA_BLEND_VARIABLE	20 /* alpha_blend_variable	*/
#define IF_ALPHA_BLEND_VARIABLE_VREG	287 /* alpha_blend_variable_vreg	*/
#define IF_ARRAY_TO_LUT	21 /* array_to_lut	*/
#define IF_ARRAY_TO_LUT_VREG	288 /* array_to_lut_vreg	*/
#define IF_BIND_MATCH_POINT	22 /* bind_match_point	*/
#define IF_BIND_MATCH_POINT_VREG	289 /* bind_match_point_vreg	*/
#define IF_BIND_ROI	23 /* bind_roi	*/
#define IF_BIND_ROI_VREG	290 /* bind_roi_vreg	*/
#define IF_BIT_SHIFT	24 /* bit_shift	*/
#define IF_BIT_SHIFT_VREG	291 /* bit_shift_vreg	*/
#define IF_CARTESIAN_TO_POLAR	25 /* cartesian_to_polar	*/
#define IF_CARTESIAN_TO_POLAR_VREG	292 /* cartesian_to_polar_vreg	*/
#define IF_CHAIN_ABORT	26 /* chain_abort	*/
#define IF_CHAIN_ABORT_VREG	293 /* chain_abort_vreg	*/

```

/*          FUNCTION DESCRIPTORS (continued)          */

#define IF_CHAIN_BEGIN                27 /* chain_begin                */
#define IF_CHAIN_DELETE              28 /* chain_delete                */
#define IF_CHAIN_DELETE_VREG        294 /* chain_delete_vreg          */
#define IF_CHAIN_END                  29 /* chain_end                    */
#define IF_CHAIN_EXECUTE              30 /* chain_execute                */
#define IF_CHAIN_EXECUTE_VREG        295 /* chain_execute_vreg          */
#define IF_CHAIN_RELOAD              31 /* chain_reload                */
#define IF_CHAIN_RELOAD_VREG        296 /* chain_reload_vreg           */
#define IF_CLASSIFIER_BAYES          32 /* classifier_bayes             */
#define IF_CLASSIFIER_BAYES_VREG    297 /* classifier_bayes_vreg        */
#define IF_CLASSIFIER_NEAREST_NBR    33 /* classifier_nearest_neighbour */
#define IF_CLASSIFIER_NEAREST_NBR_VREG 298 /* classifier_nearest_neighbour_vreg */
#define IF_CLOSE_PIKS                34 /* close_piks                    */
#define IF_CLOSE_PIKS_EMERGENCY      35 /* close_piks_emergency         */
#define IF_COLOUR_CONV_LIN           36 /* colour_conversion_linear      */
#define IF_COLOUR_CONV_LIN_VREG      299 /* colour_conversion_linear_vreg */
#define IF_COLOUR_CONV_MATRIX        37 /* colour_conversion_matrix      */
#define IF_COLOUR_CONV_MATRIX_VREG   300 /* colour_conversion_matrix_vreg */
#define IF_COLOUR_CONV_NONLIN        38 /* colour_conversion_nonlinear   */
#define IF_COLOUR_CONV_NONLIN_VREG   301 /* colour_conversion_nonlinear_vreg */
#define IF_COLOUR_CONV_SUBTRACTIVE   39 /* colour_conversion_subtractive */
#define IF_COLOUR_CONV_SUBTRACTIVE_VREG 302 /* colour_conversion_subtractive_vreg */
#define IF_COLOUR_LOOKUP_INTERP      40 /* colour_lookup_interpolated    */
#define IF_COLOUR_LOOKUP_INTERP_VREG 303 /* colour_lookup_interpolated_vreg */
#define IF_COMPLEMENT                41 /* complement                    */
#define IF_COMPLEMENT_ROI            42 /* complement_roi                */
#define IF_COMPLEMENT_ROI_VREG       304 /* complement_roi_vreg          */
#define IF_COMPLEMENT_VREG           305 /* complement_vreg               */
#define IF_COMPLEX_COMPOSITION        43 /* complex_composition           */
#define IF_COMPLEX_COMPOSITION_VREG  306 /* complex_composition_vreg      */
#define IF_COMPLEX_CONJUGATE         44 /* complex_conjugate             */
#define IF_COMPLEX_CONJUGATE_VREG    307 /* complex_conjugate_vreg        */
#define IF_COMPLEX_DECOMPOSITION      45 /* complex_decomposition         */
#define IF_COMPLEX_DECOMPOSITION_VREG 308 /* complex_decomposition_vreg    */
#define IF_COMPLEX_MAGNITUDE         46 /* complex_magnitude            */
#define IF_COMPLEX_MAGNITUDE_VREG    309 /* complex_magnitude_vreg       */
#define IF_ID_ARRAY_EQUAL            47 /* composite_identifier_array_equal */
#define IF_ID_ARRAY_EQUAL_VREG       310 /* composite_identifier_array_equal_vreg */
#define IF_ID_ARRAY_GET              48 /* composite_identifier_array_get */
#define IF_ID_ARRAY_GET_VREG         311 /* composite_identifier_array_get_vreg */
#define IF_ID_ARRAY_PUT              49 /* composite_identifier_array_put */
#define IF_ID_ARRAY_PUT_VREG         312 /* composite_identifier_array_put_vreg */
#define IF_ID_LIST_EMPTY             50 /* composite_identifier_list_empty */
#define IF_ID_LIST_EMPTY_VREG        313 /* composite_identifier_list_empty_vreg */
#define IF_ID_LIST_EQUAL             51 /* composite_identifier_list_equal */
#define IF_ID_LIST_EQUAL_VREG        314 /* composite_identifier_list_equal_vreg */
#define IF_ID_LIST_GET               52 /* composite_identifier_list_get */
#define IF_ID_LIST_GET_VREG          315 /* composite_identifier_list_get_vreg */
#define IF_ID_LIST_INSERT            53 /* composite_identifier_list_insert */
#define IF_ID_LIST_INSERT_VREG       316 /* composite_identifier_list_insert_vreg */

```

```

/*                                FUNCTION DESCRIPTORS (continued)                                */

#define IF_ID_LIST_REMOVE          54 /* composite_identifier_list_remove          */
#define IF_ID_LIST_REMOVE_VREG    317 /* composite_identifier_list_remove_vreg    */
#define IF_ID_RECORD_EQUAL        55 /* composite_identifier_record_equal        */
#define IF_ID_RECORD_EQUAL_VREG   318 /* composite_identifier_record_equal_vreg   */
#define IF_ID_RECORD_GET          56 /* composite_identifier_record_get          */
#define IF_ID_RECORD_GET_VREG     319 /* composite_identifier_record_get_vreg     */
#define IF_ID_RECORD_PUT          57 /* composite_identifier_record_put          */
#define IF_ID_RECORD_PUT_VREG     320 /* composite_identifier_record_put_vreg     */
#define IF_CONSTANT_PREDICATE     58 /* constant_predicate                      */
#define IF_CONSTANT_PREDICATE_VREG 321 /* constant_predicate_vreg                  */
#define IF_CONVERT_ARRAY_TO_IMAGE 59 /* convert_array_to_image                   */
#define IF_CONVERT_ARRAY_TO_IMAGE_VREG 322 /* convert_array_to_image_vreg              */
#define IF_CONVERT_IMAGE_DATATYPE 60 /* convert_image_datatype                   */
#define IF_CONVERT_IMAGE_DATATYPE_VREG 323 /* convert_image_datatype_vreg              */
#define IF_CONVERT_IMAGE_TO_ARRAY 61 /* convert_image_to_array                   */
#define IF_CONVERT_IMAGE_TO_ARRAY_VREG 324 /* convert_image_to_array_vreg              */
#define IF_CONVERT_IMAGE_TO_ROI    62 /* convert_image_to_roi                     */
#define IF_CONVERT_IMAGE_TO_ROI_VREG 325 /* convert_image_to_roi_vreg                */
#define IF_CONVERT_ROI_TO_IMAGE    63 /* convert_roi_to_image                     */
#define IF_CONVERT_ROI_TO_IMAGE_VREG 326 /* convert_roi_to_image_vreg                */
#define IF_CONVOLVE_2D             64 /* convolve_2d                             */
#define IF_CONVOLVE_2D_VREG        327 /* convolve_2d_vreg                        */
#define IF_CONVOLVE_5D             65 /* convolve_5d                             */
#define IF_CONVOLVE_5D_VREG        328 /* convolve_5d_vreg                        */
#define IF_COPY_WINDOW             66 /* copy_window                             */
#define IF_COPY_WINDOW_ROI         67 /* copy_window_roi                         */
#define IF_COPY_WINDOW_ROI_VREG    329 /* copy_window_roi_vreg                    */
#define IF_COPY_WINDOW_VREG        330 /* copy_window_vreg                        */
#define IF_CREATE_TUPLE            68 /* create_tuple                            */
#define IF_CREATE_TUPLE_VREG       331 /* create_tuple_vreg                       */
#define IF_CROSS_CORRELATION       69 /* cross_correlation                       */
#define IF_CROSS_CORRELATION_VREG  332 /* cross_correlation_vreg                  */
#define IF_DEALLOCATE_DATA_OBJECT  70 /* deallocate_data_object                  */
#define IF_DEALLOCATE_DATA_OBJECT_VREG 333 /* deallocate_data_object_vreg              */
#define IF_DEFINE_SUB_IMAGE        71 /* efine_sub_image                         */
#define IF_DEFINE_SUB_IMAGE_VREG    334 /* efine_sub_image_vreg                    */
#define IF_DIFF_MEASURES           72 /* difference_measures                      */
#define IF_DIFF_MEASURES_VREG      335 /* difference_measures_vreg                */
#define IF_DIFFUSE                 73 /* diffuse                                  */
#define IF_DIFFUSE_VREG            336 /* diffuse_vreg                            */
#define IF_DITHER                 74 /* dither                                  */
#define IF_DITHER_VREG            337 /* dither_vreg                             */
#define IF_DRAW_PIXELS            75 /* draw_pixels                             */
#define IF_DRAW_PIXELS_ROI         76 /* draw_pixels_roi                         */
#define IF_DRAW_PIXELS_ROI_VREG    338 /* draw_pixels_roi_vreg                    */
#define IF_DRAW_PIXELS_VREG        339 /* draw_pixels_vreg                        */
#define IF_DYADIC_ARITH           77 /* dyadic_arithmetic                       */
#define IF_DYADIC_ARITH_VREG       340 /* dyadic_arithmetic_vreg                  */
#define IF_DYADIC_COMPLEX          78 /* dyadic_complex                          */
#define IF_DYADIC_COMPLEX_VREG     341 /* dyadic_complex_vreg                     */

```


/*	FUNCTION DESCRIPTORS (continued)	*/
#define IF_DYADIC_LOGICAL	79 /* dyadic_logical	*/
#define IF_DYADIC_LOGICAL_ROI	80 /* dyadic_logical_roi	*/
#define IF_DYADIC_LOGICAL_ROI_VREG	342 /* dyadic_logical_roi_vreg	*/
#define IF_DYADIC_LOGICAL_VREG	343 /* dyadic_logical_vreg	*/
#define IF_DYADIC_PREDICATE	81 /* dyadic_predicate	*/
#define IF_DYADIC_PREDICATE_VREG	344 /* dyadic_predicate_vreg	*/
#define IF_EDGE_DETECT_ORTH_GRAD	82 /* edge_detection_orthogonal_gradient	*/
#define IF_EDGE_DETECT_ORTH_GRAD_VREG	345 /* edge_detection_orthogonal_gradient_vreg	*/
#define IF_EDGE_DETECT_2ND_DERIV	83 /* edge_detection_second_derivative	*/
#define IF_EDGE_DETECT_2ND_DERIV_VREG	346 /* edge_detection_second_derivative_vreg	*/
#define IF_EDGE_DETECT_TEMPL_GRAD	84 /* edge_detection_template_gradient	*/
#define IF_EDGE_DETECT_TEMPL_GRAD_VREG	347 /* edge_detection_template_gradient_vreg	*/
#define IF_EQUAL_PREDICATE	85 /* equal_predicate	*/
#define IF_EQUAL_PREDICATE_VREG	348 /* equal_predicate_vreg	*/
#define IF_EROSION_DILATE_BOOL	86 /* erosion_dilation_boolean	*/
#define IF_EROSION_DILATE_BOOL_ROI	87 /* erosion_dilation_boolean_roi	*/
#define IF_EROSION_DILATE_BOOL_ROI_VREG	349 /* erosion_dilation_boolean_roi_vreg	*/
#define IF_EROSION_DILATE_BOOL_VREG	350 /* erosion_dilation_boolean_vreg	*/
#define IF_EROSION_DILATE_GREY	88 /* erosion_dilation_grey	*/
#define IF_EROSION_DILATE_GREY_VREG	351 /* erosion_dilation_grey_vreg	*/
#define IF_ERROR_FUNCTION_SCALING	89 /* error_function_scaling	*/
#define IF_ERROR_FUNCTION_SCALING_VREG	352 /* error_function_scaling_vreg	*/
#define IF_ERROR_HANDLER	90 /* error_handler	*/
#define IF_ERROR_LOGGER	91 /* error_logger	*/
#define IF_ERROR_TEST	92 /* error_test	*/
#define IF_ERROR_TEST_VREG	353 /* error_test_vreg	*/
#define IF_EXPORT_HIST	93 /* export_histogram	*/
#define IF_EXPORT_IMAGE	94 /* export_image	*/
#define IF_EXPORT_LUT	95 /* export_lut	*/
#define IF_EXPORT_MATRIX	96 /* export_matrix	*/
#define IF_EXPORT_NBHOOD_ARRAY	97 /* export_neighbourhood_array	*/
#define IF_EXPORT_ROI_ARRAY	98 /* export_roi_array	*/
#define IF_EXPORT_STATIC_ARRAY	99 /* export_static_array	*/
#define IF_EXPORT_TUPLE	100 /* export_tuple	*/
#define IF_EXPORT_VALUE_BOUNDS	101 /* export_value_bounds	*/
#define IF_EXTRACT_PIXEL_PLANE	102 /* extract_pixel_plane	*/
#define IF_EXTRACT_PIXEL_PLANE_ROI	103 /* extract_pixel_plane_roi	*/
#define IF_EXTRACT_PIXEL_PLANE_ROI_VREG	354 /* extract_pixel_plane_roi_vreg	*/
#define IF_EXTRACT_PIXEL_PLANE_VREG	355 /* extract_pixel_plane_vreg	*/
#define IF_EXTREMA	104 /* extrema	*/
#define IF_FALSE_COLOUR	105 /* false_colour	*/
#define IF_FALSE_COLOUR_VREG	356 /* false_colour_vreg	*/
#define IF_FILL_REGION	106 /* fill_region	*/
#define IF_FILL_REGION_ROI	107 /* fill_region_roi	*/
#define IF_FILL_REGION_ROI_VREG	357 /* fill_region_roi_vreg	*/
#define IF_FILL_REGION_VREG	358 /* fill_region_vreg	*/
#define IF_FILTER_BUTTERWORTH	108 /* filter_butterworth	*/
#define IF_FILTER_BUTTERWORTH_VREG	359 /* filter_butterworth_vreg	*/
#define IF_FILTER_GAUSS	109 /* filter_gaussian	*/
#define IF_FILTER_GAUSS_VREG	360 /* filter_gaussian_vreg	*/

/*	FUNCTION DESCRIPTORS (continued)	*/
#define IF_FILTER_INVERSE	110 /* filter_inverse	*/
#define IF_FILTER_INVERSE_VREG	361 /* filter_inverse_vreg	*/
#define IF_FILTER_MATCHED	111 /* filter_matched	*/
#define IF_FILTER_MATCHED_VREG	362 /* filter_matched_vreg	*/
#define IF_FILTER_WIENER	112 /* filter_wiener	*/
#define IF_FILTER_WIENER_VREG	363 /* filter_wiener_vreg	*/
#define IF_FILTER_ZONAL	113 /* filter_zonal	*/
#define IF_FILTER_ZONAL_VREG	364 /* filter_zonal_vreg	*/
#define IF_FILTERING_HOMOMORPHIC	114 /* filtering_homomorphic	*/
#define IF_FILTERING_HOMOMORPHIC_VREG	365 /* filtering_homomorphic_vreg	*/
#define IF_FILTERING_LIN	115 /* filtering_linear	*/
#define IF_FILTERING_LIN_VREG	366 /* filtering_linear_vreg	*/
#define IF_FILTERING_MEDIAN	116 /* filtering_median	*/
#define IF_FILTERING_MEDIAN_VREG	367 /* filtering_median_vreg	*/
#define IF_FILTERING_PSEUDOMEDIAN	117 /* filtering_pseudomedian	*/
#define IF_FILTERING_PSEUDOMEDIAN_VREG	368 /* filtering_pseudomedian_vreg	*/
#define IF_FILTERING_RANK_ORDER	118 /* filtering_rank_order	*/
#define IF_FILTERING_RANK_ORDER_VREG	369 /* filtering_rank_order_vreg	*/
#define IF_FLIP_SPIN_TRANSPOSE	119 /* flip_spin_transpose	*/
#define IF_FLIP_SPIN_TRANSPOSE_ROI	120 /* flip_spin_transpose_roi	*/
#define IF_FLIP_SPIN_TRANSPOSE_ROI_VREG	370 /* flip_spin_transpose_roi_vreg	*/
#define IF_FLIP_SPIN_TRANSPOSE_VREG	371 /* flip_spin_transpose_vreg	*/
#define IF_GAMMA_CORRECTION	121 /* gamma_correction	*/
#define IF_GAMMA_CORRECTION_VREG	372 /* gamma_correction_vreg	*/
#define IF_GET_COLOUR_PIXEL	122 /* get_colour_pixel	*/
#define IF_GET_COLOUR_PIXEL_VREG	373 /* get_colour_pixel_vreg	*/
#define IF_GET_PIXEL	123 /* get_pixel	*/
#define IF_GET_PIXEL_ROI	124 /* get_pixel_roi	*/
#define IF_GET_PIXEL_ROI_VREG	374 /* get_pixel_roi_vreg	*/
#define IF_GET_PIXEL_VREG	375 /* get_pixel_vreg	*/
#define IF_GET_PIXEL_ARRAY	125 /* get_pixel_array	*/
#define IF_GET_PIXEL_ARRAY_ROI	126 /* get_pixel_array_roi	*/
#define IF_GET_PIXEL_ARRAY_ROI_VREG	376 /* get_pixel_array_roi_vreg	*/
#define IF_GET_PIXEL_ARRAY_VREG	377 /* get_pixel_array_vreg	*/
#define IF_GET_PIXEL_RECORD	127 /* get_pixel_record	*/
#define IF_GET_PIXEL_RECORD_ROI	128 /* get_pixel_record_roi	*/
#define IF_GET_PIXEL_RECORD_ROI_VREG	378 /* get_pixel_record_roi_vreg	*/
#define IF_GET_PIXEL_RECORD_VREG	379 /* get_pixel_record_vreg	*/
#define IF_HIST_MODIFICATION	129 /* histogram_modification	*/
#define IF_HIST_MODIFICATION_VREG	380 /* histogram_modification_vreg	*/
#define IF_HIST_SCALING	130 /* histogram_scaling	*/
#define IF_HIST_SCALING_VREG	381 /* histogram_scaling_vreg	*/
#define IF_HIST_SHAPE_1D	131 /* histogram_shape_1d	*/
#define IF_HIST_SHAPE_2D	132 /* histogram_shape_2d	*/
#define IF_HIST_1D	133 /* histogram_1d	*/
#define IF_HIST_1D_VREG	382 /* histogram_1d_vreg	*/
#define IF_HIST_2D	134 /* histogram_2d	*/
#define IF_HIST_2D_VREG	383 /* histogram_2d_vreg	*/
#define IF_HIT_MISS	135 /* hit_or_miss	*/
#define IF_HIT_MISS_ROI	136 /* hit_or_miss_roi	*/

/*	FUNCTION DESCRIPTORS (continued)	*/
#define IF_HIT_MISS_ROI_VREG	384 /* hit_or_miss_roi_vreg	*/
#define IF_HIT_MISS_VREG	385 /* hit_or_miss_vreg	*/
#define IF_HOUGH_TRANSFORM	137 /* hough_transform	*/
#define IF_HOUGH_TRANSFORM_VREG	386 /* hough_transform_vreg	*/
#define IF_IMAGE_BAR_CHART	138 /* image_bar_chart	*/
#define IF_IMAGE_BAR_CHART_VREG	387 /* image_bar_chart_vreg	*/
#define IF_IMAGE_CONSTANT	139 /* image_constant	*/
#define IF_IMAGE_CONSTANT_VREG	388 /* image_constant_vreg	*/
#define IF_IMAGE_GAUSS	140 /* image_gaussian	*/
#define IF_IMAGE_GAUSS_VREG	389 /* image_gaussian_vreg	*/
#define IF_IMAGE_GREY_SCALE	141 /* image_grey_scale	*/
#define IF_IMAGE_GREY_SCALE_VREG	390 /* image_grey_scale_vreg	*/
#define IF_IMAGE_RANDOM_NUMBER	142 /* image_random_number	*/
#define IF_IMAGE_RANDOM_NUMBER_VREG	391 /* image_random_number_vreg	*/
#define IF_IMPORT_HIST	143 /* import_histogram	*/
#define IF_IMPORT_HIST_VREG	392 /* import_histogram_vreg	*/
#define IF_IMPORT_IMAGE	144 /* import_image	*/
#define IF_IMPORT_IMAGE_VREG	393 /* import_image_vreg	*/
#define IF_IMPORT_LUT	145 /* import_lut	*/
#define IF_IMPORT_LUT_VREG	394 /* import_lut_vreg	*/
#define IF_IMPORT_MATRIX	146 /* import_matrix	*/
#define IF_IMPORT_MATRIX_VREG	395 /* import_matrix_vreg	*/
#define IF_IMPORT_NBHOOD_ARRAY	147 /* import_neighbourhood_array	*/
#define IF_IMPORT_NBHOOD_ARRAY_VREG	396 /* import_neighbourhood_array_vreg	*/
#define IF_IMPORT_ROI_ARRAY	148 /* import_roi_array	*/
#define IF_IMPORT_ROI_ARRAY_VREG	397 /* import_roi_array_vreg	*/
#define IF_IMPORT_STATIC_ARRAY	149 /* import_static_array	*/
#define IF_IMPORT_STATIC_ARRAY_VREG	398 /* import_static_array_vreg	*/
#define IF_IMPORT_TUPLE	150 /* import_tuple	*/
#define IF_IMPORT_TUPLE_VREG	399 /* import_tuple_vreg	*/
#define IF_IMPORT_VALUE_BOUNDS	151 /* import_value_bounds	*/
#define IF_IMPORT_VALUE_BOUNDS_VREG	400 /* import_value_bounds_vreg	*/
#define IF_IMPULSE_BOXCAR	152 /* impulse_boxcar	*/
#define IF_IMPULSE_BOXCAR_VREG	401 /* impulse_boxcar_vreg	*/
#define IF_IMPULSE_DERIV_GAUSS	153 /* impulse_derivative_of_gaussian	*/
#define IF_IMPULSE_DERIV_GAUSS_VREG	402 /* impulse_derivative_of_gaussian_vreg	*/
#define IF_IMPULSE_DIFF_GAUSS	154 /* impulse_difference_of_gaussians	*/
#define IF_IMPULSE_DIFF_GAUSS_VREG	403 /* impulse_difference_of_gaussians_vreg	*/
#define IF_IMPULSE_ELLIPTICAL	155 /* impulse_elliptical	*/
#define IF_IMPULSE_ELLIPTICAL_VREG	404 /* impulse_elliptical_vreg	*/
#define IF_IMPULSE_GAUSS	156 /* impulse_gaussian	*/
#define IF_IMPULSE_GAUSS_VREG	405 /* impulse_gaussian_vreg	*/
#define IF_IMPULSE_LAPLACE_GAUSS	157 /* impulse_laplacian_of_gaussian	*/
#define IF_IMPULSE_LAPLACE_GAUSS_VREG	406 /* impulse_laplacian_of_gaussian_vreg	*/
#define IF_IMPULSE_PYRAMID	158 /* impulse_pyramid	*/
#define IF_IMPULSE_PYRAMID_VREG	407 /* impulse_pyramid_vreg	*/
#define IF_IMPULSE_RECTANGULAR	159 /* impulse_rectangular	*/
#define IF_IMPULSE_RECTANGULAR_VREG	408 /* impulse_rectangular_vreg	*/
#define IF_IMPULSE_SINC	160 /* impulse_sinc	*/
#define IF_IMPULSE_SINC_VREG	409 /* impulse_sinc_vreg	*/

/*	FUNCTION DESCRIPTORS (continued)	*/
#define IF_INPUT_OBJECT	161 /* input_object	*/
#define IF_INPUT_OBJECT_VREG	410 /* input_object_vreg	*/
#define IF_INQUIRE_CHAIN_ENVIRONMENT	162 /* inquire_chain_environment	*/
#define IF_INQUIRE_CHAIN_STATUS	163 /* inquire_chain_status	*/
#define IF_INQUIRE_ELEMENTS	164 /* inquire_elements	*/
#define IF_INQUIRE_IMAGE	165 /* inquire_image	*/
#define IF_INQUIRE_INDEX_ASSIGNMENT	166 /* inquire_index_assignment	*/
#define IF_INQUIRE_NON_IMAGE_OBJECT	167 /* inquire_non_image_object	*/
#define IF_INQUIRE_PIKS_IMPL	168 /* inquire_piks_implementation	*/
#define IF_INQUIRE_PIKS_STATUS	169 /* inquire_piks_status	*/
#define IF_INQUIRE_REPOSITORY	170 /* inquire_repository	*/
#define IF_INQUIRE_RESAMPLING	171 /* inquire_resampling	*/
#define IF_INSERT_PIXEL_PLANE	172 /* insert_pixel_plane	*/
#define IF_INSERT_PIXEL_PLANE_ROI	173 /* insert_pixel_plane_roi	*/
#define IF_INSERT_PIXEL_PLANE_ROI_VREG	411 /* insert_pixel_plane_roi_vreg	*/
#define IF_INSERT_PIXEL_PLANE_VREG	412 /* insert_pixel_plane_vreg	*/
#define IF_LABEL_OBJECTS	174 /* label_objects	*/
#define IF_LABEL_OBJECTS_VREG	413 /* label_objects_vreg	*/
#define IF_LAWS_TEXTURE_FEATURES	175 /* laws_texture_features	*/
#define IF_LAWS_TEXTURE_FEATURES_VREG	414 /* laws_texture_features_vreg	*/
#define IF_LEVEL_SLICE	176 /* level_slice	*/
#define IF_LEVEL_SLICE_VREG	415 /* level_slice_vreg	*/
#define IF_LINE_PROFILE	177 /* line_profile	*/
#define IF_LINE_PROFILE_VREG	416 /* line_profile_vreg	*/
#define IF_LOOKUP	178 /* lookup	*/
#define IF_LOOKUP_VREG	417 /* lookup_vreg	*/
#define IF_LOOKUP_INTERP	179 /* lookup_interpolated	*/
#define IF_LOOKUP_INTERP_VREG	418 /* lookup_interpolated_vreg	*/
#define IF_LUM_GENERATION	180 /* luminance_generation	*/
#define IF_LUM_GENERATION_VREG	419 /* luminance_generation_vreg	*/
#define IF_MARKOV_SPECTRUM	181 /* markov_spectrum	*/
#define IF_MARKOV_SPECTRUM_VREG	420 /* markov_spectrum_vreg	*/
#define IF_MOMENTS	182 /* moments	*/
#define IF_MOMENTS_VREG	421 /* moments_vreg	*/
#define IF_MONADIC_ARITH	183 /* monadic_arithmetic	*/
#define IF_MONADIC_ARITH_VREG	422 /* monadic_arithmetic_vreg	*/
#define IF_MONADIC_COMPLEX	184 /* monadic_complex	*/
#define IF_MONADIC_COMPLEX_VREG	423 /* monadic_complex_vreg	*/
#define IF_MONADIC_LOGICAL	185 /* monadic_logical	*/
#define IF_MONADIC_LOGICAL_ROI	186 /* monadic_logical_roi	*/
#define IF_MONADIC_LOGICAL_ROI_VREG	424 /* monadic_logical_roi_vreg	*/
#define IF_MONADIC_LOGICAL_VREG	425 /* monadic_logical_vreg	*/
#define IF_MORPHIC_PROCESSOR	187 /* morphic_processor	*/
#define IF_MORPHIC_PROCESSOR_ROI	188 /* morphic_processor_roi	*/
#define IF_MORPHIC_PROCESSOR_ROI_VREG	426 /* morphic_processor_roi_vreg	*/
#define IF_MORPHIC_PROCESSOR_VREG	427 /* morphic_processor_vreg	*/
#define IF_MORPHOLOGY	189 /* morphology	*/
#define IF_MORPHOLOGY_ROI	190 /* morphology_roi	*/
#define IF_MORPHOLOGY_ROI_VREG	428 /* morphology_roi_vreg	*/
#define IF_MORPHOLOGY_VREG	429 /* morphology_vreg	*/

/*	FUNCTION DESCRIPTORS (continued)	*/
#define IF_NBR_COUNT	191 /* neighbour_count	*/
#define IF_NBR_COUNT_VREG	430 /* neighbour_count_vreg	*/
#define IF_NOISE_COMBINATION	192 /* noise_combination	*/
#define IF_NOISE_COMBINATION_VREG	431 /* noise_combination_vreg	*/
#define IF_OPEN_CLOSE	193 /* open_close	*/
#define IF_OPEN_CLOSE_ROI	194 /* open_close_roi	*/
#define IF_OPEN_CLOSE_ROI_VREG	432 /* open_close_roi_vreg	*/
#define IF_OPEN_CLOSE_VREG	433 /* open_close_vreg	*/
#define IF_OPEN_PIKS	195 /* open_piks	*/
#define IF_OUTLIER_REMOVAL	196 /* outlier_removal	*/
#define IF_OUTLIER_REMOVAL_VREG	434 /* outlier_removal_vreg	*/
#define IF_OUTPUT_OBJECT	197 /* output_object	*/
#define IF_PAINT_PIXELS	198 /* paint_pixels	*/
#define IF_PAINT_PIXELS_VREG	435 /* paint_pixels_vreg	*/
#define IF_PERIMETER_CODE_GENERATOR	199 /* perimeter_code_generator	*/
#define IF_POLAR_TO_CARTESIAN	200 /* polar_to_cartesian	*/
#define IF_POLAR_TO_CARTESIAN_VREG	436 /* polar_to_cartesian_vreg	*/
#define IF_POWER_LAW_SCALING	201 /* power_law_scaling	*/
#define IF_POWER_LAW_SCALING_VREG	437 /* power_law_scaling_vreg	*/
#define IF_PSEUDOCOLOUR	202 /* pseudocolour	*/
#define IF_PSEUDOCOLOUR_VREG	438 /* pseudocolour_vreg	*/
#define IF_PUT_COLOUR_PIXEL	203 /* put_colour_pixel	*/
#define IF_PUT_COLOUR_PIXEL_VREG	439 /* put_colour_pixel_vreg	*/
#define IF_PUT_PIXEL	204 /* put_pixel	*/
#define IF_PUT_PIXEL_ROI	205 /* put_pixel_roi	*/
#define IF_PUT_PIXEL_ROI_VREG	440 /* put_pixel_roi_vreg	*/
#define IF_PUT_PIXEL_VREG	441 /* put_pixel_vreg	*/
#define IF_PUT_PIXEL_ARRAY	206 /* put_pixel_array	*/
#define IF_PUT_PIXEL_ARRAY_ROI	207 /* put_pixel_array_roi	*/
#define IF_PUT_PIXEL_ARRAY_ROI_VREG	442 /* put_pixel_array_roi_vreg	*/
#define IF_PUT_PIXEL_ARRAY_VREG	443 /* put_pixel_array_vreg	*/
#define IF_PUT_PIXEL_RECORD	208 /* put_pixel_record	*/
#define IF_PUT_PIXEL_RECORD_ROI	209 /* put_pixel_record_roi	*/
#define IF_PUT_PIXEL_RECORD_ROI_VREG	444 /* put_pixel_record_roi_vreg	*/
#define IF_PUT_PIXEL_RECORD_VREG	445 /* put_pixel_record_vreg	*/
#define IF_RESCALE	210 /* rescale	*/
#define IF_RESCALE_ROI	211 /* rescale_roi	*/
#define IF_RESCALE_ROI_VREG	446 /* rescale_roi_vreg	*/
#define IF_RESCALE_VREG	447 /* rescale_vreg	*/
#define IF_RESIZE	212 /* resize	*/
#define IF_RESIZE_ROI	213 /* resize_roi	*/
#define IF_RESIZE_ROI_VREG	448 /* resize_roi_vreg	*/
#define IF_RESIZE_VREG	449 /* resize_vreg	*/
#define IF_RETURN_REPOSITORY_ID	214 /* return_repository_id	*/
#define IF_RETURN_REPOSITORY_ID_VREG	450 /* return_repository_id_vreg	*/
#define IF_ROI_COORDINATE	215 /* roi_coordinate	*/
#define IF_ROI_COORDINATE_VREG	451 /* roi_coordinate_vreg	*/
#define IF_ROI_ELLIPTICAL	216 /* roi_elliptical	*/
#define IF_ROI_ELLIPTICAL_VREG	452 /* roi_elliptical_vreg	*/
#define IF_ROI_POLYGON	217 /* roi_polygon	*/

/*	FUNCTION DESCRIPTORS (continued)	*/
#define IF_ROI_POLYGON_VREG	453 /* roi_polygon_vreg	*/
#define IF_ROI_RECTANGULAR	218 /* roi_rectangular	*/
#define IF_ROI_RECTANGULAR_VREG	454 /* roi_rectangular_vreg	*/
#define IF_ROTATE	219 /* rotate	*/
#define IF_ROTATE_ROI	220 /* rotate_roi	*/
#define IF_ROTATE_ROI_VREG	455 /* rotate_roi_vreg	*/
#define IF_ROTATE_VREG	456 /* rotate_vreg	*/
#define IF_RUBBER_BAND_SCALING	221 /* rubber_band_scaling	*/
#define IF_RUBBER_BAND_SCALING_VREG	457 /* rubber_band_scaling_vreg	*/
#define IF_SEQ_AVERAGE	222 /* sequence_average	*/
#define IF_SEQ_AVERAGE_VREG	458 /* sequence_average_vreg	*/
#define IF_SEQ_KARHUNEN_LOEVE	223 /* sequence_karhunen_loeve	*/
#define IF_SEQ_KARHUNEN_LOEVE_VREG	459 /* sequence_karhunen_loeve_vreg	*/
#define IF_SEQ_RUNNING_MEASURES	224 /* sequence_running_measures	*/
#define IF_SEQ_RUNNING_MEASURES_VREG	460 /* sequence_running_measures_vreg	*/
#define IF_SET_ERROR_HANDLER	269 /* set_error_handler	*/
#define IF_SET_GLOBALS	225 /* set_globals	*/
#define IF_SET_GLOBALS_VREG	461 /* set_globals_vreg	*/
#define IF_SET_IMAGE_ATTRS	226 /* set_image_attributes	*/
#define IF_SET_IMAGE_ATTRS_VREG	462 /* set_image_attributes_vreg	*/
#define IF_SET_INDEX_ASSIGNMENT	227 /* set_index_assignment	*/
#define IF_SET_INDEX_ASSIGNMENT_VREG	463 /* set_index_assignment_vreg	*/
#define IF_SHAPE_METRICS	228 /* shape_metrics	*/
#define IF_SPATIAL_MOMENTS_INVARIANT	229 /* spatial_moments_invariant	*/
#define IF_SPATIAL_MOMENTS_SCALED	230 /* spatial_moments_scaled	*/
#define IF_SPLIT_IMAGE	231 /* split_image	*/
#define IF_SPLIT_IMAGE_VREG	464 /* split_image_vreg	*/
#define IF_SUBSAMPLE	232 /* subsample	*/
#define IF_SUBSAMPLE_ROI	233 /* subsample_roi	*/
#define IF_SUBSAMPLE_ROI_VREG	465 /* subsample_roi_vreg	*/
#define IF_SUBSAMPLE_VREG	466 /* subsample_vreg	*/
#define IF_SYNC	234 /* synchronize	*/
#define IF_TEMPL_MATCH	235 /* template_match	*/
#define IF_TEMPL_MATCH_VREG	467 /* template_match_vreg	*/
#define IF_THRESHOLD	236 /* threshold	*/
#define IF_THRESHOLD_VREG	468 /* threshold_vreg	*/
#define IF_TRANSFORM_COSINE	237 /* transform_cosine	*/
#define IF_TRANSFORM_COSINE_VREG	469 /* transform_cosine_vreg	*/
#define IF_TRANSFORM_FOURIER	238 /* transform_fourier	*/
#define IF_TRANSFORM_FOURIER_VREG	470 /* transform_fourier_vreg	*/
#define IF_TRANSFORM_HADAMARD	239 /* transform_hadamard	*/
#define IF_TRANSFORM_HADAMARD_VREG	471 /* transform_hadamard_vreg	*/
#define IF_TRANSFORM_HARTLEY	240 /* transform_hartley	*/
#define IF_TRANSFORM_HARTLEY_VREG	472 /* transform_hartley_vreg	*/
#define IF_TRANSLATE	241 /* translate	*/
#define IF_TRANSLATE_ROI	242 /* translate_roi	*/
#define IF_TRANSLATE_ROI_VREG	473 /* translate_roi_vreg	*/
#define IF_TRANSLATE_VREG	474 /* translate_vreg	*/
#define IF_UNARY_INTEGER	243 /* unary_integer	*/
#define IF_UNARY_INTEGER_VREG	475 /* unary_integer_vreg	*/

/*	FUNCTION DESCRIPTORS (continued)	*/
#define IF_UNARY_REAL	244 /* unary_real	*/
#define IF_UNARY_REAL_VREG	476 /* unary_real_vreg	*/
#define IF_UNSHARP_MASK	245 /* unsharp_mask	*/
#define IF_UNSHARP_MASK_VREG	477 /* unsharp_mask_vreg	*/
#define IF_VALUE_BOUNDS	246 /* value_bounds	*/
#define IF_VALUE_BOUNDS_VREG	478 /* value_bounds_vreg	*/
#define IF_VREG_ALTER	247 /* vreg_alter	*/
#define IF_VREG_ALTER_VREG	479 /* vreg_alter_vreg	*/
#define IF_VREG_CLEAR	248 /* vreg_clear	*/
#define IF_VREG_CLEAR_VREG	480 /* vreg_clear_vreg	*/
#define IF_VREG_CONDITIONAL	249 /* vreg_conditional	*/
#define IF_VREG_CONDITIONAL_VREG	481 /* vreg_conditional_vreg	*/
#define IF_VREG_COPY	250 /* vreg_copy	*/
#define IF_VREG_COPY_VREG	482 /* vreg_copy_vreg	*/
#define IF_VREG_DELETE	251 /* vreg_delete	*/
#define IF_VREG_DELETE_VREG	483 /* vreg_delete_vreg	*/
#define IF_VREG_GET	252 /* vreg_get	*/
#define IF_VREG_SET	253 /* vreg_set	*/
#define IF_VREG_SET_VREG	484 /* vreg_set_vreg	*/
#define IF_VREG_WAIT	254 /* vreg_wait	*/
#define IF_VREG_WAIT_VREG	485 /* vreg_wait_vreg	*/
#define IF_WALLIS_STAT_DIFF	255 /* wallis_statistical_differencing	*/
#define IF_WALLIS_STAT_DIFF_VREG	486 /* wallis_statistical_differencing_vreg	*/
#define IF_WARP_CONTROL_POINT	256 /* warp_control_point	*/
#define IF_WARP_CONTROL_POINT_ROI	257 /* warp_control_point_roi	*/
#define IF_WARP_CONTROL_POINT_ROI_VREG	487 /* warp_control_point_roi_vreg	*/
#define IF_WARP_CONTROL_POINT_VREG	488 /* warp_control_point_vreg	*/
#define IF_WARP_LUT	258 /* warp_lut	*/
#define IF_WARP_LUT_ROI	259 /* warp_lut_roi	*/
#define IF_WARP_LUT_ROI_VREG	489 /* warp_lut_roi_vreg	*/
#define IF_WARP_LUT_VREG	490 /* warp_lut_vreg	*/
#define IF_WARP_POLYNOMIAL	260 /* warp_polynomial	*/
#define IF_WARP_POLYNOMIAL_ROI	261 /* warp_polynomial_roi	*/
#define IF_WARP_POLYNOMIAL_ROI_VREG	491 /* warp_polynomial_roi_vreg	*/
#define IF_WARP_POLYNOMIAL_VREG	492 /* warp_polynomial_vreg	*/
#define IF_WINDOWING_FUNCTION	262 /* windowing_function	*/
#define IF_WINDOWING_FUNCTION_VREG	493 /* windowing_function_vreg	*/
#define IF_WINDOW_LEVEL	263 /* window_level	*/
#define IF_WINDOW_LEVEL_VREG	494 /* window_level_vreg	*/
#define IF_WINDOW_STATISTICS	264 /* window_statistics	*/
#define IF_WINDOW_STATISTICS_VREG	495 /* window_statistics_vreg	*/
#define IF_Z_MERGE	265 /* z_merge	*/
#define IF_Z_MERGE_VREG	496 /* z_merge_vreg	*/
#define IF_ZOOM	266 /* zoom	*/
#define IF_ZOOM_ROI	267 /* zoom_roi	*/
#define IF_ZOOM_ROI_VREG	497 /* zoom_roi_vreg	*/
#define IF_ZOOM_VREG	498 /* zoom_vreg	*/
#define IF_3D_SLICE	268 /* 3d_slice	*/
#define IF_3D_SLICE_VREG	499 /* 3d_slice_vreg	*/

```

/*                                FUNCTION DESCRIPTORS (concluded)                                */

#define IC_CREATE_UNBND_IMAGE_COPY      1000 /* create_unbounded_image_copy */
#define IC_GENERATE_2D_ROI_RECTANGULAR 1001 /* generate_2d_roi_rectangular */
#define IC_GENERATE_ROI_COORDINATE      1002 /* generate_roi_coordinate */
#define IC_GENERATE_ROI_ELLIPTICAL      1003 /* generate_roi_elliptical */
#define IC_GENERATE_ROI_POLYGON         1004 /* generate_roi_polygon */
#define IC_GENERATE_ROI_RECTANGULAR     1005 /* generate_roi_rectangular */
#define IC_GENERATE_ND_1_TUPLE          1006 /* generate_nd_1_tuple */
#define IC_GENERATE_ND_3_TUPLE          1007 /* generate_nd_3_tuple */
#define IC_GENERATE_ND_4_TUPLE          1008 /* generate_nd_4_tuple */
#define IC_GENERATE_ND_5_TUPLE          1009 /* generate_nd_5_tuple */
#define IC_GENERATE_RD_3_TUPLE          1010 /* generate_rd_3_tuple */
#define IC_GENERATE_RD_4_TUPLE          1011 /* generate_rd_4_tuple */
#define IC_GENERATE_RD_5_TUPLE          1012 /* generate_rd_5_tuple */
#define IC_GENERATE_SD_1_TUPLE          1013 /* generate_sd_1_tuple */
#define IC_GENERATE_SD_3_TUPLE          1014 /* generate_sd_3_tuple */
#define IC_GENERATE_SD_4_TUPLE          1015 /* generate_sd_4_tuple */
#define IC_GENERATE_SD_5_TUPLE          1016 /* generate_sd_5_tuple */
#define IC_PREPARE_COLOUR_IMAGE         1017 /* prepare_colour_image */
#define IC_PREPARE_MONOCHROME_IMAGE     1018 /* prepare_mon_image */
#define IC_PREPARE_2D_ROI_RECTANGULAR  1019 /* prepare_2d_roi_rectangular */
#define IC_PREPARE_ROI                  1020 /* prepare_roi */
#define IC_TILED_IMAGE_EXPORT           1021 /* tiled_image_export */
#define IC_TILED_IMAGE_IMPORT           1022 /* tiled_image_import */

/*                                ERROR MACROS                                */

/*                                < 000 Implementation-dependent errors.                                */

#define IE_NO_ERROR                      0 /* No error. */

#define IE_PARAMETER_DATA_TYPE           1 /* Element input data parameter is incorrect
data type.
Abort operation and issue warning */

#define IE_ENUM_PARAMETER_RANGE          2 /* Enumerated option parameter is out of
range. Abort operation and issue warning. */

#define IE_OPTION                        3 /* Signed integer option parameter is out of
range. Abort operation and issue warning. */

#define IE_INCOMPATIBLE_ATTRS            4 /* Incompatible attributes specified will not be
set. Warning will be issued and program
will continue. */

#define IE_ATTEMPT_ATTR_CHANGE           5 /* Attempt to change existing attributes not
accepted. Warning will be issued and
program will continue. */

#define IE_IMAGE_ID                     11 /* Image identifier is invalid.
Abort operation and issue warning. */

```



```

/*                                ERROR MACROS (continued)                                */

#define IE_NON_ID                12 /* Non-image identifier is invalid.
                                   Abort operation and issue warning. */

#define IE_ID_NOT_IMAGE          13 /* Identifier is not to an image.
                                   Abort operation and issue warning. */

#define IE_ID_NOT_NON            14 /* Identifier is not to a non-image.
                                   Abort operation and issue warning. */

#define IE_SOURCE_IMAGE_EMPTY    15 /* Source image does not contain data.
                                   Abort operation and issue warning. */

#define IE_SOURCE_NON_EMPTY      16 /* Source non-image object does not contain
                                   data. Abort operation and issue warning. */

#define IE_SUB_IMAGE_REASSIGN     17 /* Attempt to change image identifier of
                                   existing sub image.
                                   Abort operation and issue warning. */

#define IE_SOURCE_DEST_IMAGE_SIZE 18 /* Mismatch between source and destination
                                   image dimension sizes. Abort operation and
                                   issue warning. */

#define IE_SOURCE_DEST_NON_SIZE  19 /* Mismatch between source and destination
                                   non_image dimension sizes. Abort operation
                                   and issue warning. */

#define IE_SOURCE_IMAGE_INPUT_STRUCTURE 21 /* Mismatch between source image and
                                   element input structure.
                                   Abort operation and issue warning. */

#define IE_SOURCE_NON_INPUT_STRUCTURE 22 /* Mismatch between source non-image and
                                   element input structure.
                                   Abort operation and issue warning. */

#define IE_DEST_IMAGE_OUTPUT_STRUCTURE 23 /* Mismatch between destination image and
                                   element output structure.
                                   Abort operation and issue warning. */

#define IE_DEST_NON_OUTPUT_STRUCTURE 24 /* Mismatch between destination non-image
                                   and element output structure.
                                   Abort operation and issue warning. */

#define IE_SOURCE_DEST_IMAGE_STRUCTURE 25 /* Mismatch between source image and
                                   destination image structure.
                                   Abort operation and issue warning. */

#define IE_SOURCE_DEST_NON_STRUCTURE 26 /* Mismatch between source non-image and
                                   destination non-image structures.
                                   Abort operation and issue warning.

