```
   First draft of the API definition for libminc
   written by Bert Vincent and John Sled and Leila Baghdadi
   THRS APR 24/2003
   THRS MAY 01/2003
   THRS MAY 02/2003
   WED  MAY 07/2003 Bert
   TUS  MAY 20/2003 Leila & John
   THU  MAY 29/2003 Bert
        * Added complex type as a peer to integer, real, record, etc.
        * Proposed changing the term "midge" to a more standard "class"
        * Added functions for getting/setting labels in la belled volumes

   MON  JUN 02/2003 Leila & John
        * Sorted methods in Alphabetical order

   WED  JUN 04/2003 Leila & John

   THU  JUN 05/2003 Bert

   MON  JUN 09/2003 Bert
        * Added volume property list functions
        * Added miopen_volume
        * Added "units" functions

   WED  JUN 11/2003 Leila & John
        * Added the name argument to all attribute fucntions
        * Created a new flag mivoxel_order_t
        * Cleaned the description of micreate_volume function
        * Moved miget_volume_dimensions from the volume group into
          the dimension funstions group. Added a new flag miorder_t.
        * Added hyper_cube functions

   THU  JUN 12/2003 Bert
        * Added miclose_volume()

   THU  JUN 19/2003 Leila & John
        * Added valid min/max and range fucntions
        * Added the hyper_cube with icv functions.
```



```
********************************************************************************
**********************ATTRIBUTE/GROUP FUNCTIONS (7)*****************************

********************************************************************************
********************************************************************************
```

```
NAME
micreate_group − create a new group
SYNOPSIS
#include <minc.h>
int micreate_group ( mihandle_t      volume,
                     const char      *path,
                     const char      *name)



DESCRIPTION
This method creates a new empty group with the specified path and name.
RETURN VALUE
micreate_group returns MI_NOERROR if it successfully creates a group or
MIERROR otherwise
```



```
NAME
midelete_attr − delete the attribute given its name
SYNOPSIS
#include <minc.h>
int midelete_attr ( mihandle_t          volume,
                    const char          *path,
                    const char          *name)



DESCRIPTION
This methods deletes the attribute with the specified name. It also clears
memory and releases handles.
RETURN VALUE
midelete_attr returns MI_NOERROR if it successfully deletes the attribute
or MIERROR otherwise
```

```
NAME
midelete_group − delete an existing group
SYNOPSIS
#include <minc.h>
int midelete_group ( mihandle_t      volume,
                     const char      *path,
                     const char      *name)


DESCRIPTION
This method deletes the group with the given group name.
RETURN VALUE
midelete_group returns MI_NOERROR if it successfully deletes a group
or MIERROR otherwise
```



```
NAME
miget_attr_dimension − get the dimension of the attribute
SYNOPSIS
#include <minc.h>
int miget_attr_dimension ( mihandle_t           volume,
                           const char           *path,
                           const char           *name,
                           int                  *length )


DESCRIPTION
This method gets the dimension of the given attribute name.
RETURN VALUE
miget_attr_dimension returns MI_NOERROR if it successfully gets the
the dimension of a given attribute name or MIERROR otherwise
```

```
NAME
miget_attr_type − get the data type of the attribute
SYNOPSIS
#include <minc.h>
int miget_attr_type ( mihandle_t               volume,
                      const char               *path,
                      const char               *name,
                      mitype_t                 *attr_data_type)


DESCRIPTION
This method gets data_type the given attribute name. For the definition
of mitype_t see miget_data_type().
RETURN VALUE
miget_attr_type returns MI_NOERROR if it successfully gets the
data type of a given attribute name or MIERROR otherwise
In the initial implementation, attributes are restricted to type of
either MI_TYPE_DOUBLE or MI_TYPE_CHAR.
```



```
NAME
miget_attr_values − get the value(s) of an attribute
SYNOPSIS
#include <minc.h>
int miget_attr_values ( mihandle_t             volume,
                        mitype_t               attr_data_type,
                        const char             *path,
                        const char             *name,
                        int                    length,
                        void                   *values)

DESCRIPTION
This method returns the values of a given attribute name
(of type double or string)in array values.
RETURN VALUE
miget_attr_values returns MI_NOERROR if it successfully gets the
attribute value(s) or MIERROR otherwise
```

```
NAME
miset_attr_values − set the value(s) of the attribute
SYNOPSIS
#include <minc.h>
int miset_attr_values ( mihandle_t              volume,
                        mitype_t                attr_data_type,
                        const char              *path,
                        const char              *name,
                        int                     length,
                        void                    *values)


DESCRIPTION
This method sets the value(s) of type double or string for the
specfied attribute name.If the attribute does not exist,
then it gets created it first.
RETURN VALUE
miset_attr_values returns MI_NOERROR if it successfully sets the
attribute value(s) or MIERROR otherwise
```



```
********************************************************************************
********************DATA*TYPE/SPACE*FUNCTIONS*(4)***************************
********************************************************************************
********************************************************************************
```

```
NAME
miget_data_class − get the class type of data
SYNOPSIS
#include <minc.h>
int miget_data_class ( mihandle_t       volume,
                       miclass_t        *volume_class)

DESCRIPTION
vol_class is the interpretation of the numerical values of the volume
independent of the numerical type used to represent the data.  The
class type is defined as follows.
tyepdef enum {
MI_CLASS_REAL    = 0,
MI_CLASS_INT     = 1,
MI_CLASS_LABEL   = 2,
MI_CLASS_COMPLEX = 3,
MI_CLASS_RECORD  = 4
} miclass_t;
where MI_CLASS_LABEL is used for enumerated data in which a
description is associated with each value and MI_CLASS_RECORD is used
for aggregate datatypes consisting of multiple values.

RETURN VALUE
miget_data_class returns the data class of the volume or MI_ERROR if an
error occurs.
```



```
NAME
miget_data_type − get the volume's data type
SYNOPSIS
#include <minc.h>
int  miget_data_type( mihandle_t       volume,
                      mitype_t         *volume_data_type)
DESCRIPTION
miget_data_type gets the date−type of the volume, which in this case refers
to the actual format in which the data is stored on disk. Note that volume
of type string is not supported.
The mitype_t type is defined as follows:
typedef enum {
        MI_TYPE_BYTE = 1,       /* 8−bit signed integer */
        MI_TYPE_CHAR = 2,       /* ASCII text */
        MI_TYPE_SHORT = 3,      /* 16−bit signed integer */
        MI_TYPE_INT = 4,        /* 32−bit signed integer */
        MI_TYPE_FLOAT = 5,      /* 32−bit floating point */
        MI_TYPE_DOUBLE = 6,     /* 64−bit floating point */
        MI_TYPE_STRING = 7,     /* string */
        MI_TYPE_UBYTE = 100,    /* 8−bit unsigned integer */
        MI_TYPE_USHORT = 101,   /* 16−bit unsigned integer */
        MI_TYPE_UINT = 102,     /* 32−bit unsigned integer */
        MI_TYPE_SCOMPLEX = 1000,   /* 16−bit signed integer complex */
        MI_TYPE_ICOMPLEX = 1001,   /* 32−bit signed integer complex */
        MI_TYPE_FCOMPLEX = 1002,   /* 32−bit floating point complex */
        MI_TYPE_DCOMPLEX = 1003,   /* 64−bit floating point complex */
        MI_TYPE_UNKNOWN  = −1     /* when the type is a record */
} mitype_t;
typedef struct {
        short real;
        short imag;
} miscomplex_t;
typedef struct {
        int real;
        int imag;
} miicomplex_t;
typedef struct {
        float real;
        float imag;
} mifcomplex_t;
typedef struct {
        double real;
        double imag;
} midcomplex_t;
RETURN VALUE
miget_data_type returns the data−type of the volume or MI_ERROR if an
error occurs.
