J. Appl. Cryst. (1993). **26**, 482–494

CIF Applications. IV. *CIFtbx*: a Tool Box for Manipulating CIFs*

BY SYDNEY R. HALL

Crystallography Centre, University of Western Australia, Nedlands 6009, *Australia*

(*Received* 12 *December* 1992; *accepted* 15 *January* 1993)

Abstract

CIFtbx is a subroutine library which provides simple commands for reading and writing CIF data. This library is referred to as the *CIF* (*software*) *tool box*. *CIFtbx* routines may be applied to any Fortran program. The library is written in Fortran77 and may be installed without modification on most computers. *CIFtbx* is public-domain software.

Introduction

This paper is part of a series on CIF applications. Other programs in this series are *QUASAR* (Hall & Sievers, 1993) for generating a requested CIF from existing CIF data; *CIFIO* (Hall, 1993*a*) for reading and writing CIFs in a custom application and *CYCLOPS* (Hall, 1993*b*) for validating CIF data names in an ASCII file. The purpose of the *CIFtbx* software is similar to *CIFIO*, *i.e.* it may be used to convert ASCII CIF data into an internal binary representation and the reverse. It differs in that it is designed for use with external software applications.

CIFtbx is intended for use by programmers developing software to access and generate CIFs. The application of *CIFtbx* requires only a rudimentary knowledge of the CIF syntax. Typically, a single *CIFtbx* command may be used to move a data item to or from a local program variable. No prior knowledge of the CIF content or structure is needed. The recovery of looped data, text lines or looped text packets is easily controlled by the programmer. In addition, the *CIFtbx* routines automatically check the structure of the input and output CIFs. It is anticipated that *CIFtbx* will make access to data stored in CIF format easier than for any other format.

In addition to data access commands, *CIFtbx* will validate data against one or more CIF dictionaries. This feature is useful for checking data conformance (using standard dictionaries such as `cifdic.C91`) or for local customized applications such as *CIFIO* (Hall, 1993*a*).

CIFtbx overview

The *CIFtbx* routines are applied in the same way as Fortran library functions. The *CIFtbx* commands are identified by names that have a trailing underline character. Here is a brief summary of the *CIFtbx* commands. Additional control variables are detailed in the next section.

General

`init_`	initialize input/output device numbers
`dict_`	use dictionary to validate data names

Read CIF data

`open_`	open input CIF
`data_`	select data block containing requested data
`test_`	get data type and loop number of item
`name_`	get data name of next data item
`numb_`	get numerical data item and its e.s.d.
`char_`	get character string or text data item

Write CIF data

`pfile_`	open output CIF
`pdata_`	output data block line
`ploop_`	output data name in a `loop_` structure
`pnumb_`	output data name and number with its e.s.d.
`pchar_`	output data name and a character string
`ptext_`	output data name and text line
`close_`	close output CIF.

The *CIFtbx* commands are in three categories: *general* commands, which apply to reading and writing CIFs; commands to *read* data from a CIF; and commands to *write* CIF data. Note that the read and write commands are logically independent and may be applied simultaneously to copy and update CIFs.

Most of the *CIFtbx* routines are defined as Fortran LOGICAL functions. This means that on invocation they are returned with a value of either *true* or *false* depending on whether the invocation is successful or unsuccessful. For example, the function `open_` is returned as *true* if the input CIF exists and has been opened, and is returned as *false* if the CIF cannot be opened. It is the responsibility of the programmer to test these functions and take action appropriate to the truth state.

The *CIFtbx* commands will now be described in detail. To assist in this description, a listing of a test Fortran application is provided in Table 1. This program contains four separate examples of how the commands are applied. These are trivial applications but they adequately illustrate how these commands may be applied for more complex situations. Other files used with this test application are an input CIF named `test.cif` (see Table 2); an input request file `test.req` (see Table 3); the output CIF `test.new` (see Table 4) and the output listing `test.lst` (see Table 5).

* This paper is one of a series of papers on CIF applications. Offprints are available from The Technical Editor, 5 Abbey Square, Chester CH1 2HU, England. See text of paper for availability of program(s) by email.