```

/* ERROR MACROS (continued) */		
#define IE_SOURCE_IMAGE_STRUCTURES	27	/*Mismatch between multiple source image structures. Abort operation and issue warning. */
#define IE_SOURCE_NON_STRUCTURES	28	/*Mismatch between multiple source non-image structures. Abort operation and issue warning. */
#define IE_DEST_IMAGE_STRUCTURES	29	/*Mismatch between multiple destination image structures. Abort operation and issue warning. */
#define IE_DEST_NON_STRUCTURES	30	/*Mismatch between multiple destination non-image structures. Abort operation and issue warning. */
#define IE_SOURCE_IMAGE_DEST_NON	31	/*Mismatch between source image and destination non-image structure. Abort operation and issue warning. */
#define IE_SOURCE_NON_DEST_IMAGE	32	/*Mismatch between source non-image and destination image structure. Abort operation and issue warning. */
#define IE_SOURCE_IMAGE_DEST_NON_TYPE	40	/*Mismatch between source image and destination non-image data type. Abort operation and issue warning. */
#define IE_SOURCE_IMAGE_INPUT_TYPE	41	/*Mismatch between source image and element input data type. Abort operation and issue warning. */
#define IE_SOURCE_NON_INPUT_TYPE	42	/*Mismatch between source non-image and element input data type. Abort operation and issue warning. */
#define IE_DEST_IMAGE_OUTPUT_TYPE	43	/*Mismatch between destination image and element output data type. Abort operation and issue warning. */
#define IE_DEST_NON_OUTPUT_TYPE	44	/*Mismatch between destination non-image and element output data type. Abort operation and issue warning. */
#define IE_SOURCE_DEST_IMAGE_TYPE	45	/*Mismatch between source image and destination image data types. Abort operation and issue warning. */
#define IE_SOURCE_DEST_NON_TYPE	46	/*Mismatch between source non-image and destination non-image data types. Abort operation and issue warning. */

```

/*                                ERROR MACROS (continued)                                */

#define IE_SOURCE_IMAGE_TYPES      47 /*Mismatch between multiple source image
                                     data types.
                                     Abort operation and issue warning. */

#define IE_SOURCE_NON_TYPES        48 /*Mismatch between multiple source
                                     non-image data types.
                                     Abort operation and issue warning. */

#define IE_DEST_IMAGE_TYPES        49 /*Mismatch between multiple destination
                                     image data types.
                                     Abort operation and issue warning. */

#define IE_DEST_NON_TYPES          50 /*Mismatch between multiple destination
                                     non-image data types.
                                     Abort operation and issue warning. */

#define IE_DEST_IMAGE_NON_TYPES    51 /*Mismatch between destination image and
                                     destination non-image data types.
                                     Abort operation and issue warning. */

#define IE_SOURCE_IMAGE_NON_TYPE    52 /*Mismatch between source image and source
                                     non-image data types.
                                     Abort operation and issue warning. */

#define IE_DEST_IMAGE_SOURCE_NON_TYPE 53 /*Mismatch between destination image and
                                     source non-image data type.
                                     Abort operation and issue warning. */

#define IE_SOURCE_IMAGE_INPUT_COLOUR 54 /*Mismatch between source image and
                                     element input colour space.
                                     Abort operation and issue warning. */

#define IE_DEST_IMAGE_OUTPUT_COLOUR 55 /*Mismatch between destination image and
                                     element output colour space.
                                     Abort operation and issue warning. */

#define IE_SOURCE_DEST_IMAGE_COLOUR 56 /*Mismatch between source image and
                                     destination image colour space.
                                     Abort operation and issue warning. */

#define IE_HETEROGENEOUS_BANDS    61 /*Element does not support heterogeneous
                                     bands. Abort operation and issue warning. */

#define IE_ELEMENT_NOT_CHAINABLE   62 /*Element is not chainable.
                                     Abort operation and issue warning. */

#define IE_INT_EXCESS              71 /*Specified integer size exceeds
                                     implementation integer size.
                                     Abort operation and issue warning. */

```



```

/*                                ERROR MACROS (continued)                                */

#define IE_REAL_EXCESS              72 /* Specified real arithmetic size exceeds
                                         implementation real arithmetic size.
                                         Abort operation and issue warning. */

#define IE_IMAGE_EXCESS            73 /* Specified image size exceeds
                                         implementation image size.
                                         Abort operation and issue warning. */

#define IE_ARRAY_EXCESS            74 /* Specified array size exceeds implementation
                                         array size.
                                         Abort operation and issue warning. */

#define IE_INPUT_PARAMETER_RANGE  101 /* Element input data parameter is out of
                                         range. Abort operation and issue warning. */

#define IE_UPPER_LOWER_PARAMETER  102 /* Upper parameter bound is smaller than
                                         lower parameter bound.
                                         Abort operation and issue warning. */

#define IE_SOURCE_ZERO_LESS       103 /* Source image contains pixel of zero or
                                         negative amplitude.
                                         Abort operation and issue warning. */

#define IE_SOURCE_NEGATIVE        104 /* Source image contains pixel of negative
                                         amplitude.
                                         Abort operation and issue warning. */

#define IE_ARRAY_ROW_SIZE         111 /* Array row dimension is not a positive
                                         integer.
                                         Abort operation and issue warning. */

#define IE_ARRAY_COLUMN_SIZE      112 /* Array column dimension is not a positive
                                         integer.
                                         Abort operation and issue warning. */

#define IE_ARRAY_ROW_NOT_ODD      113 /* Array row dimension is not a positive odd
                                         integer.
                                         Abort operation and issue warning. */

#define IE_ARRAY_COLUMN_NOT_ODD   114 /* Array column dimension is not a positive
                                         odd integer.
                                         Abort operation and issue warning. */

#define IE_ARRAY_ROW_NOT_EVEN     115 /* Array row dimension is not a positive even
                                         integer.
                                         Abort operation and issue warning. */

#define IE_ARRAY_COLUMN_NOT_EVEN  116 /* Array column dimension is not a positive
                                         even integer.
                                         Abort operation and issue warning. */

```

```

/*                                ERROR MACROS (continued)                                */

#define IE_TRUE_MASK_NOT_ODD      117 /*Number of TRUE state mask array
                                         elements is not a positive odd integer.
                                         Abort operation and issue warning.      */

#define IE_LUT_SEGMENT_RANGE      121 /*Lookup table segment value is out of range.
                                         Abort operation and issue warning.      */

#define IE_LUT_INDEX_RANGE        122 /*Lookup table index is out of range.
                                         Abort operation and issue warning.      */

#define IE_LUT_SIZE                123 /*Lookup table size is insufficient.
                                         Abort operation and issue warning.      */

#define IE_SEED_NOT_FALSE         131 /*Seed pixel is not in FALSE state.
                                         Abort operation and issue warning.      */

#define IE_BLOCK_SIZE_NOT_POWER_2 132 /*Block size dimensions not a power of two.
                                         Abort operation and issue warning.      */

#define IE_SIZE_NOT_MULTIPLE      133 /*j-k pixel plane sizes are not integer
                                         multiples of the block size.
                                         Abort operation and issue warning.      */

#define IE_COVARIANCE_NOT_SYMMETRIC 134 /*Covariance matrix is not symmetric.
                                         Abort operation and issue warning.      */

#define IE_INTERNAL_TYPE_CONV      151 /*Illegal internal data type conversion.
                                         Abort operation and issue warning.      */

#define IE_IMPORT_TYPE_CONV        152 /*Illegal import data type conversion.
                                         Abort operation and issue warning.      */

#define IE_EXPORT_TYPE_CONV        153 /*Illegal export data type conversion.
                                         Abort operation and issue warning.      */

#define IE_ID_ARRAYS_UNEQUAL      161 /*Composite image identifier arrays are of
                                         different length.
                                         Abort operation and issue warning.      */

#define IE_ID_LISTS_UNEQUAL       162 /*Composite image identifier lists are of
                                         different length.
                                         Abort operation and issue warning.      */

#define IE_ID_RECORD_UNEQUAL      163 /*Composite image identifier records are of
                                         different length.
                                         Abort operation and issue warning.      */

#define IE_ID_LIST_EMPTY          164 /*Composite image identifier list is empty.
                                         Abort operation and issue warning.      */

```

```

/*                                ERROR MACROS (continued)                                */

#define IE_ID_RECORD_EMPTY          165 /* Composite image identifier record is empty.
                                         Abort operation and issue warning. */

#define IE_ID_RECORD_NAMES          166 /* Composite image identifier record names
                                         are not unique.
                                         Abort operation and issue warning. */

#define IE_HIST_OVERFLOW            171 /* Histogram bin overflow.
                                         Abort operation and issue warning. */

#define IE_DIVISION_BY_ZERO         172 /* Attempted division by zero-valued constant.
                                         Abort operation and issue warning. */

#define IE_CHAIN_TYPE_INVALID       201 /* Chain type is invalid.
                                         Abort operation and issue warning. */

#define IE_CHAIN_ID_INVALID         202 /* Chain identifier is invalid.
                                         Abort operation and issue warning. */

#define IE_CHAIN_ID_NOT_IMAGE       203 /* Chain identifier is not to an image.
                                         Abort operation and issue warning. */

#define IE_CHAIN_RESOURCE_PARAMETER 204 /* Chain resource type parameter invalid.
                                         Abort operation and issue warning. */

#define IE_CHAIN_PARAMETER          205 /* Chain parameter is invalid.
                                         Abort operation and issue warning. */

#define IE_CHAIN_EXECUTING          206 /* Chain already executing.
                                         Abort operation and issue warning. */

#define IE_CHAIN_DELETED            207 /* Chain already deleted.
                                         Abort operation and issue warning. */

#define IE_CHAIN_NOT_EXECUTING      208 /* Chain is not executing.
                                         Abort operation and issue warning. */

#define IE_CHAIN_INCOMPLETE         209 /* Chain definition incomplete.
                                         Abort operation and issue warning. */

#define IE_CHAIN_DEFINED            210 /* Chain previously defined.
                                         Abort operation and issue warning. */

#define IE_CHAIN_NOT_BUILD_STATE    211 /* Not in CHAIN_BUILD state.
                                         Abort operation and issue warning. */

#define IE_CHAIN_EXECUTE            212 /* Attempt to begin a chain in
                                         CHAIN_EXECUTE state.
                                         Abort operation and issue warning. */

```

/*	ERROR MACROS (continued)	*/
#define IE_CHAIN_SELECTOR	213 /* Selector less than one or greater than number of chain elements. Abort operation and issue warning.	*/
#define IE_CHAIN_MISMATCH	214 /* Next chainable PIKS element does not match selected element built into chain. Abort operation and issue warning.	*/
#define IE_CHAIN_ASYNC_EXECUTING	215 /* Chain is presently on the asynchronous queue or is in execution. Abort operation and issue warning.	*/
#define IE_VREG_SOURCE_DEST_TYPES	251 /* Mismatch between source and destination virtual register data types. Abort operation and issue warning.	*/
#define IE_VREG_NOT_SET	252 /* Virtual register is not in VREG_SET state. Abort operation and issue warning.	*/
#define IE_PIKS_OPEN	301 /* State is PIKS_OPEN when PIKS_CLOSED is expected. Log the error, abort operation, invoke <i>close_piks_emergency</i> .	*/
#define IE_PIKS_CLOSED	302 /* State is PIKS_CLOSED when PIKS_OPEN is expected. Return to the calling PIKS element to resume execution with the error flag parameter set to 'error' whether or not a prior error has been recorded.	*/
#define IE_PIKS_INITIALIZATION	303 /* PIKS initialization failure. Log the error, abort operation, invoke <i>close_piks_emergency</i> .	*/
#define IE_PIKS_CLOSURE	304 /* PIKS closure failure. Abort operation and issue warning.	*/
#define IE_NO_PARAMETER_DEFAULT	305 /* Unsupported default state for parameter. Log the error, abort operation, invoke <i>close_piks_emergency</i> .	*/
#define IE_DEFAULT	306 /* Invalid default state. Abort operation and issue warning.	*/
#define IE_ELEMENT_NON_PROFILE	307 /* Element is not in implementation conformance profile. Abort operation and issue warning.	*/
#define IE_POINTER	1001 /* Pointer out of range.	*/
#define IE_ID	1002 /* Identifier not assigned.	*/

```

/*                                ERROR MACROS (concluded)                                */

#define IE_STORE                    1003 /* Storage overflow.                        */

#define IE_INPUT_OUTPUT            1004 /* Input/output failure.                    */

#define IE_ALTERATION              1005 /* Attempt to alter a permanent object.    */

#define IE_DEALLOCATION             1006 /* Attempt to deallocate a permanent object.*/

#define IE_START_POINT_INVALID     2201 /* Start index is out of range of data
                                     available.                                */

#define IE_ENUM_TYPE_INVALID       2202 /* Enumerated type out of range.          */

#define IE_REPOSITORY_UNAVAILABLE  2203 /* Identifier points to non-existent repository
                                     object.                                */

/*                                ARRAY LENGTHS                                */

#define IL_PROFILE                  15  /* conformance profile label string length */
#define IL_REAL                     3  /* implementation method of RD data type string length */

/*                                REPOSITORY NAMES                                */

/*                                impulse response function arrays                                */

#define IR_BOXCAR_5x5_HORIZONTAL   90  /* boxcar 5x5 horizontal                    */
#define IR_BOXCAR_5x5_VERTICAL    91  /* boxcar 5x5 vertical                      */
#define IR_CHEBYSHEV_1             69  /* Chebyshev 1                             */
#define IR_CHEBYSHEV_2             70  /* Chebyshev 2                             */
#define IR_CHEBYSHEV_3             71  /* Chebyshev 3                             */
#define IR_CHEBYSHEV_4             72  /* Chebyshev 4                             */
#define IR_CHEBYSHEV_5             73  /* Chebyshev 5                             */
#define IR_CHEBYSHEV_6             74  /* Chebyshev 6                             */
#define IR_CHEBYSHEV_7             75  /* Chebyshev 7                             */
#define IR_CHEBYSHEV_8             76  /* Chebyshev 8                             */
#define IR_CHEBYSHEV_9             77  /* Chebyshev 9                             */
#define IR_CORNER_NBR_AVERAGE      7  /* corner neighbour average                 */
#define IR_DECORRELATE             65  /* decorrelate                             */
#define IR_DELTA_3x3                1  /* delta 3x3                               */
#define IR_EIGHT_NBR_AVERAGE       5  /* eight neighbour average                  */
#define IR_FOUR_NBR_AVERAGE        6  /* four neighbour average                  */
#define IR_FREI_HORIZONTAL         21  /* Frei horizontal                         */
#define IR_FREI_VERTICAL           22  /* Frei vertical                           */
#define IR_HIGH_PASS_1_3x3         8  /* high pass 1 3x3                         */
#define IR_HIGH_PASS_2_3x3         9  /* high pass 2 3x3                         */
#define IR_HIGH_PASS_3_3x3        10  /* high pass 3 3x3                         */
#define IR_KIRSCH_EAST             31  /* Kirsch east                             */
#define IR_KIRSCH_NORTH            33  /* Kirsch north                           */
#define IR_KIRSCH_NORTHEAST        32  /* Kirsch northeast                       */
#define IR_KIRSCH_NORTHWEST        34  /* Kirsch northwest                       */
#define IR_KIRSCH_SOUTH            37  /* Kirsch south                           */

```

```

/*                                     impulse response function arrays (continued)                                     */

#define IR_KIRSCH_SOUTHEAST           38 /*Kirsch southeast */
#define IR_KIRSCH_SOUTHWEST          36 /*Kirsch southwest */
#define IR_KIRSCH_WEST                35 /*Kirsch west */
#define IR_LAPLACE_1                 66 /*Laplacian 1 */
#define IR_LAPLACE_2                 67 /*Laplacian 2 */
#define IR_LAPLACE_3                 68 /*Laplacian 3 */
#define IR_LAWS_1                     78 /*Laws 1 */
#define IR_LAWS_2                     79 /*Laws 2 */
#define IR_LAWS_3                     80 /*Laws 3 */
#define IR_LAWS_4                     81 /*Laws 4 */
#define IR_LAWS_5                     82 /*Laws 5 */
#define IR_LAWS_6                     83 /*Laws 6 */
#define IR_LAWS_7                     84 /*Laws 7 */
#define IR_LAWS_8                     85 /*Laws 8 */
#define IR_LAWS_9                     86 /*Laws 9 */
#define IR_LOW_PASS_3x3               4  /*Low pass 3x3 */
#define IR_NEVATIA_0                  94 /*Nevatia 0 */
#define IR_NEVATIA_120                98 /*Nevatia 120 */
#define IR_NEVATIA_150                99 /*Nevatia 150 */
#define IR_NEVATIA_180               100 /*Nevatia 180 */
#define IR_NEVATIA_210               101 /*Nevatia 210 */
#define IR_NEVATIA_240               102 /*Nevatia 240 */
#define IR_NEVATIA_270               103 /*Nevatia 270 */
#define IR_NEVATIA_30                 95 /*Nevatia 30 */
#define IR_NEVATIA_300               104 /*Nevatia 300 */
#define IR_NEVATIA_330               105 /*Nevatia 330 */
#define IR_NEVATIA_60                 96 /*Nevatia 60 */
#define IR_NEVATIA_90                 97 /*Nevatia 90 */
#define IR_PIXEL_DIFF_HORIZONTAL      11 /*pixel difference horizontal */
#define IR_PIXEL_DIFF_VERTICAL        12 /*pixel difference vertical */
#define IR_PIXEL_STACK                87 /*pixel stack */
#define IR_PREWITT_COMPASS_EAST        23 /*Prewitt compass east */
#define IR_PREWITT_COMPASS_NORTH       25 /*Prewitt compass north */
#define IR_PREWITT_COMPASS_NORTHEAST  24 /*Prewitt compass northeast */
#define IR_PREWITT_COMPASS_NORTHWEST  26 /*Prewitt compass northwest */
#define IR_PREWITT_COMPASS_SOUTH       29 /*Prewitt compass south */
#define IR_PREWITT_COMPASS_SOUTHEAST  30 /*Prewitt compass southeast */
#define IR_PREWITT_COMPASS_SOUTHWEST  28 /*Prewitt compass southwest */
#define IR_PREWITT_COMPASS_WEST        27 /*Prewitt compass west */
#define IR_PREWITT_HORIZONTAL          17 /*Prewitt horizontal */
#define IR_PREWITT_VERTICAL            18 /*Prewitt vertical */
#define IR_PYRAMID_3x3                 3  /*pyramid 3x3 */
#define IR_PYRAMID_5x5                89 /*pyramid 5x5 */
#define IR_ROBERTS_3_LEVEL_EAST        39 /*Roberts 3 level east */
#define IR_ROBERTS_3_LEVEL_NORTH       41 /*Roberts 3 level north */
#define IR_ROBERTS_3_LEVEL_NORTHEAST  40 /*Roberts 3 level northeast */
#define IR_ROBERTS_3_LEVEL_NORTHWEST  42 /*Roberts 3 level northwest */
#define IR_ROBERTS_3_LEVEL_SOUTH       45 /*Roberts 3 level south */
#define IR_ROBERTS_3_LEVEL_SOUTHEAST  46 /*Roberts 3 level southeast */
#define IR_ROBERTS_3_LEVEL_SOUTHWEST  44 /*Roberts 3 level southwest */

```

```

/*                                impulse response function arrays (concluded)                                */

#define IR_ROBERTS_3_LEVEL_WEST      43  /*Roberts 3 level west                                */
#define IR_ROBERTS_5_LEVEL_EAST      47  /*Roberts 5 level east                                */
#define IR_ROBERTS_5_LEVEL_NORTH     49  /*Roberts 5 level north                               */
#define IR_ROBERTS_5_LEVEL_NORTHEAST 48  /*Roberts 5 level northeast                           */
#define IR_ROBERTS_5_LEVEL_NORTHWEST 50  /*Roberts 5 level northwest                           */
#define IR_ROBERTS_5_LEVEL_SOUTH     53  /*Roberts 5 level south                               */
#define IR_ROBERTS_5_LEVEL_SOUTHEAST 54  /*Roberts 5 level southeast                           */
#define IR_ROBERTS_5_LEVEL_SOUTHWEST 52  /*Roberts 5 level southwest                           */
#define IR_ROBERTS_5_LEVEL_WEST      51  /*Roberts 5 level west                                */
#define IR_ROBERTS_HORIZONTAL        15  /*Roberts horizontal                                  */
#define IR_ROBERTS_VERTICAL          16  /*Roberts vertical                                    */
#define IR_SEP_PIXEL_DIFF_HORIZONTAL 13  /*separated pixel difference horizontal                */
#define IR_SEP_PIXEL_DIFF_VERTICAL   14  /*separated pixel difference vertical                  */
#define IR_SOBEL_HORIZONTAL          19  /*Sobel horizontal                                    */
#define IR_SOBEL_VERTICAL            20  /*Sobel vertical                                      */
#define IR_SPOT_4_CONNECTED           63  /*spot 4 connected                                    */
#define IR_SPOT_8_CONNECTED           64  /*spot 8 connected                                    */
#define IR_TRUNC_PYRAMID_5x5_HORIZONTAL 92 /*truncated pyramid 5x5 horizontal                    */
#define IR_TRUNC_PYRAMID_5x5_VERTICAL  93 /*truncated pyramid 5x5 vertical                      */
#define IR_UNIFORM_3x3                2   /*uniform 3x3                                          */
#define IR_UNIFORM_5x5                88  /*uniform 5x5                                          */
#define IR_UNWEIGHTED_LINE_1          55  /*unweighted line 1                                  */
#define IR_UNWEIGHTED_LINE_2          56  /*unweighted line 2                                  */
#define IR_UNWEIGHTED_LINE_3          57  /*unweighted line 3                                  */
#define IR_UNWEIGHTED_LINE_4          58  /*unweighted line 4                                  */
#define IR_WEIGHTED_LINE_1            59  /*weighted line 1                                     */
#define IR_WEIGHTED_LINE_2            60  /*weighted line 2                                     */
#define IR_WEIGHTED_LINE_3            61  /*weighted line 3                                     */
#define IR_WEIGHTED_LINE_4            62  /*weighted line 4                                     */

/*                                dither arrays                                */

#define IR_DITHER_2x2                 1   /*dither 2x2                                           */
#define IR_DITHER_4x4                 2   /*dither 4x4                                           */
#define IR_DITHER_8x8                 3   /*dither 8x8                                           */
#define IR_DITHER_16x16               4   /*dither 16x16                                        */

/*                                colour conversion matrices                                */

#define IR_A_LIN_CCIR_D65_CIE_E_RGB    73  /*appearance match - linear CCIR illuminant D65 RGB to linear CIE illuminant E RGB */
#define IR_A_LIN_CCIR_D65_EBU_C_RGB    74  /*appearance match - linear CCIR illuminant D65 RGB to linear EBU illuminant C RGB */
#define IR_A_LIN_CCIR_D65_NTSC_C_RGB   75  /*appearance match - linear CCIR illuminant D65 RGB to linear NTSC illuminant C RGB */

```


/* colour conversion matrices (continued) */		
#define IR_A_LIN_CIE_E_CCIR_D65_RGB	76	/* appearance match - linear CIE illuminant E RGB to linear CCIR illuminant D65 RGB */
#define IR_A_LIN_CIE_E_EBU_C_RGB	77	/* appearance match - linear CIE illuminant E RGB to linear EBU illuminant C RGB */
#define IR_A_LIN_CIE_E_EBU_D65_RGB	78	/* appearance match - linear CIE illuminant E RGB to linear EBU illuminant D65 RGB */
#define IR_A_LIN_CIE_E_NTSC_C_RGB	79	/* appearance match - linear CIE illuminant E RGB to linear NTSC illuminant C RGB */
#define IR_A_LIN_CIE_E_NTSC_D65_RGB	80	/* appearance match - linear CIE illuminant E RGB to linear NTSC illuminant D65 RGB */
#define IR_A_LIN_CIE_E_SMPTE_D65_RGB	81	/* appearance match - linear CIE illuminant E RGB to linear SMPTE illuminant D65 RGB */
#define IR_A_LIN_EBU_C_CCIR_D65_RGB	82	/* appearance match - linear EBU illuminant C RGB to linear CCIR illuminant C65 RGB */
#define IR_A_LIN_EBU_C_CIE_E_RGB	83	/* appearance match - linear EBU illuminant C RGB to linear CIE illuminant E RGB */
#define IR_A_LIN_EBU_C_EBU_D65_RGB	84	/* appearance match - linear EBU illuminant C RGB to linear EBU illuminant D65 RGB */
#define IR_A_LIN_EBU_C_NTSC_D65_RGB	85	/* appearance match - linear EBU illuminant C RGB to linear NTSC illuminant D65 RGB */
#define IR_A_LIN_EBU_C_SMPTE_D65_RGB	86	/* appearance match - linear EBU illuminant C RGB to linear SMPTE illuminant D65 RGB */
#define IR_A_LIN_EBU_D65_CIE_E_RGB	87	/* appearance match - linear EBU illuminant D65 RGB to linear CIE illuminant E RGB */
#define IR_A_LIN_EBU_D65_EBU_C_RGB	88	/* appearance match - linear EBU illuminant D65 RGB to linear EBU illuminant C RGB */

/*		colour conversion matrices (continued)	*/
#define IR_A_LIN_EBU_D65_NTSC_C_RGB	89	/* appearance match - linear EBU illuminant D65 RGB to linear NTSC illuminant C RGB	*/
#define IR_A_LIN_NTSC_C_CCIR_D65_RGB	90	/* appearance match - linear NTSC illuminant C RGB to linear CCIR illuminant D65 RGB	*/
#define IR_A_LIN_NTSC_C_CIE_E_RGB	91	/* appearance match - linear NTSC illuminant C RGB to linear CIE illuminant E RGB	*/
#define IR_A_LIN_NTSC_C_EBU_D65_RGB	92	/* appearance match - linear NTSC illuminant C RGB to linear EBU illuminant D65 RGB	*/
#define IR_A_LIN_NTSC_C_NTSC_D65_RGB	93	/* appearance match - linear NTSC illuminant C RGB to linear NTSC illuminant D65 RGB	*/
#define IR_A_LIN_NTSC_C_SMPTE_D65_RGB	94	/* appearance match - linear NTSC illuminant C RGB to linear SMPTE illuminant D65 RGB	*/
#define IR_A_LIN_NTSC_D65_CIE_E_RGB	95	/* appearance match - linear NTSC illuminant D65 RGB to linear CIE illuminant E RGB	*/
#define IR_A_LIN_NTSC_D65_EBU_C_RGB	96	/* appearance match - linear NTSC illuminant D65 RGB to linear EBU illuminant C RGB	*/
#define IR_A_LIN_NTSC_D65_NTSC_C_RGB	97	/* appearance match - linear NTSC illuminant D65 RGB to linear NTSC illuminant C RGB	*/
#define IR_A_LIN_SMPTE_D65_CIE_E_RGB	98	/* appearance match - linear SMPTE illuminant D65 RGB to linear CIE illuminant E RGB	*/
#define IR_A_LIN_SMPTE_D65_EBU_C_RGB	99	/* appearance match - linear SMPTE illuminant D65 RGB to linear EBU illuminant C RGB	*/
#define IR_A_LIN_SMPTE_D65_NTSC_C_RGB	100	/* appearance match - linear SMPTE illuminant D65 RGB to linear NTSC illuminant C RGB	*/
#define IR_C_LIN_CCIR_D65_CIE_E_RGB	1	/* colourmetric match - linear CCIR illuminant D65 RGB to linear CIE illuminant E RGB	*/

```

/*                                colour conversion matrices (continued)                                */

#define IR_C_LIN_CCIR_D65_EBU_C_RGB      2  /*colourmetric match - linear CCIR
                                           illuminant D65 RGB to linear EBU
                                           illuminant C RGB                                */

#define IR_C_LIN_CCIR_D65_EBU_D65_RGB    3  /*colourmetric match - linear CCIR
                                           illuminant D65 RGB to linear EBU
                                           illuminant D65 RGB                                */

#define IR_C_LIN_CCIR_D65_NTSC_C_RGB      4  /*colourmetric match - linear CCIR
                                           illuminant D65 RGB to linear NTSC
                                           illuminant C RGB                                */

#define IR_C_LIN_CCIR_D65_NTSC_D65_RGB    5  /*colourmetric match - linear CCIR
                                           illuminant D65 RGB to linear NTSC
                                           illuminant D65 RGB                                */

#define IR_C_LIN_CCIR_D65_SMPTE_D65_RGB   6  /*colourmetric match - linear CCIR
                                           illuminant D65 RGB to linear SMPTE
                                           illuminant D65 RGB                                */

#define IR_C_LIN_CCIR_D65_RGB_UVW         7  /*colourmetric match - linear CCIR
                                           illuminant D65 RGB to UVW                                */

#define IR_C_LIN_CCIR_D65_RGB_XYZ         8  /*colourmetric match - linear CCIR
                                           illuminant D65 RGB to XYZ                                */

#define IR_C_LIN_CIE_E_CCIR_D65_RGB       9  /*colourmetric match - linear CIE
                                           illuminant E RGB to linear CCIR
                                           illuminant D65 RGB                                */

#define IR_C_LIN_CIE_E_EBU_C_RGB          10 /*colourmetric match - linear CIE
                                           illuminant E RGB to linear EBU
                                           illuminant C RGB                                */

#define IR_C_LIN_CIE_E_EBU_D65_RGB        11 /*colourmetric match - linear CIE
                                           illuminant E RGB to linear EBU
                                           illuminant D65 RGB                                */

#define IR_C_LIN_CIE_E_NTSC_C_RGB         12 /*colourmetric match - linear CIE
                                           illuminant E RGB to linear NTSC
                                           illuminant C RGB                                */

#define IR_C_LIN_CIE_E_NTSC_D65_RGB       13 /*colourmetric match - linear CIE
                                           illuminant E RGB to linear NTSC
                                           illuminant D65 RGB                                */

#define IR_C_LIN_CIE_E_SMPTE_D65_RGB      14 /*colourmetric match - linear CIE
                                           illuminant E RGB to linear SMPTE
                                           illuminant D65 RGB                                */

```

/*	colour conversion matrices (continued)	*/
#define IR_C_LIN_CIE_E_RGB_UVW	15	/* colourmetric match - linear CIE illuminant E RGB to UVW */
#define IR_C_LIN_CIE_E_RGB_XYZ	16	/* colourmetric match - linear CIE illuminant E RGB to XYZ */
#define IR_C_LIN_EBU_C_CCIR_D65_RGB	17	/* colourmetric match - linear EBU illuminant C RGB to linear CCIR illuminant D65 RGB */
#define IR_C_LIN_EBU_C_CIE_E_RGB	18	/* colourmetric match - linear EBU illuminant C RGB to linear CIE illuminant E RGB */
#define IR_C_LIN_EBU_C_EBU_D65_RGB	19	/* colourmetric match - linear EBU illuminant C RGB to linear EBU illuminant D65 RGB */
#define IR_C_LIN_EBU_C_NTSC_C_RGB	20	/* colourmetric match - linear EBU illuminant C RGB to linear NTSC illuminant C RGB */
#define IR_C_LIN_EBU_C_NTSC_D65_RGB	21	/* colourmetric match - linear EBU illuminant C RGB to linear NTSC illuminant D65 RGB */
#define IR_C_LIN_EBU_C_SMPTE_D65_RGB	22	/* colourmetric match - linear EBU illuminant C RGB to linear SMPTE illuminant D65 RGB */
#define IR_C_LIN_EBU_C_RGB_UVW	23	/* colourmetric match - linear EBU illuminant C RGB to UVW */
#define IR_C_LIN_EBU_C_RGB_XYZ	24	/* colourmetric match - linear EBU illuminant C RGB to XYZ */
#define IR_C_LIN_EBU_D65_CCIR_D65_RGB	25	/* colourmetric match - linear EBU illuminant D65 RGB to linear CCIR illuminant D65 RGB */
#define IR_C_LIN_EBU_D65_CIE_E_RGB	26	/* colourmetric match - linear EBU illuminant D65 RGB to linear CIE illuminant E RGB */
#define IR_C_LIN_EBU_D65_EBU_C_RGB	27	/* colourmetric match - linear EBU illuminant D65 RGB to linear EBU illuminant C RGB */
#define IR_C_LIN_EBU_D65_NTSC_C_RGB	28	/* colourmetric match - linear EBU illuminant D65 RGB to linear NTSC illuminant C RGB */

```

/*                                colour conversion matrices (continued)                                */

#define IR_C_LIN_EBU_D65_NTSC_D65_RGB    29  /* colourmetric match - linear EBU
                                           illuminant D65 RGB to linear NTSC
                                           illuminant D65 RGB                                */

#define IR_C_LIN_EBU_D65_SMPTE_D65_RGB    30  /* colourmetric match - linear EBU
                                           illuminant D65 RGB to linear SMPTE
                                           illuminant D65 RGB                                */

#define IR_C_LIN_EBU_D65_RGB_UVW          31  /* colourmetric match - linear EBU
                                           illuminant D65 RGB to UVW                        */

#define IR_C_LIN_EBU_D65_RGB_XYZ          32  /* colourmetric match - linear EBU
                                           illuminant D65 RGB to XYZ                        */

#define IR_C_LIN_NTSC_C_CCIR_D65_RGB      33  /* colourmetric match - linear NTSC
                                           illuminant C RGB to linear CCIR
                                           illuminant D65 RGB                                */

#define IR_C_LIN_NTSC_C_CIE_E_RGB         34  /* colourmetric match - linear NTSC
                                           illuminant C RGB to linear CIE
                                           illuminant E RGB                                */

#define IR_C_LIN_NTSC_C_EBU_C_RGB         35  /* colourmetric match - linear NTSC
                                           illuminant C RGB to linear EBU
                                           illuminant C RGB                                */

#define IR_C_LIN_NTSC_C_EBU_D65_RGB       36  /* colourmetric match - linear NTSC
                                           illuminant C RGB to linear EBU
                                           illuminant D65 RGB                                */

#define IR_C_LIN_NTSC_C_NTSC_D65_RGB      37  /* colourmetric match - linear NTSC
                                           illuminant C RGB to linear NTSC
                                           illuminant D65 RGB                                */

#define IR_C_LIN_NTSC_C_SMPTE_D65_RGB     38  /* colourmetric match - linear NTSC
                                           illuminant C RGB to linear SMPTE
                                           illuminant D65 RGB                                */

#define IR_C_LIN_NTSC_C_RGB_UVW           39  /* colourmetric match - linear NTSC
                                           illuminant C RGB to UVW                        */

#define IR_C_LIN_NTSC_C_RGB_XYZ           40  /* colourmetric match - linear NTSC
                                           illuminant C RGB to XYZ                        */

#define IR_C_LIN_NTSC_D65_CCIR_D65_RGB    41  /* colourmetric match - linear NTSC
                                           illuminant D65 RGB to linear CCIR
                                           illuminant D65 RGB                                */

```

/*		colour conversion matrices (continued)	*/
#define	IR_C_LIN_NTSC_D65_CIE_E_RGB	42	/* colourmetric match - linear NTSC illuminant D65 RGB to linear CIE illuminant E RGB */
#define	IR_C_LIN_NTSC_D65_EBU_C_RGB	43	/* colourmetric match - linear NTSC illuminant D65 RGB to linear EBU illuminant C RGB */
#define	IR_C_LIN_NTSC_D65_EBU_D65_RGB	44	/* colourmetric match - linear NTSC illuminant D65 RGB to linear EBU illuminant D65 RGB */
#define	IR_C_LIN_NTSC_D65_NTSC_C_RGB	45	/* colourmetric match - linear NTSC illuminant D65 RGB to linear NTSC illuminant C RGB */
#define	IR_C_LIN_NTSC_D65_SMPTE_D65_RGB	46	/* colourmetric match - linear NTSC illuminant D65 RGB to linear SMPTE illuminant D65 RGB */
#define	IR_C_LIN_NTSC_D65_RGB_UVW	47	/* colourmetric match - linear NTSC illuminant D65 RGB to UVW */
#define	IR_C_LIN_NTSC_D65_RGB_XYZ	48	/* colourmetric match - linear NTSC illuminant D65 RGB to XYZ */
#define	IR_C_LIN_SMPTE_D65_CCIR_D65_RGB	49	/* colourmetric match - linear SMPTE illuminant D65 RGB to linear CCIR illuminant D65 RGB */
#define	IR_C_LIN_SMPTE_D65_CIE_E_RGB	50	/* colourmetric match - linear SMPTE illuminant D65 RGB to linear CIE illuminant E RGB */
#define	IR_C_LIN_SMPTE_D65_EBU_C_RGB	51	/* colourmetric match - linear SMPTE illuminant D65 RGB to linear EBU illuminant C RGB */
#define	IR_C_LIN_SMPTE_D65_EBU_D65_RGB	52	/* colourmetric match - linear SMPTE illuminant D65 RGB to linear EBU illuminant D65 RGB */
#define	IR_C_LIN_SMPTE_D65_NTSC_C_RGB	53	/* colourmetric match - linear SMPTE illuminant D65 RGB to linear NTSC illuminant C RGB */
#define	IR_C_LIN_SMPTE_D65_NTSC_D65_RGB	54	/* colourmetric match - linear SMPTE illuminant D65 RGB to linear NTSC illuminant D65 RGB */

```

/*                                colour conversion matrices (continued)                                */

#define IR_C_LIN_SMPTE_D65_RGB_UVW      55  /* colourmetric match - linear SMPTE      */
                                           illuminant D65 RGB to UVW                */

#define IR_C_LIN_SMPTE_D65_RGB_XYZ      56  /* colourmetric match - linear SMPTE      */
                                           illuminant D65 RGB to XYZ                */

#define IR_C_UVW_LIN_CCIR_D65_RGB        57  /* colourmetric match - UVW to linear      */
                                           CCIR illuminant D65 RGB                 */

#define IR_C_UVW_LIN_CIE_E_RGB           58  /* colourmetric match - UVW to linear      */
                                           CIE illuminant E RGB                   */

#define IR_C_UVW_LIN_EBU_C_RGB           59  /* colourmetric match - UVW to linear      */
                                           EBU illuminant C RGB                   */

#define IR_C_UVW_LIN_EBU_D65_RGB         60  /* colourmetric match - UVW to linear      */
                                           EBU illuminant D65 RGB                 */

#define IR_C_UVW_LIN_NTSC_C_RGB          61  /* colourmetric match - UVW to linear      */
                                           NTSC illuminant C RGB                  */

#define IR_C_UVW_LIN_NTSC_D65_RGB        62  /* colourmetric match - UVW to linear      */
                                           NTSC illuminant D65 RGB                 */

#define IR_C_UVW_LIN_SMPTE_D65_RGB       63  /* colourmetric match - UVW to linear      */
                                           SMPTE illuminant D65 RGB                */

#define IR_C_UVW_XYZ                     64  /* colourmetric match - UVW to XYZ         */

#define IR_C_XYZ_LIN_CCIR_D65_RGB        65  /* colourmetric match - XYZ to linear      */
                                           CCIR illuminant D65 RGB                 */

#define IR_C_XYZ_LIN_CIE_E_RGB           66  /* colourmetric match - XYZ to linear      */
                                           CIE illuminant E RGB                   */

#define IR_C_XYZ_LIN_EBU_C_RGB           67  /* colourmetric match - XYZ to linear      */
                                           EBU illuminant C RGB                   */

#define IR_C_XYZ_LIN_EBU_D65_RGB         68  /* colourmetric match - XYZ to linear      */
                                           EBU illuminant D65 RGB                 */

#define IR_C_XYZ_LIN_NTSC_C_RGB          69  /* colourmetric match - XYZ to linear      */
                                           NTSC illuminant C RGB                  */

#define IR_C_XYZ_LIN_NTSC_D65_RGB        70  /* colourmetric match - XYZ to linear      */
                                           NTSC illuminant D65 RGB                 */

#define IR_C_XYZ_LIN_SMPTE_D65_RGB       71  /* colourmetric match - XYZ to linear      */
                                           SMPTE illuminant D65 RGB                */

```

```

/*                                colour conversion matrices (continued)                                */

#define IR_C_XYZ_UVW                72  /* colourmetric match - XYZ to UVW */

#define IR_GAMMA_EBU_C_RGB_YUV      101 /* gamma EBU illuminant C RGB to
                                         EBU illuminant C YUV */

#define IR_GAMMA_EBU_D65_RGB_YUV    102 /* gamma EBU illuminant D65 RGB to
                                         EBU illuminant D65 YUV */

#define IR_GAMMA_NTSC_C_RGB_YIQ     103 /* gamma NTSC illuminant C RGB to
                                         NTSC illuminant C YIQ */

#define IR_GAMMA_NTSC_D65_RGB_YIQ   104 /* gamma NTSC illuminant D65 RGB
                                         to NTSC illuminant D65 YIQ */

#define IR_GAMMA_SMPTE_D65_RGB_YCBCR 105 /* gamma SMPTE illuminant D65 RGB
                                         to SMPTE illuminant D65 YCbCr */

#define IR_YUV_GAMMA_EBU_C_RGB      106 /* EBU illuminant C YUV to gamma
                                         EBU illuminant C RGB */

#define IR_YUV_GAMMA_EBU_D65_RGB    107 /* EBU illuminant D65 YUV to gamma
                                         EBU illuminant D65 RGB */

#define IR_YIQ_GAMMA_NTSC_C_RGB      108 /* NTSC illuminant C YIQ to gamma
                                         NTSC illuminant C RGB */

#define IR_YIQ_GAMMA_NTSC_D65_RGB    109 /* NTSC illuminant D65 YIQ to gamma
                                         NTSC illuminant D65 RGB */

#define IR_YCBCR_GAMMA_SMPTE_D65_RGB 110 /* SMPTE illuminant D65 YCbCr to
                                         gamma SMPTE illuminant D65 RGB*/

#define IR_ILLUMINANT_C_D50          111 /* illuminant C XYZ to illuminant D50
                                         XYZ */

#define IR_ILLUMINANT_C_D65          112 /* illuminant C XYZ to illuminant D65
                                         XYZ */

#define IR_ILLUMINANT_C_E            113 /* illuminant C XYZ to illuminant E
                                         XYZ */

#define IR_ILLUMINANT_D50_C          114 /* illuminant D50 XYZ to illuminant C
                                         XYZ */

#define IR_ILLUMINANT_D50_D65        115 /* illuminant D50 XYZ to illuminant D65
                                         XYZ */

#define IR_ILLUMINANT_D50_E          116 /* illuminant D50 XYZ to illuminant E
                                         XYZ */

```



```

/*                                colour conversion matrices (concluded)                                */

#define IR_ILLUMINANT_D65_C      117  /*illuminant D65 XYZ to illuminant C
                                   XYZ                                */

#define IR_ILLUMINANT_D65_D50    118  /*illuminant D65 XYZ to illuminant D50
                                   XYZ                                */

#define IR_ILLUMINANT_D65_E      119  /*illuminant D65 XYZ to illuminant E
                                   XYZ                                */

#define IR_ILLUMINANT_E_C        120  /*illuminant E XYZ to illuminant C
                                   XYZ                                */

#define IR_ILLUMINANT_E_D50      121  /*illuminant E XYZ to illuminant D50
                                   XYZ                                */

#define IR_ILLUMINANT_E_D65      122  /*illuminant E XYZ to illuminant D65
                                   XYZ                                */