```

```
NAME
miget_space_name, miset_space_name − get or set the space type name for
a MINC volume
SYNOPSIS
#include <minc.h>
int miget_space_name( mihandle_t volume, char **name );
int miset_space_name( mihandle_t volume, const char *name );
DESCRIPTION
miget_space_name retrieves the "space" name of the given volume,
returning a pointer to a string.  The memory allocated
by this function should be released with a call to mifree_name().
miset_space_name will set the space name of the volume.  The new name
must be no greater than 128 characters in length, including the
trailing zero byte.
Space names are used to define the coordinate system of the volume.  Three
standard values are defined by MINC:
MI_NATIVE    "native____"
MI_TALAIRACH "talairach_"
MI_CALLOSAL  "callosal__"
"Native" space specifies the coordinate system of a particular
scanner.  Talairach and callosal are standard coordinate systems for
brain images.
If not explicitly set, the space will be type MI_NATIVE by default.
RETURN VALUE
miget_space_name returns the length of the name retrieved, including
the terminating zero byte.  miset_space_name will return MI_NOERROR
on success.  Both functions return MI_ERROR if an error occurs.
SEE ALSO
mifree_name
```



```
*****************************************************************************
**********************DIMENSION*FUNCTIONS*(3d)*******************************

*****************************************************************************
*****************************************************************************
```

```
NAME
miget_volume_from_dimension − to figure out whether a dimensions is
associated with a volume or not
SYNOPSIS
#include <minc.h>
int miget_volume_from_dimension ( midimhandle_t         dimension,
                                  mihandle_t            *volume)

DESCRIPTION
This method returns the volume handle associated with A given dimension.
RETURN VALUE
miget_volume_from_dimension returns MIERROR if the specified handle
is not associated with tghe volume and MI_NOERROR otherwise.
```



```
NAME
micopy_dimension − create copy of the given dimension
SYNOPSIS
#include <minc.h>
int micopy_dimension ( midimhandle_t       dimension,
                       midimhandle_t       *new_dim_ptr)

DESCRIPTION
Creates a copy of the specified dimension and returns the handle to the copy
RETURN VALUE
micopy_dimension returns MI_NOERROR if it successfully copies all the
attributes of the provided dimension and MI_ERROR otherwise
```

```
NAME
micreate_dimension − define a new dimension in a MINC volume
SYNOPSIS
#include <minc.h>
int micreate_dimension ( const char     *name,
                         midimclass_t   class, midimattr_t      attr,
                         unsigned long  size, midimhandle_t     *new_dim_ptr);

DESCRIPTION
This function defines a dimension that can be used in the definition
of a new MINC volume (see the create_volume function).  The name may
be an arbitrary string of up to 128 alphanumeric characters. Any of
the "standard" names retained from MINC 1.0 retain their default
behaviors: MIxspace, MIyspace, and MIzspace default to spatial
dimensions, and MItime default to be a time dimension.  MItfrequency
is a temporal frequency axis, and MIxfrequency, MIyfrequency, and
MIzfrequency are spatial frequency axes.  Any other name may be used.
When initially defined, a regularly−sampled dimension will have a
"start" value of zero, and a "separation" or "step" value of 1.0.  An
irregular dimension will be initialized with all offsets equal to
zero.
The type midimclass_t is defined as follows:
typedef enum {
MI_DIMCLASS_ANY        = 0,             /* Don't care (or unknown) */
MI_DIMCLASS_SPATIAL    = 1,             /* Space */
MI_DIMCLASS_TIME       = 2,             /* Time */
MI_DIMCLASS_SFREQUENCY = 3,             /* Spatial frequency */
MI_DIMCLASS_TFREQUENCY = 4,             /* Temporal frequency */
MI_DIMCLASS_USER       = 5              /* Arbitrary user−defined axis */
} midimclass_t;
The type midimattr_t is a bit field of dimension attributes, defined as
follows:
typedef unsigned int dimattr_t;
#define MI_DIMATTR_ALL 0
#define MI_DIMATTR_REGULARLY_SAMPLED 0x1
#define MI_DIMATTR_NOT_REGULARLY_SAMPLED 0x2
The "size" argument may range from 0 to 2^32, which should provide
enough range to represent detail on the order of 10 Angstroms in
typical medical imaging applications.
If successful, the function will return a handle to the newly−defined
dimension in the location specified by "new_dim_ptr".
RETURN VALUE
micreate_dimension returns MI_NOERROR on success, MI_ERROR on failure.
```



```
NAME
mifree_dimension_handle − delete the dimension definition associated
with the given handle
SYNOPSIS
#include <minc.h>
int mifree_dimension_handle ( midimhandle_t       dim_ptr)
DESCRIPTION
Deletes the dimension definition (i.e., dimension handle and dimension itself)
only if the dimension is NOT associated with a volume.
RETURN VALUE
mifree_dimension_handle returns MI_NOERROR if it successfully deletes
a dimension or MI_ERROR otherwise.
```

```
NAME
miget_volume_dimensions − retrieve the list of dimensions defined in a MINC
volume, according to their class and attribute.
SYNOPSIS
#include <minc.h>
int miget_volume_dimensions ( mihandle_t        volume,
                              midimclass_t      class,
                              midimattr_t       attr,
                              miorder_t         order,
                              int               array_length,
                              midimhandle_t     dimensions[]);


DESCRIPTION
This function is used to retrieve an array of dimension handles for a
MINC volume.  It will place the handles of the first "array_length"
dimensions into the "dimensions[]" array, returning only those dimension
whose characteristics match the "class" and "attr" parameters.
The miorder_t is an enumerated type flag which determines whether the
dimenions ordering is from file or is from the apparent order given by
the user.
typedef enum {
        MI_ORDER_FILE      = 0
        MI_ORDER_APPARENT  = 1
} miorder_t;
The following example will return up to 3 spatial dimensions:
    int result;
    midimhandle_t dimensions[3];
    mihandle_t volume;
    result = miget_vol_dimensions (volume, MI_DIMCLASS_SPATIAL,
                                   MI_DIMATTR_ALL, MI_ORDER_FILE,
                                   3, dimensions);

RETURN VALUE
miget_volume_dimensions returns the number of dimension handles copied
to the array, or MI_ERROR on failure.
```



```
NAME
miset_apparent_dimension_order − set apparent dimension order
SYNOPSIS
#include <minc.h>
int miset_apparent_dimension_order ( mihandle_t         volume,
                                     int                array_length,
                                     midimhandle_t      dimensions[])


DESCRIPTION
This method sets an apparent dimnension order. The user can sort the
dimensions in any desired order. If the user specfies fewer dimensions
than the existing ones, then they are assumed to be added to the last.
For example, given (z,y,x) for the fie dimension order of (x,y,z,t)
will result in (t,z,y,x) and so on.