0021-8898/93/030482-13$06.00

Table 1. *An example application of the CIFtbx tools*

```
C            CIF Tool Box Application 'tbx_ex.f'
C            -----------------------
C
             include                          'ciftbx.f'
C
             include                          'ciftbx.cmn'
C
             logical                          f1,f2,f3
             character*4                      type
             character*32                     name
             character*80                     line
             character*4                      label(6)
             character*26                     alpha
             real                             cela,celb,celc,siga,sigb,sigc
             real                             x,y,z,u,sx,sy,sz,su
             real                             numb,sdev,dum
             real                             xf(6),yf(6),zf(6),uij(6,6)
             integer                          i,j,nsite
             data alpha/'abcdefghijklmnopqrstuvwxyz'/
C
C                    ........................ Example 1 ..................................
C
C   This example illustrates how to extract non-loop and loop items. Note carefully how the logical functions
C   numb_ and char_ signal if the request has been successful or not. Note how the logical variables text_ and
C   loop_ are used to control the text lines and the data loops.
C
C    .......  Assign the CIFtbx files
C
             f1 = init_( 1, 2, 3, 6 )
C
C    .......  Request dictionary validation check
C
             if(dict_('cifdic.C91','valid')) goto 100
             write(6,'(/a/)')                 ' Requested Core dictionary not present'
C
C    .......  Open the CIF to be accessed
C
100          name='test.cif'
             write(6,'(/2a/)')                ' Read data from CIF ',name
             if(open_(name))                  goto 120
             write(6,'(a///)')                ' >>>>>>>>> CIF cannot be opened'
             stop
C
C    .......  Assign the data block to be accessed
C
120          if(data_(' '))                   goto 130
             write(6,'(/a/)')                 ' >>>>>>> No data_ statement found'
             stop
130          write(6,'(/a,a/)')               ' Access items in data block ',bloc_
C
C    .......  Extract some cell dimensions; test all is OK
C
             f1 = numb_('_cell_length_a', cela, siga)
             f2 = numb_('_cell_length_b', celb, sigb)
             f3 = numb_('_cell_length_c', celc, sigc)
             if(.not.(f1.and.f2.and.f3))
         *   write(6,'(a)')                   ' Cell dimension(s) missing!'
             write(6,'(/a,3f10.4)')           ' Cell ',cela,celb,celc
             write(6,'(a,3f10.4/)')           '      ',siga,sigb,sigc
C
C    .......  Extract space group notation (expected char string)
C
             f1 = char_('_symmetry_space_group_name_Hall', name)
             write(6,'(a,a/)')                ' Space group ',name(1:long_)
C
C    .......  List the audit record (possible text line sequence)
C
             write(6,'(a/)')                  ' Audit record'
140          f1 = char_('_audit_update_record', line)
             write(6,'(a)')                   line
             if(text_)                        goto 140
C
C    .......  Extract atom site data in a loop
C
             write(6,'(/a/)')                 ' Atom sites'
160          f1 = char_('_atom_site_label', name)
             f2   numb_('_atom_site_fract_x', x, sx)
             f2   numb_('_atom_site_fract_y', y, sy)
```