```

B.1.2 Data types in compilation order

```

/*
                                BASIC DATA TYPES
*/

/*
                                external image pixel data types
*/

typedef      unsigned char    Imbool;      /* BI data type
*/

typedef      unsigned char    Imuint;      /* NI data type
*/

typedef      int              Imint;        /* SI data type
*/

typedef      int              Imfixed;      /* TI representation of real data type
*/

typedef      float            Imfloat;      /* RF float representation of real data type
*/

typedef struct {
    Imfloat      xReal;      /* real part
*/
    Imfloat      xImaginary; /* imaginary part
*/
} Imscomplex; /* CF data type
*/

/*
                                parameter data types
*/

typedef      unsigned char    Ipbool;      /* BP data type
*/

typedef      unsigned long int Ipuint;      /* NP data type
*/

typedef      int              Ipint;        /* SP data type
*/

typedef      float            Ipfloat;      /* RP real data type
*/

typedef struct {
    Ipfloat      xReal;      /* real part
*/
    Ipfloat      xImaginary; /* imaginary part
*/
} Ipscomplex; /* CP data type
*/

/*
                                object identifiers
*/

typedef      void             *Idnchain;     /* IPI-PIKS internal chain object
*/

typedef      void             *Idnid_array; /* IPI-PIKS internal image id array
*/

typedef      void             *Idnid_list;  /* IPI-PIKS internal image id list
*/

typedef      void             *Idnid_record; /* IPI-PIKS internal image id record
*/

typedef      void             *Idnhist;     /* IPI-PIKS internal histogram arrays
*/

typedef      void             *Idnimage;    /* IPI-PIKS internal image objects
*/

typedef      void             *Idnlut;      /* IPI-PIKS internal lookup tables
*/

```

```

/*                                object identifiers (concluded)                                */

typedef      void      *Idnmatrix;      /* IPI-PIKS internal matrix objects */
typedef      void      *Idnnbhood;      /* IPI-PIKS internal neighbourhood arrays */
typedef      void      *Idnpixel_record; /* IPI-PIKS internal pixel record */
typedef      void      *Idnrepository;   /* IPI-PIKS repository object */
typedef      void      *Idnroi;          /* IPI-PIKS internal region of interest objects */
typedef      void      *Idnstatic;       /* IPI-PIKS internal static arrays */
typedef      void      *Idntuple;        /* IPI-PIKS internal tuples */
typedef      void      *Idnbounds;       /* IPI-PIKS internal value bounds lists */
typedef      void      *Idnvreg;         /* IPI-PIKS internal virtual register objects */
typedef      void      *Ipnarg;          /* external function reloadable arguments */
typedef      FILE      *Ipnerror;        /* external error file */
typedef      void      *Ipnport;         /* external input/output port */

/*                                ENUMERATED DATA TYPES                                */

typedef enum {
    IBINARY_EVENT_ON,      /* yes/on */
    IBINARY_EVENT_OFF,     /* no/off */
} Ipebinary_event;

typedef enum {
    ICOMPLEX_REAL_IMAGINARY, /* real/imaginary to/from complex */
    ICOMPLEX_MAGNITUDE_PHASE, /* magnitude/phase to/from complex */
} Ipecomplex_conv;

typedef enum {
    ICOMPLEX_MAGNITUDE,     /* magnitude output */
    ICOMPLEX_SQUARED,       /* magnitude squared output */
} Ipemagnitude;

typedef enum {
    IVALIDITY_VALID,        /* data is valid and available */
    IVALIDITY_INVALID,      /* data is not valid */
} Ipevalidity;

typedef enum {
    IDERIV_HORIZONTAL,      /* horizontal derivative */
    IDERIV_VERTICAL,        /* vertical derivative */
} Ipederiv;

```

```

/*                      ENUMERATED DATA TYPES (continued)                      */

typedef enum {
    IDIMENSION_1D,          /* one dimension          */
    IDIMENSION_2D,          /* two dimensions         */
    IDIMENSION_3D,          /* three dimensions       */
    IDIMENSION_4D,          /* four dimensions        */
    IDIMENSION_5D,          /* five dimensions        */
} Ipedimension;

typedef enum {
    IOPERATION_EROSION,     /* erosion                */
    IOPERATION_DILATE,      /* dilation               */
} Ipeerosion;

typedef enum {
    IFLIP_TOP_BOTTOM,       /* top-to-bottom flip    */
    IFLIP_LEFT_RIGHT,       /* left-to-right flip    */
    ISPIN_90_DEGREES_CCW,   /* 90 degrees counterclockwise spin */
    ISPIN_180_DEGREES_CCW, /* 180 degrees counterclockwise spin */
    ISPIN_270_DEGREES_CCW, /* 270 degrees counterclockwise spin */
    ITRANPOSE_UPLEFT_LOWRIGHT, /* transpose about upper left to lower right */
    ITRANPOSE_LOWLEFT_UPRIGHT, /* transpose about lower left to upper right */
} Ipeflip;

typedef enum {
    IFREQUENCY_SHIFT_ORIGIN, /* constant term at origin */
    IFREQUENCY_SHIFT_CENTRE, /* constant term centre    */
} Ipefrequency_shift;

typedef enum {
    ICORRECT_LIN_GAMMA,     /* linear to gamma correction */
    ICORRECT_GAMMA_LIN,     /* gamma to linear correction  */
} Ipegamma;

typedef enum {
    IINDEX_HORIZONTAL,      /* horizontal dimension      */
    IINDEX_VERTICAL,        /* vertical dimension        */
    IINDEX_DEPTH,           /* depth dimension           */
    IINDEX_TEMPORAL,        /* temporal dimension        */
    IINDEX_BAND,            /* band dimension            */
} Ipeindex;

typedef enum {
    IBACKGROUND_ZERO,       /* zero background          */
    IBACKGROUND_IMAGE,      /* image background         */
} Ipebackground;

typedef enum {
    IERROR_REPORT_ACTIVE,   /* error reporting active    */
    IERROR_REPORT_INACTIVE, /* error reporting inactive  */
} Ipeerror_report_mode;

```

```

/*                                     ENUMERATED DATA TYPES (continued)                                     */

typedef enum {
    IINDEX_ASSIGNMENT_ON,          /* index assignment operative          */
    IINDEX_ASSIGNMENT_OFF         /* index assignment inoperative       */
} Ipeindex_mode;

typedef enum {
    IMATCH_POINT_ON,              /* match point adjustment operative   */
    IMATCH_POINT_OFF             /* match point adjustment inoperative */
} Ipematch_mode;

typedef enum {
    IROI_CONTROL_ON,              /* ROI control operative              */
    IROI_CONTROL_OFF             /* ROI control inoperative            */
} Iperoi_control_mode;

typedef enum {
    IROI_PROCESS_ON,              /* ROI processing operative            */
    IROI_PROCESS_OFF             /* ROI processing inoperative          */
} Iperoi_process_mode;

typedef enum {
    ISYNCHRONICITY_MODE_SYNCH,    /* synchronous mode                   */
    ISYNCHRONICITY_MODE_ASYNCH    /* asynchronous mode                   */
} Ipesynchronicity_mode;

typedef enum {
    IOPERATION_OPEN,              /* open option                         */
    IOPERATION_CLOSE             /* close option                        */
} Ipeopen_close;

typedef enum {
    IPOLARITY_TRUE,               /* TRUE state                          */
    IPOLARITY_FALSE              /* FALSE state                         */
} Ipepolarity;

typedef enum {
    IPROCESS_MODE_0,              /* processing mode 0                   */
    IPROCESS_MODE_1,              /* processing mode 1                   */
    IPROCESS_MODE_2,              /* processing mode 2                   */
    IPROCESS_MODE_3,              /* processing mode 3                   */
    IPROCESS_MODE_4              /* processing mode 4                   */
} Ipeprocess;

typedef enum {
    IREFLECTION_NO,               /* no reflection                       */
    IREFLECTION_YES               /* reflection                           */
} Ipereflexion;

```

```

/*                      ENUMERATED DATA TYPES (concluded)                      */

typedef enum {
    IROTATION_NO,                      /* no rotation                      */
    IROTATION_YES                      /* rotation                          */
} Iperotation;

typedef enum {
    ICHAIN_BUILD,                     /* building chain state             */
    ICHAIN_EXECUTE,                   /* executing chain state            */
    ICHAIN_NOT                         /* not in any chain state           */
} Ipechain_state;

typedef enum {
    IERROR_SUBSTATE,                  /* error substate condition exists  */
    IERROR_NO                         /* no error condition               */
} Ipeerror_state;

typedef enum {
    IPIKS_OPERATION_CLOSED,           /* PIKS closed                      */
    IPIKS_OPERATION_OPEN              /* PIKS open                        */
} Ipepiks_operation_state;

typedef enum {
    ISYNCHRONICITY_STATE_SYNCH,       /* synchronous state                */
    ISYNCHRONICITY_STATE_ASYNCH       /* asynchronous state                */
} Ipesynchronicity_state;

typedef enum {
    ITRANSFORM_FORWARD,               /* forward transform                */
    ITRANSFORM_INVERSE                /* inverse transform                 */
} Ipetransform;

typedef enum {
    IVREG_CHC_DIRECT,                 /* direct parameter specification    */
    IVREG_CHC_VREG                    /* virtual register representation    */
} Ipevreg_chc;

typedef enum {
    IVREG_STATE_CLEARED,               /* virtual register cleared          */
    IVREG_STATE_SET                    /* virtual register set              */
} Ipevreg_state;

/*                      FUNCTION POINTER DECLARATION                      */

typedef void (*Iptferror_function) /* error handler function pointer */
(Ipint iCode,
Ipint iElement,
Idnchain nChain,
Iperror nError,
Ipesynchronicity_state eSynchronicityState);

```

```

/*                                UNIONS                                */

typedef union {
    Idnmatrix          nMatrix;          /* matrix identifier          */
    Idnnbhood          nNbhood;         /* neighbourhood array identifier */
    Idnstatic           nStatic;         /* static array identifier     */
} Ipsarray_id;

typedef union {
    Imbool             *tbPixel;         /* external Boolean pixel value pointer */
    Imuint             *tuPixel;         /* external unsigned integer pixel value pointer */
    Imint              *tiPixel;         /* external signed integer pixel pointer */
    Imfixed            *trPixelFixed;    /* external fixed point pixel pointer */
    Imfloat            *trPixelFloat;    /* external floating point pixel pointer */
    Imscomplex         *txPixel;         /* external complex pixel value pointer */
} Ipspiks_pixel_types;

typedef union {
    Idnid_array         nIdArray;         /* composite identifier array identifier */
    Idnid_list          nIdList;          /* composite identifier list identifier */
    Idnid_record        nIdRecord;        /* composite identifier record identifier */
    Idnhist             nHist;           /* histogram identifier         */
    Idnlut             nLUT;            /* lookup table identifier      */
    Idnmatrix          nMatrix;          /* matrix identifier           */
    Idnnbhood          nNbhood;         /* neighbourhood array identifier */
    Idnpixel_record     nPixel;          /* pixel record identifier      */
    Idnroi             nROI;            /* region of interest identifier */
    Idnstatic           nStatic;         /* static array identifier     */
    Idntuple           nTuple;          /* tuple identifier            */
    Idnbounds          nBounds;         /* value bounds collection identifier */
    Idnvreg            nVReg;           /* virtual register identifier   */
} Ipsnon_image_object;

typedef union {
    Idnchain           nChain;           /* chain identifier            */
    Idnid_array         nIdArray;         /* composite identifier array identifier */
    Idnid_list          nIdList;          /* composite identifier list identifier */
    Idnid_record        nIdRecord;        /* composite identifier record identifier */
    Idnhist             nHist;           /* histogram identifier         */
    Idnimage           nImage;          /* image identifier            */
    Idnlut             nLUT;            /* lookup table identifier      */
    Idnmatrix          nMatrix;          /* matrix identifier           */
    Idnnbhood          nNbhood;         /* neighbourhood array identifier */
    Idnpixel_record     nPixel;          /* pixel record identifier      */
    Idnrepository       nRepository;      /* repository identifier       */
    Idnroi             nROI;            /* region of interest identifier */
    Idnstatic           nStatic;         /* static array identifier     */
    Idntuple           nTuple;          /* tuple identifier            */
    Idnbounds          nBounds;         /* value bounds collection identifier */
    Idnvreg            nVReg;           /* virtual register identifier   */
} Ipsobject;

```



```

/*                                UNIONS (continued)                                */

typedef union {
    Ipuint      *tuArith;          /* unsigned integer value pointer */
    Ipinint     *tiArith;          /* signed integer value pointer   */
    Ipfloata    *trArithFloat;    /* floating real value pointer     */
} Ipsparameter_arith;

typedef union {
    Ipbool      *tbParameter;      /* Boolean value pointer          */
    Ipuint      *tuParameter;      /* non-negative integer value pointer */
    Ipinint     *tiParameter;      /* signed integer value pointer    */
    Ipfloata    *trParameterFloat; /* floating real value pointer     */
    Ipscomplex  *txParameter;      /* complex data pointer           */
    char        *tcParameter;      /* character pointer              */
} Ipsparameter_basic;

typedef union {
    Ipbool      *tbColour;         /* Boolean value pointer          */
    Ipuint      *tuColour;         /* non-negative integer value pointer */
    Ipinint     *tiColour;         /* signed integer value pointer    */
    Ipfloata    *trColourFloat;    /* floating real value pointer     */
} Ipsparameter_colour;

typedef union {
    Ipbool      *tbEdge;           /* Boolean output value pointer    */
    Ipfloata    *trEdgeFloat;      /* floating real output value pointer */
} Ipsparameter_edge;

typedef union {
    Ipbool      *tbLogical;        /* Boolean value pointer          */
    Ipuint      *tuLogical;        /* unsigned integer value pointer  */
    Ipinint     *tiLogical;        /* signed integer value pointer    */
} Ipsparameter_logical;

typedef union {
    Ipuint      *tuNumeric;        /* unsigned integer value pointer  */
    Ipinint     *tiNumeric;        /* signed integer value pointer    */
    Ipfloata    *trNumericFloat;   /* floating real value pointer     */
    Ipscomplex  *txNumeric;        /* complex data pointer           */
} Ipsparameter_numeric;

typedef union {
    Ipbool      *tbPixel;          /* Boolean value pointer          */
    Ipuint      *tuPixel;          /* unsigned integer value pointer  */
    Ipinint     *tiPixel;          /* signed integer value pointer    */
    Ipfloata    *trPixelFloat;     /* floating real value pointer     */
    Ipscomplex  *txPixel;          /* complex data pointer           */
} Ipsparameter_pixel;

```

```

/*                                UNIONS (continued)                                */

typedef union {
    Ipsobject          snVReg;          /* PIKS internal identifier structure */
    void               *tVReg;         /* external object pointer           */
    Ipsparameter_basic stVReg;         /* basic parameter type value pointer structure */
} Ipsparameter_vreg;

typedef union {
    Ipnarg             sArgs;          /* argument parameter collection identifier */
    Idnvreg            nVReg;         /* virtual register identifier           */
} Ipsarg_vreg;

typedef union {
    Ipsparameter_arith stArith;        /* arithmetic parameter pointer union */
    Idnvreg            nVReg;         /* virtual register identifier           */
} Ipsparameter_arith_vreg;

typedef union {
    Ipsarray_id        snArrayId;      /* matrix, neighbourhood or static array identifier */
    Idnvreg            nVReg;         /* virtual register identifier           */
} Ipsarray_id_vreg;

typedef union {
    Ipebackground      eOption;        /* level slice background option */
    Idnvreg            nVReg;         /* virtual register identifier           */
} Ipsbackground_vreg;

typedef union {
    Ipsparameter_basic stParameter;    /* basic parameter pointer union */
    Idnvreg            nVReg;         /* virtual register identifier           */
} Ipsparameter_basic_vreg;

typedef union {
    Ipbool             *tbaBPValue;    /* BP array data pointer */
    Idnvreg            nVReg;         /* virtual register identifier           */
} Ipsabool_vreg;

typedef union {
    Ipbool             bBPValue;       /* BP parameter value */
    Idnvreg            nVReg;         /* virtual register identifier           */
} Ipsbool_vreg;

typedef union {
    Idnchain           nChain;         /* chain identifier */
    Idnvreg            nVReg;         /* virtual register identifier           */
} Ipschain_vreg;

typedef union {
    char               *tzChar;        /* character string pointer */
    Idnvreg            nVReg;         /* virtual register identifier           */
} Ipschar_vreg;

```

```

/*
UNIONS (continued)
*/

typedef union {
    Ipsparameter_colour    stColour;    /* colour parameter pointer union */
    Idnvreg                nVReg;       /* virtual register identifier */
} Ipsparameter_colour_vreg;

typedef union {
    Ipecomplex_conv        eOption;     /* complex conversion option */
    Idnvreg                nVReg;       /* virtual register identifier */
} Ipscomplex_conv_vreg;

typedef union {
    Idnid_array            nIdArray;     /* composite id array identifier */
    Idnvreg                nVReg;       /* virtual register identifier */
} Ipsid_array_vreg;

typedef union {
    Idnid_list             nIdList;      /* composite id list identifier */
    Idnvreg                nVReg;       /* virtual register identifier */
} Ipsid_list_vreg;

typedef union {
    Idnid_record           nIdRecord;    /* composite id record identifier */
    Idnvreg                nVReg;       /* virtual register identifier */
} Ipsid_record_vreg;

typedef union {
    Ipscomplex             sxData;       /* complex data value */
    Idnvreg                nVReg;       /* virtual register identifier */
} Ipscomplex_vreg;

typedef union {
    Ipederiv              eDirection;    /* direction option */
    Idnvreg                nVReg;       /* virtual register identifier */
} Ipsderiv_vreg;

typedef union {
    Ipedimension           eDimension;   /* dimension option */
    Idnvreg                nVReg;       /* virtual register identifier */
} Ipsdimension_vreg;

typedef union {
    Ipsparameter_edge      stEdge;       /* edge output parameter pointer union */
    Idnvreg                nVReg;       /* virtual register identifier */
} Ipsparameter_edge_vreg;

typedef union {
    Ipeerosion             eErodeDilate; /* erosion or dilation option */
    Idnvreg                nVReg;       /* virtual register identifier */
} Ipserosion_vreg;

```

```

/*                                UNIONS (continued)                                */

typedef union {
    Ipspiks_pixel_types    stPixel;          /* external physical image pixel data pointer */
    Idnvreg                nVReg;           /* virtual register identifier */
} Ipspiks_pixel_types_vreg;

typedef union {
    Ipeflip                eOption;         /* flip, spin, transpose option */
    Idnvreg                nVReg;           /* virtual register identifier */
} Ipsflip_vreg;

typedef union {
    Ipefrequency_shift     eOption;         /* frequency shift option */
    Idnvreg                nVReg;           /* virtual register identifier */
} Ipsfrequency_shift_vreg;

typedef union {
    Ipegamma               eOption;         /* gamma conversion option */
    Idnvreg                nVReg;           /* virtual register identifier */
} Ipsgamma_vreg;

typedef union {
    Idnhist                nHist;           /* histogram identifier */
    Idnvreg                nVReg;           /* virtual register identifier */
} Ipshist_vreg;

typedef union {
    Idnimage               nImage;          /* image identifier */
    Idnvreg                nVReg;           /* virtual register identifier */
} Ipsimage_vreg;

typedef union {
    Ipeindex               eIndex;          /* image index assignment */
    Idnvreg                nVReg;           /* virtual register identifier */
} Ipsindex_vreg;

typedef union {
    Ipsparameter_logical   stLogical;       /* logical parameter pointer union */
    Idnvreg                nVReg;           /* virtual register identifier */
} Ipsparameter_logical_vreg;

typedef union {
    Idnlut                 nLUT;            /* lookup table identifier */
    Idnvreg                nVReg;           /* virtual register identifier */
} Ipslut_vreg;

typedef union {
    Ipemagnitude           eMagnitude;     /* magnitude options */
    Idnvreg                nVReg;           /* virtual register identifier */
} Ipsmagnitude_vreg;

```

```

/*                                UNIONS (continued)                                */

typedef union {
    Idnmatrix          nMatrix;          /* matrix identifier          */
    Idnvreg            nVReg;            /* virtual register identifier */
} Ipsmatrix_vreg;

typedef union {
    Idnnbhood          nNbhood;          /* neighbourhood array identifier */
    Idnvreg            nVReg;            /* virtual register identifier */
} Ipsnbhood_vreg;

typedef union {
    Idnnbhood          *tnaNbhood;       /* set of neighbourhood array identifiers */
    Idnvreg            nVReg;            /* virtual register identifier */
} Ipsanbhood_vreg;

typedef union {
    Ipuint             *tuaNPValue;       /* NP parameter array pointer */
    Idnvreg            nVReg;            /* virtual register identifier */
} Ipsaint_vreg;

typedef union {
    Ipuint             uNPValue;          /* NP parameter value */
    Idnvreg            nVReg;            /* virtual register identifier */
} Ipsuint_vreg;

typedef union {
    Ipsparameter_numeric stNumeric;       /* numeric parameter pointer union */
    Idnvreg            nVReg;            /* virtual register identifier */
} Ipsparameter_numeric_vreg;

typedef union {
    Ipsobject          snObject;          /* object identifier union */
    Idnvreg            nVReg;            /* virtual register identifier */
} Ipsobject_vreg;

typedef union {
    Ipeopen_close      eOption;           /* open close operation option */
    Idnvreg            nVReg;            /* virtual register identifier */
} Ipsopen_close_vreg;

typedef union {
    Ipsparameter_pixel stPixel;           /* pixel parameter pointer union */
    Idnvreg            nVReg;            /* virtual register identifier */
} Ipsparameter_pixel_vreg;

typedef union {
    Ipepolarity         ePolarity;        /* polarity option */
    Idnvreg            nVReg;            /* virtual register identifier */
} Ipspolarity_vreg;

```

```

/*                                UNIONS (continued)                                */

typedef union {
    Ipnport          nPort;          /* input/output port identifier          */
    Idnvreg          nVReg;          /* virtual register identifier          */
} Ipsport_vreg;

typedef union {
    Ipereflection    eOption;        /* reflection option                    */
    Idnvreg          nVReg;          /* virtual register identifier          */
} Ipsreflection_vreg;

typedef union {
    Idnroi           nROI;           /* ROI identifier                      */
    Idnvreg          nVReg;          /* virtual register identifier          */
} Ipsroi_vreg;

typedef union {
    Iperotation      eOption;        /* rotation option                    */
    Idnvreg          nVReg;          /* virtual register identifier          */
} Ipsrotation_vreg;

typedef union {
    Ipfloating       *traRPFloat;    /* float RP parameter pointer          */
    Idnvreg          nVReg;          /* virtual register identifier          */
} Ipsafloating_vreg;

typedef union {
    Ipfloating       rRPFloat;       /* RP parameter value                  */
    Idnvreg          nVReg;          /* virtual register identifier          */
} Ipsfloating_vreg;

typedef union {
    Ipintr           iSPValue;       /* SP parameter value                  */
    Idnvreg          nVReg;          /* virtual register identifier          */
} Ipsintr_vreg;

typedef union {
    Idnstatic        nStatic;        /* static array identifier              */
    Idnvreg          nVReg;          /* virtual register identifier          */
} Ipsstatic_vreg;

typedef union {
    Ipetransform     eTransform;     /* transform option                    */
    Idnvreg          nVReg;          /* virtual register identifier          */
} Ipstransform_vreg;

typedef union {
    Idntuple         nTuple;         /* tuple identifier                    */
    Idnvreg          nVReg;          /* virtual register identifier          */
} Ipstuple_vreg;

```

```

/*                                UNIONS (concluded)                                */

typedef union {
    Idnbounds          nBounds;          /* value bounds identifier          */
    Idnvreg             nVReg;            /* virtual register identifier      */
} Ipsbounds_vreg;

typedef union {
    Ipsparameter_vreg  sValue;           /* virtual register value data types union */
    Idnvreg             nVReg;            /* virtual register identifier      */
} Ipsparameter_vreg_vreg;

/*                                STRUCTURES                                */

typedef struct {
    Ipuint              uArrayLength;     /* identifier array length          */
} Ipsattrs_id_array;

typedef struct {
    Ipuint              uRecordLength;     /* identifier array record length    */
    char                **tatzNames;       /* pointer to array of name string pointers */
} Ipsattrs_id_record;

typedef struct {
    Ipuint              uHistSize;         /* histogram size                   */
    Ipfloat             rLowerBound;        /* lower amplitude bound            */
    Ipfloat             rUpperBound;       /* upper amplitude bound            */
} Ipsattrs_hist;

typedef struct {
    Idntuple            nImageSize;        /* image sizes ND 5-tuple identifier */
    Idntuple            nBandType;         /* band data types SD B-tuple identifier */
    Idntuple            nBandPrecision;    /* band precision ND B-tuple identifier */
    Ipfloat             rXWhite;           /* X tristimulus value white point  */
    Ipfloat             rYWhite;           /* Y tristimulus value white point  */
    Ipfloat             rZWhite;           /* Z tristimulus value white point  */
    Ipin                iImageStructureOption; /* image structure option          */
    Ipin                iColourOption;     /* colour space option              */
} Ipsattrs_image;

typedef struct {
    Ipuint              uTableEntries;     /* number of table entries          */
    Ipuint              uTableBands;       /* number of table bands            */
    Ipin                iInputTypeOption;  /* table input data type (internal) option */
    Ipin                iOutputTypeOption; /* table output data type (internal) option */
} Ipsattrs_lut;

typedef struct {
    Ipuint              uColumnSize;       /* matrix column size               */
    Ipuint              uRowSize;          /* matrix row size                  */
    Ipin                iTypeOption;       /* matrix data type (internal) option */
} Ipsattrs_matrix;

```

```

/*                                STRUCTURES (continued)                                */

typedef struct {
    Idntuple      nArraySize;      /* array size ND 5-tuple identifier */
    Idntuple      nKeyPixel;       /* array key pixel SD 5-tuple identifier */
    Ipint         iScaleFactor;    /* array scale factor */
    Ipint         iLabelOption;    /* array semantic label option */
    Ipint         iTypeOption;     /* array data type (internal) option */
} Ipsattrs_nbhood;

typedef struct {
    Ipuint        uRecordLength;   /* pixel record length */
    Idntuple      nTypeOption;     /* band data types SD B-tuple identifier */
} Ipsattrs_pixel_record;

typedef struct {
    Idntuple      nROISize;        /* ROI size ND 5-tuple identifier */
    Ipint         iStructureOption; /* ROI structure option */
    Ipepolarity   ePolarityOption; /* ROI polarity option */
} Ipsattrs_roi;

typedef struct {
    Idntuple      nROISize;        /* ROI array size ND 5-tuple identifier */
} Ipsattrs_roi_array;

typedef struct {
    Idntuple      nROISize;        /* ROI coordinate size ND 5-tuple identifier */
    Ipuint        uCollectionSize; /* number of coordinates */
    Ipepolarity   ePolarityOption; /* ROI polarity option */
} Ipsattrs_roi_coordinate;

typedef struct {
    Idntuple      nROISize;        /* ROI elliptical array size ND 5-tuple identifier */
    Idntuple      nEllipseCentre;  /* ellipse centre RD 5-tuple identifier */
    Idntuple      nEllipseLength;  /* ellipse axial lengths RD 5-tuple identifier */
    Idntuple      nIndexManip;     /* index manipulation CS 5-tuple identifier */
    Ipedimension  eDimensionOption; /* ellipse dimension option */
    Ipepolarity   ePolarityOption; /* ROI polarity option */
} Ipsattrs_roi_elliptical;

typedef struct {
    Idntuple      nROISize;        /* ROI generic size ND 5-tuple identifier */
} Ipsattrs_roi_generic;

typedef struct {
    Idntuple      nROISize;        /* ROI polygon size ND 5-tuple identifier */
    Ipuint        uPolygonArraySize; /* number of polygon array entries */
    Idntuple      nIndexManip;     /* index manipulation CS 5-tuple identifier */
    Ipepolarity   ePolarityOption; /* ROI polarity option */
} Ipsattrs_roi_polygon;

```



```

/*                      STRUCTURES (continued)                      */

typedef struct {
    Idntuple      nROIsize;      /* ROI rectangular size ND 5-tuple identifier */
    Idntuple      nRectangleStart; /* rectangle start position ND 5-tuple identifier */
    Idntuple      nRectangleEnd;  /* rectangle end position ND 5-tuple identifier */
    Idntuple      nIndexManip;   /* index manipulation CS 5-tuple identifier */
    Ipedimension  eDimensionOption; /* rectangle dimension option */
    Ipepolarity   ePolarityOption; /* ROI polarity option */
} Ipsattrs_roi_rectangular;

typedef struct {
    Idntuple      nArraySize;    /* static array size ND 5-tuple identifier */
    Ipin          iLabelOption;  /* static array semantic label option */
    Ipin          iTypeOption;   /* static array data type (internal) option */
} Ipsattrs_static;

typedef struct {
    Ipoint        uTupleSize;    /* tuple size */
    Ipin          iTypeOption;   /* tuple data type (internal) option */
} Ipsattrs_tuple;

typedef struct {
    Ipoint        uBoundsSize;   /* collection size */
    Ipsparameter_arith stLowerBound; /* lower amplitude bound pointer union */
    Ipsparameter_arith stUpperBound; /* upper amplitude bound pointer union */
    Ipin          iTypeOption;   /* pixel data type (parameter) option */
} Ipsattrs_bounds;

typedef struct {
    Ipin          bImmediate;    /* immediate mode flag */
    Ipin          iTypeOption;   /* virtual register data type (vreg) option */
} Ipsattrs_vreg;

typedef struct {
    Idntuple      nMatch;        /* match point SD 5-tuple */
    Idntuple      nROI;          /* ROI identifier */
    Idntuple      nOffset;       /* ROI offset SD 5-tuple */
} Ipsbind_image;

typedef struct {
    Ipoint        uHorizontal;    /* horizontal coordinate value */
    Ipoint        uVertical;      /* vertical coordinate value */
    Ipoint        uDepth;         /* depth coordinate value */
    Ipoint        uTemporal;      /* temporal coordinate value */
    Ipoint        uBand;          /* band coordinate value */
    union {
        Ipin      bValue;
        Ipoint     uValue;
        Ipin      iValue;
    } sPixel;
} Ipsdraw_pixels;

```

```

/*                                STRUCTURES (continued)                                */

typedef struct {
    Ipoint    uHorizontal;    /* horizontal coordinate value    */
    Ipoint    uVertical;     /* vertical coordinate value      */
    Ipoint    uDepth;        /* depth coordinate value         */
    Ipoint    uTemporal;     /* temporal coordinate value      */
    Ipoint    uBand;         /* band coordinate value          */
    Ibool     bPixelValue;   /* ROI pixel value                */
} Ipsdraw_pixels_roi;

typedef struct {
    char      zProfile[IL_PROFILE]; /* conformance profile label    */
    Idntuple  nDimensionLimit;     /* maximum dimension sizes identifier */
    Ipoint    iNDPrecision;        /* storage precision of ND data type */
    Ipoint    iSDPrecision;        /* storage precision of SD data type */
    Ipoint    iRDPrecision;        /* storage precision of RD data type */
    char      zRD[IL_REAL];        /* implementation method of RD data type */
    char      *tzInformation;       /* implementation information pointer */
    Ipebinary_event eAsynch;       /* asynchronous support            */
} Ipspiks_impl;

typedef struct {
    Iperoi_control_mode    eROIControlMode; /* PIKS ROI control mode    */
    Iperoi_process_mode    eROIProcessMode; /* PIKS ROI processing mode */
    Ipematch_mode          eMatchPointMode; /* PIKS match point control mode */
    Ipeindex_mode          eIndexAssignmentMode; /* PIKS index assignment mode */
    Ipesynchronicity_mode  eSynchronicityMode; /* PIKS synchronicity mode */
    Ipeerror_report_mode   eErrorReportMode; /* PIKS error report mode */
    Ipoint                 iImageResampleMode; /* PIKS image resampling selection mode */
    Ipoint                 iROIResampleMode; /* PIKS ROI resampling selection */
} Ipspiks_modes;

typedef struct {
    Ipepiks_operation_state    eOperationState; /* PIKS operational state    */
    Ipechain_state             eChainState;     /* PIKS chain state          */
    Ipeerror_state             eErrorState;      /* PIKS error state          */
    Ipesynchronicity_state     eSynchronicityState; /* PIKS synchronicity state */
} Ipspiks_states;

typedef struct {
    Ipevreg_chc    eChc; /* data source choice option */
    Ipsparameter_arith_vreg sData; /* data source */
} Ipsparameter_arith_vreg_chc;

typedef struct {
    Ipevreg_chc    eChc; /* data source choice option */
    Ipsbackground_vreg sOption; /* level slice background option source */
} Ipsbackground_vreg_chc;

```

```

/*                                STRUCTURES (continued)                                */

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsparameter_basic_vreg sData;      /* basic parameter data source        */
} Ipsparameter_basic_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsabool_vreg        sData;        /* Boolean data array source          */
} Ipsabool_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsbool_vreg         sData;        /* Boolean data value source          */
} Ipsbool_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipschar_vreg         sCharacter;    /* character source                    */
} Ipschar_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsparameter_colour_vreg sData;    /* colour data parameter source       */
} Ipsparameter_colour_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipscomplex_conv_vreg sOption;      /* option source                      */
} Ipscomplex_conv_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipscomplex_vreg      sData;        /* complex data value source          */
} Ipscomplex_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsderiv_vreg        sOption;      /* derivative option source           */
} Ipsderiv_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsdimension_vreg    sOption;      /* dimension option source            */
} Ipsdimension_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsparameter_edge_vreg sData;      /* edge parameter data source        */
} Ipsparameter_edge_vreg_chc;

```

```

/*                                STRUCTURES (continued)                                */

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipserosion_vreg      sOption;       /* erosion option source              */
} Ipserosion_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipspiks_pixel_types_vreg sData;     /* external image pixel data source   */
} Ipspiks_pixel_types_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsflip_vreg         sOption;       /* flip, spin, transpose option source */
} Ipsflip_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsfrequency_shift_vreg sOption;    /* frequency shift option source      */
} Ipsfrequency_shift_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsgamma_vreg        sOption;       /* gamma conversion option source     */
} Ipsgamma_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsindex_vreg        sOption;       /* index assignment option source     */
} Ipsindex_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsparameter_logical_vreg sAttrs;   /* logical data value source          */
} Ipsparameter_logical_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsmagnitude_vreg    sOption;       /* magnitude option source            */
} Ipsmagnitude_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsanbhood_vreg      sArrays;       /* neighbourhood array set source     */
} Ipsanbhood_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsaint_vreg         sData;         /* NP data array source              */
} Ipsaint_vreg_chc;

```

```

/*                                STRUCTURES (continued)                                */

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsuint_vreg         sData;         /* NP data value source              */
} Ipsuint_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsparameter_numeric_vreg sData;    /* numeric parameter data source      */
} Ipsparameter_numeric_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsopen_close_vreg   sData;         /* open close option source          */
} Ipsopen_close_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsparameter_pixel_vreg sData;      /* pixel parameter source            */
} Ipsparameter_pixel_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipspolarity_vreg     sPolarity;     /* polarity option source            */
} Ipspolarity_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsreflection_vreg    sOption;       /* reflection option source          */
} Ipsreflection_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsrotation_vreg     sOption;       /* rotate option source              */
} Ipsrotation_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsafloat_vreg       sData;         /* RP data array source              */
} Ipsafloat_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsfloat_vreg        sData;         /* RP data value source              */
} Ipsfloat_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsint_vreg          sData;         /* SP data source                    */
} Ipsint_vreg_chc;

```

```

/*                                STRUCTURES (concluded)                                */

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipstransform_vreg    sOption;       /* transform option source            */
} Ipstransform_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsparameter_vreg_vreg sValue;     /* virtual register virtual register data source */
} Ipsparameter_vreg_vreg_chc;

/*                                STRUCTURE-DEPENDENT UNIONS                                */

typedef union {
    Ipsattrs_bounds      *tsAttrs;      /* value bounds collection attributes pointer */
    Idnvreg              nVReg;         /* virtual register identifier              */
} Ipsattrs_bounds_vreg;

typedef union {
    Ipsattrs_hist        *tsAttrs;      /* histogram attributes pointer          */
    Idnvreg              nVReg;         /* virtual register identifier              */
} Ipsattrs_hist_vreg;

typedef union {
    Ipsattrs_id_array    *tsAttrsArray; /* composite id array attributes pointer  */
    Idnvreg              nVReg;         /* virtual register identifier              */
} Ipsattrs_id_array_vreg;

typedef union {
    Ipsattrs_id_record    *tsAttrsRecord; /* composite id record attributes pointer */
    Idnvreg              nVReg;         /* virtual register identifier              */
} Ipsattrs_id_record_vreg;

typedef union {
    Ipsattrs_image       *tsAttrs;      /* image attributes pointer              */
    Idnvreg              nVReg;         /* virtual register identifier              */
} Ipsattrs_image_vreg;

typedef union {
    Ipsattrs_lut          *tsAttrs;      /* lookup table attributes pointer        */
    Idnvreg              nVReg;         /* virtual register identifier              */
} Ipsattrs_lut_vreg;

typedef union {
    Ipsattrs_matrix       *tsAttrs;      /* matrix attributes pointer              */
    Idnvreg              nVReg;         /* virtual register identifier              */
} Ipsattrs_matrix_vreg;

```

```

/*          STRUCTURE-DEPENDENT UNIONS (continued)          */

typedef union {
    Ipsattrs_nbhood      *tsAttr;      /* neighbourhood array attributes pointer */
    Idnvreg              nVReg;        /* virtual register identifier */
} Ipsattrs_nbhood_vreg;

typedef union {
    Ipsattrs_id_array    *tsIdArray;   /* identifier array attributes pointer */
    Ipsattrs_id_record   *tsIdRecord;  /* identifier record attributes pointer */
    Ipsattrs_hist        *tsHist;      /* histogram attributes pointer */
    Ipsattrs_lut         *tsLUT;       /* lookup table attributes pointer */
    Ipsattrs_matrix      *tsMatrix;    /* matrix attributes pointer */
    Ipsattrs_nbhood      *tsNbhood;    /* neighbourhood array attributes pointer */
    Ipsattrs_pixel_record *tsPixel;    /* pixel record attributes pointer */
    Ipsattrs_roi_array   *tsROIArray;  /* ROI array attributes pointer */
    Ipsattrs_roi_coordinate *tsROICoordinate; /* ROI coordinate attributes pointer */
    Ipsattrs_roi_elliptical *tsROIElliptical; /* ROI elliptical attributes pointer */
    Ipsattrs_roi_generic *tsROIGeneric; /* ROI generic attributes pointer */
    Ipsattrs_roi_polygon *tsROIPolygon; /* ROI polygon attributes pointer */
    Ipsattrs_roi_rectangular *tsROIRectangular; /* ROI rectangular attributes pointer */
    Ipsattrs_static      *tsStatic;    /* static array attributes pointer */
    Ipsattrs_tuple       *tsTuple;     /* tuple attributes pointer */
    Ipsattrs_bounds      *tsBounds;    /* value bounds collection attributes pointer */
    Ipsattrs_vreg        *tsVReg;      /* virtual register attributes pointer */
} Ipsattrs_non_image;

typedef union {
    Ipsattrs_pixel_record *tsAttr;      /* pixel record attributes pointer */
    Idnvreg              nVReg;        /* virtual register identifier */
} Ipsattrs_pixel_record_vreg;

typedef union {
    Ipsattrs_roi         *tsAttr;      /* ROI attributes pointer */
    Idnvreg              nVReg;        /* virtual register identifier */
} Ipsattrs_roi_vreg;

typedef union {
    Ipsattrs_static      *tsAttr;      /* static array attributes pointer */
    Idnvreg              nVReg;        /* virtual register identifier */
} Ipsattrs_static_vreg;

typedef union {
    Ipsattrs_tuple       *tsAttr;      /* tuple attributes pointer */
    Idnvreg              nVReg;        /* virtual register identifier */
} Ipsattrs_tuple_vreg;

typedef union {
    Ipsattrs_vreg        *tsAttr;      /* virtual register attributes pointer */
    Idnvreg              nVReg;        /* virtual register identifier */
} Ipsattrs_vreg_vreg;

```

```

/*                      STRUCTURE-DEPENDENT UNIONS (concluded)                      */

typedef union {
    Ipsdraw_pixels      *tsaPixel;      /* pixel position and value */
    Idnvreg             nVReg;          /* virtual register identifier */
} Ipsdraw_pixels_vreg;

typedef union {
    Ipsdraw_pixels_roi  *tsaPixel;      /* pixel position and value */
    Idnvreg             nVReg;          /* virtual register identifier */
} Ipsdraw_pixels_roi_vreg;

typedef union {
    Ipspiks_modes       sMode;          /* PIKS modes settings */
    Idnvreg             nVReg;          /* virtual register identifier */
} Ipspiks_modes_vreg;

/*                      STRUCTURE-DEPENDENT UNION-DEPENDENT STRUCTURES                */

typedef struct {
    Ipevreg_chc         eChc;          /* data source choice option */
    Ipsattrs_bounds_vreg sAttrs;        /* value bounds attributes source */
} Ipsattrs_bounds_vreg_chc;

typedef struct {
    Ipevreg_chc         eChc;          /* data source choice option */
    Ipsattrs_hist_vreg  sAttrs;        /* histogram attributes source */
} Ipsattrs_hist_vreg_chc;

typedef struct {
    Ipevreg_chc         eChc;          /* data source choice option */
    Ipsattrs_id_array_vreg sAttrs;      /* composite id array attributes source */
} Ipsattrs_id_array_vreg_chc;

typedef struct {
    Ipevreg_chc         eChc;          /* data source choice option */
    Ipsattrs_id_record_vreg sAttrs;     /* composite id record attributes source */
} Ipsattrs_id_record_vreg_chc;

typedef struct {
    Ipevreg_chc         eChc;          /* data source choice option */
    Ipsattrs_image_vreg sAttrs;        /* image attributes source */
} Ipsattrs_image_vreg_chc;

typedef struct {
    Ipevreg_chc         eChc;          /* data source choice option */
    Ipsattrs_lut_vreg   sAttrs;        /* lookup table attributes source */
} Ipsattrs_lut_vreg_chc;

```



```

/*          STRUCTURE-DEPENDENT UNION-DEPENDENT STRUCTURES (concluded)          */

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsattrs_matrix_vreg sAttrs;        /* matrix attributes source           */
} Ipsattrs_matrix_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsattrs_nbhood_vreg sAttrs;        /* neighbourhood array attributes source */
} Ipsattrs_nbhood_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsattrs_pixel_record_vreg sAttrs;  /* pixel record attributes source      */
} Ipsattrs_pixel_record_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsattrs_roi_vreg    sAttrs;        /* ROI attributes source              */
} Ipsattrs_roi_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsattrs_static_vreg saAttrs;       /* static array attributes source      */
} Ipsattrs_static_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsattrs_tuple_vreg  sAttrs;        /* tuple attributes source            */
} Ipsattrs_tuple_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsattrs_vreg_vreg   sAttrs;        /* virtual register attributes source  */
} Ipsattrs_vreg_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsdraw_pixels_vreg  sData;         /* draw pixels image data source      */
} Ipsdraw_pixels_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipsdraw_pixels_roi_vreg sData;      /* draw pixels ROI data source        */
} Ipsdraw_pixels_roi_vreg_chc;

typedef struct {
    Ipevreg_chc          eChc;          /* data source choice option          */
    Ipspiks_modes_vreg   sOption;       /* PIKS modes option source          */
} Ipspiks_modes_vreg_chc;

```

B.1.3 Functions

```

/*                      FUNCTION PROTOTYPES                      */

extern void IvAbortAsynchExecution (                /* no return value output */
    void                                           /* no input arguments      */
);

extern Ipfloor *ItraAccumulator (
    Idnimage      nSourceImage, /* source image identifier */
    Ipfloor      *traSum,      /* sum of pixels data pointer */
    Ipeprocess    eMode,       /* processing mode option    */
    Ipoint        iApplBufferSize, /* data transfer buffer size or option */
    Ipoint        uStartPoint,  /* index offset within internal data */
    Ipoint        *tuDataSize   /* OUT post transfer data size pointer */
);

extern Idnimage InAdaptiveHistEqual (
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage,   /* destination image identifier */
    Ipoint        uPoints,      /* number of histogram reference points/index */
    Ipoint        uWindowSize,  /* window size */
    Ipoint        uOffset      /* number of lookup table source offset entries */
);

extern Idnvreg InAdaptiveHistEqualVReg (
    Idnvreg      nVreg,        /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage,  /* destination image identifier */
    Ipsuint_vreg_chc sPoints,    /* number of histogram reference points/index */
    Ipsuint_vreg_chc sWindowSize, /* window size */
    Ipsuint_vreg_chc sOffset     /* number of lookup table source offset entries */
);

extern Idnchain InAllocateChain (
    void
);

extern Idnvreg InAllocateChainVReg (
    Idnvreg      nVreg
);

extern Idnid_array InAllocateIdArray (
    Ipsattrs_id_array sAttrs
);

extern Idnvreg InAllocateIdArrayVReg (
    Idnvreg      nVreg,
    Ipsattrs_id_array_vreg_chc sAttrs
);