RETURN VALUE
miset_apparent_dimension_order returns MI_NOERROR if i successfully
sets the apparent dimension order or MI_ERROR otherwise
```

```
NAME
miset_apparent_dimension_order_by_name − set apparent dimension order
by name
SYNOPSIS
#include <minc.h>
int miset_apparent_dimension_order_by_name ( mihandle_t         volume,
                                             int                array_length,
                                             char               **names)


DESCRIPTION
This method sets an apparent dimension order by dimension name. Note that
all dimension names must be different or an error occurs.
RETURN VALUE
miset_apparent_dimension_order_by_name returns MI_NOERROR if i successfully
sets the apparent dimension order by name or MI_ERROR otherwise
```



```
NAME
miget_dimension_apparent_voxel_order − gets the apparent order of voxels
SYNOPSIS
#include <minc.h>
int miget_dimension_apparent_voxel_order ( midimhandle_t        dimension,
                                           miflipping_t         *file_order,
                                           miflipping_t         *sign)


DESCRIPTION
This method gets the apparent order of voxels for the specified dimension
and the sign of the step values.
RETURN VALUE
miget_dimension_apparent_voxel_order returns MI_NOERROR if i successfully
gets the apparent voxel order of the specified dimension or MI_ERROR otherwise
```

```
NAME
miset_dimension_apparent_voxel_order − sets the apparent order of voxels
SYNOPSIS
#include <minc.h>
int miset_dimension_apparent_voxel_order ( midimhandle_t        dimension,
                                           miflipping_t         flipping_order)

DESCRIPTION
This method sets the apparent order of voxels for the specified dimension.
The miflipping_t is an enumerated type as follows:
tyepdef enum {
        MI_FILE_ORDER           = 0,   /* no flip */
        MI_COUNTER_FILE_ORDER   = 1,   /* flip    */
        MI_POSITIVE             = 2,   /* check step if positive −> no flip
                                                         negative −> flip    */
        MI_NEGATIVE             = 3    /* check step if positive −> flip
                                                         negative −> no flip */
} miflipping_t;
RETURN VALUE
miset_dimension_apparent_voxel_order returns MI_NOERROR if i successfully
sets the apparent voxel order of the specified dimension or MI_ERROR otherwise
```



```
NAME
miget_dimension_class, miset_dimension_class − get or set the class of
a MINC dimension.
SYNOPSIS
#include <minc.h>
int miget_dimension_class ( midimhandle_t       dimension,
                            midimclass_t        *class );

int miset_dimension_class ( midimhandle_t       dimension,
                            midimclass_t        class);

DESCRIPTION
The "class" of a MINC dimension defines the general type of a dimension,
whether it is a spatial dimension, a time dimension, or a frequency dimension
as transformed from either space or time.  User−defined dimension are also
permitted, with no default handling assumed.
The definition of midimclass_t is as follows:
typedef enum {
MI_DIMCLASS_ANY         = 0,            /* Don't care (or unknown) */
MI_DIMCLASS_SPATIAL     = 1,            /* Space */
MI_DIMCLASS_TIME        = 2,            /* Time */
MI_DIMCLASS_SFREQUENCY  = 3,            /* Spatial frequency */
MI_DIMCLASS_TFREQUENCY  = 4,            /* Temporal frequency */
MI_DIMCLASS_USER        = 5             /* Arbitrary user−defined axis */
} midimclass_t;
While the MINC library does not enforce any particular semantics based
upon dimension class, individual MINC programs may define default
behaviors for certain classes of dimensions.
RETURN VALUE
Returns MI_NOERROR on success, or MI_ERROR on failure.
```

```
NAME
miget_dimension_cosines, miset_dimension_cosines − Get or set the dimension's
cosine vector.
SYNOPSIS
#include <minc.h>
int miget_dimension_cosines ( midimhandle_t dimension,
                              double        cosines[3]);
int  miset_dimension_cosines ( midimhandle_t dimension,
                               const double  cosines[3]);
DESCRIPTION
Spatial dimension in MINC volumes may be associated with a vector of direction
cosines which define the precise orientation of the axis relative to "true"
x, y, or z coordinates.
The direction cosine vector always consists of exactly three values
which correspond to the x, y, and z directions, respectively.  This is
true regardless of the ordering of dimensions in a specific volume or
data object.
Because of the direction cosines, it is possible for MINC volumes to
define non−orthogonal dimensions.
These functions fail if the dimension is not of the spatial class.
RETURN VALUE
Returns MI_NOERROR on success, or MI_ERROR on failure.
```



```
NAME
miget_dimension_name, miset_dimension_name − get or set the identifier
of a MINC dimension
SYNOPSIS
#include <minc.h>
int miget_dimension_name ( midimhandle_t dimension, char **name_ptr );
int miset_dimension_name ( midimhandle_t dimension, const char *name );
DESCRIPTION
miget_dimension_name retrieves the name of the given dimension, allocating
the space needed.  The memory allocated by this function should be released
with a call to mifree_name().
miset_dimension_name will rename an existing dimension.  The new name
must be no greater than 128 characters in length, including the
trailing zero byte.
RETURN VALUE
miget_dimension_name returns the length of the name retrieved, including
the terminating zero byte.  miset_dimension_name will return MI_NOERROR
on success.  Both functions return MI_ERROR if an error occurs.
SEE ALSO
mifree_name
```

```
NAME
miget_dimension_offsets, miset_dimension_offsets − get or set the absolute
world coordinates of points along a MINC dimension
SYNOPSIS
#include <minc.h>
int miget_dimension_offsets ( midimhandle_t     dimension, double       offsets[
],
                              unsigned long     array_length,
                              unsigned long     start_position);

int miset_dimension_offsets ( midimhandle_t     dimension, const double offsets[
],
                              unsigned long     array_length,
                              unsigned long     start_position);

DESCRIPTION
These functions get or set the dimension offsets, that is, the
absolute world coordinates of each sampled point along the dimension.
The caller may retrieve up to "array_length" values, starting at the
integer index "start_position".  Thus an arbitrary contiguous subset
of the dimension’s offsets may be retrieved or stored.  An error is
returned if the "start_position" exceeds the total size of the
dimension.  If the value of "start_position" is legal, but the sum of
"start_position" and "array_length" exceeds the size of the dimension,
the function will get or set offsets up to the size of the dimension.
Any extra positions in the offsets[] array will be ignored.
It is explicitly legal to call this function for a regularly sampled
dimension.  The result will be a list of values calculated from the
"start" and "separation" values of the dimension.  However, it is
not possible to set offsets on a regularly sampled dimension.
RETURN VALUE
Returns the number of offset values read or written , or MI_ERROR if
an error is detected.
```



```
NAME
miget_dimension_sampling_flag, miset_dimension_sampling_flag − get or set
the sampling flag for a MINC dimension
SYNOPSIS
#include <minc.h>
int miget_dimension_sampling_flag ( midimhandle_t       dimension,
                                    BOOLEAN             *sampling_flag);

int miset_dimension_sampling_flag ( midimhandle_t       dimension,
                                    BOOLEAN             sampling_flag);

DESCRIPTION
The miget_dimension_sampling flag function retrieves the value
of the "sampling" flag for a given MINC dimension.  This flag
is true (non−zero) if the dimension is sampled at regular
intervals, and false if the dimension is sampled irregularly.
