Table 5.6.2.6. *Formal parameters for high-level CBFlib functions*

array	Pointer to image array data
axis_id	Axis ID
center1	Displacement along the slow axis
center2	Displacement along the fast axis
compression	Compression type
coordinate1	*x* component
coordinate2	*y* component
coordinate3	*z* component
crystal_id	ASCII crystal ID
day	Timestamp day (1–31)
detector	Detector handle
diffrn_id	ASCII diffraction ID
distance	Distance
div_x_source	Value of **_diffrn_radiation.div_x_source**
div_x_y_source	Value of **_diffrn_radiation.div_x_y_source**
div_y_source	Value of **_diffrn_radiation.div_y_source**
element_id	ASCII element ID
element_number	Detector element counting from 0
elements	Count of elements
elsigned	Set to nonzero if the destination array elements are signed
elsize	Size in bytes of each destination array element
file	File descriptor
gain	Detector gain in counts per photon
gain_esd	Gain e.s.d. value
goniometer	Goniometer handle
handle	CBF handle
hour	Timestamp hour (0–23)
increment	Increment value
index1	Slow index
index2	Fast index
initial1	*x* component of the initial vector
initial2	*y* component of the initial vector
initial3	*z* component of the initial vector
minute	Timestamp minute (0–59)
month	Timestamp month (1–12)
ndim1	Slow array dimension
ndim2	Fast array dimension
normal1	*x* component of the normal vector
normal2	*y* component of the normal vector
normal3	*z* component of the normal vector
overload	Overload value
polarizn_source_norm	Polarization normal
polarizn_source_ratio	Polarization ratio
precision	Timestamp precision in seconds
projected_area	Apparent area in mm^2
ratio	Goniometer setting (0 = beginning of exposure, 1 = end)
real1	*x* component of the real-space vector
real2	*y* component of the real-space vector
real3	*z* component of the real-space vector
reciprocal1	*x* component of the reciprocal-space vector
reciprocal2	*y* component of the reciprocal-space vector
reciprocal3	*z* component of the reciprocal-space vector
reserved	Unused; any value other than 0 is invalid
second	Timestamp second (0–60.0)
start	Start value
time	Timestamp in seconds since 1 January 1970 or integration time in seconds
timezone	Time zone difference from universal time in minutes or CBF_NOTIMEZONE
vector1	*x* component of the rotation axis
vector2	*y* component of the rotation axis
vector3	*z* component of the rotation axis
wavelength	Wavelength in Å
year	Pointer to the destination timestamp year

The values of the divergence parameters, represented by the data names **_diffrn_radiation.div_x_source**, ***.div_y_source** and ***.div_x_y_source**, can be retrieved or set. The functions

```
int cbf_get_divergence (cbf_handle handle,
     double *div_x_source, double *div_y_source,
     double *div_x_y_source);
int cbf_set_divergence (cbf_handle handle,
     double div_x_source, double div_y_source,
     double div_x_y_source);
```

operate on the DIFFRN_RADIATION category.

The values of **_diffrn.id** and **_diffrn.crystal_id** can be retrieved or set:

```
int cbf_get_diffrn_id (cbf_handle handle,
     const char **diffrn_id);
int cbf_set_diffrn_id (cbf_handle handle,
     const char *diffrn_id);
int cbf_get_crystal_id (cbf_handle handle,
     const char **crystal_id);
int cbf_set_crystal_id (cbf_handle handle,
     const char *crystal_id);
```

Changing **_diffrn.id** also modifies the corresponding ***.diffrn_id** entries in the DIFFRN_SOURCE, DIFFRN_RADIATION, DIFFRN_DETECTOR and DIFFRN_MEASUREMENT categories.

The starting value and increment of an axis may be retrieved or set:

```
int cbf_get_axis_setting (cbf_handle handle,
     unsigned int reserved, const char *axis_id,
     double *start, double *increment);
int cbf_set_axis_setting (cbf_handle handle,
     unsigned int reserved, const char *axis_id,
     double start, double increment);
```

The `cbf_set_axis_setting` call is used during the creation of a CBF/imgCIF file to store the goniometer settings and rotation. The `cbf_get_axis_setting` is not generally useful when interpreting a file as there are no standard identifiers and the arrangement of the experimental axes is not consistent. Much more useful are the goniometer geometry functions described below.

The number of detector elements can be retrieved:

```
int cbf_count_elements (cbf_handle handle,
     unsigned int *elements);
```

This is the number of rows in the DIFFRN_DETECTOR_ELEMENT category. For each element, counting from 0, the detector identifier (the ***.detector_id** entry) can be retrieved and the gain and overload values in the ARRAY_INTENSITIES category retrieved or set:

```
int cbf_get_element_id (cbf_handle handle,
     unsigned int element_number,
     const char **element_id);
int cbf_get_gain (cbf_handle handle,
     unsigned int element_number,
     double *gain, double *gain_esd);
int cbf_set_gain (cbf_handle handle,
     unsigned int element_number, double gain,
     double gain_esd);
int cbf_get_overload (cbf_handle handle,
     unsigned int element_number, double *overload);
int cbf_set_overload (cbf_handle handle,
     unsigned int element_number, double overload);
```

For each element, counting from 0, the values of the parameters of the detector can be retrieved and some can be set. The value of **_diffrn_detector_element.id** is retrieved as *element_id*. The value of **_diffrn_data_frame.array_id** can be retrieved as *array_name*. The values of **_array_intensities.gain** and **_array_intensities.gain_esd** are retrieved as *gain*