Table 1 (*cont.* 1)

```
              f2 = numb_('_atom_site_fract_z', z, sz)
              f3 = numb_('_atom_site_U_iso_or_equiv', u, su)
              write(6,'(1x,a4,8f8.4)')          name,x,y,z,u,sx,sy,sz,su
              if(loop_)                         goto 160
C
C                     ........................... Example 2 ....................................
C
C   In this example, two separate data blocks are accessed. The first contains looped publication authors and text
C   addresses. The second part of this example shows how data from two different loops may be merged. Data items
C   from different loops may NOT be accessed simultaneously as this causes the CIFtbx loop counters to be reset to
C   the start of the loop (see Example 3).
C
C    .......  List the author addresses from publication data block
C
              if(data_('publication'))
        *     write(6,'(//a,a/)')               ' Access items in data block ',bloc_
              write(6,'(/a)')                   ' Author list'
C
210           f1 = char_('_publ_author_name', line)
              write(6,'(/1x,a)')                line(1:long_)
C
220           f1 = char_('_publ_author_address', line)
              if(line(1:10).eq.'      ')        goto 230
              write(6,'(1x,a)')                 line(1:50)
230           if(text_)                         goto 220
              if(loop_)                         goto 210
C
C    .......  Read and store the atom site data from other data block
C
              f1 = data_('mumbo_jumbo')
              write(6,'(///a,a/)')              ' Access items in data block ',bloc_
C
              nsite = 0
240           nsite = nsite+1
              f1 = char_('_atom_site_label', label(nsite))
              f2 = numb_('_atom_site_fract_x', xf(nsite), sx)
              f2 = numb_('_atom_site_fract_y', yf(nsite), sy)
              f2 = numb_('_atom_site_fract_z', zf(nsite), sz)
              do 250 i=1,6
250           uij(nsite,i)=0.0
              if(loop_)                         goto 240
C
C    .......  Read the Uij loop and store in the site list
C
260           f1 = char_('_atom_site_aniso_label', name)
              do 270 i=1,nsite
              if(label(i).eq.name)              goto 280
270           continue
              write(6,'(a)')                    ' Label mismatch between atom lists'
280           f1 = numb_('_atom_site_aniso_U_11', uij(i,1), dum)
              f1 = numb_('_atom_site_aniso_U_22', uij(i,2), dum)
              f1 = numb_('_atom_site_aniso_U_33', uij(i,3), dum)
              f1 = numb_('_atom_site_aniso_U_12', uij(i,4), dum)
              f1 = numb_('_atom_site_aniso_U_13', uij(i,5), dum)
              f1 = numb_('_atom_site_aniso_U_23', uij(i,6), dum)
              if(loop_)                         goto 260
C
C    .......  List the atom site data
C
              write(6,'(/a/)')                  ' Atom coordinates and Uij'
              do 290 i=1,nsite
              if(uij(i,1).gt.0.0001)            goto 285
              write(6,'(1x,a,3f8.4)')           label(i),xf(i),yf(i),zf(i)
              goto 290
285           write(6,'(1x,a,9f8.4)')           label(i),xf(i),yf(i),zf(i),
        *                                       (uij(i,j),j=1,6)
290           continue
C
C                     ........................... Example 3 ....................................
C
C   This example serves to illustrate how a general list of data requests may be handled. The logical function
C   test_ is used to identify the nature of the requested data item and then numb_ and char_ are invoked when
C   applicable. The supplied list of requests on 'test.req' is not of particular significance. The requests are
C   intentionally jumbled up to show what happens if a non-loop item is called within a loop. [WARNING: CIFtbx
C   interprets this as a signal to end the loop and the next call for a loop item will extract data from its first
C   packet! Look at the output listing to see what happens.]
```

Table 1 (*cont.* 2)

```
C
C    .......  Loop over the data request file
C
300           read(5,'(a)',end=400)              name
C
              f1 = test_(name)
              write(6,'(/a,3x,a,i5)')            name,type_,long_
C
              if(type_.ne.'numb')                goto 320
              f1 = numb_(name, numb, sdev)
              write(6,'(2f10.4)')                numb,sdev
              goto 300
C
320           if(type_.ne.'char')                goto 340
              f1 = char_(name, line)
              write(6,'(a)')                     line(1:long_)
              goto 300
C
340           if(type_.ne.'text')                goto 300
350           f1 = char_(name, line)
              write(6,'(a)')                     line
              if(text_)                          goto 350
              goto 300
C
C                     ........................... Example 4 ....................................
C
C   In this example, a new CIF is created. Note that it will not overwrite an existing CIF of the same name. Note
C   also that reading an existing CIF and writing a new CIF is possible at the same time, so that it is feasible
C   to use these tools to update or modify an existing CIF.
C
C    .......  Open a new CIF
C
400           if(pfile_('test.new'))             goto 450
              write(6,'(//a/)')                  ' Output CIF by this name exists already!'
              goto 500
C
C    .......  Insert a data block code
C
450           f1 = pdata_('whoops_a_daisy')
C
C    .......  Enter various single data items to show how
C
              f1 = pchar_('_audit_creation_method','using CIFtbx')
              f1 = pchar_('_audit_creation_extra1','using_CIFtbx')
              f1 = pchar_('_audit_creation_extra2',"Terry O'Connell")
              f1 = pchar_('_audit_creation_extra3','Terry O"Connell')
C
              f1 = ptext_('_audit_creation_record',' Text data may be ')
              f1 = ptext_('_audit_creation_record',' entered like this')
              f1 = ptext_('_audit_creation_record',' or in a loop.')
C
              f1 = pnumb_('_cell_measurement_temperature', 293., 0.)'
              f1 = pnumb_('_cell_volume', 1759.0, 13.)
              f1 = pnumb_('_cell_length_junk', 8.753535535243l3,0.)
              f1 = pnumb_('_cell_length_c', 19.737, .003)
C
C    .......  Enter some looped data
C
              f1 = ploop_('_atom_type_symbol')
              f1 = ploop_('_atom_type_oxidation_number')
              f1 = ploop_('_atom_type_number_in_cell')
              do 470 i=1,10
              f1 = pchar_(' ',alpha(1:i))
              f1 = pnumb_(' ',float(i),float(i)*0.1)
470           f1 = pnumb_(' ',float(i)*8.64523,0.)
C
C    .......  Do it again but as contiguous data with text data
C
              f1 = ploop_('_atom_type_symbol')
              f1 = ploop_('_atom_type_oxidation_number')
              f1 = ploop_('_some_silly_text')
              do 480 i=1,3
              f1 = pchar_(' ',alpha(1:i))
              1 = pnumb_(' ',float(i),float(i)*0.1)
480           f1 = ptext_(' ',' Hi Ho the diddly oh!')
C
500           call close_
              stop
              end
```