If a dimension has regular sampling, the miget_dimension_step
function may be used to retrieve the sampling interval, and the
miget_dimension_start function may be used to retrieve the origin
value along the axis.
If a dimension has irregular sampling, the miget_dimension_offsets
function may be used to retrieve the positions of each sample along
that axis.
RETURN VALUE
These functions returns MI_NOERROR on success, or MI_ERROR if an
error is detected (for example, if a parameter is invalid).
```

```
NAME
miget_dimension_separation, miset_dimension_separation − set/get the
sampling interval for a single dimension
SYNOPSIS
#include <minc.h>
int miget_dimension_separation ( midimhandle_t          dimension,
                                 mivoxel_order_t        voxel_order,
                                 double                 *separation_ptr );

int miset_dimension_separation ( midimhandle_t          dimension,
                                 mivoxel_order_t        voxel_order,
                                 double                 separation );

DESCRIPTION
Gets or sets the constant sampling interval defined on a regularly−sampled
dimension.  While it is legal to call these functions for an irregularly−
sampled dimension, the values will be ignored. The mivoxel_order_t is an
enumerated type which is defined as follows
typedef enum {
        MI_ORIGINAL_ORDER   = 0,
        MI_APPARENT_ORDER  = 1
} mivoxel_order_t;
This flag specifies whether the voxel order is original from file or
is an apparent one which can be the default (i.e., the same as the
original file) or the order that is specified by the user.
If not explicitly set, the separation will have a default value of one.
RETURN VALUE
Returns MI_NOERROR on success, or MI_ERROR on failure.
SEE ALSO
miget_dimension_separations, miset_dimension_separations
```



```
NAME
miget_dimension_separations, miset_dimension_separations − get/set the
sampling intervals for a list of dimensions.
SYNOPSIS
#include <minc.h>
int miget_dimension_separations ( const midimhandle_t   dimensions[],
                                  mivoxel_order_t       voxel_order,
                                  int                   array_length,
                                  double                separations[] );

int miset_dimension_separations ( const midimhandle_t   dimensions[],
                                  mivoxel_order_t       voxel_order,
                                  int                   array_length,
                                  const double          separations[] );

DESCRIPTION
These functions get or set the scalar separation (sampling interval)
associated with each of the dimensions in the input "dimensions[]"
array.  The "array_length" parameter specifies the size of both the
input and output arrays. While it is legal to call these functions for
an irregularly− sampled dimension, the values will be ignored.
RETURN VALUE
Returns the number of separations copied to (or from) the array, or
MI_ERROR if an error occurs.
SEE ALSO
miget_dimension_separation, miset_dimension_separation
```

```
NAME
miget_dimension_size, miset_dimension_size − get or set the length of a
MINC dimension
SYNOPSIS
#include <minc.h>
int miget_dimension_size ( midimhandle_t        dimension,
                           unsigned long        *size_ptr );

int miset_dimension_size ( midimhandle_t        dimension,
                           unsigned long        size );

DESCRIPTION
These functions get or set the size (or length) of a MINC 2 dimension
object used in creating a new volume.  The size of a dimension
associated with an existing volume cannot be changed.  One can, however,
make a copy of an existing dimension and set the size of the copy.
RETURN VALUE
Returns MI_NOERROR on success, MI_ERROR on failure.
```



```
NAME
miget_dimension_sizes − retrieve the sizes of an array of dimension handles
SYNOPSIS
#include <minc.h>
int miget_dimension_sizes ( const midimhandle_t         dimensions[],
                            int                         array_length,
                            unsigned long               sizes[] );

DESCRIPTION
This function will copy the lengths of each of the dimensions listed in the
"dimensions[]" array into the "sizes[]" array.  The parameter "array_length"
specifies the length of both of the arrays.
RETURN VALUE
Returns MI_NOERROR on success, or MI_ERROR on failure.
SEE ALSO
miget_dimension_size, miset_dimension_size
```

```
NAME
miget_dimension_start, miset_dimension_start − get or set the origin
of a MINC dimension
SYNOPSIS
#include <minc.h>
int miget_dimension_start ( midimhandle_t       dimension,
                            mivoxel_order_t     voxel_order,
                            double              *start_ptr);

int miset_dimension_start ( midimhandle_t       dimension,
                            mivoxel_order_t     voxel_order,
                            double              start);

DESCRIPTION
These functions get or set the origin of the dimension in world
coordinates. While a "start" value may be legally associated with any
dimension, it is considered meaningless when associated with an
irregularly sampled dimension.

RETURN VALUE
Returns MI_NOERROR on success, or MI_ERROR on failure.
```



```
NAME
miget_dimension_starts, miset_dimension_starts − get or set the start
values
SYNOPSIS
#include <minc.h>
int miget_dimension_starts ( const midimhandle_t        dimensions[],
                             mivoxel_order_t            voxel_order,
                             int                        array_length,
                             double                     starts[]);

int miset_dimension_starts ( const midimhandle_t        dimensions[],
                             mivoxel_order_t            voxel_order,
                             int                        array_length,
                             const                      double starts[]);

DESCRIPTION
These functions get or set the start value for an array of
regularly−sampled dimensions.  The start value defines the origin of
that dimension.  While it is legal to call these functions for an
irregularly−sampled dimension, the values will be ignored.
If not explicitly set, the start value defaults to zero.
RETURN VALUE
These functions return the number of start values copied to or from
the starts[] array, or MI_ERROR on failure.
```

```
NAME
miget_dimension_units, miset_dimension_units − get or set the unit string
for a MINC dimension
SYNOPSIS
#include <minc.h>
int miget_dimension_units ( midimhandle_t dimension, char **units_ptr );
int miset_dimension_units ( midimhandle_t dimension, const char *units );
DESCRIPTION
miget_dimension_units retrieves the units of the given dimension,
allocating the space needed for the string.  The memory allocated by
this function should be released with a call to mifree_name().
miset_dimension_name will set the units for an existing dimension.
The new string must be no greater than 128 characters in length,
including the trailing zero byte.
Typical values for units include "mm" or "cm" for spatial dimensions
and "seconds" or "msec" for time dimensions.
RETURN VALUE
miget_dimension_units returns the length of the string retrieved,
including the terminating zero byte.  miset_dimension_units will
return MI_NOERROR on success.  Both functions return MI_ERROR if an
error occurs.
SEE ALSO
mifree_name
```



```
NAME
miget_dimension_width, miset_dimension_width − get or set the full−width
half−maximum value for points along a MINC dimension
SYNOPSIS
#include <minc.h>
int miget_dimension_width ( midimhandle_t       dimension,
                            mivoxel_order_t     voxel_order,
                            double              *width_ptr );

int miset_dimension_width ( midimhandle_t       dimension,
                            mivoxel_order_t     voxel_order,
                            double              width );

DESCRIPTION
These functions get or set the dimension width, that is, the
full−width half−maximum values of each sampled point along the
dimension.
These functions are used to set a constant width for regularly−sampled
dimensions.
If not explicitly set, the width will be assumed to be equal to the
dimension's step size.
RETURN VALUE
Returns MI_NOERROR on success, or MI_ERROR on failure.  Will fail if
called for an irregularly−sampled dimension.