```

```

/*                                FUNCTION PROTOTYPES (continued)                                */

extern Idnid_list InAllocateIdList (                                /* OUT composite identifier list identifier */
    void                                /* no input arguments */
);

extern Idnvreg InAllocateIdListVReg (                                /* OUT composite identifier list identifier */
    Idnvreg                                nVReg                                /* output virtual register identifier */
);

extern Idnid_record InAllocateIdRecord (                                /* OUT composite identifier record identifier */
    Ipsattrs_id_record                                sAttrs                                /* composite identifier record attributes */
);

extern Idnvreg InAllocateIdRecordVReg (                                /* OUT composite identifier record identifier */
    Idnvreg                                nVReg,                                /* output virtual register identifier */
    Ipsattrs_id_record_vreg_chc                                sAttrs                                /* composite id record attributes */
);

extern Idnhist InAllocateHist (                                /* OUT histogram identifier */
    Ipsattrs_hist                                sAttrs                                /* histogram attributes */
);

extern Idnvreg InAllocateHistVReg (                                /* OUT histogram identifier */
    Idnvreg                                nVReg,                                /* output virtual register identifier */
    Ipsattrs_hist_vreg_chc                                sAttrs                                /* histogram attributes */
);

extern Idnimage InAllocateImage (                                /* OUT image identifier */
    Ipsattrs_image                                sAttrs                                /* image attributes */
);

extern Idnvreg InAllocateImageVReg (                                /* OUT image identifier */
    Idnvreg                                nVReg,                                /* output virtual register identifier */
    Ipsattrs_image_vreg_chc                                sAttrs                                /* image attributes */
);

extern Idnlut InAllocateLUT (                                /* OUT lookup table identifier */
    Ipsattrs_lut                                sAttrs                                /* lookup table attributes */
);

extern Idnvreg InAllocateLUTVReg (                                /* OUT lookup table identifier */
    Idnvreg                                nVReg,                                /* output virtual register identifier */
    Ipsattrs_lut_vreg_chc                                sAttrs                                /* lookup table attributes */
);

extern Idnmatrix InAllocateMatrix (                                /* OUT matrix identifier */
    Ipsattrs_matrix                                sAttrs                                /* matrix attributes */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InAllocateMatrixVReg (                                     /* OUT matrix identifier */
    Idnvreg      nVReg,                                                /* output virtual register identifier */
    Ipsattrs_matrix_vreg_chc  sAttrs                                  /* matrix attributes */
);

extern Idnnbhood InAllocateNbhoodArray (                               /* OUT neighbourhood array identifier */
    Ipsattrs_nbhood          sAttrs                                  /* neighbourhood array attributes */
);

extern Idnvreg InAllocateNbhoodArrayVReg (                             /* OUT neighbourhood array identifier */
    Idnvreg      nVReg,                                                /* output virtual register identifier */
    Ipsattrs_nbhood_vreg_chc  sAttrs                                  /* neighbourhood array attributes */
);

extern Idnpixel_record InAllocatePixelRecord (                         /* OUT pixel record identifier */
    Ipsattrs_pixel_record    sAttrs                                  /* pixel record attributes */
);

extern Idnvreg InAllocatePixelRecordVReg (                             /* OUT pixel record identifier */
    Idnvreg      nVReg,                                                /* output virtual register identifier */
    Ipsattrs_pixel_record_vreg_chc  sAttrs                          /* pixel record attributes */
);

extern Idnroi InAllocateROI (                                           /* OUT ROI identifier */
    Ipsattrs_roi            sAttrs                                  /* ROI attributes */
);

extern Idnvreg InAllocateROIIVReg (                                     /* OUT ROI identifier */
    Idnvreg      nVReg,                                                /* output virtual register identifier */
    Ipsattrs_roi_vreg_chc    sAttrs                                  /* ROI attributes */
);

extern Idnstatic InAllocateStaticArray (                               /* OUT static array identifier */
    Ipsattrs_static          sAttrs                                  /* static array attributes */
);

extern Idnvreg InAllocateStaticArrayVReg (                             /* OUT static array identifier */
    Idnvreg      nVReg,                                                /* output virtual register identifier */
    Ipsattrs_static_vreg_chc  sAttrs                                  /* static array attributes */
);

extern Idntuple InAllocateTuple (                                       /* OUT tuple identifier */
    Ipsattrs_tuple          sAttrs                                  /* tuple attributes */
);

extern Idnvreg InAllocateTupleVReg (                                     /* OUT tuple identifier */
    Idnvreg      nVReg,                                                /* output virtual register identifier */
    Ipsattrs_tuple_vreg_chc    sAttrs                                  /* tuple attributes */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnbounds InAllocateValueBounds (          /* OUT value bounds identifier */
    Ipsattrs_bounds          sAttrs          /* value bounds collection attributes */
);

extern Idnvreg InAllocateValueBoundsVReg (          /* OUT value bounds identifier */
    Idnvreg          nVReg,          /* output virtual register identifier */
    Ipsattrs_bounds_vreg_chc          sAttrs          /* value bounds collection attributes */
);

extern Idnvreg InAllocateVReg (          /* OUT virtual register identifier */
    Ipsattrs_vreg          sAttrs          /* virtual register attributes */
);

extern Idnvreg InAllocateVRegVReg (          /* OUT virtual register identifier */
    Idnvreg          nId,          /* output virtual register identifier */
    Ipsattrs_vreg_vreg_chc          sAttrs          /* virtual register attributes */
);

extern Idnimage InAlphaBlendConstant (          /* OUT destination image identifier */
    Idnimage          nSourceImage1,          /* first input source image identifier */
    Idnimage          nSourceImage2,          /* second input source image identifier */
    Idnimage          nDestImage,          /* destination image identifier */
    Ipfloat          rAlphaFactor          /* alpha factor */
);

extern Idnvreg InAlphaBlendConstantVReg (          /* OUT destination image identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg          snSourceImage1,          /* first input source image identifier */
    Ipsimage_vreg          snSourceImage2,          /* second input source image identifier */
    Ipsimage_vreg          snDestImage,          /* destination image identifier */
    Ipsfloat_vreg_chc          sAlphaFactor          /* alpha factor */
);

extern Idnimage InAlphaBlendVariable (          /* OUT destination image identifier */
    Idnimage          nSourceImage1,          /* first input source image identifier */
    Idnimage          nSourceImage2,          /* second input source image identifier */
    Idnimage          nSourceImage3,          /* third input source image identifier */
    Idnimage          nSourceImage4,          /* fourth input source image identifier */
    Idnimage          nDestImage          /* destination image identifier */
);

extern Idnvreg InAlphaBlendVariableVReg (          /* OUT destination image identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg          snSourceImage1,          /* first input source image identifier */
    Ipsimage_vreg          snSourceImage2,          /* second input source image identifier */
    Ipsimage_vreg          snSourceImage3,          /* third input source image identifier */
    Ipsimage_vreg          snSourceImage4,          /* fourth input source image identifier */
    Ipsimage_vreg          snDestImage          /* destination image identifier */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnlut InArrayToLUT (                                /* OUT lookup table identifier */
    Idnlut          nLUT,                                   /* lookup table identifier      */
    Ipsparameter_pixel stArray,                             /* lookup table value array pointer union */
    Ipoint          uEntryOffset,                          /* lookup table entry offset    */
    Ipoint          uBandOffset,                          /* lookup table band offset     */
    Ipoint          uColumns,                             /* number of array columns     */
    Ipoint          uRows,                                /* number of array rows        */
);

extern Idnvreg InArrayToLUTVReg (                          /* OUT lookup table identifier */
    Idnvreg          nVReg,                                /* output virtual register identifier or NULL */
    Ipslut_vreg      snLUT,                                /* lookup table identifier      */
    Ipsparameter_pixel_vreg_chc sArray,                   /* lookup table value array     */
    Ipsuint_vreg_chc sEntryOffset,                        /* lookup table entry offset    */
    Ipsuint_vreg_chc sBandOffset,                        /* lookup table band offset     */
    Ipsuint_vreg_chc sColumns,                           /* number of array columns     */
    Ipsuint_vreg_chc sRows,                              /* number of array rows        */
);

extern Idnimage InBindMatchPoint (                        /* OUT source image identifier */
    Idnimage          nSourceImage,                       /* source image identifier      */
    Idntuple          nMatch,                             /* match point RD 5-tuple identifier */
);

extern Idnvreg InBindMatchPointVReg (                    /* OUT source image identifier */
    Idnvreg          nVReg,                                /* output virtual register identifier or NULL */
    Ipsimage_vreg     snSourceImage,                     /* source image identifier      */
    Ipstuple_vreg     snMatch,                            /* match point RD 5-tuple identifier */
);

extern Idnimage InBindROI (                              /* OUT source image identifier */
    Idnimage          nSourceImage,                       /* source image identifier      */
    Idnroi            nROI,                               /* ROI identifier              */
    Idntuple          nOffset,                            /* offset SD 5-tuple identifier */
);

extern Idnvreg InBindROIIVReg (                          /* OUT source image identifier */
    Idnvreg          nVReg,                                /* output virtual register identifier or NULL */
    Ipsimage_vreg     snSourceImage,                     /* source image identifier      */
    Ipsroi_vreg       snROI,                             /* ROI identifier              */
    Ipstuple_vreg     snOffset,                           /* offset SD 5-tuple identifier */
);

extern Idnimage InBitShift (                             /* OUT destination image identifier */
    Idnimage          nSourceImage,                       /* source image identifier      */
    Idnimage          nDestImage,                         /* destination image identifier  */
    Ipoint            uBits,                              /* number of bits to be shifted */
    Ipint             iOption,                            /* bit shift option            */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InBitShiftVReg (                                           /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipsuint_vreg_chc sBits,        /* number of bits to be shifted */
    Ipsint_vreg_chc  sOption       /* bit shift option */
);

extern Idnimage InCartesianToPolar ( /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Ipfloat           rMinimumRho,  /* minimum value of rho */
    Ipfloat           rMaximumRho,  /* maximum value of rho */
    Ipfloat           rMinimumTheta, /* minimum value of theta */
    Ipfloat           rMaximumTheta /* maximum value of theta */
);

extern Idnvreg InCartesianToPolarVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipsfloat_vreg_chc sMinimumRho, /* minimum value of rho */
    Ipsfloat_vreg_chc sMaximumRho, /* maximum value of rho */
    Ipsfloat_vreg_chc sMinimumTheta, /* minimum value of theta */
    Ipsfloat_vreg_chc sMaximumTheta /* maximum value of theta */
);

extern Idnchain InChainAbort ( /* OUT chain identifier */
    Idnchain          nChain /* chain identifier */
);

extern Idnvreg InChainAbortVReg ( /* OUT chain identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipschain_vreg    snVReg /* chain identifier */
);

extern Idnchain InChainBegin ( /* OUT chain identifier */
    Idnchain          nChain /* chain identifier */
);

extern Idnchain InChainDelete ( /* OUT chain identifier */
    Idnchain          nChain /* chain identifier */
);

extern Idnvreg InChainDeleteVReg ( /* OUT chain identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipschain_vreg    snVReg /* chain identifier */
);

```



```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnchain InChainEnd (                      /* OUT chain identifier */
    Idnchain          nChain                      /* chain identifier      */
);

extern Idnchain InChainExecute (                  /* OUT chain identifier */
    Idnchain          nChain                      /* chain identifier      */
);

extern Idnvreg InChainExecuteVReg (              /* OUT chain identifier */
    Idnvreg          nVReg,                      /* output virtual register identifier or NULL */
    Ipschain_vreg    snVReg                      /* chain identifier      */
);

extern Idnchain InChainReload (                  /* OUT chain identifier */
    Idnchain          nChain,                    /* chain identifier      */
    Ipuint            uElement,                  /* element selector      */
    Ipnarg            nArgs                      /* argument parameter collection identifier */
);

extern Idnvreg InChainReloadVReg (              /* OUT chain identifier */
    Idnvreg          nVReg,                      /* output virtual register identifier or NULL */
    Ipschain_vreg    snVReg,                    /* chain identifier      */
    Ipsuint_vreg_chc sElement,                  /* element selector      */
    Ipsarg_vreg      snArgs                     /* argument parameter collection identifier */
);

extern Idnimage InClassifierBayes (              /* OUT destination image identifier */
    Idnimage          nSourceImage,             /* source image identifier */
    Idnimage          nDestImage,               /* destination image identifier */
    Ipfloating         *traProbability,          /* class a priori probability array pointer */
    Ipfloating         *traMeanVector,           /* class mean vector array pointer */
    Ipfloating         *traCovariance,           /* class covariance matrix array pointer */
    Ipuint            uClasses,                 /* number of classes */
    Ipuint            iApplBufferSize,          /* data transfer buffer size or option of each array */
    Ipuint            uStartPoint,              /* index offset within internal data of each array */
    Ipuint            *tuDataSize              /* OUT post transfer data size pointer */
);

extern Idnvreg InClassifierBayesVReg (          /* OUT destination image identifier */
    Idnvreg          nVReg,                      /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,             /* source image identifier */
    Ipsimage_vreg    snDestImage,               /* destination image identifier */
    Ipsafloat_vreg_chc sProbability,             /* class a priori probability array */
    Ipsafloat_vreg_chc sMeanVector,             /* class mean vector array */
    Ipsafloat_vreg_chc sCovariance,             /* class covariance matrix array */
    Ipsuint_vreg_chc sClasses                   /* number of classes */
);

```



```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnimage InClassifierNearestNbr (                                /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    lpfloor           *traTraining, /* training pattern set array pointer */
    lpuint            uClasses,     /* number of classes */
    lpuint            uPatterns,    /* number of patterns per class */
    lpint             iOption,      /* distance measure option */
    lpint             iApplBufferSize, /* data transfer buffer size or option */
    lpuint            uStartPoint,   /* index offset within internal data */
    lpuint            *tuDataSize   /* OUT post transfer data size pointer */
);

extern Idnvreg InClassifierNearestNbrVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    lpsimage_vreg    snSourceImage, /* source image identifier */
    lpsimage_vreg    snDestImage,   /* destination image identifier */
    lpsafloat_vreg_chc sTraining, /* training pattern set array */
    lpsuint_vreg_chc  sClasses,    /* number of classes */
    lpsuint_vreg_chc  sPatterns,   /* number of patterns per class */
    lpsint_vreg_chc   sOption      /* distance measure option */
);

extern void IvClosePIKS ( /* no return value output */
    void /* no input arguments */
);

extern void IvClosePIKSEmergency ( /* no return value output */
    void /* no input arguments */
);

extern Idnimage InColourConvLin ( /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Idnmatrix         nColourMatrix /* colour conversion matrix identifier or NULL */
);

extern Idnvreg InColourConvLinVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    lpsimage_vreg    snSourceImage, /* source image identifier */
    lpsimage_vreg    snDestImage,   /* destination image identifier */
    lpsmatrix_vreg    snColourMatrix /* colour conversion matrix identifier or NULL */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern void IvColourConvMatrix (                                           /* no return value output */
    Idnmatrix nXYZMatrixRGB, /* OUT XYZ-RGB conversion matrix identifier */
    Idnmatrix nRGBMatrixXYZ, /* OUT RGB-XYZ conversion matrix identifier */
    Ipfloor rRedXChrom, /* red primary x chromaticity coordinate */
    Ipfloor rRedYChrom, /* red primary y chromaticity coordinate */
    Ipfloor rGreenXChrom, /* green primary x chromaticity coordinate */
    Ipfloor rGreenYChrom, /* green primary y chromaticity coordinate */
    Ipfloor rBlueXChrom, /* blue primary x chromaticity coordinate */
    Ipfloor rBlueYChrom, /* blue primary y chromaticity coordinate */
    Ipfloor rWhiteXChrom, /* white reference x chromaticity coordinate */
    Ipfloor rWhiteYChrom /* white reference y chromaticity coordinate */
);

extern void IvColourConvMatrixVReg (                                       /* no return value output */
    Idnvreg nXYZMatrixRGB, /* OUT XYZ-RGB conversion matrix identifier */
    Idnvreg nRGBMatrixXYZ, /* OUT RGB-XYZ conversion matrix identifier */
    Ipsfloat_vreg_chc sRedXChrom, /* red primary x chromaticity coordinate */
    Ipsfloat_vreg_chc sRedYChrom, /* red primary y chromaticity coordinate */
    Ipsfloat_vreg_chc sGreenXChrom, /* green primary x chromaticity coordinate */
    Ipsfloat_vreg_chc sGreenYChrom, /* green primary y chromaticity coordinate */
    Ipsfloat_vreg_chc sBlueXChrom, /* blue primary x chromaticity coordinate */
    Ipsfloat_vreg_chc sBlueYChrom, /* blue primary y chromaticity coordinate */
    Ipsfloat_vreg_chc sWhiteXChrom, /* white reference x chromaticity coordinate */
    Ipsfloat_vreg_chc sWhiteYChrom /* white reference y chromaticity coordinate */
);

extern Idnimage InColourConvNonlin (                                       /* OUT destination image identifier */
    Idnimage nSourceImage, /* source image identifier */
    Idnimage nDestImage, /* destination image identifier */
    Ipfloor rXWhite, /* X tristimulus value white point or 0 */
    Ipfloor rYWhite, /* Y tristimulus value white point or 0 */
    Ipfloor rZWhite, /* Z tristimulus value white point or 0 */
    Ipoint iRGBOption, /* RGB colour option */
    Ipoint iInputOption, /* colour conversion input option */
    Ipoint iOutputOption /* colour conversion output option */
);

extern Idnvreg InColourConvNonlinVReg ( /* OUT destination image identifier */
    Idnvreg nVReg, /* output virtual register identifier of NULL */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage, /* destination image identifier */
    Ipsfloat_vreg_chc sXWhite, /* X tristimulus value white point or 0 */
    Ipsfloat_vreg_chc sYWhite, /* Y tristimulus value white point or 0 */
    Ipsfloat_vreg_chc sZWhite, /* Z tristimulus value white point or 0 */
    Ipsint_vreg_chc sRGBOption, /* RGB colour option */
    Ipsint_vreg_chc sInputOption, /* colour conversion input option */
    Ipsint_vreg_chc sOutputOption /* colour conversion output option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnimage InColourConvSubtractive ( /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage,   /* destination image identifier */
    lpsfloat      rBlackFactor, /* blackness factor */
    lpuint        uIntegerSize, /* maximum integer size */
    lpsparameter_arith stMethod, /* method data pointer union */
    lpint         iDirectionOption, /* colour conversion, direction option */
    lpint         iMethodOption /* colour conversion, method option */
);

extern Idnvreg InColourConvSubtractiveVReg ( /* OUT destination image identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    lpsimage_vreg snSourceImage, /* source image identifier */
    lpsimage_vreg snDestImage,   /* destination image identifier */
    lpsfloat_vreg_chc sBlackFactor, /* blackness factor */
    lpsuint_vreg_chc sIntegerSize, /* maximum integer size */
    lpsparameter_arith_vreg_chc sMethod, /* method data */
    lpsint_vreg_chc sDirectionOption, /* colour conversion, direction option */
    lpsint_vreg_chc sMethodOption /* colour conversion, method option */
);

extern Idnimage InColourLookupInterp ( /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage,   /* destination image identifier */
    Idnlut        nTable,       /* colour lookup table identifier */
    Idntuple      nCellCount,   /* number of cells ND B-tuple identifier */
    Idntuple      nCellSize,    /* size of cells RD B-tuple identifier */
    Idntuple      nDimensions /* array dimension ND B-tuple identifier */
);

extern Idnvreg InColourLookupInterpVReg ( /* OUT destination image identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    lpsimage_vreg snSourceImage, /* source image identifier */
    lpsimage_vreg snDestImage,   /* destination image identifier */
    lpslut_vreg   snTable,       /* colour lookup table identifier */
    lpstuple_vreg snCellCount,   /* number of cells ND B-tuple identifier */
    lpstuple_vreg snCellSize,    /* size of cells RD B-tuple identifier */
    lpstuple_vreg snDimensions /* array dimension ND B-tuple identifier */
);

extern Idnimage InComplement ( /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage,   /* destination image identifier */
);

extern Idnvreg InComplementVReg ( /* OUT destination image identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    lpsimage_vreg snSourceImage, /* source image identifier */
    lpsimage_vreg snDestImage,   /* destination image identifier */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InComplementROIVReg ( /* OUT destination ROI identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsroi_vreg      snSourceROI, /* source ROI identifier */
    Ipsroi_vreg      snDestROI /* destination ROI identifier */
);

extern Idnvreg InComplementVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg     snSourceImage, /* source image identifier */
    Ipsimage_vreg     snDestImage /* destination image identifier */
);

extern Idnimage InComplexComposition ( /* OUT destination image identifier */
    Idnimage          nSourceImage1, /* first source image identifier */
    Idnimage          nSourceImage2, /* second source image identifier */
    Idnimage          nDestImage, /* destination image identifier */
    Ipecomplex_conv    eOption /* complex conversion option */
);

extern Idnvreg InComplexCompositionVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg     snSourceImage1, /* first source image identifier */
    Ipsimage_vreg     snSourceImage2, /* second source image identifier */
    Ipsimage_vreg     snDestImage, /* destination image identifier */
    Ipscomplex_conv_vreg_chc sOption /* complex conversion option */
);

extern Idnimage InComplexConjugate ( /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage /* destination image identifier */
);

extern Idnvreg InComplexConjugateVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg     snSourceImage, /* source image identifier */
    Ipsimage_vreg     snDestImage /* destination image identifier */
);

extern void IvComplexDecomposition ( /* no return value output */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestFirst, /* OUT first destination image identifier */
    Idnimage          nDestSecond, /* OUT second destination image identifier */
    Ipecomplex_conv    eOption /* complex conversion option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern void IvComplexDecompositionVReg (                                /* no return value output */
    Ipsimage_vreg          snSourceImage, /* source image identifier */
    Idnvreg                nDestFirst,    /* OUT first destination image identifier */
    Idnvreg                nDestSecond,   /* OUT second destination image identifier */
    Ipscomplex_conv_vreg_chc sOption      /* complex conversion option */
);

extern Idnimage InComplexMagnitude ( /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Ipemagnitude      eOption       /* complex magnitude option */
);

extern Ipsimage_vreg InComplexMagnitudeVReg ( /* OUT destination image identifier */
    Ipsimage_vreg          snSourceImage, /* source image identifier */
    Ipsimage_vreg          snDestImage,   /* destination image identifier */
    Ipsmagnitude_vreg_chc sOption        /* complex magnitude option */
);

extern Ipbool IbIdArrayEqual ( /* OUT equality flag */
    Idnid_array          nFirst, /* first image identifier array identifier */
    Idnid_array          nSecond /* second image identifier array identifier */
);

extern Idnvreg InIdArrayEqualVReg ( /* OUT equality flag virtual register identifier */
    Idnvreg          nVReg, /* output virtual register identifier */
    Ipsid_array_vreg snFirst, /* first image identifier array identifier */
    Ipsid_array_vreg snSecond /* second image identifier array identifier */
);

extern Idnimage InIdArrayGet ( /* OUT image identifier */
    Idnid_array          nIdArray, /* composite image identifier array identifier */
    Ipuint               uIndex    /* array index value */
);

extern Idnvreg InIdArrayGetVReg ( /* OUT image identifier */
    Idnvreg          nVReg, /* output virtual register identifier */
    Ipsid_array_vreg snIdArray, /* composite image identifier array identifier */
    Ipsuint_vreg_chc sIndex    /* array index value */
);

extern Idnid_array InIdArrayPut ( /* OUT image identifier array identifier */
    Idnimage          nImage, /* source image identifier */
    Idnid_array       nIdArray, /* composite image identifier array identifier */
    Ipuint            uIndex    /* array index value */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InIdArrayPutVReg (                                           /* OUT image identifier array identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg    snImage,        /* source image identifier */
    Ipsid_array_vreg snIdArray,      /* composite image identifier array identifier */
    Ipsuint_vreg_chc sIndex          /* array index value */
);

extern Ipbool IbIdListEmpty (                                              /* OUT empty flag */
    Idnid_list       nList           /* composite image identifier list identifier */
);

extern Idnvreg InIdListEmptyVReg (                                         /* OUT empty flag */
    Idnvreg          nVReg,          /* output virtual register identifier */
    Ipsid_list_vreg  snList          /* composite image identifier list identifier */
);

extern Ipbool IbIdListEqual (                                              /* OUT equality flag */
    Idnid_list       nFirst,         /* first image identifier list identifier */
    Idnid_list       nSecond        /* second image identifier list identifier */
);

extern Idnvreg InIdListEqualVReg (                                         /* OUT equality flag */
    Idnvreg          nVReg,          /* output virtual register identifier */
    Ipsid_list_vreg  snFirst,        /* first image identifier list identifier */
    Ipsid_list_vreg  snSecond       /* second image identifier list identifier */
);

extern Idnimage InIdListGet (                                              /* OUT image identifier */
    Idnid_list       nIdList,        /* composite image identifier list identifier */
    Ipint            iOption         /* composite list pointer option */
);

extern Idnvreg InIdListGetVReg (                                           /* OUT image identifier */
    Idnvreg          nVReg,          /* output virtual register identifier */
    Ipsid_list_vreg  snIdList,       /* composite image identifier list identifier */
    Ipsint_vreg_chc sOption         /* composite list pointer option */
);

extern Idnid_list InIdListInsert (                                         /* OUT composite identifier list identifier */
    Idnimage         nImage,         /* source image identifier */
    Idnid_list       nIdList,        /* list identifier */
    Ipint            iOption         /* list pointer option */
);

extern Idnvreg InIdListInsertVReg (                                        /* OUT composite identifier list identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg    snImage,        /* source image identifier */
    Ipsid_list_vreg  snIdList,       /* list identifier */
    Ipsint_vreg_chc sOption         /* list pointer option */
);

```



```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnid_list InIdListRemove (                                     /* OUT composite image identifier list identifier */
    Idnid_list          nList                                       /* composite image identifier list identifier */
);

extern Idnvreg InIdListRemoveVReg (                                  /* OUT composite image identifier list identifier */
    Idnvreg             nVReg,                                       /* output virtual register identifier or NULL */
    Ipsid_list_vreg     snList                                       /* composite image identifier list identifier */
);

extern Ipbool IbIdRecordEqual (                                     /* OUT equality flag */
    Idnid_record        nFirst,                                       /* 1st image identifier record identifier */
    Idnid_record        nSecond                                       /* 2nd image identifier record identifier */
);

extern Idnvreg InIdRecordEqualVReg (                                /* OUT equality flag */
    Idnvreg             nVReg,                                       /* output virtual register identifier */
    Ipsid_record_vreg   snFirst,                                       /* 1st image identifier record identifier */
    Ipsid_record_vreg   snSecond                                       /* 2nd image identifier record identifier */
);

extern Idnimage InIdRecordGet (                                     /* OUT image identifier */
    Idnid_record        nIdRecord,                                       /* composite image identifier record identifier */
    char                *tzName                                           /* attribute name pointer */
);

extern Idnvreg InIdRecordGetVReg (                                  /* OUT image identifier */
    Idnvreg             nVReg,                                       /* output virtual register identifier */
    Ipsid_record_vreg   snIdRecord,                                       /* composite image identifier record identifier */
    Ipschar_vreg_chc    sName                                           /* attribute name */
);

extern Idnid_record InIdRecordPut (                                  /* OUT image identifier record identifier */
    Idnimage            nImage,                                       /* source image identifier */
    Idnid_record        nIdRecord,                                       /* composite image identifier record identifier */
    char                *tzName                                           /* attribute name pointer */
);

extern Idnvreg InIdRecordPutVReg (                                  /* OUT image identifier record identifier */
    Idnvreg             nVReg,                                       /* output virtual register identifier or NULL */
    Ipsimage_vreg       snImage,                                       /* source image identifier */
    Ipsid_record_vreg   snIdRecord,                                       /* composite image identifier record identifier */
    Ipschar_vreg_chc    sName                                           /* attribute name */
);

extern Ipbool IbConstantPredicate (                                  /* OUT predicate flag */
    Idnimage            nSourceImage,                                       /* source image identifier */
    Ipfloat             rPixelValue,                                       /* pixel value */
    Ipfloat             rPixelTolerance                                       /* pixel tolerance */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InConstantPredicateVReg ( /* OUT predicate flag */
    Idnvreg          nVReg, /* output virtual register identifier */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsfloat_vreg_chc sPixelValue, /* pixel value */
    Ipsfloat_vreg_chc sPixelTolerance /* pixel tolerance */
);

extern Idnimage InConvertArrayToImage ( /* OUT destination image identifier */
    Idnimage      nImage, /* destination image identifier */
    Ipsarray_id   snArray /* array identifier */
);

extern Idnvreg InConvertArrayToImageVReg ( /* OUT destination image identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg snImage, /* destination image identifier */
    Ipsarray_id_vreg snArray /* array identifier */
);

extern Idnimage InConvertImageDatatype ( /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage /* destination image identifier */
);

extern Idnvreg InConvertImageDatatypeVReg ( /* OUT destination image identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage /* destination image identifier */
);

extern Ipsarray_id IsnConvertImageToArray ( /* OUT array identifier */
    Idnimage      nImage, /* source image identifier */
    Ipsarray_id   snArray /* array identifier */
);

extern Idnvreg InConvertImageToArrayVReg ( /* OUT array identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg snImage, /* source image identifier */
    Ipsarray_id_vreg snArray /* array identifier */
);

extern Idnroi InConvertImageToROI ( /* OUT ROI identifier */
    Idnimage      nImage, /* source image identifier */
    Idnroi        nROI /* ROI identifier */
);

extern Idnvreg InConvertImageToROIvReg ( /* OUT ROI identifier */
    Ipsimage_vreg snImage, /* source image identifier */
    Ipsroi_vreg   snROI /* ROI identifier */
);

```



```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnimage InConvertROIToImage (          /* OUT destination image identifier */
    Idnimage          nImage,          /* destination image identifier */
    Idnroi            nROI             /* ROI identifier */
);

extern Idnvreg InConvertROIToImageVReg (        /* OUT destination image identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg    snImage,        /* destination image identifier */
    Ipsroi_vreg      snROI           /* ROI identifier */
);

extern Idnimage InConvolve2D (                /* OUT destination image identifier */
    Idnimage          nSourceImage,   /* source image identifier */
    Idnimage          nDestImage,     /* destination image identifier */
    Idnnbhood         nImpulse,       /* impulse response array identifier */
    Ipint             iOption         /* convolution option */
);

extern Idnvreg InConvolve2DVReg (             /* OUT destination image identifier */
    Ipsimage_vreg     snSourceImage,   /* source image identifier */
    Ipsimage_vreg     snDestImage,     /* destination image identifier */
    Ipsnbhood_vreg    snImpulse,       /* impulse response array identifier */
    Ipsint_vreg_chc   sOption         /* convolution option */
);

extern Idnimage InConvolve5D (                /* OUT destination image identifier */
    Idnimage          nSourceImage,   /* source image identifier */
    Idnimage          nDestImage,     /* destination image identifier */
    Idnnbhood         nImpulse       /* impulse response array identifier */
);

extern Idnvreg InConvolve5DVReg (             /* OUT destination image identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg     snSourceImage,   /* source image identifier */
    Ipsimage_vreg     snDestImage,     /* destination image identifier */
    Ipsnbhood_vreg    snImpulse       /* impulse response array identifier */
);

extern Idnimage InCopyWindow (                /* OUT destination image identifier */
    Idnimage          nSourceImage,   /* source image identifier */
    Idnimage          nDestImage,     /* destination image identifier */
    Idntuple          nSourceOffset,   /* source index offset SD 5-tuple identifier */
    Idntuple          nDestOffset,     /* destination index offset SD 5-tuple identifier */
    Idntuple          nWindowSize     /* window size ND 5-tuple identifier */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InCopyWindowVReg (                                           /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipstuple_vreg    snSourceOffset, /* source index offset SD 5-tuple identifier */
    Ipstuple_vreg    snDestOffset,  /* destination index offset SD 5-tuple identifier */
    Ipstuple_vreg    snWindowSize   /* window size ND 5-tuple identifier */
);

extern Idnroi InCopyWindowROI (                                           /* OUT destination ROI identifier */
    Idnroi           nSourceROI,    /* source ROI identifier */
    Idnroi           nDestROI,      /* destination ROI identifier */
    Idntuple         nSourceOffset, /* source index offset SD 5-tuple identifier */
    Idntuple         nDestOffset,   /* destination index offset SD 5-tuple identifier */
    Idntuple         nWindowSize    /* window size ND 5-tuple identifier */
);

extern Idnvreg InCopyWindowROIIVReg ( /* OUT destination ROI identifier */
    Idnvreg      nVReg, /* output virtual register identifier */
    Ipsroi_vreg  snSourceROI, /* source ROI identifier */
    Ipsroi_vreg  snDestROI,   /* destination ROI identifier */
    Ipstuple_vreg snSourceOffset, /* source index offset SD 5-tuple identifier */
    Ipstuple_vreg snDestOffset,   /* destination index offset SD 5-tuple identifier */
    Ipstuple_vreg snWindowSize    /* window size ND 5-tuple identifier */
);

extern Idntuple InCreateTuple ( /* OUT tuple data identifier */
    Ipsparameter_basic stTupleData, /* tuple data pointer union */
    Idntuple           nTupleId,    /* tuple identifier */
    Ipsattrs_tuple     sTupleAttrs  /* tuple attributes */
);

extern Idnvreg InCreateTupleVReg ( /* OUT tuple data identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    Ipsparameter_basic_vreg_chc sTupleData, /* tuple data */
    Ipstuple_vreg snTupleId, /* tuple identifier */
    Ipsattrs_tuple_vreg_chc sTupleAttrs /* tuple attributes */
);

extern Idnstatic InCrossCorrelation ( /* OUT cross-correlation array identifier */
    Idnimage      nSourceImage1, /* first source image identifier */
    Idnimage      nSourceImage2, /* second source image identifier */
    Idnstatic     nCorrelation,   /* cross-correlation RD array identifier */
    Ipoint        uIndex3,       /* third index value */
    Ipoint        uIndex4,       /* fourth index value */
    Ipoint        uIndex5,       /* fifth index value */
    Ipoint        uColumns,      /* number of array columns */
    Ipoint        uRows          /* number of array rows */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InCrossCorrelationVReg (                                /* OUT cross-correlation array identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage1, /* first source image identifier */
    Ipsimage_vreg    snSourceImage2, /* second source image identifier */
    Ipsstatic_vreg   snCorrelation, /* cross-correlation array identifier */
    Ipsuint_vreg_chc sIndex3,     /* third index value */
    Ipsuint_vreg_chc sIndex4,     /* fourth index value */
    Ipsuint_vreg_chc sIndex5,     /* fifth index value */
    Ipsuint_vreg_chc sColumns,    /* number of array columns */
    Ipsuint_vreg_chc sRows        /* number of array rows */
);

extern void IvDeallocateDataObject (                                /* no return value output */
    Ipsobject          snObject /* data identifier */
);

extern void IvDeallocateDataObjectVReg (                          /* no return value output */
    Ipsobject_vreg     snObject /* data identifier */
);

extern Idnimage InDefineSubImage (                                /* OUT destination sub image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination sub image identifier */
    Idntuple          nOffset,      /* index offset ND 5-tuple */
    Idntuple          nSubImageSize /* sub image size ND 5-tuple */
);

extern Idnvreg InDefineSubImageVReg (                            /* OUT destination sub image identifier */
    Idnvreg          nVReg,         /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,   /* destination sub image identifier */
    Ipstuple_vreg     snOffset,      /* index offset ND 5-tuple */
    Ipstuple_vreg     snSubImageSize /* sub image size ND 5-tuple */
);

extern Ipfloat IrDiffMeasures (                                /* OUT distance measure value */
    Idnimage          nSourceImage1, /* first source image identifier */
    Idnimage          nSourceImage2, /* second source image identifier */
    Ipint             iOption        /* difference measures option */
);

extern Idnvreg InDiffMeasuresVReg (                            /* OUT distance measure value */
    Idnvreg          nVReg,         /* output virtual register identifier */
    Ipsimage_vreg    snSourceImage1, /* first source image identifier */
    Ipsimage_vreg    snSourceImage2, /* second source image identifier */
    Ipsint_vreg_chc  sOption        /* difference measures option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnimage InDiffuse (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier          */
    Idntuple          nAmplitudeLevel, /* destination image identifier     */
); /* destination amplitude ND B-tuple identifier */

extern Idnvreg InDiffuseVReg (
    Idnvreg          nVReg, /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier          */
    Ipstuple_vreg    snAmplitudeLevel, /* destination image identifier     */
); /* destination amplitude ND B-tuple identifier */

extern Idnimage InDither (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier          */
    Idnnbhood         nDither,      /* destination image identifier     */
    Idntuple          nAmplitudeLevel, /* dither array identifier         */
); /* destination amplitude ND B-tuple identifier */

extern Idnvreg InDitherVReg (
    Idnvreg          nVReg, /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier          */
    Ipsnbhood_vreg   snDither,      /* destination image identifier     */
    Ipstuple_vreg    snAmplitudeLevel, /* dither array identifier         */
); /* destination amplitude ND B-tuple identifier */

extern Idnimage InDrawPixels (
    Ipsdraw_pixels    *tsaPixels, /* OUT destination image identifier */
    Idnimage          nDestImage, /* pointer to array of pixel collection structures */
    Ipuint            uEntries,    /* destination image identifier     */
); /* number of array entries */

extern Idnvreg InDrawPixelsVReg (
    Ipsdraw_pixels_vreg_chc    saPixels, /* OUT destination image identifier */
    Ipsimage_vreg              snDestImage, /* pixel collection structure array choice */
    Ipsuint_vreg_chc           sEntries,    /* destination image identifier     */
); /* number of array entries */

extern Idnroi InDrawPixelsROI (
    Ipsdraw_pixels_roi    *tsaPixels, /* OUT destination ROI identifier */
    Idnroi                nDestROI,   /* pointer to array of data collection structures */
    Ipuint                uEntries,    /* destination ROI identifier     */
); /* number of array entries */

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InDrawPixelsROIvReg (                                     /* OUT destination ROI identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsdraw_pixels_roi_vreg_chc saPixels,                             /* data collection structure array choice */
    Ipsroi_vreg      snDestROI,                                       /* destination ROI identifier */
    Ipsuint_vreg_chc sEntries                                         /* number of array entries */
);

extern Idnimage InDyadicArith (                                       /* OUT destination image identifier */
    Idnimage          nSourceImage1, /* first source image identifier */
    Idnimage          nSourceImage2, /* second source image identifier */
    Idnimage          nDestImage,    /* destination image identifier */
    Ipint             iOption        /* dyadic arithmetic combination option */
);

extern Idnvreg InDyadicArithVReg (                                    /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage1, /* first source image identifier */
    Ipsimage_vreg    snSourceImage2, /* second source image identifier */
    Ipsimage_vreg    snDestImage,    /* destination image identifier */
    Ipsint_vreg_chc  sOption        /* dyadic arithmetic combination option */
);

extern Idnimage InDyadicComplex (                                    /* OUT destination image identifier */
    Idnimage          nSourceImage1, /* first source image identifier */
    Idnimage          nSourceImage2, /* second source image identifier */
    Idnimage          nDestImage,    /* destination image identifier */
    Ipint             iOption        /* dyadic complex combination option */
);

extern Idnvreg InDyadicComplexVReg (                                /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage1, /* first source image identifier */
    Ipsimage_vreg    snSourceImage2, /* second source image identifier */
    Ipsimage_vreg    snDestImage,    /* destination image identifier */
    Ipsint_vreg_chc  sOption        /* dyadic complex combination option */
);

extern Idnimage InDyadicLogical (                                    /* OUT destination image identifier */
    Idnimage          nSourceImage1, /* first source image identifier */
    Idnimage          nSourceImage2, /* second source image identifier */
    Idnimage          nDestImage,    /* destination image identifier */
    Ipint             iOption        /* dyadic logical combination option */
);

extern Idnvreg InDyadicLogicalVReg (                                /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage1, /* first source image identifier */
    Ipsimage_vreg    snSourceImage2, /* second source image identifier */
    Ipsimage_vreg    snDestImage,    /* destination image identifier */
    Ipsint_vreg_chc  sOption        /* dyadic logical combination option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnroi InDyadicLogicalROI ( /* OUT destination ROI identifier */
    Idnroi      nSourceROI1, /* first source ROI identifier */
    Idnroi      nSourceROI2, /* second source ROI identifier */
    Idnroi      nDestROI,    /* destination ROI identifier */
    Ipint       iOption      /* dyadic logical combination option */
);

extern Idnvreg InDyadicLogicalROIvReg ( /* OUT destination ROI identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    Ipsroi_vreg  snSourceROI1, /* first source ROI identifier */
    Ipsroi_vreg  snSourceROI2, /* second source ROI identifier */
    Ipsroi_vreg  snDestROI,    /* destination ROI identifier */
    Ipsint_vreg_chc sOption /* dyadic logical combination option */
);

extern Idnimage InDyadicPredicate ( /* OUT destination image identifier */
    Idnimage      nSourceImage1, /* first source image identifier */
    Idnimage      nSourceImage2, /* second source image identifier */
    Idnimage      nDestImage,    /* destination image identifier */
    Ipint         iOption      /* dyadic predicate combination option */
);

extern Idnvreg InDyadicPredicateVReg ( /* OUT destination image identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage1, /* first source image identifier */
    Ipsimage_vreg snSourceImage2, /* second source image identifier */
    Ipsimage_vreg snDestImage,    /* destination image identifier */
    Ipsint_vreg_chc sOption /* dyadic predicate combination option */
);

extern Idnimage InEdgeDetectOrthGrad ( /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage,    /* destination image identifier */
    Idnnbhood     nHorizontalGrad, /* horizontal gradient impulse array identifier */
    Idnnbhood     nVerticalGrad,   /* vertical gradient impulse array identifier */
    Ipsparameter_arith stThreshold, /* threshold value pointer union */
    Ipsparameter_edge stBelowValue, /* below threshold output value pointer union */
    Ipsparameter_edge stAboveValue, /* above threshold output value pointer union */
    Ipint         iEdgeOption, /* edge detection, orthogonal option */
    Ipereflection  eHorizontalOption, /* horizontal impulse rotation option */
    Ipereflection  eVerticalOption,  /* vertical impulse rotation option */
    Ipint         iTypeOption      /* BD or RD destination data type option */
);

```



```

/*
FUNCTION PROTOTYPES (continued)
*/

extern Idnvreg InEdgeDetectOrthGradVReg (
    Idnvreg          nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier */
    Ipsnbhood_vreg   snHorizontalGrad, /* destination image identifier */
    Ipsnbhood_vreg   snVerticalGrad, /* horizontal gradient impulse array identifier */
    Ipsparameter_arith_vreg_chc sThreshold, /* vertical gradient impulse array identifier */
    Ipsparameter_edge_vreg_chc sBelowValue, /* threshold value */
    Ipsparameter_edge_vreg_chc sAboveValue, /* below threshold output value */
    Ipsint_vreg_chc sEdgeOption, /* above threshold output value */
    Ipsreflection_vreg_chc sHorizontalOption, /* edge detection, orthogonal option */
    Ipsreflection_vreg_chc sVerticalOption, /* horizontal impulse rotation option */
    Ipsint_vreg_chc sTypeOption /* vertical impulse rotation option */
    /* BD or RD destination data type option */
);

extern Idnimage InEdgeDetect2ndDeriv (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Idnnbhood         n2ndDeriv,    /* destination image identifier */
    Ipsparameter_edge stNoCrossing, /* second derivative impulse array identifier */
    Ipsparameter_edge stZeroCrossing, /* no zero crossing output value pointer union */
    Ipint             iDerivOption, /* zero crossing output value pointer union */
    Ipint             iTypeOption, /* edge detection option */
    /* BD or RD destination data type option */
);

extern Idnvreg InEdgeDetect2ndDerivVReg (
    Idnvreg          nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier */
    Ipsnbhood_vreg   sn2ndDeriv,    /* destination image identifier */
    Ipsparameter_edge_vreg_chc sNoCrossing, /* second derivative impulse array identifier */
    Ipsparameter_edge_vreg_chc sZeroCrossing, /* no zero crossing output value */
    Ipsint_vreg_chc sDerivOption, /* zero crossing output value */
    Ipsint_vreg_chc sTypeOption, /* edge detection option */
    /* BD or RD destination data type option */
);

extern Idnimage InEdgeDetectTemplGrad (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Idnnbhood         *tnaArrays,   /* destination image identifier */
    Ipuint            uNumber,       /* template gradient array identifier set */
    Ipsparameter_arith stThreshold, /* number of template gradient arrays */
    Ipsparameter_edge stBelowValue, /* threshold value pointer union */
    Ipsparameter_edge stAboveValue, /* below threshold output value pointer union */
    Ipint             iGradOption, /* above threshold output value pointer union */
    Ipint             iTypeOption, /* edge detection option */
    /* BD or RD destination data type option */
);

