

H5TB_MAKE_TABLE

[Expand all](#) [Collapse all](#)

- [Jump to ...](#)
- [Summary](#)
- [Description](#)
- [Example](#)
- [Switch language ...](#)
- [C](#)
- [C++](#)
- [FORTRAN](#)
- [JAVA](#)

[Summary](#)
[Description](#)
[Example](#)
[JAVA](#)
[FORTRAN](#)
[C++](#)
[C](#)

H5TB_MAKE_TABLE

Creates and writes a table

Procedure:

H5TB_MAKE_TABLE (table_title, loc_id, dset_name, nfields, nrecords, type_size, field_names, field_offset, field_types, chunk_size, fill_data, compress, data)

Signature:

```
herr_t H5Tmake_table( const char *table_title, hid_t loc_id, const char *dset_name, hsize_t nfields,  
                    const hsize_t nrecords, size_t type_size, const char *field_names [ ], const size_t *field_offset,  
                    const hid_t *field_types, hsize_t chunk_size, void *fill_data, int compress, const void *data )
```

```

subroutine h5tbmake_table_f(table_title, loc_id, dset_name, nfields, &
                          nrecords, type_size, field_names, field_offset, &
                          field_types, chunk_size, compress, errcode)

  implicit none
  character(LEN=*), intent(IN) :: table_title      ! name of the table
  integer(HID_T), intent(IN) :: loc_id            ! file or group identifier
  character(LEN=*), intent(IN) :: dset_name       ! name of the dataset
  integer(HSIZE_T), intent(IN) :: nfields        ! fields
  integer(HSIZE_T), intent(IN) :: nrecords       ! records
  integer(SIZE_T), intent(IN) :: type_size      ! type size
  character(LEN=*), dimension(nfields), intent(IN) :: field_names
                                                ! field names
  integer(SIZE_T), dimension(nfields), intent(IN) :: field_offset
                                                ! field offset
  integer(HID_T), dimension(nfields), intent(IN) :: field_types
                                                ! field types

  integer(HSIZE_T), intent(IN) :: chunk_size     ! chunk size
  integer, intent(IN) :: compress               ! compress
  integer :: errcode                           ! error code
end subroutine h5tbmake_table_f

```

Note: h5tbmake_table_f only creates the table, it does not write data to it.

Parameters:

<i>const char</i> *table_title	IN: The title of the table
<i>hid_t</i> loc_id	IN: Identifier of the file or group to create the table within
<i>const char</i> *dset_name	IN: The name of the dataset to create
<i>hsize_t</i> nfields	IN: The number of fields
<i>const hsize_t</i> nrecords	IN: The number of records
<i>hsize_t</i> type_size	IN: The size in bytes of the structure associated with the table This value is obtained with <code>sizeof</code> .
<i>const char</i> *field_names[]	IN: An array containing the names of the fields
<i>const size_t</i> *field_offset	IN: An array containing the offsets of the fields
<i>const hid_t</i> *field_types	IN: An array containing the type of the fields
<i>hsize_t</i> chunk_size	IN: The chunk size
<i>void</i> *fill_data	IN: Fill values data
<i>int</i> compress	IN: Flag that turns compression on or off
<i>const void</i> *data	IN: Buffer with data to be written to the table

Description:

H5TB_MAKE_TABLE creates and writes a dataset named `dset_name` attached to the object specified by the identifier `loc_id`.

Returns:

Returns a non-negative value if successful; otherwise returns a negative value.

Example:

```
typedef struct Particle
{
    char    name[16];
    int     lati;
    int     longi;
    float   pressure;
    double  temperature;
} Particle;

Particle  dst_buf[NRECORDS];

/* Calculate the size and the offsets of our struct members in memory */
size_t dst_size = sizeof( Particle );
size_t dst_offset[NFIELDS] = { HOFFSET( Particle, name ),
                               HOFFSET( Particle, lati ),
                               HOFFSET( Particle, longi ),
                               HOFFSET( Particle, pressure ),
                               HOFFSET( Particle, temperature )};

size_t dst_sizes[NFIELDS] = { sizeof( dst_buf[0].name),
                              sizeof( dst_buf[0].lati),
                              sizeof( dst_buf[0].longi),
                              sizeof( dst_buf[0].pressure),
                              sizeof( dst_buf[0].temperature)};

/* Define an array of Particles */
Particle p_data[NRECORDS] = {
    {"zero",0,0, 0.0f, 0.0},
    {"one",10,10, 1.0f, 10.0},
    {"two", 20,20, 2.0f, 20.0},
    {"three",30,30, 3.0f, 30.0},
    {"four", 40,40, 4.0f, 40.0},
    {"five", 50,50, 5.0f, 50.0},
    {"six", 60,60, 6.0f, 60.0},
    {"seven",70,70, 7.0f, 70.0}
};

/* Define field information */
const char *field_names[NFIELDS] =
    { "Name","Latitude", "Longitude", "Pressure", "Temperature" };
hid_t      field_type[NFIELDS];
hid_t      string_type;
hid_t      file_id;
hsize_t    chunk_size = 10;
int        *fill_data = NULL;
int        compress = 0;
int        i;

/* Initialize field_type */
string_type = H5Tcopy( H5T_C_S1 );
H5Tset_size( string_type, 16 );
field_type[0] = string_type;
field_type[1] = H5T_NATIVE_INT;
```

```
field_type[2] = H5T_NATIVE_INT;
field_type[3] = H5T_NATIVE_FLOAT;
field_type[4] = H5T_NATIVE_DOUBLE;

/* Create a new file using default properties. */
file_id = H5Fcreate( "ex_table_01.h5", H5F_ACC_TRUNC, H5P_DEFAULT, H5P_DEFAULT );

/*-----
 * H5Tbmake_table
 *-----
 */
```

```
H5TBmake_table( "Table Title", file_id, TABLE_NAME,NFIELDS,NRECORDS,  
                dst_size,field_names, dst_offset, field_type,  
                chunk_size, fill_data, compress, p_data );
```

Please see [Tables](#) for Fortran programming hints.

```
hl / fortran / test / tsttable.F90 [694:700]
```

```
hdf5_1_12  H5FFV/hdf5
```

```
f_ptr1 = C_LOC(p_data(1)%name(1:1))  
f_ptr2 = C_NULL_PTR
```

```
CALL h5tbmake_table_f("Table Title",file_id, table_name, nfields, nrecords, &  
    dst_size, field_names, dst_offset, field_type, &  
    chunk_size, f_ptr2, compress, f_ptr1, errcode )
```

--- Last Modified: June 25, 2020 | 02:53 PM