SEE ALSO
miget_dimension_widths, miset_dimension_widths
```

```
NAME
miget_dimension_widths, miset_dimension_widths − get or set the full−width
half−maximum values for points along a MINC dimension
SYNOPSIS
#include <minc.h>
int miget_dimension_widths ( midimhandle_t      dimension,
                             mivoxel_order_t    voxel_order,
                             unsigned long      array_length,
                             unsigned long      start_position,
                             double             widths[]);

int miset_dimension_widths ( midimhandle_t      dimension,
                             mivoxel_order_t    voxel_order,
                             unsigned long      array_length,
                             unsigned long      start_position,
                             const double       widths[]);

DESCRIPTION
These functions get or set the dimension widths, that is, the
full−width half−maximum values of each sampled point along the
dimension.
The caller may retrieve up to "array_length" values, starting at the
integer index "start_position".  Thus an arbitrary contiguous subset
of the dimension’s widths may be retrieved or stored.  An error is
returned if the "start_position" exceeds the total size of the
dimension.  If the value of "start_position" is legal, but the sum of
"start_position" and "array_length" exceeds the size of the dimension,
the function will get or set widths up to the size of the dimension.
Any extra positions in the widths[] array will be ignored.
It is explicitly legal to call this function for a regularly sampled
dimension.  The result will be a list of constant width values.
However, it is not possible to set widths on a regularly sampled
dimension.
RETURN VALUE
Returns the number of offset values read or written , or MI_ERROR if
an error is detected.
SEE ALSO
miget_dimension_width, miset_dimension_width
```



```
********************************************************************************
************************FREE FUNCTIONS (2)**************************************

********************************************************************************
********************************************************************************
```

```
NAME
mifree_name, mifree_names − free the storage allocated for strings by
MINC functions
SYNOPSIS
#include <minc.h>
int mifree_name ( char *name );
int mifree_names ( char **names );
DESCRIPTION
Frees the space allocated for string storage by MINC function such as
miget_dimension_name and miget_space_name.
RETURN VALUE
Returns MI_NOERROR on success, or MI_ERROR on failure.
SEE ALSO
miget_dimension_name, miget_space_name
```



```
********************************************************************************
************************HYPER_CUBE FUNCTIONS (8)********************************

********************************************************************************
********************************************************************************
```

```
NAME
miget_hyper_cube_normalized − get a normalized hyper_cube
SYNOPSIS
#include <minc.h>
int miget_hyper_cube_normalized ( mihandle_t    volume,
                                  mitype_t      buffer_data_type,
                                  long          voxel_offsets[],
                                  long          sizes[],
                                  double        min,
                                  double        max,
                                  void          *buffer)


DESCRIPTION
The real values in the volume from the inetrval min through max
is mapped to the maximum representable range for the requested
data type. Float type is NOT an allowed data type.
RETURN VALUE
miget_hyper_cube_normalized returns MI_NOERROR if it successfully returns
a normalized hyper_cube with specified size and type and MI_ERROR otherwise
```



```
NAME
miget_hyper_cube_size − calculate the size of the hyper_cube
SYNOPSIS
#include <minc.h>
int miget_hyper_cube_size ( mitype_t            volume_data_type,
                            int                 number_of_dimensions,
                            int                 sizes_of_dimensions[],
                            misize_t            *size)

DESCRIPTION
calculate the size of the hyper_cube. i.e., the amount of memory in BYTES
which is needed to store the hyper_cube
RETURN VALUE
miget_hyper_cube_size returns MI_NOERROR if it successfully calculates the
size of the hyper_cube with specified dimensions and MI_ERROR otherwise
```

```
NAME
miget_hyper_cube_with_icv − get hyper_cube with the specified icv
SYNOPSIS
#include <minc.h>
int miget_hyper_cube_with_icv ( int             icv,
                                mitype_t        buffer_data_type,
                                long            voxel_offsets[],
                                long            sizes[],
                                void            *buffer)

DESCRIPTION
This method gets the hyper_cube with the specified icv.
RETURN VALUE
miget_hyper_cube_with_icv returns MI_NOERROR if it successfully gets
the hyper_cube with specified icv and MI_ERROR otherwise
```



```
NAME
miset_hyper_cube_with_icv − set hyper_cube with the specified icv
SYNOPSIS
#include <minc.h>
int miset_hyper_cube_with_icv ( int             icv,
                                mitype_t        buffer_data_type,
                                long            voxel_offsets[],
                                long            sizes[],
                                void            *buffer)

DESCRIPTION
This method sets the hyper_cube with the specified icv.
RETURN VALUE
miset_hyper_cube_with_icv returns MI_NOERROR if it successfully sets
the hyper_cube with specified icv and MI_ERROR otherwise
```

```
NAME
miget_real_value_hyper_cube − get a real value hyper_cube
SYNOPSIS
#include <minc.h>
int miget_real_value_hyper_cube( mihandle_t     volume,
                                 mitype_t       buffer_data_type,
                                 long           voxel_offsets[],
                                 long           sizes[],
                                 void           *buffer)

DESCRIPTION
This method converts the data from the type that is in the file in a value
preserving way.If real value exceeds range of the requested data type, the
result is UNDEFINED. The void pointer is pointing to memory which has to be
allocated by the programmer in advance.
RETURN VALUE
miget_real_value_hyper_cube returns MI_NOERROR if it successfully returns
the real value hyper_cube and MI_ERROR otherwise
```



```
NAME
miset_real_value_hyper_cube − set a real value hyper_cube
SYNOPSIS
#include <minc.h>
int miset_real_value_hyper_cube( mihandle_t     volume,
                                 mitype_t       buffer_data_type,
                                 long           voxel_offsets[],
                                 long           sizes[],
                                 void           *buffer)

DESCRIPTION
This method sets a real value hyper_cube with the specified type and size. The
void pointer would get casted to the appropriate type once it is used. The data_
type
argument will be used to ensure type compatibility with the hyper_cube data type
.
RETURN VALUE
miset_real_value_hyper_cube returns MI_NOERROR if it successfully sets
the hyper_cube with specified size and type and MI_ERROR otherwise
```

```
NAME
miget_voxel_value_hyper_cube − get a voxel value hyper_cube
SYNOPSIS
#include <minc.h>
int miget_voxel_value_hyper_cube( mihandle_t    volume,
                                  mitype_t      buffer_data_type,
                                  long          voxel_offsets[],
                                  long          sizes[],
                                  void          *buffer)

DESCRIPTION
This method returns a voxel value hyper_cube.
RETURN VALUE
miget_voxel_value_hyper_cube returns MI_NOERROR if it successfully gets
the hyper_cube with specified size and type and MI_ERROR otherwise
```



```
NAME
miset_voxel_value_hyper_cube − set a voxel value hyper_cube
SYNOPSIS
#include <minc.h>
int miset_voxel_value_hyper_cube( mihandle_t    volume,
                                  mitype_t      buffer_data_type,
                                  long          voxel_offsets[],
                                  long          sizes[],
                                  void          *buffer)

DESCRIPTION
This method sets a voxel value hyper_cube with the specified type and size. If
the type does not match a simple C cast will be applied.