```



```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InEdgeDetectTemplGradVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage, /* destination image identifier */
    Ipsanbhood_vreg_chc sArrays, /* template gradient array identifier set */
    Ipsuint_vreg_chc sNumber, /* number of template gradient arrays */
    Ipsparameter_arith_vreg_chc sThreshold, /* threshold value */
    Ipsparameter_edge_vreg_chc sBelowValue, /* below threshold output value */
    Ipsparameter_edge_vreg_chc sAboveValue, /* above threshold output value */
    Ipsint_vreg_chc sGradOption, /* edge detection option */
    Ipsint_vreg_chc sTypeOption /* BD or RD destination data type option */
);

extern Ipbool IbEqualPredicate ( /* OUT predicate flag */
    Idnimage          nSourceImage1, /* first source image identifier */
    Idnimage          nSourceImage2, /* second source image identifier */
    Ipfloat           rTolerance /* pixel tolerance */
);

extern Idnvreg InEqualPredicateVReg ( /* OUT predicate flag */
    Idnvreg          nVReg, /* output virtual register identifier */
    Ipsimage_vreg    snSourceImage1, /* first source image identifier */
    Ipsimage_vreg    snSourceImage2, /* second source input image identifier */
    Ipsfloat_vreg_chc sTolerance /* pixel tolerance */
);

extern Idnimage InErosionDilateBool ( /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage, /* destination image identifier */
    Idnnbhood         nArray, /* structuring element array identifier */
    Ipuint            uIterations, /* number of iterations */
    Ippeerosion        eErosionOption, /* erosion/dilation operation option */
    Ipint             iProcessOption /* processing option */
);

extern Idnvreg InErosionDilateBoolVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage, /* destination image identifier */
    Ipsnbhood_vreg    snArray, /* structuring element array identifier */
    Ipsuint_vreg_chc sIterations, /* number of iterations */
    Ipserosion_vreg_chc sErosionOption, /* erosion/dilation operation option */
    Ipsint_vreg_chc sProcessOption /* processing option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnroi InErosionDilateBoolROI (                                     /* OUT destination ROI identifier */
    Idnroi          nSourceROI, /* source ROI identifier */
    Idnroi          nDestROI,  /* destination ROI identifier */
    Idnnbhood       nArray,    /* structuring element array identifier */
    Ipuint          uIterations, /* number of iterations */
    Ipeerosion      eErosionOption, /* erosion/dilation operation option */
    Ipin            iProcessOption /* processing option */
);

extern Idnvreg InErosionDilateBoolROIvReg ( /* OUT destination ROI identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsroi_vreg      snSourceROI, /* source ROI identifier */
    Ipsroi_vreg      snDestROI,  /* destination ROI identifier */
    Ipsnbhood_vreg   snArray,    /* structuring element array identifier */
    Ipsuint_vreg_chc sIterations, /* number of iterations */
    Ipserosion_vreg_chc sErosionOption, /* erosion/dilation operation option */
    Ipsint_vreg_chc  sProcessOption /* processing option */
);

extern Idnimage InErosionDilateGrey ( /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Idnnbhood         nStructure,   /* structuring element array identifier */
    Idnnbhood         nMask,        /* mask array identifier */
    Ipuint            uIterations,  /* number of iterations */
    Ipeerosion        eErosionOption, /* erosion/dilation operation option */
    Ipin              iProcessOption /* processing option */
);

extern Idnvreg InErosionDilateGreyVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,   /* destination image identifier */
    Ipsnbhood_vreg   snStructure,   /* structuring element array identifier */
    Ipsnbhood_vreg   snMask,        /* mask array identifier */
    Ipsuint_vreg_chc sIterations,  /* number of iterations */
    Ipserosion_vreg_chc sErosionOption, /* erosion/dilation operation option */
    Ipsint_vreg_chc  sProcessOption /* processing option */
);

extern Idnimage InErrorFunctionScaling ( /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Ipfloating         rFactor      /* Gaussian spread factor */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InErrorFunctionScalingVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage, /* destination image identifier */
    Ipsfloat_vreg_chc sFactor /* Gaussian spread factor */
);

extern void IvErrorHandler ( /* no return value output */
    Ipint          iCode, /* error identifier code */
    Ipint          iElement, /* descriptor of element detecting error */
    Idnchain       nChain, /* identifier of chain detecting error */
    Ipnterror      nError, /* error file identifier */
    Ipsynchronicity_state eSynchronicityState /* synchronicity state */
);

extern void IvErrorLogger ( /* no return value output */
    Ipint          iCode, /* error identifier code */
    Ipint          iElement, /* descriptor of function detecting error */
    Idnchain       nChain, /* identifier of chain detecting error */
    Ipnterror      nError, /* error file identifier */
    Ipsynchronicity_state eSynchronicityState /* synchronicity state */
);

extern Ipbool IbErrorTest ( /* OUT error flag */
    void /* no input arguments */
);

extern Idnvreg InErrorTestVReg ( /* OUT error flag */
    Idnvreg          nVReg /* output virtual register identifier */
);

extern Ipuint *ItuaExportHist ( /* OUT external histogram data array pointer */
    Idnhist          nHist, /* histogram identifier */
    Ipuint           *tuaData /* external data value array pointer */
);

extern Ipspiks_pixel_types IstExportImage ( /* OUT external image pixel data pointer union */
    Idnimage          nSourceImage, /* internal source image identifier */
    Ipspiks_pixel_types stDestImage, /* external image pixel data pointer union */
    Ipuint            uPrecision, /* number of bits of destination pixel */
    Ipint             iType, /* destination data type option */
    Ipint             iApplBufferSize, /* data transfer buffer size or option */
    Ipuint            uStartPoint, /* index offset within internal data */
    Ipuint            *tuDataSize /* OUT post transfer data size pointer */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Ipsparameter_pixel IstExportLUT ( /* OUT external data array pointer union */
    Idnlut          nLUT, /* lookup table data identifier */
    Ipsparameter_pixel stData, /* external data array pointer union */
    Ipint           iApplBufferSize, /* data transfer buffer size or option */
    Ipuint          uStartPoint, /* index offset within internal data */
    Ipuint          *tuDataSize /* OUT post transfer data size pointer */
);

extern Ipsparameter_numeric IstExportMatrix ( /* OUT external data array pointer union */
    Idnmatrix      nMatrix, /* matrix identifier */
    Ipsparameter_numeric stData /* external data array pointer union */
);

extern Ipsparameter_pixel IstExportNbhoodArray ( /* OUT neighbourhood array pointer union */
    Idnnbhood      nNbhood, /* neighbourhood array identifier */
    Ipsparameter_pixel stData, /* neighbourhood array data pointer union */
    Ipint          *tiScaleFactor /* OUT scale factor */
);

extern Ipbool *ItbaExportROIArray ( /* OUT external data value array pointer */
    Idnroi         nROI, /* ROI array identifier */
    Ipbool         *tbaData, /* external data values array pointer */
    Ipint          iApplBufferSize, /* data transfer buffer size or option */
    Ipuint         uStartPoint, /* index offset within internal data */
    Ipuint         *tuDataSize /* OUT post transfer data size pointer */
);

extern Ipsparameter_pixel IstExportStaticArray ( /* OUT external data array pointer union */
    Idnstatic      nArray, /* static array identifier */
    Ipsparameter_pixel stData, /* external array of data array pointer union */
    Ipint          iApplBufferSize, /* data transfer buffer size or option */
    Ipuint         uStartPoint, /* index offset within internal data */
    Ipuint         *tuDataSize /* OUT post transfer data size pointer */
);

extern Ipsparameter_basic IstExportTuple ( /* OUT external data array pointer union */
    Idntuple       nTuple, /* tuple identifier */
    Ipsparameter_basic stData /* external data value array pointer union */
);

extern Ipsparameter_arith IstExportValueBounds ( /* OUT to external data array pointer union */
    Idnbounds      nValueBounds, /* value bound identifier */
    Ipsparameter_arith stData, /* external data value array pointer union */
    Ipint          iApplBufferSize, /* data transfer buffer size or option */
    Ipuint         uStartPoint, /* index offset within internal data */
    Ipuint         *tuDataSize /* OUT post transfer data size pointer */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnimage InExtractPixelPlane (                                     /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage,   /* destination image identifier */
    Ipoint        uDepth,       /* depth index value */
    Ipoint        uTemporal,    /* temporal index value */
    Ipoint        uBand         /* band index value */
);

extern Idnvreg InExtractPixelPlaneVReg (                               /* OUT destination image identifier */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage,   /* destination image identifier */
    Ipsuint_vreg_chc sDepth,     /* depth index value */
    Ipsuint_vreg_chc sTemporal,  /* temporal index value */
    Ipsuint_vreg_chc sBand      /* band index value */
);

extern Idnroi InExtractPixelPlaneROI (                                /* OUT destination ROI identifier */
    Idnroi      nSourceROI, /* source ROI identifier */
    Idnroi      nDestROI,   /* destination ROI identifier */
    Ipoint      uDepth,     /* depth index value */
    Ipoint      uTemporal,  /* temporal index value */
    Ipoint      uBand       /* band index value */
);

extern Idnvreg InExtractPixelPlaneROIvReg (                          /* OUT destination ROI identifier */
    Idnvreg     nVReg,      /* output virtual register identifier or NULL */
    Ipsroi_vreg snSourceROI, /* source ROI identifier */
    Ipsroi_vreg snDestROI,   /* destination ROI identifier */
    Ipsuint_vreg_chc sDepth, /* depth index value */
    Ipsuint_vreg_chc sTemporal, /* temporal index value */
    Ipsuint_vreg_chc sBand   /* band index value */
);

extern void IvExtrema ( /* no return value output */
    Idnimage      nSourceImage, /* source image identifier */
    Ipfpoint      *traMinimum,   /* OUT minimum extrema pixel array pointers */
    Ipfpoint      *traMaximum,   /* OUT maximum extrema pixel array pointers */
    Ipeprocess    eOption,       /* processing mode option */
    Ipoint        iApplBufferSize, /* data transfer buffer size or option of each array */
    Ipoint        uStartPoint,    /* index offset within internal data of each array */
    Ipoint        *tuDataSize     /* OUT post transfer data size pointer */
);

extern Idnimage InFalseColour ( /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage,   /* destination image identifier */
    Idnmatrix      nColour      /* colour conversion matrix identifier */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InFalseColourVReg (                                           /* OUT destination image identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,   /* destination image identifier */
    Ipsmatrix_vreg   snColour       /* colour conversion matrix identifier */
);

extern Idnimage InFillRegion (                                               /* OUT destination image identifier */
    Idnimage         nSourceImage, /* source image identifier */
    Idnimage         nDestImage,   /* destination image identifier */
    Ipoint           uSeed1,       /* first index value - seed pixel within contour */
    Ipoint           uSeed2,       /* second index value - seed pixel within contour */
    Ipoint           uIndex3,      /* third index value */
    Ipoint           uIndex4,      /* fourth index value */
    Ipoint           uIndex5       /* fifth index value */
);

extern Idnvreg InFillRegionVReg (                                           /* OUT destination image identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,   /* destination image identifier */
    Ipsuint_vreg_chc sSeed1,        /* first index value - seed pixel within contour */
    Ipsuint_vreg_chc sSeed2,        /* second index value - seed pixel within contour */
    Ipsuint_vreg_chc sIndex3,       /* third index value */
    Ipsuint_vreg_chc sIndex4,       /* fourth index value */
    Ipsuint_vreg_chc sIndex5       /* fifth index value */
);

extern Idnroi InFillRegionROI (                                             /* OUT destination ROI identifier */
    Idnroi           nSourceROI,    /* source ROI identifier */
    Idnroi           nDestROI,      /* destination ROI identifier */
    Ipoint           uSeed1,        /* first index value - seed pixel within contour */
    Ipoint           uSeed2,        /* second index value - seed pixel within contour */
    Ipoint           uIndex3,       /* third index value */
    Ipoint           uIndex4,       /* fourth index value */
    Ipoint           uIndex5       /* fifth index value */
);

extern Idnvreg InFillRegionROIIVReg (                                       /* OUT destination ROI identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsroi_vreg      snSourceROI,    /* source ROI identifier */
    Ipsroi_vreg      snDestROI,      /* destination ROI identifier */
    Ipsuint_vreg_chc sSeed1,        /* first index value - seed pixel within contour */
    Ipsuint_vreg_chc sSeed2,        /* second index value - seed pixel within contour */
    Ipsuint_vreg_chc sIndex3,       /* third index value */
    Ipsuint_vreg_chc sIndex4,       /* fourth index value */
    Ipsuint_vreg_chc sIndex5       /* fifth index value */
);

```



```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnstatic InFilterButterworth (                                     /* OUT filter transfer function array identifier */
    Idnstatic          nFilter,                                         /* filter transfer function array identifier */
    Idntuple           nSize,                                           /* array size ND 5-tuple */
    Ipfloat            rOrder,                                           /* filter order */
    Ipfloat            rCutoff,                                          /* cutoff frequency */
    Ipint              iOption                                          /* filter (Butterworth) type option */
);

extern Idnvreg InFilterButterworthVReg (                                /* OUT filter transfer function array identifier */
    Idnvreg           nVReg,                                           /* output virtual register identifier or NULL */
    Ipsstatic_vreg    snFilter,                                         /* filter transfer function array identifier */
    Ipstuple_vreg     snSize,                                           /* array size ND 5-tuple */
    Ipsfloat_vreg_chc sOrder,                                           /* filter order */
    Ipsfloat_vreg_chc sCutoff,                                          /* cutoff frequency */
    Ipsint_vreg_chc   sOption                                          /* filter (Butterworth) type option */
);

extern Idnstatic InFilterGauss (                                       /* OUT filter transfer function array identifier */
    Idnstatic          nFilter,                                         /* filter transfer function array identifier */
    Idntuple           nSize,                                           /* array size ND 5-tuple */
    Ipfloat            rColumnSpread, /* column index spread factor */
    Ipfloat            rRowSpread   /* row index spread factor */
);

extern Idnvreg InFilterGaussVReg (                                    /* OUT filter transfer function array identifier */
    Idnvreg           nVReg,                                           /* output virtual register identifier or NULL */
    Ipsstatic_vreg    snFilter,                                         /* filter transfer function array identifier */
    Ipstuple_vreg     snSize,                                           /* array size ND 5-tuple */
    Ipsfloat_vreg_chc sColumnSpread, /* column index spread factor */
    Ipsfloat_vreg_chc sRowSpread   /* row index spread factor */
);

extern Idnstatic InFilterInverse (                                     /* OUT filter transfer function array identifier */
    Idnstatic          nBlur,                                           /* image blur transfer function array identifier */
    Idnstatic          nTransfer,                                         /* filter transfer function array identifier */
    Idntuple           nSize,                                           /* array size ND 5-tuple */
    Ipuint             uColumnCutoff, /* column index cutoff frequency */
    Ipuint             uRowCutoff   /* row index cutoff frequency */
);

extern Idnvreg InFilterInverseVReg (                                /* OUT filter transfer function array identifier */
    Idnvreg           nVReg,                                           /* output virtual register identifier or NULL */
    Ipsstatic_vreg    snBlur,                                           /* image blur transfer function array identifier */
    Ipsstatic_vreg    snTransfer,                                         /* filter transfer function array identifier */
    Ipstuple_vreg     snSize,                                           /* array size ND 5-tuple */
    Ipsuint_vreg_chc  sColumnCutoff, /* column index cutoff frequency */
    Ipsuint_vreg_chc  sRowCutoff   /* row index cutoff frequency */
);

```



```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnstatic InFilterMatched (                                           /* OUT filter transfer function array identifier */
    Idnstatic      nFourier, /* reference image Fourier transform array id */
    Idnstatic      nNoise,  /* noise power spectrum array identifier */
    Idnstatic      nFilter, /* filter transfer function array identifier */
    Idntuple       nSize,   /* array size ND 5-tuple */
    Ipuint         uColumnOffset, /* column index offset */
    Ipuint         uRowOffset,  /* row index offset */
);

extern Idnvreg InFilterMatchedVReg ( /* OUT filter transfer function array identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    Ipsstatic_vreg snFourier, /* reference image Fourier transform array id */
    Ipsstatic_vreg snNoise,  /* noise power spectrum array identifier */
    Ipsstatic_vreg snFilter, /* filter transfer function array identifier */
    Ipstuple_vreg snSize,   /* array size ND 5-tuple */
    Ipsuint_vreg_chc sColumnOffset, /* column index offset */
    Ipsuint_vreg_chc sRowOffset,  /* row index offset */
);

extern Idnstatic InFilterWiener ( /* OUT filter transfer function array identifier */
    Idnstatic      nBlur, /* image blur transfer function array identifier */
    Idnstatic      nImagePower, /* image power spectrum array identifier */
    Idnstatic      nNoisePower, /* noise power spectrum array identifier */
    Idnstatic      nFilter, /* filter transfer function array identifier */
    Idntuple       nSize, /* array size ND 5-tuple */
);

extern Idnvreg InFilterWienerVReg ( /* OUT filter transfer function array identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    Ipsstatic_vreg snBlur, /* image blur transfer function array identifier */
    Ipsstatic_vreg snImagePower, /* image power spectrum array identifier */
    Ipsstatic_vreg snNoisePower, /* noise power spectrum array identifier */
    Ipsstatic_vreg snFilter, /* filter transfer function array identifier */
    Ipstuple_vreg snSize, /* array size ND 5-tuple */
);

extern Idnstatic InFilterZonal ( /* OUT filter transfer function array identifier */
    Idnstatic      nFilter, /* filter transfer function array identifier */
    Idntuple       nSize, /* array size ND 5-tuple */
    Ipuint         uColumnCutoff, /* column index cutoff frequency */
    Ipuint         uRowCutoff, /* row index cutoff frequency */
    Ipint          iOption, /* filter (zonal) type option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InFilterZonalVReg (                                           /* OUT filter transfer function array identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsstatic_vreg   snFilter,       /* filter transfer function array identifier */
    Ipstuple_vreg     snSize,        /* array size 5-tuple */
    Ipsuint_vreg_chc  sColCutoff,    /* column index cutoff frequency */
    Ipsuint_vreg_chc  sRowCutoff,    /* row index cutoff frequency */
    Ipsint_vreg_chc   sOption        /* filter (zonal) type option */
);

extern Idnimage InFilteringHomomorphic (                                     /* OUT destination image identifier */
    Idnimage          nSourceImage,  /* source image identifier */
    Idnimage          nDestImage,    /* destination image identifier */
    Idnstatic         nWindow,       /* windowing function array identifier */
    Idnstatic         nFilter        /* filter transfer function array identifier */
);

extern Idnvreg InFilteringHomomorphicVReg (                                 /* OUT destination image identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg     snSourceImage, /* source image identifier */
    Ipsimage_vreg     snDestImage,   /* destination image identifier */
    Ipsstatic_vreg     snWindow,     /* windowing function array identifier */
    Ipsstatic_vreg     snFilter      /* filter transfer function array identifier */
);

extern Idnimage InFilteringLin (                                           /* OUT destination image identifier */
    Idnimage          nSourceImage,  /* source image identifier */
    Idnimage          nDestImage,    /* destination image identifier */
    Idnstatic         nWindow,       /* window function array identifier */
    Idnstatic         nFilter        /* filter transfer function array identifier */
);

extern Idnvreg InFilteringLinVReg (                                         /* OUT destination image identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg     snSourceImage, /* source image identifier */
    Ipsimage_vreg     snDestImage,   /* destination image identifier */
    Ipsstatic_vreg     snWindow,     /* window function array identifier */
    Ipsstatic_vreg     snFilter      /* filter transfer function array identifier */
);

extern Idnimage InFilteringMedian (                                         /* OUT destination image identifier */
    Idnimage          nSourceImage,  /* source image identifier */
    Idnimage          nDestImage,    /* destination image identifier */
    Idnnbhood         nMask,        /* mask array identifier */
    Ipint             iOption        /* filtering process option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InFilteringMedianVReg (                                     /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipsnbhood_vreg   snMask,        /* mask array identifier */
    Ipsint_vreg_chc  siOption       /* filtering process option */
);

extern Idnimage InFilteringPseudomedian (                               /* OUT destination image identifier */
    Idnimage         nSourceImage, /* source image identifier */
    Idnimage         nDestImage,   /* destination image identifier */
    Ipoint           uMaskWidth,   /* mask width */
    Ipoint           uMaskHeight  /* mask height */
);

extern Idnvreg InFilteringPseudomedianVReg (                          /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipsuint_vreg_chc sMaskWidth,   /* mask width */
    Ipsuint_vreg_chc sMaskHeight  /* mask height */
);

extern Idnimage InFilteringRankOrder (                                /* OUT destination image identifier */
    Idnimage         nSourceImage, /* source image identifier */
    Idnimage         nDestImage,   /* destination image identifier */
    Idnnbhood        nMask,        /* mask array identifier */
    Ipoint           uRank,        /* rank value */
    Ipoint           iOption       /* filtering process option */
);

extern Idnvreg InFilteringRankOrderVReg (                             /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipsnbhood_vreg   snMask,        /* mask array identifier */
    Ipsuint_vreg_chc sRank,        /* rank value */
    Ipsint_vreg_chc  sOption       /* filtering process option */
);

extern Idnimage InFlipSpinTranspose (                                  /* OUT destination image identifier */
    Idnimage         nSourceImage, /* source image identifier */
    Idnimage         nDestImage,   /* destination image identifier */
    Ipeflip          eOption       /* flip, spin, transpose option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InFlipSpinTransposeVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage, /* destination image identifier */
    Ipsflip_vreg_chc sOption /* flip, spin, transpose option */
);

extern Idnroi InFlipSpinTransposeROI ( /* OUT destination ROI identifier */
    Idnroi          nSourceROI, /* source ROI identifier */
    Idnroi          nDestROI, /* destination ROI identifier */
    Ipeflip         eOption /* flip, spin, transpose option */
);

extern Idnvreg InFlipSpinTransposeROIvReg ( /* OUT destination ROI identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsroi_vreg      snSourceROI, /* source ROI identifier */
    Ipsroi_vreg      snDestROI, /* destination ROI identifier */
    Ipsflip_vreg_chc sOption /* flip, spin, transpose option */
);

extern Idnimage InGammaCorrection ( /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage, /* destination image identifier */
    Ipfloat           rFirst, /* first constant */
    Ipfloat           rSecond, /* second constant */
    Ipfloat           rThird, /* third constant */
    Ipfloat           rFourth, /* fourth constant */
    Ipfloat           rBreak, /* break point */
    Ipegamma          eOption /* gamma correction option */
);

extern Idnvreg InGammaCorrectionVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage, /* destination image identifier */
    Ipsfloat_vreg_chc sFirst, /* first constant */
    Ipsfloat_vreg_chc sSecond, /* second constant */
    Ipsfloat_vreg_chc sThird, /* third constant */
    Ipsfloat_vreg_chc sFourth, /* fourth constant */
    Ipsfloat_vreg_chc sBreak, /* break point */
    Ipsgamma_vreg_chc sOption /* gamma correction option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern void IvGetColourPixel (
    Idnimage          nSourceImage, /* no return value output */
    Ipsparameter_colour stBand0,    /* source image identifier */
    Ipsparameter_colour stBand1,    /* OUT band 0 pixel value pointer union */
    Ipsparameter_colour stBand2,    /* OUT band 1 pixel value pointer union */
    Ipsparameter_colour stBand3,    /* OUT band 2 pixel value pointer union */
    Ipoint            uHorizontal,   /* OUT band 3 pixel value pointer union */
    Ipoint            uVertical,     /* horizontal index value */
    Ipoint            uDepth,        /* vertical index value */
    Ipoint            uTemporal,     /* depth index value */
    /* temporal index value */
);

extern void IvGetColourPixelVReg (
    Ipsimage_vreg      snSourceImage, /* no return value output */
    Idnvreg            nBand0VReg,    /* source image identifier */
    Idnvreg            nBand1VReg,    /* OUT band 0 pixel virtual register identifier */
    Idnvreg            nBand2VReg,    /* OUT band 1 pixel virtual register identifier */
    Idnvreg            nBand3VReg,    /* OUT band 2 pixel virtual register identifier */
    Ipsuint_vreg_chc   sHorizontal,   /* OUT band 3 pixel virtual register identifier */
    Ipsuint_vreg_chc   sVertical,     /* horizontal index value */
    Ipsuint_vreg_chc   sDepth,        /* vertical index value */
    Ipsuint_vreg_chc   sTemporal,     /* depth index value */
    /* temporal index value */
);

extern Ipsparameter_pixel IstGetPixel (
    Idnimage          nSourceImage, /* OUT pixel value pointer union */
    Ipsparameter_pixel stPixel,     /* source image identifier */
    Ipoint            uHorizontal,   /* pixel value pointer union */
    Ipoint            uVertical,     /* horizontal index value */
    Ipoint            uDepth,        /* vertical index value */
    Ipoint            uTemporal,     /* depth index value */
    Ipoint            uBand,        /* temporal index value */
    /* band index value */
);

extern Idnvreg InGetPixelVReg (
    Idnvreg          nVReg, /* OUT pixel value */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier */
    Ipsuint_vreg_chc sHorizontal,   /* source image identifier */
    Ipsuint_vreg_chc sVertical,     /* horizontal index value */
    Ipsuint_vreg_chc sDepth,        /* vertical index value */
    Ipsuint_vreg_chc sTemporal,     /* depth index value */
    Ipsuint_vreg_chc sBand,        /* temporal index value */
    /* band index value */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Ipbool IbGetPixelROI (
    Idnroi          nSourceROI, /* OUT ROI pixel value */
    Ipbool          *tbPixel,   /* source ROI identifier */
    Ipoint          uHorizontal, /* pixel value pointer */
    Ipoint          uVertical,   /* horizontal index value */
    Ipoint          uDepth,      /* vertical index value */
    Ipoint          uTemporal,   /* depth index value */
    Ipoint          uBand,       /* temporal index value */
    Ipoint          /* band index value */
);

extern Idnreg InGetPixelROIReg (
    Idnreg          nVReg,      /* OUT ROI pixel value */
    Ipsroi_vreg     snSourceROI, /* output virtual register identifier */
    Ipsuint_vreg_chc sHorizontal, /* source ROI identifier */
    Ipsuint_vreg_chc sVertical,   /* horizontal index value */
    Ipsuint_vreg_chc sDepth,      /* vertical index value */
    Ipsuint_vreg_chc sTemporal,   /* depth index value */
    Ipsuint_vreg_chc sBand,       /* temporal index value */
    Ipsuint_vreg_chc /* band index value */
);

extern Ipsparameter_pixel IstGetPixelArray (
    Idnimage          nSourceImage, /* OUT pixel array pointer union */
    Ipsparameter_pixel stPixels,     /* source image identifier */
    Ipoint            uOffset1,      /* pixel values array pointer union */
    Ipoint            uOffset2,      /* first index offset */
    Ipoint            uValue3,       /* second index offset */
    Ipoint            uValue4,       /* third index value */
    Ipoint            uValue5,       /* fourth index value */
    Ipoint            uColumns,      /* fifth index value */
    Ipoint            uRows,         /* number of array columns */
    Ipoint            /* number of array rows */
);

extern Idnreg InGetPixelArrayVReg (
    Ipsimage_vreg     snSourceImage, /* OUT pixel array */
    Idnreg            nPixelsVReg,   /* source image identifier */
    Ipsuint_vreg_chc sOffset1,       /* output array virtual register identifier */
    Ipsuint_vreg_chc sOffset2,       /* first index offset */
    Ipsuint_vreg_chc sValue3,        /* second index offset */
    Ipsuint_vreg_chc sValue4,        /* third index value */
    Ipsuint_vreg_chc sValue5,        /* fourth index value */
    Ipsuint_vreg_chc sColumns,       /* fifth index value */
    Ipsuint_vreg_chc sRows,          /* number of array columns */
    Ipsuint_vreg_chc /* number of array rows */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Ipbool *ItbaGetPixelArrayROI (                                     /* OUT pixel array pointer */
    Idnroi          nSourceROI, /* source ROI identifier */
    Ipbool          *tbaPixels, /* pixel values array pointer */
    Ipoint          uOffset1,   /* first index offset */
    Ipoint          uOffset2,   /* second index offset */
    Ipoint          uValue3,    /* third index value */
    Ipoint          uValue4,    /* fourth index value */
    Ipoint          uValue5,    /* fifth index value */
    Ipoint          uColumns,   /* number of array columns */
    Ipoint          uRows      /* number of array rows */
);

extern Idnvreg InGetPixelArrayROIvReg ( /* OUT pixel array */
    Ipsroi_vreg    snSourceROI, /* source ROI identifier */
    Idnvreg        nPixelsVReg, /* output array virtual register identifier */
    Ipsuint_vreg_chc sOffset1, /* first index offset */
    Ipsuint_vreg_chc sOffset2, /* second index offset */
    Ipsuint_vreg_chc sValue3,  /* third index value */
    Ipsuint_vreg_chc sValue4,  /* fourth index value */
    Ipsuint_vreg_chc sValue5,  /* fifth index value */
    Ipsuint_vreg_chc sColumns, /* number of array columns */
    Ipsuint_vreg_chc sRows     /* number of array rows */
);

extern Ipsparameter_pixel IstGetPixelRecord ( /* OUT pixel record pointer union */
    Idnimage        nSourceImage, /* source image identifier */
    Ipsparameter_pixel stPixel, /* pixel record pointer union */
    Idntuple        nType, /* data type SD B-tuple */
    Ipoint          uHorizontal, /* horizontal index value */
    Ipoint          uVertical, /* vertical index value */
    Ipoint          uDepth, /* depth index value */
    Ipoint          uTemporal /* temporal index value */
);

extern Idnvreg InGetPixelRecordVReg ( /* OUT pixel record */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Idnvreg          nPixelVReg, /* pixel record virtual register identifier */
    Ipsuple_vreg     snType, /* data type B-tuple */
    Ipsuint_vreg_chc sHorizontal, /* horizontal index value */
    Ipsuint_vreg_chc sVertical, /* vertical index value */
    Ipsuint_vreg_chc sDepth, /* depth index value */
    Ipsuint_vreg_chc sTemporal /* temporal index value */
);

```



```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Ipbool *ItbaGetPixelRecordROI (                                     /* OUT pixel record pointer */
    Idnroi                      nSourceROI, /* source ROI identifier */
    Ipbool                      *tbaPixel, /* pixel record value pointer */
    Ipuint                      uHorizontal, /* horizontal index value */
    Ipuint                      uVertical, /* vertical index value */
    Ipuint                      uDepth, /* depth index value */
    Ipuint                      uTemporal /* temporal index value */
);

extern Idnreg InGetPixelRecordROIvReg ( /* OUT pixel record */
    Ipsroi_vreg          snSourceROI, /* source ROI identifier */
    Idnreg              nPixelVReg, /* pixel record virtual register identifier */
    Ipsuint_vreg_chc    sHorizontal, /* horizontal index value */
    Ipsuint_vreg_chc    sVertical, /* vertical index value */
    Ipsuint_vreg_chc    sDepth, /* depth index value */
    Ipsuint_vreg_chc    sTemporal /* temporal index value */
);

extern Idnimage InHistModification ( /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage, /* destination image identifier */
    Ipsparameter_arith sUpperValue, /* output histogram upper value */
    Ipsparameter_arith sLowerValue, /* output histogram lower value */
    Ipfloat           rSpreadFactor, /* exponential spread factor */
    Ipuint            uEntries, /* number of lookup table source offset entries */
    Ipint             iOption /* histogram modification option */
);

extern Idnreg InHistModificationVReg ( /* OUT destination image identifier */
    Idnreg          nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage, /* destination image identifier */
    Ipsparameter_arith_vreg_chc sUpperValue, /* output histogram upper value */
    Ipsparameter_arith_vreg_chc sLowerValue, /* output histogram lower value */
    Ipsfloat_vreg_chc    sSpreadFactor, /* exponential spread factor */
    Ipsuint_vreg_chc    sEntries, /* number of lookup table source offset entries */
    Ipsint_vreg_chc     sOption /* histogram option */
);

extern Idnimage InHistScaling ( /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage, /* destination image identifier */
    Ipsparameter_arith stLowerValue, /* lower output value pointer union */
    Ipsparameter_arith stUpperValue, /* upper output value pointer union */
    Ipfloat           rLowerClip, /* lower input clip fraction */
    Ipfloat           rUpperClip, /* upper input clip fraction */
    Ipuint            uBins /* number of histogram bins */
);

```

```

/*                                     FUNCTION PROTOTYPES (continued)                                     */

extern Idnvreg InHistScalingVReg (                                     /* OUT destination image identifier */
    Idnvreg          nVReg,                                     /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,                             /* source image identifier */
    Ipsimage_vreg    snDestImage,                             /* destination image identifier */
    Ipsparameter_arith_vreg_chc sLowerValue,                 /* lower output value */
    Ipsparameter_arith_vreg_chc sUpperValue,                 /* upper output value */
    Ipsfloat_vreg_chc sLowerClip,                             /* lower input clip fraction */
    Ipsfloat_vreg_chc sUpperClip,                             /* upper input clip fraction */
    Ipsuint_vreg_chc sBins                                   /* number of histogram bins */
);

extern Ipsfloat *ItraHistShape1D (                                     /* OUT shape array identifier */
    Idnhist          nHist,                                     /* histogram identifier */
    Ipsfloat          *traShape,                             /* histogram shape array pointer */
    Ipoint           iApplBufferSize,                         /* data transfer buffer size or option */
    Ipsuint           uStartPoint,                             /* index offset within internal data */
    Ipsuint           *tuDataSize                             /* OUT post transfer data size pointer */
);

extern Ipsfloat *ItraHistShape2D (                                     /* OUT shape array identifier */
    Idnhist          nHist,                                     /* histogram identifier */
    Ipsfloat          *traShape,                             /* histogram shape array pointer */
    Ipoint           iApplBufferSize,                         /* data transfer buffer size or option */
    Ipsuint           uStartPoint,                             /* index offset within internal data */
    Ipsuint           *tuDataSize                             /* OUT post transfer data size pointer */
);

extern Idnhist InHist1D (                                     /* OUT histogram identifier */
    Idnimage         nSourceImage,                             /* source image identifier */
    Idnhist          nHist,                                     /* histogram identifier */
    Ipoint           uBins,                                    /* number of histogram bins */
    Ipsfloat          rLowerBound,                             /* lower amplitude bound */
    Ipsfloat          rUpperBound,                             /* upper amplitude bound */
    Ipoint           iOption                                   /* histogram computation option */
);

extern Idnvreg InHist1DVReg (                                     /* OUT histogram identifier */
    Idnvreg          nVReg,                                     /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,                             /* source image identifier */
    Ipshist_vreg     snHist,                                    /* histogram identifier */
    Ipsuint_vreg_chc sBins,                                    /* number of histogram bins */
    Ipsfloat_vreg_chc sLowerBound,                             /* lower amplitude bound */
    Ipsfloat_vreg_chc sUpperBound,                             /* upper amplitude bound */
    Ipsint_vreg_chc  sOption                                   /* histogram computation option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnhist InHist2D (
    Idnimage          nSourceImage, /* OUT histogram identifier */
    Idnhist           nHist,        /* source image identifier */
    Ipuint            uBins,        /* histogram identifier */
    Ipfloat           rLowerBound,  /* number of histogram bins in each dimension */
    Ipfloat           rUpperBound,  /* lower amplitude bound */
    Ipuint            uHorizontal,  /* upper amplitude bound */
    Ipuint            uVertical,    /* horizontal offset */
    );                             /* vertical offset */

extern Idnvreg InHist2DVReg (
    Idnvreg           nVReg,        /* OUT histogram identifier */
    Ipsimage_vreg     snSourceImage, /* output virtual register identifier or NULL */
    Ipshist_vreg      snHist,       /* source image identifier */
    Ipsuint_vreg_chc  sBins,        /* histogram identifier */
    Ipsfloat_vreg_chc sLowerBound,  /* number of histogram bins in each dimension */
    Ipsfloat_vreg_chc sUpperBound,  /* lower amplitude bound */
    Ipsuint_vreg_chc sHorizontal,   /* upper amplitude bound */
    Ipsuint_vreg_chc sVertical,     /* horizontal offset */
    );                             /* vertical offset */

extern Idnimage InHitMiss (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Ipuint            iIterations,   /* destination image identifier */
    Ipuint            *tuCycles,     /* number of iterations */
    Ipuint            iOption,       /* OUT number of cycles */
    );                             /* hit or miss transformation option */

extern Idnvreg InHitMissVReg (
    Idnvreg           nVReg,        /* OUT destination image identifier */
    Ipsimage_vreg     snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg     snDestImage,   /* source image identifier */
    Ipsint_vreg_chc   sIterations,   /* destination image identifier */
    Idnvreg           nCycles,       /* number of iterations */
    Ipsint_vreg_chc   sOption,       /* OUT cycles virtual register identifier */
    );                             /* hit or miss transformation option */

extern Idnroi InHitMissROI (
    Idnroi            nSourceROI,    /* OUT destination ROI identifier */
    Idnroi            nDestROI,      /* source ROI identifier */
    Ipuint            iIterations,   /* destination ROI identifier */
    Ipuint            *tuCycles,     /* number of iterations */
    Ipuint            iOption,       /* OUT number of cycles */
    );                             /* hit or miss transformation option */

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InHitMissROIIVReg (                                     /* OUT destination ROI identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsroi_vreg      snSourceROI,                                     /* source ROI identifier */
    Ipsroi_vreg      snDestROI,                                       /* destination ROI identifier */
    Ipsint_vreg_chc  sIterations,                                     /* number of iterations */
    Idnvreg          nCycles,                                          /* OUT cycles virtual register identifier */
    Ipsint_vreg_chc  sOption                                          /* hit or miss transformation option */
);

extern Idnstatic InHoughTransform (                                   /* OUT Hough array identifier */
    Idnimage         nSourceImage,                                    /* source image identifier */
    Idnstatic         nHough,                                         /* Hough array identifier */
    Ipoint           uColumns,                                        /* number of array columns */
    Ipoint           uRows                                           /* number of array rows */
);

extern Idnvreg InHoughTransformVReg (                               /* OUT Hough array identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,                                    /* source image identifier */
    Ipsstatic_vreg    snHough,                                        /* Hough array identifier */
    Ipsuint_vreg_chc sColumns,                                        /* number of array columns */
    Ipsuint_vreg_chc sRows                                           /* number of array rows */
);

extern Idnimage InImageBarChart (                                   /* OUT destination image identifier */
    Idnimage         nDestImage,                                    /* destination image identifier */
    Idntuple         nSize,                                           /* image size ND 5-tuple identifier */
    Ipoint           nMaximum,                                        /* maximum integer size */
    Ipoint           iStructure,                                       /* image structure option */
    Ipoint           iType                                           /* destination data type (internal) option */
);

extern Idnvreg InImageBarChartVReg (                               /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,                                    /* destination image identifier */
    Ipstuple_vreg     snSize,                                         /* image size ND 5-tuple identifier */
    Ipsuint_vreg_chc sMaximum,                                        /* maximum integer size */
    Ipsint_vreg_chc  sStructure,                                       /* image structure option */
    Ipsint_vreg_chc  sType                                           /* destination data type (internal) option */
);

extern Idnimage InImageConstant (                                   /* OUT destination image identifier */
    Idnimage         nDestImage,                                    /* destination image identifier */
    Idntuple         nSize,                                           /* image size ND 5-tuple identifier */
    Ipsparameter_pixel sValue,                                       /* value of constant pointer union */
    Ipoint           iType                                           /* destination data type (internal) option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InImageConstantVReg (                                     /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,                                     /* destination image identifier */
    Ipstuple_vreg    snSize,                                          /* image size ND 5-tuple identifier */
    Ipsparameter_pixel_vreg_chc sValue,                             /* value of constant */
    Ipsint_vreg_chc  sType                                           /* destination data type (internal) option */
);

extern Idnimage InImageGauss (                                       /* OUT destination image identifier */
    Idnimage         nDestImage,                                     /* destination image identifier */
    Idntuple         nSize,                                          /* image size ND 5-tuple identifier */
    Ipfloat          rHorizontalCentre, /* horizontal centre */
    Ipfloat          rVerticalCentre,   /* vertical centre */
    Ipfloat          rHorizontalSDev,   /* horizontal standard deviation */
    Ipfloat          rVerticalSDev     /* vertical standard deviation */
);

extern Idnvreg InImageGaussVReg (                                     /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,                                     /* destination image identifier */
    Ipstuple_vreg    snSize,                                          /* image size ND 5-tuple identifier */
    Ipsfloat_vreg_chc sHorizontalCentre, /* horizontal centre */
    Ipsfloat_vreg_chc sVerticalCentre,   /* vertical centre */
    Ipsfloat_vreg_chc sHorizontalSDev,   /* horizontal standard deviation */
    Ipsfloat_vreg_chc sVerticalSDev     /* vertical standard deviation */
);

extern Idnimage InImageGreyScale (                                   /* OUT destination image identifier */
    Idnimage         nDestImage,                                     /* destination image identifier */
    Idntuple         nSize,                                          /* image size ND 5-tuple identifier */
    Ipuint           uMaximum,                                       /* maximum positive integer size */
    Ipint            iGreyOption,                                     /* grey scale orientation option */
    Ipint            iTypeOption,                                    /* destination data type option */
);

extern Idnvreg InImageGreyScaleVReg (                               /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,                                     /* destination image identifier */
    Ipstuple_vreg    snSize,                                          /* image size ND 5-tuple identifier */
    Ipsuint_vreg_chc sMaximum,                                       /* maximum positive integer size */
    Ipsint_vreg_chc  sGreyOption,                                     /* grey scale orientation option */
    Ipsint_vreg_chc  sTypeOption,                                    /* destination data type option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnimage InImageRandomNumber ( /* OUT destination image identifier */
    Idnimage      nDestImage, /* destination image identifier */
    Idntuple      nSize,      /* image size ND 5-tuple identifier */
    Ipfloating    rMean,      /* probability density mean */
    Ipfloating    rSDev,      /* probability standard deviation */
    Ipfloating    rMaximum,   /* pixel amplitude maximum value */
    Ipfloating    rMinimum,   /* pixel amplitude minimum value */
    Ipoint        iOption     /* probability density option */
);

extern Idnvreg InImageRandomNumberVReg ( /* OUT destination image identifier */
    Idnvreg      nVReg,      /* output virtual register identifier or NULL */
    Ipsimage_vreg snDestImage, /* destination image identifier */
    Ipsstuple_vreg snSize,    /* image size 5-tuple identifier */
    Ipsfloat_vreg_chc sMean,  /* probability density mean */
    Ipsfloat_vreg_chc sSDev,  /* probability standard deviation */
    Ipsfloat_vreg_chc sMaximum, /* pixel amplitude maximum value */
    Ipsfloat_vreg_chc sMinimum, /* pixel amplitude minimum value */
    Ipsint_vreg_chc sOption    /* probability density option */
);

extern Idnhist InImportHist ( /* OUT histogram object identifier */
    Ipoint      *tuaData, /* external data value array pointer */
    Idnhist     nHist,    /* histogram object identifier */
    Ipsattrs_hist sAttrs  /* histogram attributes */
);

extern Idnvreg InImportHistVReg ( /* OUT histogram object identifier */
    Idnvreg      nVReg,      /* output virtual register identifier or NULL */
    Ipsaint_vreg_chc sData,  /* external data value array */
    Ipshist_vreg snHist,    /* histogram object identifier */
    Ipsattrs_hist_vreg_chc sAttrs /* histogram attributes */
);

extern Idnimage InImportImage ( /* OUT destination image identifier */
    Ipspiks_pixel_types stSourceImage, /* external source image data pointer union */
    Idnimage      nDestImage, /* destination image identifier */
    Ipoint        uPrecision,   /* number of source bits per pixel */
    Ipoint        iType,        /* source data type option */
    Ipoint        iApplBufferSize, /* data transfer buffer size or option */
    Ipoint        uStartPoint,   /* index offset within internal data */
    Ipoint        *tuDataSize    /* OUT post transfer data size pointer */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InImportImageVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipspiks_pixel_types_vreg_chc sSourceImage, /* external source image data */
    Ipsimage_vreg      snDestImage, /* destination image identifier */
    Ipsuint_vreg_chc    sPrecision, /* number of source bits per pixel */
    Ipsint_vreg_chc     sType /* source data type option */
);

extern Idnlut InImportLUT ( /* OUT destination lookup table identifier */
    Ipsparameter_pixel stData, /* external table of data values pointer union */
    Idnlut             nLUT, /* lookup data identifier */
    Ipsattrs_lut       sAttrs, /* lookup table attributes */
    Ipint              iApplBufferSize, /* data transfer buffer size or option */
    Ipuint             uStartPoint, /* index offset within internal data */
    Ipuint             *tuDataSize /* OUT post transfer data size pointer */
);

extern Idnvreg InImportLUTVReg ( /* OUT destination lookup table identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsparameter_pixel_vreg_chc sData, /* external table of data values */
    Ipslut_vreg      snLUT, /* lookup data identifier */
    Ipsattrs_lut_vreg_chc sAttrs /* lookup table attributes */
);

extern Idnmatrix InImportMatrix ( /* OUT destination matrix identifier */
    Ipsparameter_numeric stData, /* external table of data values pointer union */
    Idnmatrix            nMatrix, /* matrix identifier */
    Ipsattrs_matrix      sAttrs /* matrix attributes */
);

extern Idnvreg InImportMatrixVReg ( /* OUT destination matrix identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsparameter_numeric_vreg_chc sData, /* external table of data values */
    Ipsmatrix_vreg      snMatrix, /* matrix identifier */
    Ipsattrs_matrix_vreg_chc sAttrs /* matrix attributes */
);

extern Idnnbhood InImportNbhoodArray ( /* OUT destination array identifier */
    Ipsparameter_pixel stData, /* neighbourhood array data pointer union */
    Idnnbhood          nNbhood, /* neighbourhood array identifier */
    Ipsattrs_nbhood    sAttrs /* neighbourhood array attributes */
);

extern Idnvreg InImportNbhoodArrayVReg ( /* OUT destination array identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsparameter_pixel_vreg_chc sData, /* neighbourhood array data */
    Ipsnbhood_vreg      snNbhood, /* neighbourhood array identifier */
    Ipsattrs_nbhood_vreg_chc sAttrs /* neighbourhood array attributes */
);