Table 2. *The test CIF input to the program* `tbx_ex.f`

```
data_mumbo_jumbo

_audit_creation_date                                    91-03-20
_audit_creation_method                                  from_xtal_archive_file_using_CIFIO
_audit_update_record
;  91-04-09     text and data added by Tony Willis.
   91-04-15     rec'd by co-editor with diagram as manuscript HL7.
   91-04-17     adjustments based on first referee's report.
   91-04-18     adjustments based on second referee's report.
;

_chemical_name_systematic
     trans-3-Benzoyl-2-(tert-butyl)-4-(iso-butyl)-1,3-oxazolidin-5-one
_chemical_formula_moiety                                'C18 H25 N O3'
_chemical_formula_sum                                   'C18 H25 N O3'
_chemical_formula_weight                                303.40
_chemical_melting_point                                 ?

####_cell_length_a                                      5.959(1)
_cell_length_b                                          14.956(1)
_cell_length_c                                          19.737(3)
_cell_angle_alpha                                       90
_cell_angle_beta                                        90
_cell_angle_gamma                                       90
_cell_volume                                            1759.0(3)
_cell_formula_units_Z                                   4
_cell_measurement_temperature                           293
_cell_measurement_reflns_used                           25
cell_measurement_theta_min                              25
_cell_measurement_theta_max                             31

_symmetry_cell_setting                                  orthorhombic
_symmetry_space_group_name_H-M                          'P 21 21 21'
_symmetry_space_group_name_Hall                         P_2ac_2ab

loop_
_atom_type_symbol
_atom_type_oxidation_number
_atom_type_number_in_cell
_atom_type_scat_dispersion_REAL                         #<< capitals to test case insensitivity
_atom_type_scat_dispersion_imag
_atom_type_scat_source
S    0    6     .319     .557     'Int Tab Vol III p202 Tab. 3.3.1a'
O    0    6     .047     .032     'Cromer,D.T. & Mann,J.B. 1968 AC A24,321.'
C    0    20    .017     .009     'Cromer,D.T. & Mann,J.B. 1968 AC A24,321.'
RU   0    1     -.105    3.296    'Cromer,D.T. & Mann,J.B. 1968 AC A24,321.'

loop_
_atom_site_label
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_U_iso_or_equiv
_atom_site_thermal_displace_type
_atom_site_calc_flag
_atom_site_calc_attached_atom
_atom_site_type_symbol
s     .20200     .79800     .91667     .030(3)     Uij     ?     ?     s
o     .49800     .49800     .66667     .02520      Uiso    ?     ?     o
c1    .48800     .09600     .03800     .03170      Uiso    ?     ?     c

loop_
_atom_site_aniso_label
_atom_site_aniso_U_11
_atom_site_aniso_U_22
_atom_site_aniso_U_33
_atom_site_aniso_U_12
_atom_site_aniso_U_13
_atom_site_aniso_U_23
_atom_site_aniso_type_symbol
s .035(4) .025(3) .025(3) .013(1) .00000 .00000 s

loop_ _blat1 _blat2 1 2 3 4 5 6 a b c d 7 8 9 0
```