RETURN VALUE
miset_voxel_value_hyper_cube returns MI_NOERROR if it successfully sets the
voxel value hyper_cube with specified size and type and MI_ERROR otherwise
```

```
*****************************************************************************
*****************************************************************************
                    IMAGE CONVERSION VARIABLE FUNCTIONS (12)
*****************************************************************************
*****************************************************************************
```



```
NAME
miicv_volume_attach − attach a MINC ICV object to a MINC 2.0 volume.
SYNOPSIS
#include <minc.h>
int miicv_volume_attach ( int           icv,
                          mihandle_t    volume);

DESCRIPTION
This function attaches a MINC image conversion variable (ICV) object
to a MINC 2.0 volume.
Given the flexibility of the MINC format, there are many different
possible choices available for details such as dimension order, data
type, and data range. Accounting for all of the possible combinations
of these items could make MINC programming too complex or unwieldy in
many situations.
MINC ICV objects are a solution to this problem.  They are essentially
a specification of the properties that the programmer would like the
data to have. The ICV is responsible for making any necessary
conversions, hiding the details of the actual data format from the
programmer.
A program may allocate any number of ICV objects and may configure and
attach them independently, allowing the program to have several
different views of the volume’s data at the same time.
Note that ICV properties cannot be modified while a variable is
attached to the ICV. If a file and variable are already attached to
the ICV, they will be automatically detached before the new variable
is attached.
NOTE
This interface is being extended to allow use with the new MINC 2.0
interface, however, the existing ICV interface will be retained as
well.
RETURN VALUE
MI_NOERROR on success, or MI_ERROR on failure.
```

```
NAME
miicv_volume_detach − detach an image conversion variable from a volume.
SYNOPSIS
#include <minc.h>
int miicv_volume_detach ( int   icv);

DESCRIPTION
Deletes the association between an image conversion variable (ICV)
and a MINC2.0 volume object.
RETURN VALUE
MI_NOERROR on success, or MI_ERROR on failure.
SEE ALSO
miicv_volume_attach
```



```
NAME
miicv_volume_get, miicv_volume_put
SYNOPSIS
#include <minc.h>
int miicv_volume_get(int                icv,
                     mihandle_t         volume,
                     const unsigned long start[],
                     const unsigned long count[],
                     void               *value_ptr);

int miicv_volume_put(int                icv,
                     mihandle_t         volume,
                     const unsigned long start[],
                     const unsigned long count[],
                     void               *value_ptr);

DESCRIPTION
These functions actually read or write data through the ICV to the attached
variable and volume.
All value/range conversions and dimension conversions are applied before the
data is read or written.
RETURN VALUES
MI_NOERROR on success, MI_ERROR on failure.
```

```
NAME
miicv_create − Create an image conversion variable and initialize
properties to their defaults
SYNOPSIS
#include <minc.h>
int miicv_create(void);
DESCRIPTION
RETURN VALUE
An ICV identifier greater than or equal to zero, or MI_ERROR if an
error occurs.
```



```
NAME
miicv_free − Free an image conversion variable
SYNOPSIS
#include <minc.h>
int miicv_free(int      icv);

DESCRIPTION
Frees an image conversion variable allocated by miicv_create
RETURN VALUE
MI_NOERROR on success, MI_ERROR on failure
```

```
NAME
miicv_setdbl, miicv_setint, miicv_setstr − Set ICV properties
SYNOPSIS
int miicv_setdbl(int         icv,
                 int         icv_prop_id,
                 double      value);
int miicv_setint(int         icv,
                 int         icv_prop_id,
                 int         value);
int miicv_setstr(int         icv,
                 int         icv_prop_id,
                 char        *value);

DESCRIPTION
Sets an ICV property to the requested value.  It is illegal to attempt
to set a numeric property to a string value or vice−versa, but any numeric
value may be set as either a floating−point or as an integer value.
RETURN VALUE
MI_NOERROR on success, MI_ERROR on failure
```



```
NAME
miicv_inqdbl, miicv_inqint, miicv_inqstr − Query ICV properties
SYNOPSIS
int miicv_inqdbl(int            icv,
                 int            icv_prop_id,
                 double         *value_ptr);
int miicv_inqint(int            icv,
                 int            icv_prop_id,
                 int            *value_ptr);
int miicv_inqstr(int            icv,
                 int            icv_prop_id,
                 char           *value);

DESCRIPTION
Query the value of an ICV property.  It is illegal to request the
value of numeric property as a string value or vice−versa, but any
numeric value may be requested in either floating−point or integer
format.
RETURN VALUE
MI_NOERROR on success, MI_ERROR on failure
```

```
*****************************************************************************
*************************LABEL FUNCTIONS (3)*********************************
*****************************************************************************
*****************************************************************************
```



```
NAME
midefine_label − define a label value for a labelled volume.
SYNOPSIS
#include <minc.h>
int midefine_label ( mihandle_t         volume,
                     int                value,
                     const char         *name );

DESCRIPTION
This function associates a label name with an integer value for the given
volume. Functions which read and write voxel values will read/write
in integer values, and must call miget_label_name() to discover the
descriptive text string which corresponds to the integer value.
RETURN VALUE
MI_NOERROR on success, or MI_ERROR on failure.
```

```
NAME
miget_label_name − convert a label type to a text string
SYNOPSIS
#include <minc.h>
int miget_label_name ( mihandle_t       volume,
                       int              value,
                       char             **name );

DESCRIPTION
For a labelled volume, this function retrieves the text name
associated with a given integer value.
The name pointer returned must be freed by calling mifree_name().
RETURN VALUE
MI_NOERROR on success, MI_ERROR on failure.
```



```
NAME
miget_label_value − translate a label name into the corresponding integer
value.
SYNOPSIS
#include <minc.h>
int miget_label_value ( mihandle_t      volume,
                        const char      *name,
                        int             *value_ptr );

DESCRIPTION
This function is the inverse of miget_label_name. It is called to determine
what integer value, if any, corresponds to the given text string.
RETURN VALUE
MI_NOERROR on success, MI_ERROR on failure.
```

```
*****************************************************************************
************************SLICE FUNCTIONS (8)**********************************
*****************************************************************************
*****************************************************************************
```



```
NAME
miget_slice_max − get  maximum real value of all the slice
SYNOPSIS
#include <minc.h>
int miget_slice_max ( midimhandle_t     dimensions[],
                      unsigned long     start_positions[],
                      unsigned long     array_length,
                      double            *slice_max)


DESCRIPTION
This method returns the slice_max with the minimum real value of
the corresponding slice. Note that this function is not defined for
floating point data type.
RETURN VALUE
miget_slice_max returns MI_NOERROR if it successfully returns the slice_max
or MIERROR otherwise
```

```
NAME
miset_slice_max − set maximum real value for the slice
SYNOPSIS
#include <minc.h>
int miset_slice_max ( midimhandle_t     dimensions[],
                      unsigned long     start_positions[],
                      unsigned long     array_length,
                      const double      slice_max)


DESCRIPTION
This method sets the slice_max with the minimum real value for
the corresponding slice. Note that this function is not defined for
floating point data type.
RETURN VALUE
miset_slice_max returns MI_NOERROR if it successfully sets the slice_max
or MIERROR otherwise
```



```
NAME
miget_slice_min − get the minimum real value of the slice
SYNOPSIS
#include <minc.h>
int miget_slice_min ( midimhandle_t     dimensions[],
                      unsigned long     start_positions[],
                      unsigned long     array_length,
                      double            *slice_min)


DESCRIPTION
This method returns the slice_min with the minimum real value of
the corresponding slice. Note that this function is not defined
for floating point data type.