```



```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnroi InImportROIArray (                                           /* OUT destination ROI identifier */
    Ipbool      *tbaData,          /* external data value array pointer */
    Idnroi      nROI,              /* ROI array identifier */
    Ipsattrs_roi sAttrs,           /* ROI attributes */
    Ipint       iApplBufferSize,   /* data transfer buffer size or option */
    Ipuint      uStartPoint,       /* index offset within internal data */
    Ipuint      *tuDataSize        /* OUT post transfer data size pointer */
);

extern Idnvreg InImportROIArrayVReg (                                       /* OUT destination ROI identifier */
    Idnvreg      nVReg,            /* output virtual register identifier or NULL */
    Ipsabool_vreg_chc sData,       /* external data value array */
    Ipsroi_vreg   snROI,           /* ROI array identifier */
    Ipsattrs_roi_vreg_chc sAttrs   /* ROI attributes */
);

extern Idnstatic InImportStaticArray (                                       /* OUT destination static array identifier */
    Ipsparameter_pixel stData,      /* external static array data pointer union */
    Idnstatic          nArray,       /* static array identifier */
    Ipsattrs_static    sAttrs,       /* static array attributes */
    Ipint              iApplBufferSize, /* data transfer buffer size or option */
    Ipuint              uStartPoint,    /* index offset within internal data */
    Ipuint              *tuDataSize    /* OUT post transfer data size pointer */
);

extern Idnvreg InImportStaticArrayVReg (                                       /* OUT destination static array identifier */
    Idnvreg      nVReg,            /* output virtual register identifier or NULL */
    Ipsparameter_pixel_vreg_chc sData, /* external static array data */
    Ipsstatic_vreg snArray,          /* static array identifier */
    Ipsattrs_static_vreg_chc sAttrs   /* static array attributes */
);

extern Idntuple InImportTuple (                                               /* OUT tuple identifier */
    Ipsparameter_basic stTupleData, /* external tuple data pointer union */
    Idntuple           nTupleId,     /* tuple identifier */
    Ipsattrs_tuple     sTupleAttrs   /* static array attributes */
);

extern Idnvreg InImportTupleVReg (                                           /* OUT tuple identifier */
    Idnvreg      nVReg,            /* output virtual register identifier or NULL */
    Ipsparameter_basic_vreg_chc sTupleData, /* external tuple data */
    Ipstuple_vreg snTupleId,         /* tuple identifier */
    Ipsattrs_tuple_vreg_chc sTupleAttrs   /* static array attributes */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnbounds InImportValueBounds ( /* OUT value bounds collection identifier */
    Ipsparameter_arith    stData, /* external data value collection pointer union */
    Idnbounds             nValueBounds, /* value bounds collection identifier */
    Ipsattrs_bounds       sAttrs, /* value bounds collection attributes */
    Ipoint                iApplBufferSize, /* data transfer buffer size or option */
    Ipoint                uStartPoint, /* index offset within internal data */
    Ipoint                *tuDataSize /* OUT post transfer data size pointer */
);

extern Idnvreg InImportValueBoundsVReg ( /* OUT value bounds collection identifier */
    Idnvreg              nVReg, /* output virtual register identifier or NULL */
    Ipsparameter_arith_vreg_chc sData, /* external data value collection */
    Ipsbounds_vreg       snValue, /* value bounds collection identifier */
    Ipsattrs_bounds_vreg_chc sAttrs /* value bounds collection attributes */
);

extern Idnnbhood InImpulseBoxcar ( /* OUT impulse response array identifier */
    Idnnbhood           nArray, /* impulse response array identifier */
    Idntuple            nSize, /* array size ND 5-tuple identifier */
    Idntuple            nKey, /* key pixel SD 5-tuple identifier */
    Ipederiv            eOption /* derivative option */
);

extern Idnvreg InImpulseBoxcarVReg ( /* OUT impulse response array identifier */
    Idnvreg              nVReg, /* output virtual register identifier or NULL */
    Ipsnbhood_vreg       snArray, /* impulse response array identifier */
    Ipstuple_vreg        snSize, /* array size ND 5-tuple identifier */
    Ipstuple_vreg        snKey, /* key pixel SD 5-tuple identifier */
    Ipsderiv_vreg_chc    seOption /* derivative option */
);

extern Idnnbhood InImpulseDerivGauss ( /* OUT impulse response array identifier */
    Idnnbhood           nArray, /* impulse response array identifier */
    Idntuple            nSize, /* array size ND 5-tuple identifier */
    Idntuple            nKey, /* key pixel SD 5-tuple identifier */
    Ipfloating           rColumnSDev, /* column index standard deviation */
    Ipfloating           rRowSDev, /* row index standard deviation */
    Ipederiv            eOption /* derivative option */
);

extern Idnvreg InImpulseDerivGaussVReg ( /* OUT impulse response array identifier */
    Idnvreg              nVReg, /* output virtual register identifier or NULL */
    Ipsnbhood_vreg       snArray, /* impulse response array identifier */
    Ipstuple_vreg        snSize, /* array size ND 5-tuple identifier */
    Ipstuple_vreg        snKey, /* key pixel SD 5-tuple identifier */
    Ipsfloat_vreg_chc    sColumnSDev, /* column index standard deviation */
    Ipsfloat_vreg_chc    sRowSDev, /* row index standard deviation */
    Ipsderiv_vreg_chc    seOption /* derivative option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnnbhood InImpulseDiffGauss ( /* OUT impulse response array identifier */
    Idnnbhood          nArray, /* impulse response array identifier */
    Idntuple           nSize, /* array size ND 5-tuple identifier */
    Idntuple           nKey, /* key pixel SD 5-tuple identifier */
    Ipfloat            rColumnWide, /* column index wide standard deviation */
    Ipfloat            rRowWide, /* row index wide standard deviation */
    Ipfloat            rColumnNarrow, /* column index narrow standard deviation */
    Ipfloat            rRowNarrow /* row index narrow standard deviation */
);

extern Idnvreg InImpulseDiffGaussVReg ( /* OUT impulse response array identifier */
    Idnvreg            nVReg, /* output virtual register identifier or NULL */
    Ipsnbhood_vreg     snArray, /* impulse response array identifier */
    Ipstuple_vreg      snSize, /* array size ND 5-tuple identifier */
    Ipstuple_vreg      snKey, /* key pixel SD 5-tuple identifier */
    Ipsfloat_vreg_chc  sColumnWide, /* column index wide standard deviation */
    Ipsfloat_vreg_chc  sRowWide, /* row index wide standard deviation */
    Ipsfloat_vreg_chc  sColumnNarrow, /* column index narrow standard deviation */
    Ipsfloat_vreg_chc  sRowNarrow /* row index narrow standard deviation */
);

extern Idnnbhood InImpulseElliptical ( /* OUT impulse response array identifier */
    Idnnbhood          nArray, /* impulse response array identifier */
    Idntuple           nSize, /* array size ND 5-tuple identifier */
    Idntuple           nKey /* key pixel SD 5-tuple identifier */
);

extern Idnvreg InImpulseEllipticalVReg ( /* OUT impulse response array identifier */
    Idnvreg            nVReg, /* output virtual register identifier or NULL */
    Ipsnbhood_vreg     snArray, /* impulse response array identifier */
    Ipstuple_vreg      snSize, /* array size ND 5-tuple identifier */
    Ipstuple_vreg      snKey /* key pixel SD 5-tuple identifier */
);

extern Idnnbhood InImpulseGauss ( /* OUT impulse response array identifier */
    Idnnbhood          nArray, /* impulse response array identifier */
    Idntuple           nSize, /* array size ND 5-tuple identifier */
    Idntuple           nKey, /* key pixel SD 5-tuple identifier */
    Ipfloat            rHorizontalSDev, /* horizontal standard deviation */
    Ipfloat            rVerticalSDev /* vertical standard deviation */
);

extern Idnvreg InImpulseGaussVReg ( /* OUT impulse response array identifier */
    Idnvreg            nVReg, /* output virtual register identifier or NULL */
    Ipsnbhood_vreg     snArray, /* impulse response array identifier */
    Ipstuple_vreg      snSize, /* array size ND 5-tuple identifier */
    Ipstuple_vreg      snKey, /* key pixel SD 5-tuple identifier */
    Ipsfloat_vreg_chc  sHorizaontalSDev, /* horizontal standard deviation */
    Ipsfloat_vreg_chc  sVerticalSDev /* vertical standard deviation */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnnbhood InImpulseLaplaceGauss ( /* OUT impulse response array identifier */
    Idnnbhood      nArray, /* impulse response array identifier */
    Idntuple       nSize, /* array size ND 5-tuple identifier */
    Idntuple       nKey, /* key pixel SD 5-tuple identifier */
    Ipfloat        rSDev /* standard deviation */
);

extern Idnvreg InImpulseLaplaceGaussVReg ( /* OUT impulse response array identifier */
    Idnvreg        nVReg, /* output virtual register identifier or NULL */
    Ipsnbhood_vreg snArray, /* impulse response array identifier */
    Ipstuple_vreg  snSize, /* array size ND 5-tuple identifier */
    Ipstuple_vreg  snKey, /* key pixel SD 5-tuple identifier */
    Ipsfloat_vreg_chc sSDev /* standard deviation */
);

extern Idnnbhood InImpulsePyramid ( /* OUT impulse response array identifier */
    Idnnbhood      nArray, /* impulse response array identifier */
    Idntuple       nSize, /* array size ND 5-tuple identifier */
    Idntuple       nKey, /* key pixel SD 5-tuple identifier */
    Ipint          iOption /* impulse, pyramid shape option */
);

extern Idnvreg InImpulsePyramidVReg ( /* OUT impulse response array identifier */
    Idnvreg        nVReg, /* output virtual register identifier or NULL */
    Ipsnbhood_vreg snArray, /* impulse response array identifier */
    Ipstuple_vreg  snSize, /* array size ND 5-tuple identifier */
    Ipstuple_vreg  snKey, /* key pixel SD 5-tuple identifier */
    Ipsint_vreg_chc sOption /* impulse, pyramid shape option */
);

extern Idnnbhood InImpulseRectangular ( /* OUT impulse response array identifier */
    Idnnbhood      nArray, /* impulse response array identifier */
    Idntuple       nSize, /* array size ND 5-tuple identifier */
    Idntuple       nKey /* key pixel SD 5-tuple identifier */
);

extern Idnvreg InImpulseRectangularVReg ( /* OUT impulse response array identifier */
    Idnvreg        nVReg, /* output virtual register identifier or NULL */
    Ipsnbhood_vreg snArray, /* impulse response array identifier */
    Ipstuple_vreg  snSize, /* array size ND 5-tuple identifier */
    Ipstuple_vreg  snKey /* key pixel SD 5-tuple identifier */
);

extern Idnnbhood InImpulseSinc ( /* OUT impulse response array identifier */
    Idnnbhood      nArray, /* impulse response array identifier */
    Idntuple       nSize, /* array size ND 5-tuple identifier */
    Idntuple       nKey, /* key pixel SD 5-tuple identifier */
    Ipfloat        rHorizontalScale, /* horizontal scale factor */
    Ipfloat        rVerticalScale /* vertical scale factor */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InImpulseSincVReg (                                           /* OUT impulse response array identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsnbhood_vreg   snArray,        /* impulse response array identifier */
    Ipstuple_vreg    snSize,         /* array size ND 5-tuple identifier */
    Ipstuple_vreg    snKey,          /* key pixel SD 5-tuple identifier */
    Ipsfloat_vreg_chc sHorizontalScale, /* horizontal scale factor */
    Ipsfloat_vreg_chc sVerticalScale   /* vertical scale factor */
);

extern Ipsobject IsnInputObject (                                           /* OUT PIKS identifier */
    Ipnport          nExternal,      /* external identifier */
    Ipsobject        snInternal,     /* PIKS identifier */
    Ipint            iTransferOption, /* source data transfer option */
    Ipint            iSyntaxOption,  /* source data transfer syntax option */
    Ipint            iApplBufferSize, /* data transfer buffer size or option */
    Ipuint           uStartPoint,     /* index offset within internal data */
    Ipuint           *tuDataSize      /* OUT post transfer data size pointer */
);

extern Idnvreg InInputObjectVReg (                                           /* OUT PIKS identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsport_vreg     snExternal,     /* external identifier */
    Ipsobject_vreg   snInternal,     /* PIKS identifier */
    Ipsint_vreg_chc  sTransfer,      /* source data transfer option */
    Ipsint_vreg_chc  sSyntax        /* source data transfer syntax option */
);

extern void IvInquireChainEnvironment (                                       /* no return value output */
    char             *tzInformation,  /* OUT chaining attribute list */
    Ipevalidity      *teValid,        /* OUT data validity indicator */
    Ipint            iApplBufferSize, /* data transfer buffer size or option */
    Ipuint           uStartPoint,     /* index offset within internal data */
    Ipuint           *tuDataSize      /* OUT post transfer data size pointer */
);

extern void IvInquireChainStatus (                                           /* no return value output */
    Idnchain         nChain,          /* chain identifier */
    Ipuint           *uStatus,        /* OUT chain status */
    Ipevalidity      *teValid        /* OUT data validity indicator */
);

extern void IvInquireElements (                                              /* no return value output */
    Ipint            *tiaInput,       /* array of element registration numbers pointer */
    Ipbool           *tbaOutput,      /* OUT array of registration responses pointer */
    Ipuint           uSize,           /* array size */
    char             *tzLabel,        /* OUT private label information */
    Ipevalidity      *teValid        /* OUT data validity indicator */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern void IvInquireImage (
    Idnimage          nSourceImage, /* no return value output */
    Ipsattrs_image    *tsAttrs,     /* source image identifier */
    Ipsbind_image     *tsBind,      /* OUT image attributes */
    Ipevalidity       *teValid,     /* OUT image match point and ROI bindings */
    );                             /* OUT data validity indicator */

extern void IvInquireIndexAssignment (
    char              *tcFirst,     /* no return value output */
    char              *tc2nd,       /* OUT first operator index assignment */
    char              *tcThird,     /* OUT second operator index assignment */
    char              *tcFourth,    /* OUT third operator index assignment */
    char              *tcFifth,     /* OUT fourth operator index assignment */
    Ipevalidity       *teValid,     /* OUT fifth operator index assignment */
    );                             /* OUT data validity indicator */

extern void IvInquireNonImageObject (
    Ipsnon_image_object snObject,   /* no return value output */
    Ipsattrs_non_image  stData,     /* object identifier */
    Ipoint              iType,      /* OUT non-image attributes */
    Ipevalidity         *teValid,   /* object type option */
    );                             /* OUT data validity indicator */

extern void IvInquirePIKSImp (
    Ipspiks_impl        *tsData,    /* no return value output */
    Ipevalidity         *teValid,   /* OUT implementation information */
    Ipoint              iApplBufferSize, /* OUT data validity indicator */
    Ipoint              uStartPoint,  /* data transfer buffer size or option */
    Ipoint              *tuDataSize,  /* index offset within internal data */
    );                             /* OUT post transfer data size pointer */

extern void IvInquirePIKSStatus (
    Ipspiks_states      *tsStates,  /* no return value output */
    Ipspiks_modes       *tsModes,   /* OUT inquire PIKS states pointer */
    Ipevalidity         *teValid,   /* OUT inquire PIKS modes pointer */
    );                             /* OUT data validity indicator */

extern void IvInquireRepository (
    Ipoint              *tiaInput,   /* no return value output */
    Ipoint              *tbaOutput,  /* repository registration numbers array pointer */
    Ipoint              uSize,       /* OUT registration responses array pointer */
    char               *tzLabel,    /* array size */
    Ipoint              iClassOption, /* OUT private label information */
    Ipevalidity         *teValid,   /* repository class option */
    );                             /* OUT data validity indicator */

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern void IvInquireResampling ( /* no return value output */
    Ipint      *tiaInput, /* resampling registration numbers array pointer */
    Ipbool     *tbaOutput, /* OUT resampling responses array pointer */
    Ipuint     uSize, /* array size */
    char       *tzLabel, /* OUT private label information */
    Ipevalidity *teValid /* OUT data validity indicator */
);

extern Idnimage InInsertPixelPlane ( /* OUT destination image identifier */
    Idnimage     nSourceImage, /* source image identifier */
    Idnimage     nDestImage, /* destination image identifier */
    Ipuint       uDepth, /* depth index value */
    Ipuint       uTemporal, /* temporal index value */
    Ipuint       uBand /* band index value */
);

extern Idnvreg InInsertPixelPlaneVReg ( /* OUT destination image identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage, /* destination image identifier */
    Ipsuint_vreg_chc sDepth, /* depth index value */
    Ipsuint_vreg_chc sTemporal, /* temporal index value */
    Ipsuint_vreg_chc sBand /* band index value */
);

extern Idnroi InInsertPixelPlaneROI ( /* OUT destination ROI identifier */
    Idnroi      nSourceROI, /* source ROI identifier */
    Idnroi      nDestROI, /* destination ROI identifier */
    Ipuint      uDepth, /* depth index value */
    Ipuint      uTemporal, /* temporal index value */
    Ipuint      uBand /* band index value */
);

extern Idnvreg InInsertPixelPlaneROIvReg ( /* OUT destination ROI identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    Ipsroi_vreg snSourceROI, /* source ROI identifier */
    Ipsroi_vreg snDestROI, /* destination ROI identifier */
    Ipsuint_vreg_chc sDepth, /* depth index value */
    Ipsuint_vreg_chc sTemporal, /* temporal index value */
    Ipsuint_vreg_chc sBand /* band index value */
);

```



```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnimage InLabelObjects (                                           /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage,   /* destination image identifier */
    Ipoint        uIndex3,      /* third index value */
    Ipoint        uIndex4,      /* fourth index value */
    Ipoint        uIndex5,      /* fifth index value */
    Ipoint        uLimit,       /* object number limit */
    Ipoint        *tuObjects    /* OUT number of objects labeled */
);

extern Idnvreg InLabelObjectsVReg (                                       /* OUT destination image identifier */
    Idnvreg      nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage,   /* destination image identifier */
    Ipsuint_vreg_chc sIndex3,    /* third index value */
    Ipsuint_vreg_chc sIndex4,    /* fourth index value */
    Ipsuint_vreg_chc sIndex5,    /* fifth index value */
    Ipsuint_vreg_chc sLimit,     /* number limit */
    Idnvreg      nObjectsVReg    /* OUT number of objects labeled */
);

extern Idnimage InLawsTextureFeatures (                                   /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage,   /* destination image identifier */
    Idnnbhood     nFilter,      /* low pass filter response array identifier */
    Idnnbhood     nMask        /* feature mask response array identifier */
);

extern Idnvreg InLawsTextureFeaturesVReg (                               /* OUT destination image identifier */
    Idnvreg      nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage,   /* destination image identifier */
    Ipsnbhood_vreg snFilter,     /* low pass filter response array identifier */
    Ipsnbhood_vreg snMask       /* feature mask response array identifier */
);

extern Idnimage InLevelSlice (                                           /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage,   /* destination image identifier */
    Ipsparameter_arith stUpper,  /* upper slice amplitude pointer union */
    Ipsparameter_arith stLower,  /* lower slice amplitude pointer union */
    Ipsparameter_arith stSlice,  /* slice output value pointer union */
    Ipebackground  eOption      /* level slice background option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InLevelSliceVReg (                                           /* OUT destination image identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg    nSourceImage,   /* source image identifier */
    Ipsimage_vreg    nDestImage,     /* destination image identifier */
    Ipsparameter_arith_vreg_chc    sUpper, /* upper slice amplitude */
    Ipsparameter_arith_vreg_chc    sLower, /* lower slice amplitude */
    Ipsparameter_arith_vreg_chc    sSlice, /* slice output value */
    Ipsbackground_vreg_chc    sOption /* level slice background option */
);

extern Ipfloor *ItraLineProfile (                                           /* OUT array of pixels along line pointer */
    Idnimage          nSourceImage,   /* source image identifier */
    Ipfloor           *traDestArray, /* array of pixels along line objects pointer */
    Idntuple          nStart,         /* start point ND 5-tuple */
    Idntuple          nEnd,          /* end point ND 5-tuple */
    Ipuint            uSamples,       /* number of samples along line */
    Ipint             iApplBufferSize, /* data transfer buffer size or option */
    Ipuint            uStartPoint,    /* index offset within internal data */
    Ipuint            *tuDataSize     /* OUT post transfer data size pointer */
);

extern Idnvreg InLineProfileVReg (                                           /* OUT array of pixels along line pointer */
    Ipsimage_vreg    snSourceImage,   /* source image identifier */
    Idnvreg          nDestVReg,       /* pixels along line virtual register identifier */
    Ipstuple_vreg    snStart,         /* start point ND 5-tuple */
    Ipstuple_vreg    snEnd,          /* end point ND 5-tuple */
    Ipsuint_vreg_chc sSamples         /* number of samples along line */
);

extern Idnimage InLookup (                                                  /* OUT destination image identifier */
    Idnimage          nSourceImage,   /* source image identifier */
    Idnimage          nDestImage,     /* destination image identifier */
    Idnlut            nLUT,           /* lookup table identifier */
    Ipuint            uOffset,        /* source offset entry */
    Ipint             iOption         /* lookup table mode option */
);

extern Idnvreg InLookupVReg (                                              /* OUT destination image identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage,   /* source image identifier */
    Ipsimage_vreg    snDestImage,     /* destination image identifier */
    Ipslut_vreg      snLUT,           /* lookup table identifier */
    Ipsuint_vreg_chc sOffset,         /* source offset entry */
    Ipsint_vreg_chc  sOption         /* lookup table mode option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnimage InLookupInterp ( /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage,   /* destination image identifier */
    Idnlut        nLUT,         /* lookup table identifier */
    Ipoint        uOffset       /* source offset bits */
);

extern Idnvreg InLookupInterpVReg ( /* OUT destination image identifier */
    Idnvreg      nVReg,         /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage,   /* destination image identifier */
    Ipslut_vreg   snLUT,         /* lookup table identifier */
    Ipsuint_vreg_chc sOffset     /* source offset bits */
);

extern Idnimage InLumGeneration ( /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage,   /* destination image identifier */
    Ipoint        iOption       /* luminance RGB colour option */
);

extern Idnvreg InLumGenerationVReg ( /* OUT destination image identifier */
    Idnvreg      nVReg,         /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage,   /* destination image identifier */
    Ipsint_vreg_chc sOption      /* luminance RGB colour option */
);

extern Idnstatic InMarkovSpectrum ( /* OUT power spectrum static array identifier */
    Idnstatic      nFilter,      /* power spectrum static array identifier */
    Idntuple        nSize,       /* array size ND 5-tuple identifier */
    Ipfloating      rColumn,     /* column index correlation factor */
    Ipfloating      rRow,        /* row index correlation factor */
    Ipoint          iOption      /* Markov spectrum symmetry option */
);

extern Idnvreg InMarkovSpectrumVReg ( /* OUT power spectrum static array identifier */
    Idnvreg      nVReg,         /* output virtual register identifier or NULL */
    Ipsstatic_vreg snFilter,     /* power spectrum static array identifier */
    Ipsstuple_vreg snSize,       /* array size ND 5-tuple identifier */
    Ipsfloat_vreg_chc sColumn,   /* column index correlation factor */
    Ipsfloat_vreg_chc sRow,      /* row index correlation factor */
    Ipsint_vreg_chc sOption      /* Markov spectrum symmetry option */
);

extern void IvMoments ( /* no return value output */
    Idnimage      nSourceImage, /* source image identifier */
    Ipfloating    *trMean,      /* OUT mean value */
    Ipfloating    *trSDev       /* OUT standard deviation value */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern void IvMomentsVReg (
    Ipsimage_vreg      snSourceImage, /* no return value output */
    Idnvreg            nMean,          /* source image identifier */
    Idnvreg            nSDev,          /* OUT mean value virtual register identifier */
);                                     /* OUT deviation value virtual register identifier*/

extern Idnimage InMonadicArith (
    Idnimage            nSourceImage, /* OUT destination image identifier */
    Idnimage            nDestImage,   /* source image identifier */
    Ipsparameter_arith  stConstant,    /* destination image identifier */
    Ipint              iOption,        /* value of constant pointer union */
);                                     /* monadic, arithmetic combination option */

extern Idnvreg InMonadicArithVReg (
    Idnvreg            nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg      snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg      snDestImage,   /* source image identifier */
    Ipsparameter_arith_vreg_chc  sConstant, /* destination image identifier */
    Ipsint_vreg_chc    sOption,        /* value of constant */
);                                     /* monadic, arithmetic combination option */

extern Idnimage InMonadicComplex (
    Idnimage            nSourceImage, /* OUT destination image identifier */
    Idnimage            nDestImage,   /* source image identifier */
    Ipscomplex          sxConstant,    /* destination image identifier */
    Ipint              iOption,        /* value of constant */
);                                     /* monadic, complex combination option */

extern Idnvreg InMonadicComplexVReg (
    Idnvreg            nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg      snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg      snDestImage,   /* source image identifier */
    Ipscomplex_vreg_chc  sConstant,    /* destination image identifier */
    Ipsint_vreg_chc    sOption,        /* value of constant */
);                                     /* monadic, complex combination option */

extern Idnimage InMonadicLogical (
    Idnimage            nSourceImage, /* OUT destination image identifier */
    Idnimage            nDestImage,   /* source image identifier */
    Ipsparameter_logical  stConstant,  /* destination image identifier */
    Ipint              iOption,        /* value of constant pointer union */
);                                     /* monadic, logical combination option */

extern Idnvreg InMonadicLogicalVReg (
    Idnvreg            nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg      snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg      snDestImage,   /* source image identifier */
    Ipsparameter_logical_vreg_chc  sConstant, /* destination image identifier */
    Ipsint_vreg_chc    sOption,        /* value of constant */
);                                     /* monadic, logical combination option */

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnroi InMonadicLogicalROI ( /* OUT destination ROI identifier */
    Idnroi          nSourceROI, /* source ROI identifier */
    Idnroi          nDestROI,  /* destination ROI identifier */
    Ipsparameter_logical stConstant, /* value of constant pointer union */
    Ipint           iOption      /* monadic, logical combination option */
);

extern Idnvreg InMonadicLogicalROIvReg ( /* OUT destination ROI identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsroi_vreg      snSourceROI, /* source ROI identifier */
    Ipsroi_vreg      snDestROI,  /* destination ROI identifier */
    Ipsparameter_logical_vreg_chc sConstant, /* value of constant */
    Ipsint_vreg_chc  sOption      /* monadic, logical combination option */
);

extern Idnimage InMorphicProcessor ( /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Idnlut            nLUT,         /* hit or miss lookup table identifier */
    Ipint             iterations,    /* number of iterations */
    Ipuint            *tuCycles     /* OUT number of cycles */
);

extern Idnvreg InMorphicProcessorVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg     snSourceImage, /* source image identifier */
    Ipsimage_vreg     snDestImage,   /* destination image identifier */
    Ipslut_vreg       snLUT,         /* hit or miss lookup table identifier */
    Ipsint_vreg_chc   sIterations,    /* number of iterations */
    Idnvreg           nCycles        /* OUT cycles virtual register identifier */
);

extern Idnroi InMorphicProcessorROI ( /* OUT destination ROI identifier */
    Idnroi          nSourceROI, /* source ROI identifier */
    Idnroi          nDestROI,   /* destination ROI identifier */
    Idnlut          nLUT,       /* hit or miss lookup table identifier */
    Ipint           iterations, /* number of iterations */
    Ipuint          *tuCycles   /* OUT number of cycles */
);

extern Idnvreg InMorphicProcessorROIvReg ( /* OUT destination ROI identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsroi_vreg      snSourceROI, /* source ROI identifier */
    Ipsroi_vreg      snDestROI,   /* destination ROI identifier */
    Ipslut_vreg       snLUT,       /* hit or miss lookup table identifier */
    Ipsint_vreg_chc   sIterations, /* number of iterations */
    Idnvreg           nCycles      /* OUT cycles virtual register identifier */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnimage InMorphology (
    Idnimage      nSourceImage, /* OUT destination image identifier */
    Idnimage      nDestImage,   /* source image identifier          */
    Ipint         iIterations,   /* destination image identifier     */
    Ipuint        *tuCycles,     /* number of iterations            */
    Ipint         iOption,       /* OUT number of cycles            */
    /* morphological option */
);

extern Idnvreg InMorphologyVReg (
    Idnvreg      nVReg, /* OUT destination image identifier */
    Ipsimage_vreg snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg snDestImage,   /* source image identifier          */
    Ipsint_vreg_chc sIterations, /* destination image identifier     */
    Idnvreg        nCycles,       /* number of iterations            */
    Ipsint_vreg_chc sOption,      /* OUT cycles virtual register identifier */
    /* morphological option */
);

extern Idnroi InMorphologyROI (
    Idnroi      nSourceROI, /* OUT destination ROI identifier */
    Idnroi      nDestROI,   /* source ROI identifier          */
    Ipint       iIterations, /* destination ROI identifier     */
    Ipuint      *tuCycles,  /* number of iterations            */
    Ipint       iOption,    /* OUT number of cycles            */
    /* morphological option */
);

extern Idnvreg InMorphologyROIvReg (
    Idnvreg      nVReg, /* OUT destination ROI identifier */
    Ipsroi_vreg snSourceROI, /* output virtual register identifier or NULL */
    Ipsroi_vreg snDestROI,   /* source ROI identifier          */
    Ipsint_vreg_chc sIterations, /* destination ROI identifier     */
    Idnvreg        nCycles,       /* number of iterations            */
    Ipsint_vreg_chc sOption,      /* OUT cycles virtual register identifier */
    /* morphological option */
);

extern Idnimage InNbrCount (
    Idnimage      nSourceImage, /* OUT destination image identifier */
    Idnimage      nDestImage,   /* source image identifier          */
    Idnbhhood     nArray,       /* destination image identifier     */
    /* structuring element array identifier */
);

extern Idnvreg InNbrCountVReg (
    Idnvreg      nVReg, /* OUT destination image identifier */
    Ipsimage_vreg snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg snDestImage,   /* source image identifier          */
    Ipsnbhood_vreg snArray,      /* destination image identifier     */
    /* structuring element array identifier */
);

```



```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnimage InNoiseCombination ( /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage, /* destination image identifier */
    Ipsfloat      rRatio, /* signal-to-noise ratio */
    Ipinint       iOption /* probability density option */
);

extern Idnvreg InNoiseCombinationVReg ( /* OUT destination image identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage, /* destination image identifier */
    Ipsfloat_vreg_chc sRatio, /* signal-to-noise ratio */
    Ipsint_vreg_chc sOption /* probability density option */
);

extern Idnimage InOpenClose ( /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage, /* destination image identifier */
    Idnnbhood     nArray, /* structuring element array identifier */
    Ipeopen_close eOperationOption, /* open close operation option */
    Iperotation   eErosionOption, /* erosion structuring element rotation option */
    Iperotation   eDilateOption /* dilation structuring element rotation option */
);

extern Idnvreg InOpenCloseVReg ( /* OUT destination image identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage, /* destination image identifier */
    Ipsnbhood_vreg snArray, /* structuring element array identifier */
    Ipsopen_close_vreg_chc sOperationOption, /* open close operation option */
    Ipsrotation_vreg_chc sErosionOption, /* erosion structuring element rotation option */
    Ipsrotation_vreg_chc sDilateOption /* dilation structuring element rotation option */
);

extern Idnroi InOpenCloseROI ( /* OUT destination ROI identifier */
    Idnroi      nSourceROI, /* source ROI identifier */
    Idnroi      nDestROI, /* destination ROI identifier */
    Idnnbhood   nArray, /* structuring element array identifier */
    Ipeopen_close eOperationOption, /* open close operation option */
    Iperotation   eErosionOption, /* erosion structuring element rotation option */
    Iperotation   eDilateOption /* dilation structuring element rotation option */
);

```



```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InOpenCloseROIvReg ( /* OUT destination ROI identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsroi_vreg      snSourceROI, /* source ROI identifier */
    Ipsroi_vreg      snDestROI, /* destination ROI identifier */
    Ipsnbhood_vreg   snArray, /* structuring element array identifier */
    Ipsopen_close_vreg_chc sOperationOption, /* open close operation option */
    Ipsrotation_vreg_chc sErosionOption, /* erosion structuring element rotation option */
    Ipsrotation_vreg_chc sDilateOption /* dilation structuring element rotation option */
);

extern void IvOpenPIKS ( /* no return value output */
    Ipnerror          nFile /* error file identifier */
);

extern Idnimage InOutlierRemoval ( /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage, /* destination image identifier */
    Idnnbhood         nImpulse, /* impulse response array identifier */
    Ipfloating         rThreshold /* threshold value */
);

extern Idnvreg InOutlierRemovalVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg     snSourceImage, /* source image identifier */
    Ipsimage_vreg     snDestImage, /* destination image identifier */
    Ipsnbhood_vreg    snImpulse, /* impulse response array identifier */
    Ipsfloat_vreg_chc sThreshold /* threshold value */
);

extern Ipnport InOutputObject ( /* OUT external identifier */
    Ipsobject         nInternal, /* PIKS identifier */
    Ipnport           nExternal, /* data destination identifier */
    Ipint             iTransferOption, /* destination data transfer option */
    Ipint             iSyntaxOption, /* data destination syntax option */
    Ipint             iApplBufferSize, /* data transfer buffer size or option */
    Ipuint            uStartPoint, /* index offset within internal data */
    Ipuint            *tuDataSize /* OUT post transfer data size pointer */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnimage InPaintPixels (                                           /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage,  /* destination image identifier */
    Idnnbhood     nBrush,      /* brush array identifier */
    Ipoint        *tuaControl,  /* control point array pointer */
    Ipoint        uIndex3,     /* third index value */
    Ipoint        uIndex4,     /* fourth index value */
    Ipoint        uIndex5,     /* fifth index value */
    Ipoint        uPoints,     /* number of control points */
    Ipoint        iPaintingOption, /* painting option */
    Ipoint        iBrushOption  /* brush option */
);

extern Idnvreg InPaintPixelsVReg (                                       /* OUT destination image identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage,  /* destination image identifier */
    Ipsnbhood_vreg snBrush,      /* brush array identifier */
    Ipsauint_vreg_chc sControl,  /* control point array */
    Ipsuint_vreg_chc sIndex3,   /* third index value */
    Ipsuint_vreg_chc sIndex4,   /* fourth index value */
    Ipsuint_vreg_chc sIndex5,   /* fifth index value */
    Ipsuint_vreg_chc sPoints,   /* number of control points */
    Ipsint_vreg_chc sPaintingOption, /* painting option */
    Ipsint_vreg_chc sBrushOption /* brush option */
);

extern Ipoint *ItuaPerimeterCodeGenerator ( /* OUT crack codes array pointer */
    Idnimage      nSourceImage, /* source image identifier */
    Ipoint        *tuaCodes,     /* crack codes array pointer */
    Ipoint        uIndex3,     /* third index value */
    Ipoint        uIndex4,     /* fourth index value */
    Ipoint        uIndex5,     /* fifth index value */
    Ipoint        uLimit,      /* object number limit */
    Ipoint        iApplBufferSize, /* data transfer buffer size or option */
    Ipoint        uStartPoint,  /* index offset within internal data */
    Ipoint        *tuDataSize   /* OUT post transfer data size pointer */
);

extern Idnimage InPolarToCartesian ( /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage,  /* destination image identifier */
    Ifloat        rMinimumRho,  /* minimum value of rho */
    Ifloat        rMaximumRho,  /* maximum value of rho */
    Ifloat        rMinimumTheta, /* minimum value of theta */
    Ifloat        rMaximumTheta /* maximum value of theta */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InPolarToCartesianVReg (                                     /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipsfloat_vreg_chc sMinimumRho,  /* minimum value of rho */
    Ipsfloat_vreg_chc sMaximumRho,  /* maximum value of rho */
    Ipsfloat_vreg_chc sMinimumTheta, /* minimum value of theta */
    Ipsfloat_vreg_chc sMaximumTheta /* maximum value of theta */
);

extern Idnimage InPowerLawScaling (                                     /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Ipfloat           rExponent      /* power law exponent */
);

extern Idnvreg InPowerLawScalingVReg (                               /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipsfloat_vreg_chc sExponent      /* power law exponent */
);

extern Idnimage InPseudocolour (                                     /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Idnlut            snLUT,        /* lookup table identifier */
    Ipoint            uOffset       /* number of lookup table source offset entries */
);

extern Idnvreg InPseudocolourVReg (                               /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipslut_vreg      snLUT,        /* lookup table identifier */
    Ipsuint_vreg_chc sOffset       /* number of lookup table source offset entries */
);

extern Idnimage InPutColourPixel (                                     /* OUT destination image identifier */
    Idnimage          nDestImage, /* destination image identifier */
    Ipsparameter_colour stBand0,   /* band 0 pixel value pointer union */
    Ipsparameter_colour stBand1,   /* band 1 pixel value pointer union */
    Ipsparameter_colour stBand2,   /* band 2 pixel value pointer union */
    Ipsparameter_colour stBand3,   /* band 3 pixel value pointer union */
    Ipoint            uHorizontal, /* horizontal index value */
    Ipoint            uVertical,   /* vertical index value */
    Ipoint            uDepth,      /* depth index value */
    Ipoint            uTemporal    /* temporal index value */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InPutColourPixelVReg (                                     /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,                                     /* destination image identifier */
    Ipsparameter_colour_vreg_chc sBand0,                             /* band 0 pixel value virtual register identifier */
    Ipsparameter_colour_vreg_chc sBand1,                             /* band 1 pixel value virtual register identifier */
    Ipsparameter_colour_vreg_chc sBand2,                             /* band 2 pixel value virtual register identifier */
    Ipsparameter_colour_vreg_chc sBand3,                             /* band 3 pixel value virtual register identifier */
    Ipsuint_vreg_chc  sHorizontal,                                    /* horizontal index value */
    Ipsuint_vreg_chc  sVertical,                                      /* vertical index value */
    Ipsuint_vreg_chc  sDepth,                                        /* depth index value */
    Ipsuint_vreg_chc  sTemporal,                                     /* temporal index value */
);

extern Idnimage InPutPixel (                                           /* OUT destination image identifier */
    Idnimage        nDestImage,                                       /* destination image identifier */
    Ipsparameter_pixel stPixel,                                       /* pixel value pointer union */
    Ipuint           uHorizontal,                                     /* horizontal index value */
    Ipuint           uVertical,                                       /* vertical index value */
    Ipuint           uDepth,                                           /* depth index value */
    Ipuint           uTemporal,                                       /* temporal index value */
    Ipuint           uBand,                                           /* band index value */
);

extern Idnvreg InPutPixelVReg (                                       /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,                                     /* destination image identifier */
    Ipsparameter_pixel_vreg_chc sPixel,                               /* pixel value */
    Ipsuint_vreg_chc  sHorizontal,                                    /* horizontal index value */
    Ipsuint_vreg_chc  sVertical,                                      /* vertical index value */
    Ipsuint_vreg_chc  sDepth,                                        /* depth index value */
    Ipsuint_vreg_chc  sTemporal,                                     /* temporal index value */
    Ipsuint_vreg_chc  sBand,                                         /* band index value */
);

extern Idnroi InPutPixelROI (                                         /* OUT destination ROI identifier */
    Idnroi           nDestROI,                                       /* destination ROI identifier */
    Ipbool           bPixel,                                         /* pixel value */
    Ipuint           uHorizontal,                                     /* horizontal index value */
    Ipuint           uVertical,                                       /* vertical index value */
    Ipuint           uDepth,                                           /* depth index value */
    Ipuint           uTemporal,                                       /* temporal index value */
    Ipuint           uBand,                                           /* band index value */
);

```

```

/*                                     FUNCTION PROTOTYPES (continued)                                     */

extern Idnvreg InPutPixelROIvReg (                                     /* OUT destination ROI identifier */
    Idnvreg          nVReg,                                     /* output virtual register identifier or NULL */
    Ipsroi_vreg      snDestROI,                               /* destination ROI identifier */
    Ipsbool_vreg_chc sPixel,                                  /* pixel value */
    Ipsuint_vreg_chc sHorizontal,                             /* horizontal index value */
    Ipsuint_vreg_chc sVertical,                               /* vertical index value */
    Ipsuint_vreg_chc sDepth,                                  /* depth index value */
    Ipsuint_vreg_chc sTemporal,                               /* temporal index value */
    Ipsuint_vreg_chc sBand                                    /* band index value */
);

extern Idnimage InPutPixelArray (                                     /* OUT destination image identifier */
    Idnimage          nDestImage,                             /* destination image identifier */
    Ipsparameter_pixel stPixels,                               /* pixel values array pointer union */
    Ipuint            uOffset1,                               /* first index offset */
    Ipuint            uOffset2,                               /* second index offset */
    Ipuint            uValue3,                                /* third index value */
    Ipuint            uValue4,                                /* fourth index value */
    Ipuint            uValue5,                                /* fifth index value */
    Ipuint            uColumns,                               /* number of array columns */
    Ipuint            uRows                                  /* number of array rows */
);

extern Idnvreg InPutPixelArrayVReg (                               /* OUT destination image identifier */
    Idnvreg          nVReg,                                     /* output virtual register identifier or NULL */
    Ipsimage_vreg     snDestImage,                             /* destination image identifier */
    Ipsparameter_pixel_vreg_chc sPixels,                       /* pixel values array */
    Ipsuint_vreg_chc  sOffset1,                               /* first index offset */
    Ipsuint_vreg_chc  sOffset2,                               /* second index offset */
    Ipsuint_vreg_chc  sValue3,                                /* third index value */
    Ipsuint_vreg_chc  sValue4,                                /* fourth index value */
    Ipsuint_vreg_chc  sValue5,                                /* fifth index value */
    Ipsuint_vreg_chc  sColumns,                               /* number of array columns */
    Ipsuint_vreg_chc  sRows                                   /* number of array rows */
);

extern Idnroi InPutPixelArrayROI (                                 /* OUT destination ROI identifier */
    Idnroi           nDestROI,                                 /* destination ROI identifier */
    Ipsbool          *tbaPixels,                              /* pixel values array pointer */
    Ipuint           uOffset1,                                 /* first index offset */
    Ipuint           uOffset2,                                 /* second index offset */
    Ipuint           uValue3,                                  /* third index value */
    Ipuint           uValue4,                                  /* fourth index value */
    Ipuint           uValue5,                                  /* fifth index value */
    Ipuint           uColumns,                                 /* number of array columns */
    Ipuint           uRows                                    /* number of array rows */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InPutPixelArrayROIvReg ( /* OUT destination ROI identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsroi_vreg      snDestROI, /* destination ROI identifier */
    Ipsabool_vreg_chc sPixels, /* pixel values array */
    Ipsuint_vreg_chc sOffset1, /* first index offset */
    Ipsuint_vreg_chc sOffset2, /* second index offset */
    Ipsuint_vreg_chc sValue3, /* third index value */
    Ipsuint_vreg_chc sValue4, /* fourth index value */
    Ipsuint_vreg_chc sValue5, /* fifth index value */
    Ipsuint_vreg_chc sColumns, /* number of array columns */
    Ipsuint_vreg_chc sRows, /* number of array rows */
);

extern Idnimage InPutPixelRecord ( /* OUT destination image identifier */
    Idnimage          nDestImage, /* destination image identifier */
    Ipsparameter_pixel stPixels, /* pixel values array pointer union */
    Idntuple          nType, /* data type SP B-tuple */
    Ipoint            uHorizontal, /* horizontal index value */
    Ipoint            uVertical, /* vertical index value */
    Ipoint            uDepth, /* depth index value */
    Ipoint            uTemporal /* temporal index value */
);

extern Idnvreg InPutPixelRecordVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg      snDestImage, /* destination image identifier */
    Ipsparameter_pixel_vreg_chc sPixels, /* pixel values array */
    Ipstuple_vreg      snType, /* data type SD B-tuple */
    Ipsuint_vreg_chc sHorizontal, /* horizontal index value */
    Ipsuint_vreg_chc sVertical, /* vertical index value */
    Ipsuint_vreg_chc sDepth, /* depth index value */
    Ipsuint_vreg_chc sTemporal /* temporal index value */
);

extern Idnroi InPutPixelRecordROI ( /* OUT destination ROI identifier */
    Idnroi          nDestROI, /* destination ROI identifier */
    Ipbool          *tbaPixels, /* pixel values array pointer union */
    Ipoint          uHorizontal, /* horizontal index value */
    Ipoint          uVertical, /* vertical index value */
    Ipoint          uDepth, /* depth index value */
    Ipoint          uTemporal /* temporal index value */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InPutPixelRecordROIvReg (                                /* OUT destination ROI identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsroi_vreg      snDestROI,                                       /* destination ROI identifier */
    Ipsabool_vreg_chc sPixels,                                         /* pixel values array */
    Ipsuint_vreg_chc sHorizontal,                                       /* horizontal index value */
    Ipsuint_vreg_chc sVertical,                                         /* vertical index value */
    Ipsuint_vreg_chc sDepth,                                           /* depth index value */
    Ipsuint_vreg_chc sTemporal,                                        /* temporal index value */
);

extern Idnimage InRescale (                                           /* OUT destination image identifier */
    Idnimage          nSourceImage,                                     /* source image identifier */
    Idnimage          nDestImage,                                       /* destination image identifier */
    Idntuple          nScale,                                           /* scale factor RD 5-tuple identifier */
    Idntuple          nUpper,                                           /* upper value RD 5-tuple identifier */
    Idntuple          nLower,                                           /* lower value RD 5-tuple identifier */
);

extern Idnvreg InRescaleVReg (                                       /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg     snSourceImage,                                     /* source image identifier */
    Ipsimage_vreg     snDestImage,                                       /* destination image identifier */
    Ipstuple_vreg     snScale,                                           /* scale factor RD 5-tuple identifier */
    Ipstuple_vreg     snUpper,                                           /* upper value RD 5-tuple identifier */
    Ipstuple_vreg     snLower,                                           /* lower value RD 5-tuple identifier */
);

extern Idnroi InRescaleROI (                                         /* OUT destination ROI identifier */
    Idnroi           nSourceROI,                                       /* source ROI identifier */
    Idnroi           nDestROI,                                       /* destination ROI identifier */
    Idntuple         nScale,                                           /* scale factor RD 5-tuple identifier */
    Idntuple         nUpper,                                           /* upper value RD 5-tuple identifier */
    Idntuple         nLower,                                           /* lower value RD 5-tuple identifier */
);

extern Idnvreg InRescaleROIvReg (                                    /* OUT destination ROI identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsroi_vreg      snSourceROI,                                       /* source ROI identifier */
    Ipsroi_vreg      snDestROI,                                       /* destination ROI identifier */
    Ipstuple_vreg     snScale,                                           /* scale factor RD 5-tuple identifier */
    Ipstuple_vreg     snUpper,                                           /* upper value RD 5-tuple identifier */
    Ipstuple_vreg     snLower,                                           /* lower value RD 5-tuple identifier */
);

extern Idnimage InResize (                                           /* OUT destination image identifier */
    Idnimage          nSourceImage,                                     /* source image identifier */
    Idnimage          nDestImage,                                       /* destination image identifier */
    Idntuple          nDestSize,                                        /* destination size ND 5-tuple identifier */
);