Table 2 (*cont.*)

```
data_publication

loop_
_publ_author_name                                    #ActaC
_publ_author_address                                 #ActaC

   'Furber, Mark'
;
Research School of Chemistry
Australian National University
GPO Box 4
Canberra, A.C.T.
Australia 2601
;
   'Mander, Lewis N.'
;
Research School of Chemistry
Australian National University
GPO Box 4
Canberra, A.C.T.
Australia 2601
;
   'Patrick, Graham L.'
;
Research School of Chemistry
Australian National University
GPO Box 4
Canberra, A.C.T.
Australia 2601
;
   'Willis, Anthony C.'
;
Research School of Chemistry
Australian National University
GPO Box 4
Canberra, A.C.T.
Australia 2601
;
```

CIFtbx tools

Here is detailed description of the *CIFtbx* tool box. Each command and control variable has been applied in the test application shown in Tables 1 to 5. Use these tables as a guide when reading the descriptions below.

`init_`

A logical function for setting the device numbers of the four *CIFtbx* files. This is an optional function that is always returned with a value of *true*. It need be invoked only if the default device numbers must be changed. The arguments are:

input CIF dev number Set input CIF device (default=1)

output CIF dev number Set output CIF device (default=2)

direct access dev number Set formatted scratch device number (default=3)

error dev number Set error message device (default=6).

`dict_`

A logical function which requests a CIF dictionary to be used for data validation. The function is returned as *true* if the named dictionary is opened, and if the entered check codes are recognizable. The command `dict_` may be used more than once if multiple dictionaries are required (the dictionary data-name lists will be concatenated). There are two arguments:

file name CIF dictionary file name

checking code Codes specifying checks to be applied to CIF data

`valid` data-name validation check

`dtype` data-type check.

`open_`

A logical function which opens an input CIF. This function is returned as *true* if the named CIF has been opened. There is one argument:

file name CIF file name.

`data_`

A logical function to select the data block from which data will be extracted. The function is returned as *true* if the named data block is found. The function has one argument. If this argument is entered as blank, the next-encountered data block (in the sequential processing of the CIF) is selected as the requested data block and the data-block name is stored in the character variable `bloc_` (see description below).

data block name Identity of data block containing the requested data.

`test_`

A logical function to identify the data type and loop block number of the named data item. The function is

Table 3. *The test request file used by the program* `tbx_ex.f`

```
_audit_creation_date
_audit_creation_method
_audit_update_record
_chemical_name_systematic
_chemical_formula_moiety
_chemical_formula_sum
_chemical_formula_weight
_chemical_melting_point
_cell_length_a
_cell_length_b
_cell_length_c
_cell_angle_alpha
_cell_angle_beta
_cell_angle_gamma
_cell_volume
_cell_formula_units_Z
_cell_measurement_temperature
_cell_measurement_reflns_used
_cell_measurement_theta_min
_cell_measurement_theta_max

_blat2
_blat1
_blat2
_blat1
_blat2
_blat1
_blat2
_blat1
_blat2
_blat1

_symmetry_cell_setting
_symmetry_space_group_name_H-M
_symmetry_space_group_name_Hall

_atom_type_symbol
_atom_type_oxidation_number
_atom_type_number_in_cell
_atom_type_scat_dispersion_real
_atom_type_scat_dispersion_imag
_atom_type_scat_source
_atom_type_symbol
_atom_type_oxidation_number
_atom_type_number_in_cell
_atom_type_number_in_cell
_atom_type_oxidation_number
_atom_type_scat_dispersion_real

_atom_site_label
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_U_iso_or_equiv
_atom_site_thermal_displace_type
_atom_site_calc_flag
_atom_site_calc_attached_atom
_atom_site_type_symbol
_atom_site_type_symbol
_atom_site_type_symbol
_atom_site_type_symbol
_rubbish_here
_atom_site_type_symbol
_atom_site_type_symbol
_atom_site_type_symbol
_symmetry_space_group_name_Hall
_atom_site_type_symbol
_atom_site_type_symbol
_atom_site_type_symbol
_atom_site_type_symbol

_atom_site_aniso_label
_atom_site_aniso_U_11
_atom_site_aniso_U_22
_atom_site_aniso_U_33
_atom_site_aniso_U_12
_atom_site_aniso_U_13
_atom_site_aniso_U_23
_atom_site_aniso_type_symbol
_atom_site_aniso_U_12
_atom_site_aniso_U_12
_atom_site_aniso_U_12
```