RETURN VALUE
miget_slice_min returns MI_NOERROR if it successfully returns the slice_min
or MIERROR otherwise
```

```
NAME
miset_slice_min − set the minimum real value for all the slice
SYNOPSIS
#include <minc.h>
int miset_slice_min ( midimhandle_t     dimensions[],
                      unsigned long     start_positions[],
                      unsigned long     array_length,
                      const double      slice_min)


DESCRIPTION
This method sets the slice_min with the minimum real value for
the corresponding slice. Note that this function is not defined for
floating point data type.
RETURN VALUE
miset_slice_min returns MI_NOERROR if it successfully sets the slice_min
or MIERROR otherwise
```



```
NAME
miget_slice_range − get the min and max real values of the slice range
SYNOPSIS
#include <minc.h>
int miget_slice_range ( midimhandle_t   dimensions[],
                        unsigned long   start_positions[],
                        unsigned long   array_length,
                        double          *slice_min,
                        double          *slice_max)


DESCRIPTION
This method returns the slice range according to their minimum and maximum
real values . Note that this function is not defined for floating point
data type.
RETURN VALUE
miget_slice_range returns MI_NOERROR if it successfully returns min and
max or MIERROR otherwise
```

```
NAME
miset_slice_range − set the min and max real values of the slice range
SYNOPSIS
#include <minc.h>
int miset_slice_range ( midimhandle_t   dimensions[],
                        unsigned long   start_positions[],
                        unsigned long   array_length,
                        const double    slice_min,
                        const double    slice_max)


DESCRIPTION
This method sets the slice range according to the minimum and maximum
real values . Note that this function is not defined forfloating point
data type.
RETURN VALUE
miset_slice_range returns MI_NOERROR if it successfully sets the slice_max
and the slice_min or MIERROR otherwise
```



```
NAME
miget_slice_scaling_flag − get the scaling flag for slices
SYNOPSIS
#include <minc.h>
int miget_slice_scaling_flag ( Boolean      *scaling_flag)

DESCRIPTION
This method gets the scaling flag for slices which determines whether
the volume slices have different scale factors.
RETURN VALUE
miget_slice_scaling_flag returns MI_NOERROR if it successfully gets
the scaling flag or MIERROR otherwise
```

```
NAME
miset_slice_scaling_flag - set the scaling flag for slices
SYNOPSIS
#include <minc.h>
int miset_slice_scaling_flag ( mihandle_t       volume,
                               Boolean          scaling_flag)

DESCRIPTION
This method sets the scaling flag for slices which determines whether
the volume slices have different scale factors.
RETURN VALUE
miset_slice_scaling_flag returns MI_NOERROR if it successfully sets
the scaling flag or MIERROR otherwise
```



```
********************************************************************************
************************VALID MIN/MAX AND RANGE FUNCTIONS (7)****************
********************************************************************************
********************************************************************************
```

```
NAME
miget_valid_max − get the valid maximum value
SYNOPSIS
#include <minc.h>
int miget_valid_max ( mihandle_t        volume,
                      double            *valid_max)

DESCRIPTION
This method gets the valid maximum value specific to the data type.
RETURN VALUE
miget_valid_max returns MI_NOERROR if it successfully gets
the valid maximum or MIERROR otherwise
```



```
NAME
miset_valid_max − set the valid maximum value
SYNOPSIS
#include <minc.h>
int miset_valid_max ( mihandle_t        volume,
                      double            valid_max)

DESCRIPTION
This method sets the valid maximum value specific to the data type.
RETURN VALUE
miset_valid_max returns MI_NOERROR if it successfully sets
the valid maximum or MIERROR otherwise
```

```
NAME
miget_valid_min − get the valid minimum value
SYNOPSIS
#include <minc.h>
int miget_valid_min ( mihandle_t        volume,
                      double            *valid_min)

DESCRIPTION
This method gets the valid minimum value specific to the data type.
RETURN VALUE
miget_valid_min returns MI_NOERROR if it successfully gets
the valid minimum or MIERROR otherwise
```



```
NAME
miset_valid_min − set the valid minimum value
SYNOPSIS
#include <minc.h>
int miset_valid_min ( mihandle_t        volume,
                      double            valid_min)

DESCRIPTION
This method sets the valid minimum value specific to the data type.
RETURN VALUE
miset_valid_min returns MI_NOERROR if it successfully sets
the valid minimum or MIERROR otherwise
```

```
NAME
miget_valid_range − get the valid range values
SYNOPSIS
#include <minc.h>
int miget_valid_range ( mihandle_t      volume,
                        double          *valid_max,
                        double          *valid_min)

DESCRIPTION
This method gets the valid range values specific to the data type.
RETURN VALUE
miget_valid_range returns MI_NOERROR if it successfully gets
the valid range values or MIERROR otherwise
```



```
NAME
miset_valid_range − set the valid range values
SYNOPSIS
#include <minc.h>
int miset_valid_range ( mihandle_t      volume,
                        double          valid_max,
                        double          valid_min)

DESCRIPTION
This method sets the valid range values specific to the data type.
RETURN VALUE
miset_valid_range returns MI_NOERROR if it successfully sets
the valid range values or MIERROR otherwise
```

```
NAME
miset_valid_range_to_default − set the range to the default value
SYNOPSIS
#include <minc.h>
int miset_valid_range_to_default ( mihandle_t   volume)


DESCRIPTION
This method sets the valid range to the dafault value specific to the
data type.
RETURN VALUE
miset_valid_range_to_default returns MI_NOERROR if it successfully sets
the valid range to default values  or MIERROR otherwise
```



```
********************************************************************************
************************VOLUME FUNCTIONS (4)************************************

********************************************************************************
********************************************************************************
```

```
NAME
micreate_volume − create a volume with the specified properties
SYNOPSIS
#include <minc.h>
int micreate_volume ( const char        *filename,
                      int               number_of_dimensions,
                      midimhandle_t     dimensions[],
                      mitype_t          volume_type,
                      miclass_t         volume_class,
                      mivolumeprops_t   create_props)

DESCRIPTION
Create a volume with the specified filename, data type, dimension handles,
type, class and compression type.
RETURN VALUE
micreate_volume returns MI_NOERROR if it successfully creates a
volume with all the specified properties and MI_ERROR otherwise
```



```
NAME
miget_volume_dimension_count − get the number of dimensions defined in a MINC
volume, according to their class and attribute.
SYNOPSIS
#include <minc.h>
int miget_volume_dimension_count ( mihandle_t           volume,
                                   midimclass_t         class,
                                   midimattr_t          attr );

DESCRIPTION
This function may be used to determine the number of dimensions with the
given class and attributes.
RETURN VALUE
The number of dimensions defined in the file, or MI_ERROR if an error
is detected.
```

```
NAME
miopen_volume − open a volume for reading
SYNOPSIS
#include <minc.h>
int miopen_volume(const char    *filename,
                  mihandle_t    *volume);

DESCRIPTION
Opens an existing MINC volume for read−only access.
RETURN VALUE
Returns MI_NOERROR on success or MI_ERROR on failure.