```



```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InResizeVReg (                                /* OUT destination image identifier */
    Idnvreg          nVReg,                                  /* output virtual register identifier or NULL */
    Ipsimage_vreg     snSourceImage,                         /* source image identifier */
    Ipsimage_vreg     snDestImage,                           /* destination image identifier */
    Ipstuple_vreg     snDestSize                             /* destination size ND 5-tuple identifier */
);

extern Idnroi InResizeROI (                                  /* OUT destination ROI identifier */
    Idnroi            nSourceROI,                             /* source ROI identifier */
    Idnroi            nDestROI,                               /* destination ROI identifier */
    Idntuple          nDestSize                              /* destination size ND 5-tuple identifier */
);

extern Idnvreg InResizeROIvReg (                             /* OUT destination ROI identifier */
    Idnvreg          nVReg,                                  /* output virtual register identifier or NULL */
    Ipsroi_vreg      snSourceROI,                            /* source ROI identifier */
    Ipsroi_vreg      snDestROI,                              /* destination ROI identifier */
    Ipstuple_vreg     snDestSize                             /* destination size ND 5-tuple identifier */
);

extern Idnrepository InReturnRepositoryId (                  /* OUT repository identifier */
    Ipint            iDescriptor,                             /* repository descriptor within class */
    Ipint            iClassOption                             /* repository class option */
);

extern Idnvreg InReturnRepositoryIdVReg (                    /* OUT repository identifier */
    Idnvreg          nVReg,                                  /* output virtual register identifier */
    Ipsint_vreg_chc  sIndex,                                 /* repository descriptor within class */
    Ipsint_vreg_chc  sClassOption                             /* repository class option */
);

extern Idnroi InROICoordinate (                              /* OUT destination ROI identifier */
    Ipuint           *tuaArray,                               /* 5-dimensional coordinate array pointer */
    Idnroi           nDestROI,                                /* destination ROI identifier */
    Idntuple         nSize,                                   /* ROI size NP 5-tuple */
    Ipuint           uEntries,                                /* number of array entries */
    Ipepolarity      ePolarityOption                         /* ROI polarity option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InROICoordinateVReg (                                     /* OUT destination ROI identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsauint_vreg_chc sArray,                                         /* 5-dimensional coordinate array */
    Ipsroi_vreg       snDestROI,                                       /* destination ROI identifier */
    Ipstuple_vreg     snSize,                                           /* ROI size NP 5-tuple */
    Ipsuint_vreg_chc  sEntries,                                        /* number of array entries */
    Ipspolarity_vreg_chc sPolarityOption /* ROI polarity option */
);

extern Idnroi InROIElliptical (                                       /* OUT destination ROI identifier */
    Idnroi          nDestROI,                                           /* destination ROI identifier */
    Idntuple        nSize,                                               /* ROI size ND 5-tuple */
    Idntuple        nCentre,                                             /* ellipse centre RD 5-tuple */
    Idntuple        nLength,                                             /* ellipse axis length RD 5-tuple */
    Idntuple        nIndexManip,                                         /* index manipulation CS 5-tuple identifier */
    Ipedimension    eDimensionOption, /* ellipse dimension option */
    Ipepolarity     ePolarityOption /* ROI polarity option */
);

extern Idnvreg InROIEllipticalVReg (                                   /* OUT destination ROI identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsroi_vreg       snDestROI,                                       /* destination ROI identifier */
    Ipstuple_vreg     snSize,                                           /* ROI size ND 5-tuple */
    Ipstuple_vreg     snCentre,                                         /* ellipse centre RD 5-tuple */
    Ipstuple_vreg     snLength,                                         /* ellipse axis length RD 5-tuple */
    Ipstuple_vreg     snIndexManip, /* index manipulation CS 5-tuple */
    Ipsdimension_vreg_chc sDimensionOption, /* ellipse dimension option */
    Ipspolarity_vreg_chc sPolarityOption /* ROI polarity option */
);

extern Idnroi InROIPolygon (                                          /* OUT destination ROI identifier */
    Ipuint          *tuaArray,                                           /* 2-dimensional coordinate array pointer */
    Idnroi          nDestROI,                                           /* destination ROI identifier */
    Idntuple        nSize,                                               /* ROI size ND 5-tuple */
    Ipuint          uEntries,                                           /* number of array entries */
    Idntuple        nIndexManip,                                         /* index manipulation CS 5-tuple identifier */
    Ipepolarity     ePolarityOption /* ROI polarity option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InROIPolygonVReg (                                           /* OUT destination ROI identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsaint_vreg_chc sArray,        /* 2-dimensional coordinate array */
    Ipsroi_vreg      snDestROI,     /* destination ROI identifier */
    Ipstuple_vreg    snSize,        /* ROI size ND 5-tuple */
    Ipsuint_vreg_chc sEntries,      /* number of array entries */
    Ipstuple_vreg    snIndexManip,  /* index manipulation CS 5-tuple identifier */
    Ipspolarity_vreg_chc sPolarityOption /* ROI polarity option */
);

extern Idnroi InROIRectangular (                                           /* OUT destination ROI identifier */
    Idnroi          nDestROI,      /* destination ROI identifier */
    Idntuple        nSize,         /* ROI size ND 5-tuple */
    Idntuple        nStart,        /* start position ND 5-tuple */
    Idntuple        nEnd,         /* end position ND 5-tuple */
    Idntuple        nIndexManip,   /* index manipulation CS 5-tuple */
    Ipedimension    eDimensionOption, /* rectangle dimension option */
    Ipepolarity     ePolarityOption /* ROI polarity option */
);

extern Idnvreg InROIRectangularVReg (                                     /* OUT destination ROI identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsroi_vreg      snDestROI,     /* destination ROI identifier */
    Ipstuple_vreg    snSize,        /* ROI size 5-tuple */
    Ipstuple_vreg    snStart,       /* start position 5-tuple */
    Ipstuple_vreg    snEnd,        /* end position 5-tuple */
    Ipstuple_vreg    snIndexManip,  /* index manipulation CS 5-tuple */
    Ipsdimension_vreg_chc sDimensionOption, /* rectangle dimension option */
    Ipspolarity_vreg_chc sPolarityOption /* ROI polarity option */
);

extern Idnimage InRotate (                                                 /* OUT destination image identifier */
    Idnimage         nSourceImage,  /* source image identifier */
    Idnimage         nDestImage,    /* destination image identifier */
    Ipfloat          rCentre1,      /* first index source rotation centre */
    Ipfloat          rCentre2,      /* second index source rotation centre */
    Ipfloat          rCentre3,      /* third index source rotation centre */
    Ipfloat          rAngleJK,     /* j-k plane rotation angle */
    Ipfloat          rAngleJL,     /* j-l plane rotation angle */
    Ipfloat          rAngleKL,     /* k-l plane rotation angle */
    Ipint            iOption        /* rotation option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InRotateVReg (
    Idnvreg          nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier */
    Ipsfloat_vreg_chc sCentre1,      /* destination image identifier */
    Ipsfloat_vreg_chc sCentre2,      /* first index source rotation centre */
    Ipsfloat_vreg_chc sCentre3,      /* second index source rotation centre */
    Ipsfloat_vreg_chc sAngleJK,      /* third index source rotation centre */
    Ipsfloat_vreg_chc sAngleJL,      /* j-k plane rotation angle */
    Ipsfloat_vreg_chc sAngleKL,      /* j-l plane rotation angle */
    Ipsint_vreg_chc  sOption         /* k-l plane rotation angle */
    /* rotation option */
);

extern Idnroi InRotateROI (
    Idnroi          nSourceROI,      /* OUT destination ROI identifier */
    Idnroi          nDestROI,        /* source ROI identifier */
    Ipfloat         rCentre1,         /* destination ROI identifier */
    Ipfloat         rCentre2,         /* first index source rotation centre */
    Ipfloat         rCentre3,         /* second index source rotation centre */
    Ipfloat         rAngleJK,         /* third index source rotation centre */
    Ipfloat         rAngleJL,         /* j-k plane rotation angle */
    Ipfloat         rAngleKL,         /* j-l plane rotation angle */
    Ipint           iOption           /* k-l plane rotation angle */
    /* rotation option */
);

extern Idnvreg InRotateROIvReg (
    Idnvreg          nVReg,          /* OUT destination ROI identifier */
    Ipsroi_vreg      snSourceROI,    /* output virtual register identifier or NULL */
    Ipsroi_vreg      snDestROI,      /* source ROI identifier */
    Ipsfloat_vreg_chc sCentre1,      /* destination ROI identifier */
    Ipsfloat_vreg_chc sCentre2,      /* first index source rotation centre */
    Ipsfloat_vreg_chc sCentre3,      /* second index source rotation centre */
    Ipsfloat_vreg_chc sAngleJK,      /* third index source rotation centre */
    Ipsfloat_vreg_chc sAngleJL,      /* j-k plane rotation angle */
    Ipsfloat_vreg_chc sAngleKL,      /* j-l plane rotation angle */
    Ipsint_vreg_chc  sOption         /* k-l plane rotation angle */
    /* rotation option */
);

extern Idnimage InRubberBandScaling (
    Idnimage          nSourceImage,  /* OUT destination image identifier */
    Idnimage          nDestImage,    /* source image identifier */
    Ipsparameter_arith stLowClip,     /* destination image identifier */
    Ipsparameter_arith stLowOutput,  /* low input clip value pointer union */
    Ipsparameter_arith stHighClip,   /* low output value pointer union */
    Ipsparameter_arith stHighOutput, /* high input clip value pointer union */
    Ipsparameter_arith stInflectionInput, /* high output value pointer union */
    Ipsparameter_arith stInflectionOutput /* inflection input value pointer union */
    /* inflection output value pointer union */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InRubberBandScalingVReg (                                /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipsparameter_arith_vreg_chc    sLowClip, /* low input clip value */
    Ipsparameter_arith_vreg_chc    sLowOutput, /* low output value */
    Ipsparameter_arith_vreg_chc    sHighClip, /* high input clip value */
    Ipsparameter_arith_vreg_chc    sHighOutput, /* high output value */
    Ipsparameter_arith_vreg_chc    sInflectionIn, /* inflection input value */
    Ipsparameter_arith_vreg_chc    sInflectionOut /* inflection output value */
);

extern Idnimage InSeqAverage (                                /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Ipint             iOption       /* dimension option */
);

extern Idnvreg InSeqAverageVReg (                                /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipsint_vreg_chc  sOption       /* dimension option */
);

extern Idnimage InSeqKarhunenLoeve (                                /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Idnmatrix         nMatrix,      /* sequence covariance matrix identifier */
    Ipuint            uIndex4,      /* fourth index value */
    Ipuint            uIndex5,      /* fifth index value */
    Ipetransform      eOption       /* transform direction option */
);

extern Idnvreg InSeqKarhunenLoeveVReg (                                /* OUT destination image identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipsmatrix_vreg    snMatrix,     /* sequence covariance matrix identifier */
    Ipsuint_vreg_chc  sIndex4,      /* fourth index value */
    Ipsuint_vreg_chc  sIndex5,      /* fifth index value */
    Ipstransform_vreg_chc    sOption /* transform direction option */
);

extern Idnimage InSeqRunningMeasures (                                /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Ipint             iDimensionOption, /* dimension option */
    Ipint             iMeasureOption  /* sequence running measure option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InSeqRunningMeasuresVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage, /* destination image identifier */
    Ipsint_vreg_chc  sDimensionOption, /* dimension option */
    Ipsint_vreg_chc  sMeasureOption /* sequence running measure option */
);

extern Iptferror_function ItfSetErrorHandler ( /* OUT old error handling function pointer */
    Iptferror_function tfNewErrorHandler /* new error handler function pointer */
);

extern void IvSetGlobals ( /* no return value output */
    Ipspiks_modes      sModes /* mode settings */
);

extern void IvSetGlobalsVReg ( /* no return value output */
    Ipspiks_modes_vreg_chc sModes /* mode settings */
);

extern Idnimage InSetImageAttrs ( /* OUT destination image identifier */
    Idnimage          nDestImage, /* destination image identifier */
    Ipsattrs_image    sAttrs /* image attributes */
);

extern Idnvreg InSetImageAttrsVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage, /* destination image identifier */
    Ipsattrs_image_vreg_chc sAttrs /* image attributes */
);

extern void IvSetIndexAssignment ( /* no return value output */
    Ipeindex          eJAssign, /* j index assignment selection */
    Ipeindex          eKAssign, /* k index assignment selection */
    Ipeindex          eLAssign, /* l index assignment selection */
    Ipeindex          eMAssign, /* m index assignment selection */
    Ipeindex          eNAssign, /* n index assignment selection */
);

extern void IvSetIndexAssignmentVReg ( /* no return value output */
    Ipsindex_vreg_chc sJAssign, /* j index assignment selection */
    Ipsindex_vreg_chc sKAssign, /* k index assignment selection */
    Ipsindex_vreg_chc sLAssign, /* l index assignment selection */
    Ipsindex_vreg_chc sMAssign, /* m index assignment selection */
    Ipsindex_vreg_chc sNAssign, /* n index assignment selection */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Ipfloa *ItraShapeMetrics ( /* OUT shape metric array pointer */
    Idnimage      nSourceImage, /* source image identifier */
    Ipfloa        *traShape,    /* shape metric array pointer */
    Ipoint        uIndex3,      /* third index value */
    Ipoint        uIndex4,      /* fourth index value */
    Ipoint        uIndex5,      /* fifth index value */
    Ipoint        uLimit,       /* object number limit */
    Ipoint        iApplBufferSize, /* data transfer buffer size or option */
    Ipoint        uStartPoint,   /* index offset within internal data */
    Ipoint        *tuDataSize    /* OUT post transfer data size pointer */
);

extern Ipfloa *ItraSpatialMomentsInvariant ( /* OUT spatial moments array pointer */
    Idnimage      nSourceImage, /* source image identifier */
    Ipfloa        *traMoments,   /* spatial moments array pointer */
    Ipoint        uIndex3,      /* third index value */
    Ipoint        uIndex4,      /* fourth index value */
    Ipoint        uIndex5       /* fifth index value */
);

extern Ipfloa *ItraSpatialMomentsScaled ( /* OUT spatial moments array pointer */
    Idnimage      nSourceImage, /* source image identifier */
    Ipfloa        *traMoments,   /* spatial moments array pointer */
    Ipoint        uIndex3,      /* third index value */
    Ipoint        uIndex4,      /* fourth index value */
    Ipoint        uIndex5       /* fifth index value */
);

extern Idnimage InSplitImage ( /* OUT destination image identifier */
    Idnimage      nSourceImage1, /* first source image identifier */
    Idnimage      nSourceImage2, /* second source image identifier */
    Idnimage      nDestImage,    /* destination image identifier */
    Ipoint        uSplitPosition, /* split position */
    Ipoint        iOption        /* split image option */
);

extern Idnimage InSplitImageVReg ( /* OUT destination image identifier */
    Idnimage      nVReg,         /* output virtual register identifier or NULL */
    Idnimage      snSourceImage1, /* first source image identifier */
    Idnimage      snSourceImage2, /* second source image identifier */
    Idnimage      snDestImage,    /* destination image identifier */
    Ipoint        sSplitPosition, /* split position */
    Ipoint        sOption        /* split image option */
);

```



```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnimage InSubsample (
    Idnimage      nSourceImage, /* OUT destination image identifier */
    Idnimage      nDestImage,  /* source image identifier          */
    Ipuint        uFactor1,    /* destination image identifier     */
    Ipuint        uFactor2,    /* source first index subsampling factor */
    Ipuint        uSourceOffset1, /* source second index subsampling factor */
    Ipuint        uSourceOffset2, /* source first index offset        */
    Ipuint        uDestOffset1, /* source second index offset       */
    Ipuint        uDestOffset2, /* destination first index offset   */
    Ipuint        uWindow1,    /* destination second index offset  */
    Ipuint        uWindow2,    /* source first index window width  */
    Ipuint        /* source second index window height */
);

extern Idnvreg InSubsampleVReg (
    Idnvreg      nVReg, /* OUT destination image identifier */
    Ipsimage_vreg snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg snDestImage,  /* source image identifier          */
    Ipsuint_vreg_chc sFactor1, /* destination image identifier     */
    Ipsuint_vreg_chc sFactor2, /* source first index subsampling factor */
    Ipsuint_vreg_chc sSourceOffset1, /* source second index subsampling factor */
    Ipsuint_vreg_chc sSourceOffset2, /* source first index offset        */
    Ipsuint_vreg_chc sSourceOffset2, /* source second index offset       */
    Ipsuint_vreg_chc sDestOffset1, /* destination first index offset   */
    Ipsuint_vreg_chc sDestOffset2, /* destination second index offset  */
    Ipsuint_vreg_chc sWindow1,    /* source first index window width  */
    Ipsuint_vreg_chc sWindow2,    /* source second index window height */
);

extern Idnroi InSubsampleROI (
    Idnroi      nSourceROI, /* OUT destination ROI identifier */
    Idnroi      nDestROI,  /* source ROI identifier          */
    Ipuint      uFactor1,   /* destination ROI identifier     */
    Ipuint      uFactor2,   /* source first index subsampling factor */
    Ipuint      uSourceOffset1, /* source second index subsampling factor */
    Ipuint      uSourceOffset2, /* source first index offset        */
    Ipuint      uSourceOffset2, /* source second index offset       */
    Ipuint      uDestOffset1, /* destination first index offset   */
    Ipuint      uDestOffset2, /* destination second index offset  */
    Ipuint      uWindow1,   /* source first index window width  */
    Ipuint      uWindow2,   /* source second index window height */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InSubsampleROIvReg (                                     /* OUT destination ROI identifier */
    Idnvreg          nVReg,                                           /* output virtual register identifier or NULL */
    Ipsroi_vreg      snSourceROI,                                     /* source ROI identifier */
    Ipsroi_vreg      snDestROI,                                       /* destination ROI identifier */
    Ipsuint_vreg_chc sFactor1,                                         /* source first index subsampling factor */
    Ipsuint_vreg_chc sFactor2,                                         /* source second index subsampling factor */
    Ipsuint_vreg_chc sSourceOffset1,                                   /* source first index offset */
    Ipsuint_vreg_chc sSourceOffset2,                                   /* source second index offset */
    Ipsuint_vreg_chc sDestOffset1,                                    /* destination first index offset */
    Ipsuint_vreg_chc sDestOffset2,                                    /* destination second index offset */
    Ipsuint_vreg_chc sWindow1,                                        /* source first index window width */
    Ipsuint_vreg_chc sWindow2                                         /* source second index window height */
);

extern void IvSynch (                                                 /* no return value output */
    void                                                    /* no input arguments */
);

extern Idnstatic InTemplMatch (                                       /* OUT cross-correlation array identifier */
    Idnimage          nSourceImage1,                                  /* first source image identifier */
    Idnimage          nSourceImage2,                                  /* second source image identifier */
    Idnstatic         nCrossCorrelation,                             /* cross-correlation array identifier */
    Ipuint            uIndex3,                                       /* third index value */
    Ipuint            uIndex4,                                       /* fourth index value */
    Ipuint            uIndex5,                                       /* fifth index value */
    Ipuint            uFirstOffsetImage1,                             /* first image first index offset */
    Ipuint            u2ndOffsetImage1,                               /* first image second index offset */
    Ipuint            uFirstOffsetImage2,                             /* second image first index offset */
    Ipuint            u2ndOffsetImage2,                               /* second image second index offset */
    Ipuint            uWindowColumns,                                 /* number of columns of search window */
    Ipuint            uWindowRows,                                   /* number of rows of search window */
    Ipuint            uTemplColumns,                                 /* number of columns of template */
    Ipuint            uTemplRows                                    /* number of rows of template */
);

```

```

/*                                     FUNCTION PROTOTYPES (continued)                                     */

extern Idnvreg InTemplMatchVReg (                                           /* OUT cross-correlation array identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage1, /* first source image identifier */
    Ipsimage_vreg    snSourceImage2, /* second source image identifier */
    Ipsstatic_vreg   snCrossCorrelation, /* cross-correlation array identifier */
    Ipsuint_vreg_chc sIndex3,        /* third index value */
    Ipsuint_vreg_chc sIndex4,        /* fourth index value */
    Ipsuint_vreg_chc sIndex5,        /* fifth index value */
    Ipsuint_vreg_chc sFirstOffsetImage1, /* first image first index offset */
    Ipsuint_vreg_chc s2ndOffsetImage1,  /* first image second index offset */
    Ipsuint_vreg_chc sFirstOffsetImage2, /* second image first index offset */
    Ipsuint_vreg_chc s2ndOffsetImage2,  /* second image second index offset */
    Ipsuint_vreg_chc sWindowColumns,    /* number of columns of search window */
    Ipsuint_vreg_chc sWindowRows,       /* number of rows of search window */
    Ipsuint_vreg_chc sTemplColumns,     /* number of columns of template */
    Ipsuint_vreg_chc sTemplRows        /* number of rows of template */
);

extern Idnimage InThreshold (                                               /* OUT destination image identifier */
    Idnimage      nSourceImage,      /* source image identifier */
    Idnimage      nDestImage,        /* destination image identifier */
    Ipsparameter_pixel stAbove,      /* above threshold clip value pointer union */
    Ipsparameter_pixel stBelow,      /* below threshold clip value pointer union */
    Ipsparameter_arith stThreshold   /* threshold value pointer union */
);

extern Idnvreg InThresholdVReg (                                           /* OUT destination image identifier */
    Idnvreg          nVReg,          /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,   /* destination image identifier */
    Ipsparameter_pixel_vreg_chc sAbove, /* above threshold clip value */
    Ipsparameter_pixel_vreg_chc sBelow, /* below threshold clip value */
    Ipsparameter_arith_vreg_chc sThreshold /* threshold value */
);

extern Idnimage InTransformCosine (                                       /* OUT destination image identifier */
    Idnimage      nSourceImage,      /* source image identifier */
    Idnimage      nDestImage,        /* destination image identifier */
    Ipuint        uHorizontalSize,   /* horizontal block size */
    Ipuint        uVerticalSize,     /* vertical block size */
    Ipetransform  eOption            /* transform direction option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InTransformCosineVReg (                                     /* OUT destination image identifier */
    Idnvreg          nVReg,                                             /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipsuint_vreg_chc sHorizontalSize, /* horizontal block size */
    Ipsuint_vreg_chc sVerticalSize,   /* vertical block size */
    Ipstransform_vreg_chc sOption      /* transform direction option */
);

extern Idnimage InTransformFourier (                                     /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Ipedimension      eOrderOption, /* order option */
    Ipetransform      eDirectionOption, /* transform direction option */
    Ipefrequency_shift eShiftOption  /* frequency shift option */
);

extern Idnvreg InTransformFourierVReg (                               /* OUT destination image identifier */
    Idnvreg          nVReg,                                             /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipsdimension_vreg_chc sOrderOption, /* order option */
    Ipstransform_vreg_chc sDirectionOption, /* transform direction option */
    Ipsfrequency_shift_vreg_chc sShiftOption /* frequency shift option */
);

extern Idnimage InTransformHadamard (                                  /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Ipuint            uBlockSize    /* block size */
);

extern Idnvreg InTransformHadamardVReg (                               /* OUT destination image identifier */
    Idnvreg          nVReg,                                             /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipsuint_vreg_chc sBlockSize    /* block size */
);

extern Idnimage InTransformHartley (                                   /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Ipedimension      eOption       /* order option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InTransformHartleyVReg (                                     /* OUT destination image identifier */
    Idnvreg          nVReg,                                             /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipsdimension_vreg_chc sOption /* order option */
);

extern Idnimage InTranslate (                                           /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Idntuple          nOffset       /* index offset RD 5-tuple identifier */
);

extern Idnvreg InTranslateVReg (                                       /* OUT destination image identifier */
    Idnvreg          nVReg,                                             /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipstuple_vreg    snOffset       /* index offset 5-tuple identifier */
);

extern Idnroi InTranslateROI (                                         /* OUT destination ROI identifier */
    Idnroi           nSourceROI,   /* source ROI identifier */
    Idnroi           nDestROI,    /* destination ROI identifier */
    Idntuple         nOffset       /* index offset RD 5-tuple identifier */
);

extern Idnvreg InTranslateROIvReg (                                    /* OUT destination ROI identifier */
    Idnvreg          nVReg,                                             /* output virtual register identifier or NULL */
    Ipsroi_vreg      snSourceROI,  /* source ROI identifier */
    Ipsroi_vreg      snDestROI,   /* destination ROI identifier */
    Ipstuple_vreg    snOffset       /* index offset RD 5-tuple identifier */
);

extern Idnimage InUnaryInteger (                                       /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Ipoint            iOption       /* unary integer conversion option */
);

extern Idnvreg InUnaryIntegerVReg (                                    /* OUT destination image identifier */
    Idnvreg          nVReg,                                             /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,  /* destination image identifier */
    Ipsint_vreg_chc  sOption       /* unary integer conversion option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnimage InUnaryReal (                                /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Ipint             iOption       /* unary real conversion option */
);

extern Idnvreg InUnaryRealVReg (                             /* OUT destination image identifier */
    Idnvreg          nVReg,        /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,   /* destination image identifier */
    Ipsint_vreg_chc  sOption       /* unary real conversion option */
);

extern Idnimage InUnsharpMask (                             /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Idnnbhood         nFilter,      /* low pass filter impulse response array identifier */
    Ipfloat           rWeight1,     /* first weight factor */
    Ipfloat           rWeight2,     /* second weight factor */
    Ipfloat           rBias        /* bias factor */
);

extern Idnvreg InUnsharpMaskVReg (                          /* OUT destination image identifier */
    Idnvreg          nVReg,        /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsimage_vreg    snDestImage,   /* destination image identifier */
    Ipsnbhood_vreg   snFilter,      /* low pass filter impulse response array identifier */
    Ipsfloat_vreg_chc sWeight1,     /* first weight factor */
    Ipsfloat_vreg_chc sWeight2,     /* second weight factor */
    Ipsfloat_vreg_chc sBias        /* bias factor */
);

extern Idnbounds InValueBounds (                             /* OUT value bounds collection identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnbounds         nValueBounds, /* value bounds collection identifier */
    Ipsparameter_arith stLowerBound, /* lower amplitude bound pointer union */
    Ipsparameter_arith stUpperBound /* upper amplitude bound pointer union */
);

extern Idnvreg InValueBoundsVReg (                          /* OUT value bounds collection identifier */
    Idnvreg          nVReg,        /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage, /* source image identifier */
    Ipsbounds_vreg    snValueBounds, /* value bounds collection identifier */
    Ipsparameter_arith_vreg_chc sLowerBound, /* lower amplitude bound */
    Ipsparameter_arith_vreg_chc sUpperBound /* upper amplitude bound */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InVRegAlter (                                /* OUT virtual register identifier */
    Idnvreg          nVReg,                                /* virtual register identifier      */
    Ipsparameter_pixel stDelta,                            /* value to add to virtual register pointer union */
    Ipbool           bMode                                /* immediate mode flag             */
);

extern Idnvreg InVRegAlterVReg (                            /* OUT virtual register identifier */
    Idnvreg          nVReg,                                /* virtual register identifier      */
    Ipsparameter_pixel_vreg_chc sDelta,                    /* value to add to virtual register */
    Ipsbool_vreg_chc sMode                                /* immediate mode flag             */
);

extern Idnvreg InVRegClear (                                /* OUT virtual register identifier */
    Idnvreg          nVReg,                                /* virtual register identifier      */
    Ipbool           bMode                                /* immediate mode flag             */
);

extern Idnvreg InVRegClearVReg (                            /* OUT virtual register identifier */
    Idnvreg          nVReg,                                /* virtual register identifier      */
    Ipsbool_vreg_chc sMode                                /* immediate mode flag             */
);

extern void IvVRegConditional (                             /* no return value output          */
    Idnvreg          nVReg,                                /* virtual register identifier      */
    Ipsparameter_basic stLower,                            /* lower test value pointer union   */
    Ipsparameter_basic stUpper,                            /* upper test value pointer union   */
    Ipbool           bFlag,                                /* logical flag variable for conditional control */
    Idnchain         nChain,                                /* chain identifier for branch execution */
    Ipbool           bMode,                                /* immediate mode flag             */
    Ipint            iReturnOption                          /* virtual register conditional return option */
);

extern void IvVRegConditionalVReg (                         /* no return value output          */
    Idnvreg          nVReg,                                /* virtual register identifier      */
    Ipsparameter_basic_vreg_chc sLower,                    /* lower value to test register against */
    Ipsparameter_basic_vreg_chc sUpper,                    /* upper value to test register against */
    Ipsbool_vreg_chc sFlag,                                /* logical flag variable for conditional control */
    Ipschain_vreg    snChain,                                /* chain to branch for execution    */
    Ipsbool_vreg_chc sMode,                                /* immediate mode flag             */
    Ipsint_vreg_chc  sReturnOption                          /* virtual register conditional return option */
);

extern Idnvreg InVRegCopy (                                /* OUT destination virtual register identifier */
    Idnvreg          nSourceVReg,                          /* source virtual register identifier */
    Idnvreg          nDestVReg,                             /* destination virtual register identifier */
    Ipbool           bMode                                /* immediate mode flag             */
);

```



```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InVRegCopyVReg ( /* OUT destination virtual register identifier */
    Idnvreg          nSourceVReg, /* source virtual register identifier */
    Idnvreg          nDestVReg,  /* destination virtual register identifier */
    Ipsbool_vreg_chc sMode      /* immediate mode flag */
);

extern Idnvreg InVRegDelete ( /* OUT virtual register identifier */
    Idnvreg          nVReg, /* virtual register identifier */
    Ipsbool          bMode  /* immediate mode flag */
);

extern Idnvreg InVRegDeleteVReg ( /* OUT virtual register identifier */
    Idnvreg          nVReg, /* virtual register identifier */
    Ipsbool_vreg_chc sMode  /* immediate mode flag */
);

extern void IvVRegGet ( /* no return value output */
    Idnvreg          nVReg, /* virtual register identifier */
    Ipsbool          bMode, /* immediate mode flag */
    Ipsparameter_vreg sValue, /* OUT value of virtual register pointer */
    Ipevreg_state    *teState, /* OUT virtual register state pointer */
    Ipsint           *tiType, /* OUT virtual register data type pointer */
);

extern Idnvreg InVRegSet ( /* OUT virtual register identifier */
    Idnvreg          nVReg, /* virtual register identifier */
    Ipsparameter_vreg sValue, /* value to be loaded */
    Ipsbool          bMode, /* immediate mode flag */
    Ipsint           iType  /* virtual register data type option */
);

extern Idnvreg InVRegSetVReg ( /* OUT virtual register identifier */
    Idnvreg          nVReg, /* virtual register identifier */
    Ipsparameter_vreg_vreg_chc sValue, /* value to be loaded */
    Ipsbool_vreg_chc sMode, /* immediate mode flag */
    Ipsint_vreg_chc sType  /* virtual register data type option */
);

extern Idnvreg InVRegWait ( /* OUT virtual register identifier */
    Idnvreg          nVReg, /* virtual register identifier */
    Ipsparameter_vreg sValue, /* value to wait for */
    Ipsbool          bMode  /* immediate mode flag */
);

extern Idnvreg InVRegWaitVReg ( /* OUT virtual register identifier */
    Idnvreg          nVReg, /* virtual register identifier */
    Ipsparameter_vreg_vreg_chc sValue, /* value to wait for */
    Ipsbool_vreg_chc sMode  /* immediate mode flag */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnimage InWallisStatDiff (                                     /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Idnnbhood         nFilter,      /* low pass filter impulse response array identifier */
    Ipfloating         rGain,        /* gain factor */
    Ipfloating         rAverage,     /* average factor */
    Ipfloating         rDeviation,   /* deviation factor */
    Ipfloating         rProportion  /* proportionality factor */
);

extern Idnvreg InWallisStatDiffVReg (                               /* OUT destination image identifier */
    Idnvreg           nVReg,        /* output virtual register identifier or NULL */
    Ipsimage_vreg     snSourceImage, /* source image identifier */
    Ipsimage_vreg     snDestImage,   /* destination image identifier */
    Ipsnbhood_vreg    snFilter,      /* low pass filter impulse response array identifier */
    Ipsfloat_vreg_chc sGain,         /* gain factor */
    Ipsfloat_vreg_chc sAverage,      /* average factor */
    Ipsfloat_vreg_chc sDeviation,    /* deviation factor */
    Ipsfloat_vreg_chc sProportion    /* proportionality factor */
);

extern Idnimage InWarpControlPoint (                                /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Ipoint            *tuaSourcePoints, /* source image control point array pointer */
    Ipoint            *tuaDestPoints,  /* destination image control point array pointer */
    Ipoint            uPoints,         /* number of control points */
    Ipoint            iApplBufferSize, /* data transfer buffer size or option of each array */
    Ipoint            uStartPoint,     /* index offset within internal data of each array */
    Ipoint            *tuDataSize     /* OUT post transfer data size pointer */
);

extern Idnvreg InWarpControlPointVReg (                             /* OUT destination image identifier */
    Idnvreg           nVReg,        /* output virtual register identifier or NULL */
    Ipsimage_vreg     snSourceImage, /* source image identifier */
    Ipsimage_vreg     snDestImage,   /* destination image identifier */
    Ipsaint_vreg_chc  sSourcePoints, /* source image control point array */
    Ipsaint_vreg_chc  sDestPoints,   /* destination image control point array */
    Ipsuint_vreg_chc  sPoints        /* number of control points */
);

```

```

/*          FUNCTION PROTOTYPES (continued)          */

extern Idnroi InWarpControlPointROI (                /* OUT destination ROI identifier */
    Idnroi          nSourceROI, /* source ROI identifier */
    Idnroi          nDestROI,  /* destination ROI identifier */
    Ipoint          *tuaSourcePoints, /* source image control point array pointer */
    Ipoint          *tuaDestPoints,  /* destination image control point array pointer */
    Ipoint          uPoints,        /* number of control points */
    Ipoint          iApplBufferSize, /* data transfer buffer size or option of each array */
    Ipoint          uStartPoint,    /* index offset within internal data of each array */
    Ipoint          *tuDataSize    /* OUT post transfer data size pointer */
);

extern Idnvreg InWarpControlPointROIvReg (           /* OUT destination ROI identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsroi_vreg      snSourceROI, /* source ROI identifier */
    Ipsroi_vreg      snDestROI,  /* destination ROI identifier */
    Ipsaint_vreg_chc sSourcePoints, /* source image control point array */
    Ipsaint_vreg_chc sDestPoints,  /* destination image control point array */
    Ipsuint_vreg_chc sPoints      /* number of control points */
);

extern Idnimage InWarpLUT (                          /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage,   /* destination image identifier */
    Idnlut            nRowAddress,   /* row address lookup table identifier */
    Idnlut            nColumnAddress, /* column address lookup table identifier */
    Ipoint            uEntries      /* number of lookup table source offset entries */
);

extern Idnvreg InWarpLUTvReg (                      /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg     snSourceImage, /* source image identifier */
    Ipsimage_vreg     snDestImage,   /* destination image identifier */
    Ipslut_vreg       snRowAddress,   /* row address lookup table identifier */
    Ipslut_vreg       snColumnAddress, /* column address lookup table identifier */
    Ipsuint_vreg_chc  sEntries      /* number of lookup table source offset entries */
);

extern Idnroi InWarpLUTROI (                        /* OUT destination ROI identifier */
    Idnroi          nSourceROI, /* source ROI identifier */
    Idnroi          nDestROI,   /* destination ROI identifier */
    Idnlut          nRowAddress, /* row address lookup table identifier */
    Idnlut          nColumnAddress, /* column address lookup table identifier */
    Ipoint          uEntries    /* number of lookup table source offset entries */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InWarpLUTROIVReg ( /* OUT destination ROI identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    Ipsroi_vreg  snSourceROI, /* source ROI identifier */
    Ipsroi_vreg  snDestROI, /* destination ROI identifier */
    Ipslut_vreg  snRowAddress, /* row address lookup table identifier */
    Ipslut_vreg  snColumnAddress, /* column address lookup table identifier */
    Ipsuint_vreg_chc sEntries /* number of lookup table source offset entries */
);

extern Idnimage InWarpPolynomial ( /* OUT destination image identifier */
    Idnimage      nSourceImage, /* source image identifier */
    Idnimage      nDestImage, /* destination image identifier */
    Ipfloat       *traRow, /* row polynomial coefficient array pointer */
    Ipfloat       *traColumn /* column polynomial coefficient array pointer */
);

extern Idnvreg InWarpPolynomialVReg ( /* OUT destination image identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg snSourceImage, /* source image identifier */
    Ipsimage_vreg snDestImage, /* destination image identifier */
    Ipsafloat_vreg_chc sRow, /* row polynomial coefficient array */
    Ipsafloat_vreg_chc sColumn /* column polynomial coefficient array */
);

extern Idnroi InWarpPolynomialROI ( /* OUT destination ROI identifier */
    Idnroi      nSourceROI, /* source ROI identifier */
    Idnroi      nDestROI, /* destination ROI identifier */
    Ipfloat     *traRow, /* row polynomial coefficient array pointer */
    Ipfloat     *traColumn /* column polynomial coefficient array pointer */
);

extern Idnvreg InWarpPolynomialROIVReg ( /* OUT destination ROI identifier */
    Idnvreg      nVReg, /* output virtual register identifier or NULL */
    Ipsroi_vreg  snSourceROI, /* source ROI identifier */
    Ipsroi_vreg  snDestROI, /* destination ROI identifier */
    Ipsafloat_vreg_chc sRow, /* row polynomial coefficient array */
    Ipsafloat_vreg_chc sColumn /* column polynomial coefficient array */
);

extern Idnstatic InWindowingFunction ( /* OUT windowing function array identifier */
    Idnstatic      nFunction, /* windowing function array identifier */
    Idntuple       nSize, /* array size ND 5-tuple */
    Ipuint         uColumnGuard, /* column guard band */
    Ipuint         uRowGuard, /* row guard band */
    Ipint          iOption /* windowing option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnvreg InWindowingFunctionVReg ( /* OUT windowing function array identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsstatic_vreg   snFunction, /* windowing function array identifier */
    Ipstuple_vreg     snSize, /* array size ND 5-tuple */
    Ipsuint_vreg_chc  sColumnGuard, /* column guard band */
    Ipsuint_vreg_chc  sRowGuard, /* row guard band */
    Ipsint_vreg_chc   sOption /* windowing option */
);

extern Idnimage InWindowLevel ( /* OUT destination image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage, /* destination image identifier */
    Ipsparameter_arith stAbove, /* above window clip value pointer union */
    Ipsparameter_arith stBelow, /* below window clip value pointer union */
    Ipsparameter_arith stWidth, /* window width pointer union */
    Ipsparameter_arith stLevel /* window level pointer union */
);

extern Idnvreg InWindowLevelVReg ( /* OUT destination image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg     snSourceImage, /* source image identifier */
    Ipsimage_vreg     snDestImage, /* destination image identifier */
    Ipsparameter_arith_vreg_chc sAbove, /* above window clip value */
    Ipsparameter_arith_vreg_chc sBelow, /* below window clip value */
    Ipsparameter_arith_vreg_chc sWidth, /* window width */
    Ipsparameter_arith_vreg_chc sLevel /* window level */
);

extern Idnimage InWindowStatistics ( /* OUT window statistics image identifier */
    Idnimage          nSourceImage, /* source image identifier */
    Idnimage          nDestImage, /* destination image identifier */
    Ipoint            uIndex3, /* third index value */
    Ipoint            uIndex4, /* fourth index value */
    Ipoint            uIndex5, /* fifth index value */
    Ipoint            uWindowSize, /* window size */
    Ipoint            iOption /* window statistic option */
);

extern Idnvreg InWindowStatisticsVReg ( /* OUT window statistics image identifier */
    Idnvreg          nVReg, /* output virtual register identifier or NULL */
    Ipsimage_vreg     snSourceImage, /* source image identifier */
    Ipsimage_vreg     snDestImage, /* destination image identifier */
    Ipsuint_vreg_chc  sIndex3, /* third index value */
    Ipsuint_vreg_chc  sIndex4, /* fourth index value */
    Ipsuint_vreg_chc  sIndex5, /* fifth index value */
    Ipsuint_vreg_chc  sWindowSize, /* window size */
    Ipsint_vreg_chc   sOption /* window statistic option */
);

```

```

/*                      FUNCTION PROTOTYPES (continued)                      */

extern Idnimage InZMerge (
    Idnimage          nSourceImage1, /* OUT destination image identifier */
    Idnimage          nSourceImage2, /* first source image (data) identifier */
    Idnimage          nSourceImage3, /* second source image (control) identifier */
    Idnimage          nDestImage,   /* third source image (control) identifier */
    Idnimage          /* destination image (data) identifier */
);

extern Idnvreg InZMergeVReg (
    Idnvreg          nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage1, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snSourceImage2, /* first source image (data) identifier */
    Ipsimage_vreg    snSourceImage3, /* second source image (control) identifier */
    Ipsimage_vreg    snDestImage,   /* third source image (control) identifier */
    Ipsimage_vreg    /* destination image (data) identifier */
);

extern Idnimage InZoom (
    Idnimage          nSourceImage, /* OUT destination image identifier */
    Idnimage          nDestImage,   /* source image identifier */
    Idntuple          nScale,       /* destination image identifier */
    /* zoom scale factor ND 5-tuple */
);

extern Idnvreg InZoomVReg (
    Idnvreg          nVReg,          /* OUT destination image identifier */
    Ipsimage_vreg    snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg    snDestImage,   /* source image identifier */
    Ipsimage_vreg    snScale,       /* destination image identifier */
    Ipsimage_vreg    /* zoom scale factor ND 5-tuple */
);

extern Idnroi InZoomROI (
    Idnroi           nSourceROI,    /* OUT destination ROI identifier */
    Idnroi           nDestROI,      /* source ROI identifier */
    Idntuple         nScale,        /* destination ROI identifier */
    /* zoom scale factor ND 5-tuple */
);

extern Idnvreg InZoomROIvReg (
    Idnvreg          nVReg,          /* OUT destination ROI identifier */
    Ipsroi_vreg      snSourceROI,   /* output virtual register identifier or NULL */
    Ipsroi_vreg      snDestROI,     /* source ROI identifier */
    Ipsroi_vreg      snScale,       /* destination ROI identifier */
    Ipsroi_vreg      /* zoom scale factor ND 5-tuple */
);