returned as *true* if the data name is present. The data attributes are stored in the system variables `type_` and `list_` (see descriptions below). The function has one argument:

data name Identity of the data item to be tested.

`name_`

A logical function that returns the data name of the next item in the CIF. The function is returned as *true* if a new data name is present in the data block and *false* if the end of the data block is reached. The function has one argument:

data name Returned name of the next data item in the data block.

`numb_`

A logical function for extracting a number and its standard deviation if present. The function is returned as *true* if a number is present. There are three arguments. If the function is returned as *false*, the variables representing arguments 2 and 3 are unaltered. If a standard deviation is not attached to the number, argument 3 is unaltered.

data name Identity of the number to be returned
real variable Returned number (type REAL)
real variable Returned standard deviation (type REAL).

`char_`

A logical variable for extracting a character string or a text line from a CIF. This function is returned as *true* if a character or text string is present. Note that if this string is of type text, this function should be called repeatedly until the logical variable `text_` is *false* (see below for more details). There are two arguments:

data name Identity of the string to be returned
character variable Returned string is of length `long_` (details below).

`pfile_`

A logical function for opening an output CIF. The function is returned as *true* if a new file is opened and

Table 4. *The output CIF generated by the program* `tbx_ex.f`

```
data_whoops_a_daisy

_audit_creation_method                'using CIFtbx'
_audit_creation_extra1                using_CIFtbx                    #< not in dictionary
_audit_creation_extra2                "Terry O'Connell"               #< not in dictionary
_audit_creation_extra3                'Terry O"Connell'               #< not in dictionary
_audit_creation_record                                                #< not in dictionary
;
Text data may be
entered like this
or in a loop.
;
_cell_measurement_temperature         293
_cell_volume                          1759(13)
_cell_length_junk                     8.753535                        #< not in dictionary
_cell_length_c                        19.736(3)

loop_
   _atom_type_symbol
   _atom_type_oxidation_number
   _atom_type_number_in_cell
a 1.0(1) 8.645230
ab 2.0(2) 17.290460
abc 3.0(3) 25.935691
abcd 4.0(4) 34.580921
abcde 5.0(5) 43.226150
abcdef 6.0(6) 51.871383
abcdefg 7.0(6) 60.516613
abcdefgh 8.0(8) 69.161842
abcdefghi 9.0(9) 77.807075
abcdefghij 10(1) 86.452301

loop_
   _atom_type_symbol
   _atom_type_oxidation_number
   _some_silly_text                                                   #< not in dictionary
a 1.0(1)
;
Hi Ho the diddly oh!
;
ab 2.0(2)
;
Hi Ho the diddly oh!
;
abc 3.0(3)
;
Hi Ho the diddly oh!
;
```

false if the requested file name already exists. There is one argument:

file name File name of the output CIF.

pdata_

A logical function for putting a data-block line into the output CIF. The function is returned as *true* if the block is output and *false* if the specified block name already exists. There is one argument:

block name Name concatenated to the `'data_'` command.

ploop_

A logical function for putting a data name into a loop structure. The function is returned as *true* if the invocation conforms with the CIF logical structure. A sequence of these functions is used to specify data names in a common loop structure (the first `ploop_` causes the `'loop_'` command to be inserted). The invocation of any other command will signal that data items are to be entered. There is one argument:

data name Name to be placed in a loop structure.

pnumb_

A logical function for putting a data name and number into the output CIF. The standard-deviation number is appended in parentheses if present. The function is returned as *true* if the name is unique; *and* if `dict_` is invoked, the name is defined in the dictionary; *and* if the invocation conforms to the CIF logical structure. If any of these conditions are not met, the function will be returned as *false*. For looped data, the data name is entered as blank and the order and number of invocations must match that of the `ploop_` invocations. There are three arguments:

data name Requested name of the output number

real variable Number to be output

real variable Standard deviation to be appended in parentheses.