```



```
NAME
miclose_volume − close an open volume, freeing the volume handle
SYNOPSIS
#include <minc.h>
int miclose_volume ( mihandle_t    volume);

DESCRIPTION
Close an existing MINC volume. If the volume was newly created, all
changes will be written to disk. In all cases this function closes
the open volume and frees memory associated with the volume handle.
RETURN VALUE
Returns MI_NOERROR on success or MI_ERROR on failure.
```

```
*****************************************************************************
************************VOXEL/REAL FUNCTIONS (6)*****************************
*****************************************************************************
*****************************************************************************
```



```
NAME
miconvert_real_to_voxel, miconvert_voxel_to_real  - conversion between
voxel and real values
SYNOPSIS
#include <minc.h>
int miconvert_real_to_voxel( mihandle_t                volume,
                             const unsigned long       location[],
                             int                       array_length,
                             double                    value,
                             double                    *voxel_ptr );

int miconvert_voxel_to_real ( mihandle_t               volume,
                              const unsigned long      location[],
                              int                      array_length,
                              double                   voxel,
                              double                   *value_ptr );

DESCRIPTION
These functions convert values between real (scaled) values and voxel
(unscaled) values.  The voxel value is the unscaled value, and
corresponds to the value actually stored in the file, whereas the
"real" value is the value at the given location after scaling has been
applied.
RETURN VALUE
Returns MI_NOERROR on success, MI_ERROR on failure.
```

```
NAME
miget_real_value -  return a specific scaled value from a volume
SYNOPSIS
#include <minc.h>
int miget_real_value ( mihandle_t              volume,
                       const unsigned long     location[],
                       int                     array_length,
                       double                  *value_ptr );

DESCRIPTION
This function retrieves the real values of a position in the
MINC volume.  The "real" value is the value at the given location
after scaling has been applied.
RETURN VALUE
Returns MI_NOERROR on success, MI_ERROR on failure.
```



```
NAME
miset_real_value - set the scaled value of a particular position
in the MINC volume.
SYNOPSIS
#include <minc.h>
int miset_real_value( mihandle_t               volume,
                      const unsigned long      location[],
                      int                      array_length,
                      double                   value );

DESCRIPTION
This function sets the  real value of a position in the MINC
volume. The "real" value is the value at the given location
after scaling has been applied.
RETURN VALUE
Returns MI_NOERROR on success, MI_ERROR on failure.
```

```
NAME
miget_voxel_value  − return a specific unscaled value from a given volume
SYNOPSIS
#include <minc.h>
int miget_voxel_value ( mihandle_t              volume,
                        const unsigned long     location[],
                        int                     array_length,
                        double                  *voxel_ptr );

DESCRIPTION
This function retrieves the real values of a position in the
MINC volume. The voxel value is the unscaled value, and corresponds
to the value actually stored in the file.
RETURN VALUE
Returns MI_NOERROR on success, MI_ERROR on failure.
```



```
NAME
miset_voxel_value − set the unscaled value of a particular position
in the MINC volume.
SYNOPSIS
#include <minc.h>
int miset_voxel_value( mihandle_t               volume,
                       const unsigned long      location[],
                       int                      array_length,
                       double                   voxel );

DESCRIPTION
This function sets the voxel value of a position in the MINC
volume.  The voxel value is the unscaled value, and corresponds to the
value actually stored in the file.
RETURN VALUE
Returns MI_NOERROR on success, MI_ERROR on failure.
```

```
*****************************************************************************
*****************************************************************************
                        VOLUME PROPERTIES FUNCTIONS (11)
*****************************************************************************
*****************************************************************************
```



```
NAME
minew_volume_props, mifree_volume_props − create or destroy a volume
property list.
SYNOPSIS
#include <minc.h>
int minew_volume_props(mivolumeprops_t          *props);
int mifree_volume_props(mivolumeprops_t          props);

DESCRIPTION
MINC volume properties objects are used to set or query the state of a number
of the internal parameters of the volume.
RETURN VALUE
```

```
NAME
miget_volume_props − Get a copy of the property list of a volume.
SYNOPSIS
#include <minc.h>
int miget_volume_props(mihandle_t              volume,
                       mivolumeprops_t         *props);

DESCRIPTION
Returns a copy of the properties associated with the volume.  Any
changes made to the properties will not take effect on the original volume,
but the resulting properties structure may be used in the creation of a new
volume.
The handle must be freed by calling mifree_volume_props().
RETURN VALUE
MI_NOERROR on success, MI_ERROR on failure
```



```
NAME
miset_props_multi_resolution, miget_props_multi_resolution − get or set the
multi−resolution properties for a property list.
SYNOPSIS
#include <minc.h>
int miset_props_multi_resolution(mivolumeprops_t        props,
                                 BOOLEAN                enable_flag,
                                 int                    thumbnail_depth);
int miget_props_multi_resolution(mivolumeprops_t        props,
                                 BOOLEAN                *enable_flag,
                                 int                    *thumbnail_depth);

DESCRIPTION
Returns the multi−resolution properties for a property list.
RETURN VALUE
Returns MI_NOERROR on success, MI_ERROR on failure
```

```
NAME
miset_props_compression_type, miget_props_compression_type − get or set
the compression type for a volume property list.
SYNOPSIS
#include <minc.h>
int miset_props_compression_type(mivolumeprops_t        props,
                                 micompression_t        compression_type);

int miget_props_compression_type(mivolumeprops_t        props,
                                 micompression_t        *compression_type);

DESCRIPTION
Set or retrieve the compression type, if any, for the volume properties.
Currently only two compression types are defined:
Enabling compression will automatically enable blocking with default
parameters (see miset_props_blocking).
typedef enum {
        MI_COMPRESS_NONE = 0,
        MI_COMPRESS_ZLIB = 1
} micompression_t ;
RETURN VALUE
MI_NOERROR on success, or MI_ERROR on failure.
```



```
NAME
miset_props_zlib_compression, miget_props_zlib_compression − get or
set the zlib compression properties for a volume property list.
SYNOPSIS
#include <minc.h>
int miset_props_zlib_compression(mivolumeprops_t        props,
                                 int                    zlib_level);
int miget_props_zlib_compression(mivolumeprops_t        props,
                                 int                    *zlib_level);

DESCRIPTION
Get or set the Zlib compression level for the volume properties.  The
compression level is an integer from 1 to 9.
RETURN VALUE
MI_NOERROR on success, or MI_ERROR on failure.
```

```
NAME
miset_props_blocking, miget_props_blocking − get or set the blocking
structure properties for a volume property list.
SYNOPSIS
#include <minc.h>
int miset_props_blocking(mivolumeprops_t        props,
                         int                    edge_count,
                         const int              *edge_lengths);

int miget_props_blocking(mivolumeprops_t        props,
                         int                    *edge_count,
                         int                    *edge_lengths,
                         int                    max_lengths);

DESCRIPTION
Gets or sets the block−structuring properties of a volume property
list.  If this option is set on a MINC volume, image data will be
stored in a series of 2D or 3D chunks rather than as a simple linear
array.
This option is enabled implicitly whenever compression is enabled − all
compressed volumes must be block−structured.
RETURN VALUE
Returns MI_NOERROR on success, MI_ERROR on failure
```