```

```

/*                      FUNCTION PROTOTYPES (concluded)                      */

extern Idnimage In3DSlice (
    Idnimage      nSourceImage, /* OUT destination image identifier */
    Idnimage      nDestImage,  /* source image identifier          */
    Ipsfloat      rSourceXOffset, /* destination image identifier     */
    Ipsfloat      rSourceYOffset, /* source x coordinate offset      */
    Ipsfloat      rSourceUOffset, /* source y coordinate offset      */
    Ipsfloat      rDestXOffset,  /* source third coordinate offset   */
    Ipsfloat      rDestYOffset,  /* destination x coordinate offset  */
    Ipsfloat      rXYAngle,      /* destination y coordinate offset */
    Ipsfloat      rXUAngle,      /* x-y plane rotation angle       */
    Ipsfloat      rYUAngle,      /* x-u plane rotation angle       */
    Ipsfloat      /* y-u plane rotation angle      */
);

extern Idnvreg In3DSliceVReg (
    Idnvreg      nVReg, /* OUT destination image identifier */
    Ipsimage_vreg snSourceImage, /* output virtual register identifier or NULL */
    Ipsimage_vreg snDestImage,  /* source image identifier          */
    Ipsfloat_vreg_chc sSourceXOffset, /* destination image identifier     */
    Ipsfloat_vreg_chc sSourceYOffset, /* source x coordinate offset      */
    Ipsfloat_vreg_chc sSourceUOffset, /* source y coordinate offset      */
    Ipsfloat_vreg_chc sDestXOffset,  /* source third coordinate offset   */
    Ipsfloat_vreg_chc sDestYOffset,  /* destination x coordinate offset  */
    Ipsfloat_vreg_chc sXYAngle,      /* destination y coordinate offset */
    Ipsfloat_vreg_chc sXUAngle,      /* x-y plane rotation angle       */
    Ipsfloat_vreg_chc sYUAngle,      /* x-u plane rotation angle       */
    Ipsfloat_vreg_chc /* y-u plane rotation angle      */
);

```


B.1.4 Convenience functions

```

/*          CONVENIENCE FUNCTION PROTOTYPES          */

extern Idnimage InPrepareColourImage (                /* OUT 4 or less band image identifier */
    Ipoint      uHorizontal,                          /* image width */
    Ipoint      uVertical,                            /* image height */
    Ipoint      uBands,                              /* image bands */
    Ipoint      iType,                               /* image pixel data type */
    Ipoint      uPrecision,                          /* image pixel precision */
    Ipoint      rXWhite,                             /* X tristimulus value white point */
    Ipoint      rYWhite,                             /* Y tristimulus value white point */
    Ipoint      rZWhite,                             /* Z tristimulus value white point */
    Ipoint      iColourOption                        /* image colour space code */
);

extern Idnimage InPrepareMonochromeImage (            /* OUT monochrome image identifier */
    Ipoint      uHorizontal,                          /* image width */
    Ipoint      uVertical,                            /* image height */
    Ipoint      iType,                               /* image pixel data type */
    Ipoint      uPrecision                          /* image pixel precision */
);

extern Idnimage InCreateUnbndImageCopy (              /* OUT destination image identifier */
    Idnimage    nSourceImage                        /* source image identifier */
);

extern Idnroi InGenerate2DROIrectangular (            /* OUT ROI identifier */
    Ipoint      uHorizontal,                          /* ROI width */
    Ipoint      uVertical,                            /* ROI height */
    Ipoint      uStartHorizontal,                     /* rectangle horizontal start */
    Ipoint      uStartVertical,                       /* rectangle vertical start */
    Ipoint      uEndHorizontal,                       /* rectangle horizontal end */
    Ipoint      uEndVertical,                         /* rectangle vertical end */
    Ipoint      ePolarity                            /* polarity of ROI content */
);

extern Idnroi InGenerateROIcoordinate (               /* OUT ROI identifier */
    Ipoint      *tuaCoordinateArray,                 /* pointer to coordinates array */
    Ipoint      uHorizontal,                          /* ROI width */
    Ipoint      uVertical,                            /* ROI height */
    Ipoint      uDepth,                              /* ROI depth */
    Ipoint      uTemporal,                           /* ROI temporal size */
    Ipoint      uSpectral,                           /* ROI spectral size */
    Ipoint      uCoordinateCount,                    /* number of coordinate sets */
    Ipoint      ePolarity                            /* polarity of ROI content */
);

```

```

/*          CONVENIENCE FUNCTION PROTOTYPES (continued)          */

extern Idnroi InGenerateROIElliptical ( /* OUT ROI identifier */
    Ipoint    uHorizontal, /* ROI width */
    Ipoint    uVertical,   /* ROI height */
    Ipoint    uDepth,      /* ROI depth */
    Ipoint    uTemporal,   /* ROI temporal size */
    Ipoint    uSpectral,    /* ROI spectral size */
    Ifloat    rCentreHorizontal, /* ellipse centre horizontal */
    Ifloat    rCentreVertical,   /* ellipse centre vertical */
    Ifloat    rCentreDepth,      /* ellipse centre depth */
    Ifloat    rCentreTemporal,   /* ellipse centre temporal dimension */
    Ifloat    rCentreSpectral,    /* ellipse centre spectral dimension */
    Ifloat    rLengthHorizontal, /* ellipse length horizontal */
    Ifloat    rLengthVertical,   /* ellipse length vertical */
    Ifloat    rLengthDepth,      /* ellipse length depth */
    Ifloat    rLengthTemporal,   /* ellipse length temporal dimension */
    Ifloat    rLengthSpectral,    /* ellipse length spectral dimension */
    Ipeindex  eJDimension, /* j dimension assignment */
    Ipeindex  eKDimension, /* k dimension assignment */
    Ipeindex  eLDimension, /* l dimension assignment */
    Ipeindex  eMDimension, /* m dimension assignment */
    Ipeindex  eNDimension, /* n dimension assignment */
    Ipedimension eDimension, /* ellipse dimension option */
    Ipepolarity ePolarity /* polarity of ROI content */
);

extern Idnroi InGenerateROIPolygon ( /* OUT ROI identifier */
    Ipoint    *tuaCoordinateArray, /* pointer to 2D coordinate array */
    Ipoint    uHorizontal, /* ROI width */
    Ipoint    uVertical,   /* ROI height */
    Ipoint    uDepth,      /* ROI depth */
    Ipoint    uTemporal,   /* ROI temporal size */
    Ipoint    uSpectral,    /* ROI spectral size */
    Ipeindex  eJDimension, /* j dimension assignment */
    Ipeindex  eKDimension, /* k dimension assignment */
    Ipeindex  eLDimension, /* l dimension assignment */
    Ipeindex  eMDimension, /* m dimension assignment */
    Ipeindex  eNDimension, /* n dimension assignment */
    Ipoint    uCoordinateCount, /* number of coordinates sets */
    Ipepolarity ePolarity /* polarity of ROI content */
);

```

```

/*          CONVENIENCE FUNCTION PROTOTYPES (continued)          */

extern Idnroi InGenerateROIRectangular (      /* OUT ROI identifier */
    Ipoint      uHorizontal,      /* ROI width */
    Ipoint      uVertical,        /* ROI height */
    Ipoint      uDepth,          /* ROI depth */
    Ipoint      uTemporal,        /* ROI temporal size */
    Ipoint      uSpectral,        /* ROI spectral size */
    Ipoint      uStartHorizontal, /* rectangle start horizontal */
    Ipoint      uStartVertical,   /* rectangle start vertical */
    Ipoint      uStartDepth,     /* rectangle start depth */
    Ipoint      uStartTemporal,   /* rectangle start temporal dimension */
    Ipoint      uStartSpectral,   /* rectangle start spectral dimension */
    Ipoint      uEndHorizontal,   /* rectangle end horizontal */
    Ipoint      uEndVertical,     /* rectangle end vertical */
    Ipoint      uEndDepth,       /* rectangle end depth */
    Ipoint      uEndTemporal,     /* rectangle end temporal dimension */
    Ipoint      uEndSpectral,     /* rectangle end spectral dimension */
    Ipeindex    eJDimension,      /* j dimension assignment */
    Ipeindex    eKDimension,      /* k dimension assignment */
    Ipeindex    eLDimension,      /* l dimension assignment */
    Ipeindex    eMDimension,      /* m dimension assignment */
    Ipeindex    eNDimension,      /* n dimension assignment */
    Ipedimension eDimension,      /* ellipse dimension option */
    Ipepolarity  ePolarity        /* polarity of ROI content */
);

extern Idnroi InPrepare2DROIRectangular (      /* OUT ROI identifier */
    Ipoint      uHorizontal,      /* ROI width */
    Ipoint      uVertical,        /* ROI height */
    Ipepolarity  ePolarityOption /* ROI polarity option */
);

extern Idnroi InPrepareROI (      /* OUT ROI identifier */
    Ipoint      uHorizontal,      /* ROI width */
    Ipoint      uVertical,        /* ROI height */
    Ipoint      uDepth,          /* ROI depth */
    Ipoint      uTemporal,        /* ROI temporal dimension */
    Ipoint      uSpectral,        /* ROI spectral dimension */
    Ipoint      iStructureOption, /* ROI structure option */
    Ipepolarity  ePolarityOption /* ROI polarity option */
);

extern Idntuple InGenerateND1Tuple (      /* OUT one parameter ND type tuple identifier */
    Ipoint      uValue1          /* first value, e.g., X */
);

```

```

/*          CONVENIENCE FUNCTION PROTOTYPES (continued)          */

extern Idntuple InGenerateND3Tuple (      /* OUT three parameter ND type tuple identifier */
    Ipoint      uValue1,                  /* first value, e.g., X */
    Ipoint      uValue2,                  /* second value, e.g., Y */
    Ipoint      uValue3,                  /* third value, e.g., Z */
);

extern Idntuple InGenerateND4Tuple (      /* OUT three parameter ND type tuple identifier */
    Ipoint      uValue1,                  /* first value, e.g., Band 1 precision */
    Ipoint      uValue2,                  /* second value, e.g., Band 2 precision */
    Ipoint      uValue3,                  /* third value, e.g., Band 3 precision */
    Ipoint      uValue4,                  /* fourth value, e.g., Band 4 precision */
);

extern Idntuple InGenerateND5Tuple (      /* OUT five parameter ND type tuple identifier */
    Ipoint      uValue1,                  /* first value, e.g., X */
    Ipoint      uValue2,                  /* second value, e.g., Y */
    Ipoint      uValue3,                  /* third value, e.g., Z */
    Ipoint      uValue4,                  /* fourth value, e.g., T */
    Ipoint      uValue5,                  /* fifth value, e.g., B */
);

extern Idntuple InGenerateRD3Tuple (      /* OUT three parameter RD type tuple identifier */
    Ipfloa      rValue1,                  /* first value, e.g., X offset */
    Ipfloa      rValue2,                  /* second value, e.g., Y offset */
    Ipfloa      rValue3,                  /* third value, e.g., Z offset */
);

extern Idntuple InGenerateRD4Tuple (      /* OUT four parameter RD type tuple identifier */
    Ipfloa      rValue1,                  /* first value, e.g., X offset */
    Ipfloa      rValue2,                  /* second value, e.g., Y offset */
    Ipfloa      rValue3,                  /* third value, e.g., Z offset */
    Ipfloa      rValue4,                  /* fourth value, e.g., T offset */
);

extern Idntuple InGenerateRD5Tuple (      /* OUT five parameter RD type tuple identifier */
    Ipfloa      rValue1,                  /* first value, e.g., X offset */
    Ipfloa      rValue2,                  /* second value, e.g., Y offset */
    Ipfloa      rValue3,                  /* third value, e.g., Z offset */
    Ipfloa      rValue4,                  /* fourth value, e.g., T offset */
    Ipfloa      rValue5,                  /* fifth value, e.g., B offset */
);

extern Idntuple InGenerateSD1Tuple (      /* OUT one parameter SD type tuple identifier */
    Ipoint      iValue1,                  /* single value, e.g., X */
);

```

```

/*                      CONVENIENCE FUNCTION PROTOTYPES (concluded)                      */

extern Idntuple InGenerateSD3Tuple ( /* OUT three parameter SD type tuple identifier */
    Ipint      iValue1, /* first value, e.g., X */
    Ipint      iValue2, /* second value, e.g., Y */
    Ipint      iValue3, /* third value, e.g., Z */
);

extern Idntuple InGenerateSD4Tuple ( /* OUT four parameter SD type tuple identifier */
    Ipint      iValue1, /* first value, e.g., Band Type 1 */
    Ipint      iValue2, /* second value, e.g., Band Type 2 */
    Ipint      iValue3, /* third value, e.g., Band Type 3 */
    Ipint      iValue4, /* fourth value, e.g., Band Type 4 */
);

extern Idntuple InGenerateSD5Tuple ( /* OUT five parameter SD type tuple identifier */
    Ipint      iValue1, /* first value, e.g., X */
    Ipint      iValue2, /* second value, e.g., Y */
    Ipint      iValue3, /* third value, e.g., Z */
    Ipint      iValue4, /* fourth value, e.g., T */
    Ipint      iValue5, /* fifth value, e.g., B */
);

extern Ipspiks_pixel_types IstTiledImageExport ( /* OUT desination image identifier */
    Idnimage    nSourceImage, /* source image identifier */
    Ipspiks_pixel_types stDestImage, /* tile source image data pointer */
    Idnimage    nTileImage, /* tile-sized image identifier */
    Idntuple    nTileSize, /* ND 5-tuple tile size */
    Idntuple    nSourceOffset, /* ND 5-tuple offset of tile in source image */
    Ipuint      uPrecision, /* number of source bits per pixel */
    Ipint      iExternalType, /* data type (external pixel) option */
);

extern Idnimage InTiledImageImport ( /* OUT desination image identifier */
    Ipspiks_pixel_types stSourceImage, /* tile source image data pointer */
    Idnimage    nDestImage, /* destination image identifier */
    Idnimage    nTileImage, /* tile-sized image identifier */
    Idntuple    nTileSize, /* ND 5-tuple tile size */
    Idntuple    nDestOffset, /* ND 5-tuple offset of tile in destination image */
    Ipuint      uPrecision, /* number of source bits per pixel */
    Ipint      iExternalType, /* data type (external pixel) option */
);

```

B.2 IPI-IIF macros, types and functions

B.2.1 Macro definitions

```

/*                                OPTION MACROS                                */

/*                                Boolean values                                */
#define IBOOL_FALSE                0 /* false Boolean value */
#define IBOOL_TRUE                 1 /* true Boolean value */

/*                                component classes                            */
#define ICOMPONENT_REQUIRED        1 /* syntax component required class */
#define ICOMPONENT_OPTIONAL        2 /* syntax component optional class */
#define ICOMPONENT_CHC             3 /* syntax component choice class */

/*                                compression algorithm options                */
#define ICOMPRESSION_FAX_G3        1 /* Fax Group 3 */
#define ICOMPRESSION_FAX_G4        2 /* Fax Group 4 */
#define ICOMPRESSION_JBIG          3 /* JBIG field */
#define ICOMPRESSION_JPEG          4 /* JPEG interchange format */
#define ICOMPRESSION_MPEG          5 /* MPEG bitstream */

/*                                compression modes                            */
#define ICOMPRESSION_UNCOMPRESSED  1 /* uncompressed mode */
#define ICOMPRESSION_ONE_DIMENSIONAL 2 /* one-dimensional mode */
#define ICOMPRESSION_TWO_DIMENSIONAL 3 /* two-dimensional mode */

/*                                entity classes                                */
#define IENTITY_SEQ                1 /* entity sequence class */
#define IENTITY_CHC                2 /* entity choice class */
#define IENTITY_SEQ_OF             3 /* entity sequence of class */
#define IENTITY_SET                4 /* entity set class */
#define IENTITY_SET_OF             5 /* entity set of class */
#define IENTITY_EQUIVALENCE        6 /* entity equivalence class */

/*                                gateway status                                */
#define IGATEWAY_OPEN_STATUS        1 /* gateway open status */
#define IGATEWAY_ERROR_STATUS       2 /* gateway error status */
#define IGATEWAY_CLOSED_STATUS      3 /* gateway closed status */

/*                                input/output access type options              */
#define IACCESS_RANDOM             1 /* random access */
#define IACCESS_SEQUENTIAL         2 /* sequential access */

/*                                input/output direction options                */
#define IDIRECTION_READ            1 /* read direction */
#define IDIRECTION_WRITE           2 /* write direction */
#define IDIRECTION_READ_WRITE      3 /* read and write direction */

/*                                input/output domain options                    */
#define IDOMAIN_PIKS               1 /* PIKS domain */
#define IDOMAIN_NON_PIKS           2 /* non-PIKS domain */

```

```

/*                                     input/output status                                     */
#define ISTATUS_CLEAR                  1 /* port clear                                     */
#define ISTATUS_PENDING                2 /* port operation pending                       */

/*                                     profile indicators                                   */
#define IPROFILE_FULL                  1 /* full profile of the IIF-DF                   */
#define IPROFILE_PIKS                  2 /* PIKS profile of the IIF-DF                   */
#define IPROFILE_COLOUR                3 /* colour profile of the IIF-DF                  */

/*                                     SYNTAX ENTITY TYPES                               */

/*                                     entities for the description of the entire IIF data format */
#define ISE_FULL_DATA_FORMAT           1 /* full data format                             */
#define ISE_FORMAT_DESCRIPTOR          2 /* format descriptor                           */
#define ISE_VERSION                    3 /* version                                     */
#define ISE_PROFILE                    4 /* profile                                     */
#define ISE_CONTENTS_HEADER            5 /* contents header                             */
#define ISE_CHAR_STRING                6 /* character string                             */
#define ISE_SPECIAL_CHAR_STRING        7 /* special character string                     */
#define ISE_CONTENTS                   8 /* contents                                    */
#define ISE_CONTENTS_ELEMENT           9 /* contents element                            */
#define ISE_CONTENTS_BODY              10 /* contents body                               */

/*                                     entities for the description of images                */
#define ISE_IMAGE                      101 /* image                                       */
#define ISE_IMAGE_STRUCTURE             102 /* image structure                           */
#define ISE_COMPOUND_IMAGE_STRUCTURE    103 /* compound image structure                   */
#define ISE_COMPOUND_IMAGE_ARRAY        104 /* compound image array                       */
#define ISE_DIMENSIONALITY              105 /* dimensionality                             */
#define ISE_DIMENSION_DESCRIPTION        106 /* dimension description                       */
#define ISE_ID                          107 /* identifier                                 */
#define ISE_SERIALIZATION                108 /* serialization                             */
#define ISE_DATA_PLACEMENT              109 /* data placement                             */
#define ISE_COMPOUND_IMAGE_RECORD        110 /* compound image record                       */
#define ISE_RECORD_CMP                  111 /* record component                           */
#define ISE_COMPOUND_IMAGE_LIST          112 /* compound image list                         */
#define ISE_COMPOUND_IMAGE_SET           113 /* compound image set                         */
#define ISE_FUND_IMAGE_STRUCTURE         114 /* fundamental image structure                */
#define ISE_BAND_RECORD                 115 /* band record                                */
#define ISE_BAND_RECORD_COMPONENT        116 /* band record component                       */
#define ISE_METRIC_ARRAY                 117 /* metric array                               */
#define ISE_METRIC_ARRAY_ELEMENT         118 /* metric array element                       */
#define ISE_PIXEL_STRUCTURE              119 /* pixel structure                             */
#define ISE_ELEMENTARY_PIXEL_STRUCTURE   120 /* elementary pixel structure                 */
#define ISE_PIXEL_BAND_RECORD            121 /* pixel band record                           */
#define ISE_PIXEL_BAND_RECORD_COMPONENT 122 /* pixel band record component                 */

```



```

/*          entities for the description of the representation of pixel values          */
#define ISE_REFERENCED_UNIT          201 /* referenced unit          */
#define ISE_DATA_UNIT                202 /* data unit          */
#define ISE_SUBDIVIDED_DATA_UNIT      203 /* subdivided data unit          */
#define ISE_SINGLE_DATA_UNIT          204 /* single data unit          */
#define ISE_BUILTIN_ENCODED_DATA_UNIT 205 /* builtin encoded data unit          */
#define ISE_BUILTIN_VALUE             206 /* builtin value          */
#define ISE_COMPLEX_VALUE             207 /* complex value          */
#define ISE_EXTERNALLY_DEF_DATA_UNIT  208 /* externally defined data unit          */
#define ISE_COMPRESSED_DATA_UNIT      209 /* compressed data unit          */
#define ISE_REGISTERED_DATA_UNIT      210 /* registered data unit          */
#define ISE_EXTERNAL_REFERENCE        211 /* external reference          */
#define ISE_EXTERNAL_ADDRESS          212 /* external address          */

/*          entities for the description of image-related data          */
#define ISE_IMAGE_RELATED_DATA        301 /* image-related data          */
#define ISE_HIST                      302 /* histogram          */
#define ISE_PARTITION_CLASS           303 /* partition class          */
#define ISE_LUT                      304 /* look up table          */
#define ISE_ROI                      305 /* region of interest          */
#define ISE_BOOL_ARRAY                306 /* Boolean array          */
#define ISE_ELLIPSE                   307 /* ellipse          */
#define ISE_INTERVAL_N_DIMENSIONAL    308 /* interval n-dimensional          */
#define ISE_INTERVAL_1_DIMENSIONAL    309 /* interval 1 dimensional          */
#define ISE_COORDINATE_N_DIMENSIONAL  310 /* coordinate n-dimensional          */
#define ISE_SET_OF_COORDINATES        311 /* set of coordinates          */
#define ISE_NBHOOD_ARRAY              312 /* neighbourhood array          */
#define ISE_INDEX_N_DIMENSIONAL       313 /* index n-dimensional          */
#define ISE_STATIC_ARRAY              314 /* static array          */
#define ISE_FEATURE_LIST               315 /* feature list          */
#define ISE_COORDINATE_AND_FEATURE     316 /* coordinate and feature          */
#define ISE_VALUE_BOUNDS_COLLECTION   317 /* value bounds collection          */
#define ISE_TRANSFORMATION_MATRIX      318 /* transformation matrix          */
#define ISE_PIXEL_RECORD               319 /* pixel record          */
#define ISE_PIXEL_RECORD_CMP          320 /* pixel record component          */
#define ISE_TUPLE                     321 /* tuple          */

```

```

/*          entities for the description of image attributes          */
#define ISE_IMAGE_ATTR          401 /* image attribute          */
#define ISE_METRIC_DESCRIPTION  402 /* metric description      */
#define ISE_DIMENSION_MAPPING  403 /* dimension mapping      */
#define ISE_MEASUREMENT_UNIT   404 /* measurement unit       */
#define ISE_DELTA_VECTOR       405 /* delta vector           */
#define ISE_METRIC_TRANSFORMATION 406 /* metric transformation   */
#define ISE_DOMAIN             407 /* domain                 */
#define ISE_CHANNEL_CHARACTERISTICS 408 /* channel characteristics */
#define ISE_COMPANDOR_DESCRIPTION 409 /* compandor description   */
#define ISE_COLOUR_REPRESENTATION 410 /* colour representation   */
#define ISE_STANDARDIZED_SPACE 411 /* standardized space      */
#define ISE_CIE_XYZ_SPACE       412 /* CIE XYZ space          */
#define ISE_CIE_YXY_SPACE       413 /* CIE Yxy space          */
#define ISE_CIE_UVW_SPACE       414 /* CIE UVW space          */
#define ISE_CIE_YUV_SPACE       415 /* CIE Yuv space          */
#define ISE_CIE_LAB_SPACE       416 /* CIE Lab space          */
#define ISE_CIE_LUV_SPACE       417 /* CIE Luv space          */
#define ISE_CIE_XYZ_COORDINATE  418 /* CIE XYZ coordinate     */
#define ISE_LIN_RGB_SPACE       419 /* linear RGB space        */
#define ISE_GAMMA_RGB_SPACE     420 /* gamma RGB space        */
#define ISE_YIQ_COLOUR_SPACE    421 /* YIQ colour space       */
#define ISE_YUV_COLOUR_SPACE    422 /* YUV colour space       */
#define ISE_YCBCR_COLOUR_SPACE  423 /* YCbCr colour space     */
#define ISE_NON_STANDARDIZED_SPACE 424 /* non-standardized space */
#define ISE_NON_STANDARDIZED_RGB 425 /* non-standardized RGB   */
#define ISE_NON_STANDARDIZED_IHS 426 /* non-standardized IHS   */
#define ISE_PRIMARIES           427 /* primaries              */
#define ISE_CIE_XY_COORDINATE    428 /* CIE xy coordinate      */
#define ISE_NON_STANDARDIZED_CMY 429 /* non-standardized CMY   */
#define ISE_NON_STANDARDIZED_CMYK 430 /* non-standardized CMYK  */
#define ISE_NON_STANDARDIZED_N_BAND 431 /* non-standardized N band */
#define ISE_COLOUR_BAND         432 /* colour band            */
#define ISE_TEST_COLOUR         433 /* test colour            */
#define ISE_PIKS_CONTROL         434 /* PIKS control           */

/*          entities for the description of image annotations          */
#define ISE_IMAGE_ANNOTATION    501 /* image annotation       */
#define ISE_LOCATION            502 /* location               */

/*          entities for the description of basic data objects          */
#define ISE_BASIC_DATA_OBJECT   601 /* basic data object      */
#define ISE_BASIC_DATA_TYPE     602 /* basic data type        */
#define ISE_COMPOUND_DATA_TYPE  603 /* compound data type     */
#define ISE_BASIC_ARRAY         604 /* basic array            */
#define ISE_BASIC_RECORD        605 /* basic record           */
#define ISE_BASIC_RECORD_CMP    606 /* basic record component */
#define ISE_BASIC_LIST          607 /* basic list             */
#define ISE_BASIC_SET           608 /* basic set              */
#define ISE_ELEMENTARY_DATA_TYPE 609 /* elementary data type   */

```

```

/*          entities for data types          */
#define ISE_ANY          700 /* Any          */
#define ISE_BOOL         701 /* Boolean        */
#define ISE_INTEGER      702 /* integer       */
#define ISE_BIT_STRING   703 /* bit string    */
#define ISE_OCTET_STRING 704 /* octet string  */
#define ISE_OBJECT_ID    706 /* object identifier */
#define ISE_REAL          709 /* real          */
#define ISE_IA5STRING     722 /* ia5string     */
#define ISE_GENERALIZED_TIME 724 /* generalized time */

/*          entities other                    */
#define ISE_PIXEL_FIELD_ENCODING 801 /* pixel field encoding */
#define ISE_DOR                  802 /* designated object reference */

/*          SYNTAX COMPONENTS                */

/*          full data format                  */
#define ISC_FORMAT_DESCRIPTOR 100 /* format descriptor */
#define ISC_CONTENTS_HEADER   101 /* contents header   */
#define ISC_CONTENTS          102 /* contents          */

/*          format descriptor                  */
#define ISC_SELF_IDENTIFICATION 200 /* self identification */
#define ISC_VERSION             201 /* version            */
#define ISC_PROFILE             202 /* profile            */

/*          version                          */
#define ISC_STANDARD            300 /* standard          */
#define ISC_PUBLICATION_DATE    301 /* publication date   */

/*          contents header                   */
#define ISC_TITLE               500 /* title             */
#define ISC_OWNER               501 /* owner             */
#define ISC_DATE_AND_TIME       502 /* date and time     */
#define ISC_MESSAGE             503 /* message           */
#define ISC_APPL_DATA           504 /* application data   */

/*          character string                  */
#define ISC_STANDARD_CHARS      600 /* standard characters */
#define ISC_SPECIAL_CHARS      601 /* special characters  */

/*          special character string           */
#define ISC_CHAR_SET_ESCAPE     700 /* character set escape */
#define ISC_CHARS               701 /* characters          */

/*          contents element                  */
#define ISC_PROLOG              900 /* prolog            */
#define ISC_BODY                901 /* body              */
#define ISC_EPILOG              902 /* epilog            */

```

```

/*                                contents body                                */
#define ISC_IMAGE                  1000 /* image                                */
#define ISC_IMAGE_RELATED_DATA    1001 /* image-related data                    */
#define ISC_IMAGE_ATTR            1002 /* image attribute                      */
#define ISC_IMAGE_ANNOTATION      1003 /* image annotation                     */
#define ISC_BASIC_DATA_OBJECT     1004 /* basic data object                    */

/*                                image                                */
#define ISC_IMAGE_STRUCTURE        10100 /* image structure                      */
#define ISC_IMAGE_DATA            10101 /* image data                          */

/*                                image structure                      */
#define ISC_STRUCTURE_TYPE        10200 /* structure type                      */
#define ISC_COMPOUND_STRUCTURE    10201 /* compound structure                  */
#define ISC_FUND_STRUCTURE        10202 /* fundamental structure              */
#define ISC_IMAGE_ATTRS          10203 /* image attributes                    */
/*                                /* image-related data, refer 1001 */
#define ISC_IMAGE_ANNOTATIONS     10205 /* image annotations                  */
#define ISC_PROCESSING_HISTORY    10206 /* processing history                  */

/*                                compound image structure            */
#define ISC_COMPOUND_IMAGE_ARRAY  10300 /* compound image array                */
#define ISC_COMPOUND_IMAGE_RECORD 10301 /* compound image record              */
#define ISC_COMPOUND_IMAGE_LIST   10302 /* compound image list                */
#define ISC_COMPOUND_IMAGE_SET    10303 /* compound image set                 */

/*                                compound image array                */
#define ISC_DIMENSIONALITY        10400 /* dimensionality                     */
#define ISC_SERIALIZATION         10401 /* serialization                      */
#define ISC_DATA_PLACEMENT        10402 /* data placement                     */
#define ISC_ELEMENT_STRUCTURE     10403 /* element structure                   */

/*                                dimensionality                      */
#define ISC_NUMBER_OF_DIMENSIONS  10500 /* number of dimensions                */
#define ISC_DIMENSION_DESCRIPTIONS 10501 /* dimension descriptions              */

/*                                dimension description                */
#define ISC_DIMENSION_ID          10600 /* dimension identifier                */
#define ISC_LOWER_BOUNDARY        10601 /* lower boundary                     */
#define ISC_UPPER_BOUNDARY        10602 /* upper boundary                     */

/*                                compound image record                */
#define ISC_NUMBER_OF_COMPONENTS  11000 /* number of components                */
/*                                /* data placement, refer 10402 */
#define ISC_RECORD_CMPS           11002 /* record components                  */

/*                                record components                    */
#define ISC_CMP_ID                11100 /* component identifier                */
#define ISC_CMP_STRUCTURE         11101 /* component structure                 */

```

```

/*                                compound image list                                */
#define ISC_NUMBER_OF_ELEMENTS      11200 /* number of elements                    */
/* data placement, refer 10402          */
/* element structure, refer 10403        */

/*                                compound image set                                */
#define ISC_NUMBER_OF_MEMBERS      11300 /* number of members                    */
/* data placement, refer 10402          */
#define ISC_MEMBER_STRUCTURE      11302 /* member structure                      */

/*                                fundamental image structure                        */
#define ISC_BAND_RECORD            11400 /* band record                          */
#define ISC_METRIC_ARRAY          11401 /* metric array                          */

/*                                band record                                        */
#define ISC_NUMBER_OF_BANDS      11500 /* number of bands                      */
/* data placement, refer 10402          */
/* record components, refer 11002        */

/*                                band record component                            */
#define ISC_BAND_ID              11600 /* band identifier                      */
/* component structure, refer 11101      */

/*                                metric array                                      */
/* dimensionality, refer 10400          */
/* serialization, refer 10401          */
/* data placement, refer 10402          */
/* element structure, refer 10403        */

/*                                metric array element                            */
#define ISC_PIXEL_STRUCTURE      11800 /* pixel structure                      */
/* band record, refer 11400            */
/* metric array, refer 11401            */

/*                                pixel structure                                  */
#define ISC_ELEMENTARY_PIXEL_STRUCTURE 11900 /* elementary pixel structure          */
#define ISC_COMPOUND_PIXEL_STRUCTURE 11901 /* compound pixel structure            */

/*                                pixel band record                                */
/* number of components, refer 11000    */
/* record components, refer 11002        */

/*                                pixel band record component                      */
/* band identifier, refer 11600          */
/* component structure, refer 11101      */

/*                                referenced unit                                  */
#define ISC_REFERENCE_LABEL      20100 /* reference label                      */
#define ISC_PIXEL_VALUES        20101 /* pixel values                          */

```

```

/*                                data unit                                */
#define ISC_SINGLE_DATA_UNIT      20200 /* single data unit                */
#define ISC_SUBDIVIDED_DATA_UNIT  20201 /* subdivided data unit             */
#define ISC_EXTERNAL_DATA_UNIT    20202 /* external data unit               */

/*                                subdivided data unit                     */
#define ISC_NUMBER_OF_PIXELS      20300 /* number of pixels                 */
#define ISC_PARTITIONS            20301 /* partitions                       */

/*                                single data unit                         */
#define ISC_BUILTIN_ENCODED_DATA_UNIT 20400 /* builtin encoded data unit        */
#define ISC_EXTERNALLY_DEF_DATA_UNIT  20401 /* externally defined data unit     */
#define ISC_COMPRESSED_DATA_UNIT      20402 /* compressed data unit             */
#define ISC_REGISTERED_DATA_UNIT      20403 /* registered data unit             */

/*                                builtin encoded data unit                 */
#define ISC_SEQ_OF_BOOL              20500 /* sequence of Boolean              */
#define ISC_SEQ_OF_OTHERS            20501 /* sequence of others               */

/*                                builtin values                           */
#define ISC_BOOL                     20600 /* Boolean                          */
#define ISC_NON_NEGATIVE_INTEGER     20601 /* non-negative integer            */
#define ISC_SIGNED_INTEGER           20602 /* signed integer                  */
#define ISC_REAL                     20603 /* real                            */
#define ISC_COMPLEX                   20604 /* complex                        */
#define ISC_ENUMERATED               20605 /* enumerated                      */

/*                                complex value                             */
/*                                /* real, refer 20603                      */
#define ISC_IMAGINARY                20701 /* imaginary                       */

/*                                compressed data unit                     */
#define ISC_DATA_REPRESENTATION      20900 /* data representation             */
#define ISC_DATA                     20901 /* data                            */

/*                                registered data unit                      */
#define ISC_REGISTRATION_ID          21000 /* registration identification      */
/*                                /* pixel values, refer 20101                */

/*                                external reference                       */
#define ISC_OBJECT_ADDRESS            21100 /* object address                  */
#define ISC_OBJECT_FORMAT             21101 /* object format                   */
#define ISC_OBJECT_INTERNAL_ID        21102 /* object internal identification  */

/*                                external address                         */
#define ISC_STRUCTURED_ADDRESS        21200 /* structured address              */
#define ISC_CLEARTEXT_ADDRESS        21201 /* cleartext address               */

```



```

/*                                image-related data                                */
#define ISC_ID                    30100 /* identifier                            */
#define ISC_USAGE                 30101 /* usage                               */
#define ISC_DATA_TYPE            30102 /* data type                           */
#define ISC_HIST                 30103 /* histogram                           */
#define ISC_LUT                 30104 /* lookup table                        */
#define ISC_ROI                 30105 /* region of interest                  */
#define ISC_NBHOOD_ARRAY        30106 /* neighbourhood array                 */
#define ISC_STATIC_ARRAY        30107 /* static array                        */
#define ISC_FEATURE_LIST        30108 /* feature list                        */
#define ISC_VALUE_BOUNDS_COLLECTION 30109 /* value bounds collection             */
#define ISC_MATRIX              30110 /* matrix                              */
#define ISC_PIXEL_RECORD        30111 /* pixel record                        */
#define ISC_TUPLE               30112 /* tuple                               */

/*                                histogram                                */
#define ISC_NUMBER_OF_CLASSES    30200 /* number of classes                   */
#define ISC_CLASS_DESCRIPTION    30201 /* class description                   */
#define ISC_CLASSES_AND_COUNTS   30202 /* classes and counts                  */

/*                                partition class                            */
/*                                /* lower boundary, refer 10601                */
/*                                /* upper boundary, refer 10602                */
#define ISC_COUNT               30302 /* count                               */

/*                                lookup table                             */
#define ISC_NUMBER_OF_INPUT_DIMENSIONS 30400 /* number of input dimensions         */
#define ISC_INPUT_DATA_TYPES         30401 /* input data types                   */
#define ISC_NUMBER_OF_OUTPUT_DIMENSIONS 30402 /* number of output dimensions        */
#define ISC_OUTPUT_DATA_TYPES        30403 /* output data types                  */
#define ISC_NUMBER_OF_ENTRIES        30404 /* number of entries                  */
#define ISC_INTERP_METHOD            30405 /* interpolation method                */
#define ISC_INPUT_VALUES             30406 /* input values                       */
#define ISC_OUTPUT_VALUES            30407 /* output values                      */

/*                                region of interest                       */
#define ISC_ROI_TYPE               30500 /* region of interest type            */
#define ISC_ARRAY                  30501 /* array                              */
#define ISC_ELLIPSE                30502 /* ellipse                            */
#define ISC_RECTANGULAR            30503 /* rectangular                        */
#define ISC_POLYGON                30504 /* polygon                            */
#define ISC_SET_OF_COORDINATES      30505 /* set of coordinates                 */
#define ISC_POLARITY_REVERSED       30506 /* polarity reversed                  */
#define ISC_INDEX_MANIPULATION      30507 /* index manipulation                 */

/*                                Boolean array                            */
/*                                /* dimensionality, refer 10400                */
/*                                /* serialization, refer 10401                */
#define ISC_BOOL_VALUES           30602 /* Boolean values                     */

```



```

/*                                Boolean values                                */
#define ISC_ROI_DIMENSIONALITY    30700 /* region of interest dimensionality */
#define ISC_ROI_SIZE              30701 /* region of interest size          */
#define ISC_ELLIPSE_CENTRE        30702 /* ellipse centre                    */
#define ISC_ELLIPSE_AXIS_LENGTH   30703 /* ellipse axis length                */
#define ISC_ELLIPSE_DIMENSIONALITY 30704 /* ellipse dimensionality             */
#define ISC_ELLIPTICAL_DIMENSIONS 30705 /* elliptical dimensions              */

/*                                interval N-dimensional                        */
/*                                /* number of dimensions, refer 10500          */
#define ISC_INTERVALS              30801 /* intervals                          */

/*                                interval 1-dimensional                        */
/*                                /* lower boundary, refer 10601              */
/*                                /* upper boundary, refer 10602              */

/*                                coordinate N-dimensional                      */
/*                                /* number of dimensions, refer 10500          */
#define ISC_VECTOR_TYPE            31001 /* vector type                        */
#define ISC_INTEGER_VECTOR         31002 /* integer vector                     */
#define ISC_REAL_VECTOR            31003 /* real vector                         */

/*                                set of coordinates                          */
#define ISC_NUMBER_OF_COORDINATES  31100 /* number of coordinates              */
#define ISC_PIXEL_COORDINATES      31101 /* pixel coordinates                  */

/*                                neighbourhood array                          */
/*                                /* dimensionality, refer 10400              */
/*                                /* serialization, refer 10401              */
/*                                /* element structure, refer 10403            */
#define ISC_ARRAY_ELEMENTS          31203 /* array elements                     */
#define ISC_SEMANTIC_LABEL          31204 /* semantic label                     */
#define ISC_KEY_PIXEL               31205 /* key pixel                          */
#define ISC_SCALE_FACTOR            31206 /* scale factor                       */

/*                                index N-dimensional                        */
/*                                /* number of dimensions, refer 31100          */
#define ISC_INDEX_VECTOR            31301 /* index vector                       */

/*                                static array                                */
/*                                /* dimensionality, refer 10400              */
/*                                /* serialization, refer 10401              */
/*                                /* element structure, refer 10403            */
/*                                /* array elements, refer 31203              */
/*                                /* semantic label, refer 31204              */

/*                                feature list                                */
/*                                /* number of coordinates, refer 31100          */
#define ISC_COORDINATES_AND_FEATURES 31501 /* coordinates and features          */

```

```

/*                                coordinate and feature                                */
#define ISC_COORDINATE              31600 /* coordinate                                */
#define ISC_FEATURE                  31601 /* feature                                    */

/*                                value bounds collection                                */
#define ISC_PIXEL_DATA_TYPE          31700 /* pixel data type                          */
/*                                /* lower boundary, refer 10601                      */
/*                                /* upper boundary, refer 10602                      */
/*                                /* number of coordinates, refer 31100                */
#define ISC_COORDINATE_LIST          31704 /* coordinate list                          */

/*                                transformation matrix                                */
#define ISC_MATRIX_SIZE              31800 /* matrix size                              */
#define ISC_MATRIX_ELEMENTS          31801 /* matrix elements                          */
#define ISC_INTEGERS                 31802 /* integers                                  */
#define ISC_REALS                    31803 /* reals                                     */

/*                                pixel record                                          */
/*                                /* number of components, refer 11000          */
/*                                /* record components, refer 11002          */

/*                                pixel record component                              */
/*                                /* identifier, refer 30100                      */
#define ISC_CMP_VALUE                32001 /* component value                          */

/*                                tuple                                                */
/*                                /* number of elements, refer 11200          */
/*                                /* element structure, refer 10403          */
#define ISC_VALUES                   32102 /* values                                   */

/*                                image attribute                                      */
#define ISC_METRIC                    40100 /* metric                                   */
#define ISC_CHANNEL                   40101 /* channel                                  */
#define ISC_COLOUR                    40102 /* colour                                   */
#define ISC_CONTROL                   40103 /* control                                  */
#define ISC_FREEFORM                  40104 /* freeform                                 */

/*                                metric description                                  */
#define ISC_COORDINATE_SYSTEM         40200 /* coordinate system                        */
#define ISC_DIMENSION_MAPPINGS        40201 /* dimension mappings                       */
#define ISC_METRIC_TRANSFORM          40202 /* metric transform                         */
#define ISC_DOMAIN                    40203 /* domain                                   */

/*                                dimension mapping                                    */
/*                                /* dimension identifier, refer 10600          */
#define ISC_COORDINATE_AXIS           40301 /* coordinate axis                          */
#define ISC_PHYSICAL_MEASUREMENT      40302 /* physical measurement                     */
#define ISC_ORIGIN                    40303 /* origin                                   */
#define ISC_DELTA_TYPE                40304 /* delta type                               */
#define ISC_SINGLE_DELTA              40305 /* single delta                             */
#define ISC_MULTIPLE_DELTAS           40306 /* multiple deltas                          */
#define ISC_PRECISION                 40307 /* precision                                */

```

```

/*                                measurement unit                                */
#define ISC_MEASUREMENT            40400 /* measurement                        */
#define ISC_BASIS                  40401 /* basis                            */
#define ISC_VERBAL_DESCRIPTION     40402 /* verbal description                */

/*                                metric transformation                            */
#define ISC_STANDARD_TRANSFORMATION 40600 /* standard transformation          */
/*                                /* matrix, refer 30110                      */

/*                                domain                                          */
#define ISC_DOMAIN_TYPE            40700 /* domain type                      */
/*                                /* verbal description, refer 40402      */

/*                                channel characteristics                        */
/*                                /* band identifier, refer 11600          */
/*                                /* verbal description, refer 40402      */
/*                                /* precision, refer 40307                */
#define ISC_CHANNEL_TYPE           40803 /* channel type                    */
#define ISC_CHANNEL_USAGE          40804 /* channel usage                   */
#define ISC_BACKGROUND_VALUE       40805 /* background value                */
#define ISC_COMPANDOR_DESCRIPTION  40806 /* compandor description           */
#define ISC_SAMPLING_FUNCTION      40807 /* sampling function                */
#define ISC_APPL_SPECIFICS         40808 /* application specifics            */

/*                                compandor description                          */
#define ISC_FUNCTION_TYPE          40900 /* function type                   */
#define ISC_TRANSFER_FACTOR        40901 /* transfer factor                 */
#define ISC_FUNCTION_PARAMETER     40902 /* function parameter              */
#define ISC_EXPANDER_FUNCTION      40903 /* expander function               */
#define ISC_APPL_SPECIFIC          40904 /* application specific             */

/*                                colour representation                          */
#define ISC_COLOUR_SPACE_TYPE      41000 /* colour space type               */
#define ISC_STANDARDIZED_SPACE     41001 /* standardized space              */
#define ISC_NON_STANDARDIZED_SPACE 41002 /* non-standardized space          */
#define ISC_TEST_COLOUR            41003 /* test colour                     */

/*                                standardized space                             */
#define ISC_CIE_XYZ                41100 /* CIE XYZ space                   */
#define ISC_CIE_YXY                41101 /* CIE Yxy space                  */
#define ISC_CIE_UVW                41102 /* CIE UVW space                  */
#define ISC_CIE_YUV                41103 /* CIE Yuv space                  */
#define ISC_CIE_LAB                41104 /* CIE Lab space                  */
#define ISC_CIE_LUV                41105 /* CIE Luv space                  */
#define ISC_LIN_RGB                41106 /* linear RGB space                */
#define ISC_GAMMA_RGB              41107 /* gamma RGB space                */
#define ISC_YIQ_COLOUR_SPACE       41108 /* YIQ colour space               */
#define ISC_YUV_COLOUR_SPACE       41109 /* YUV colour space               */
#define ISC_YCBCR_COLOUR_SPACE     41110 /* YCbCr colour space             */

```