pchar_

A logical function for putting a data name and character string into the output CIF. The function is returned as *true* if the name is unique; *and* if `dict_` is invoked, the name is defined in the dictionary; *and* if the invocation conforms

Table 5. *The output listing from the program* `tbx_ex.f`

```
Read data from CIF test.cif
Warning:              data name _atom_type_scat_dispersion_REAL              not in dictionary!
Warning:              data name _blat1                                      not in dictionary!
Warning:              data name _blat2                                      not in dictionary!

Access items in data block mumbo_jumbo

Cell dimension(s) missing!

Cell        0.0000      14.9560     19.7370
            0.0000      0.0010      0.0030

Space group             P_2ac_2ab

Audit record
91-04-09                        text and data added by Tony Willis.
91-04-15                        rec'd by co-editor with diagram as manuscript HL7.
91-04-17                        adjustments based on first referee's report.
91-04-18                        adjustments based on second referee's report.

Atom sites
s           0.2020      0.7980      0.9167      0.0300      0.0000      0.0000      0.0000      0.0030
o           0.4980      0.4980      0.6667      0.0252      0.0000      0.0000      0.0000      0.0030
c1          0.4880      0.0960      0.0380      0.0317      0.0000      0.0000      0.0000      0.0030

Access items in data block publication

Author list

Furber, Mark
Research School of Chemistry
Australian National University
GPO Box 4
Canberra, A.C.T.
Australia 2601

Mander, Lewis N.
Research School of Chemistry
Australian National University
GPO Box 4
Canberra, A.C.T.
Australia 2601

Patrick, Graham L.
Research School of Chemistry
Australian National University
GPO Box 4
Canberra, A.C.T.
Australia 2601

Willis, Anthony C.
Research School of Chemistry
Australian National University
GPO Box 4
Canberra, A.C.T.
Australia 2601
Warning:              data name _atom_type_scat_dispersion_REAL              not in dictionary!
Warning:              data name _blat1                                      not in dictionary!
Warning:              data name _blat2                                      not in dictionary!

Access items in data block mumbo_jumbo

Atom coordinates and Uij
s           0.2020      0.7980      0.9167      0.0350      0.0250      0.0250      0.0130      0.0000      0.0000
o           0.4980      0.4980      0.6667
c1          0.4880      0.0960      0.0380

_audit_creation_date                                    char 8
91-03-20

_audit_creation_method                                  char 34
from_xtal_archive_file_using_CIFIO

_audit_update_record                                    text 80
91-04-09                        text and data added by Tony Willis.
91-04-15                        rec'd by co-editor with diagram as manuscript HL7.
91-04-17                        adjustments based on first referee's report.
91-04-18                        adjustments based on second referee's report.

_chemical_name_systematic                               char 65
trans-3-Benzoyl-2-(tert-butyl)-4-(iso-butyl)-1,3-oxazolidin-5-one
```

Table 5 (*cont.* 1)

```
_chemical_formula_moiety                                  char 12
C18 H25 N O3

_chemical_formula_sum                                     char 12
C18 H25 N O3

_chemical_formula_weight                                  numb 6
     303.4000            0.0000

_chemical_melting_point                                   null 1

_cell_length_a                                            null 1

_cell_length_b                                            numb 9
      14.9560            0.0010

_cell_length_c                                            numb 9
      19.7370            0.0030

_cell_angle_alpha                                         numb 2
      90.0000            0.0030

_cell_angle_beta                                          numb 2
      90.0000            0.0030

_cell_angle_gamma                                         numb 2
      90.0000            0.0030

_cell_volume                                              numb 9
    1759.0000             0.3000

_cell_formula_units_Z                                     numb 1
       4.0000            0.3000

_cell_measurement_temperature                             numb 3
     293.0000            0.3000

_cell_measurement_reflns_used                             numb 2
      25.0000            0.3000

_cell_measurement_theta_min                               numb 2
      25.0000            0.3000

_cell_measurement_theta_max                               numb 2
      31.0000            0.3000


                                                          null 1

_blat2                                                    numb 1
       2.0000            0.3000

_blat1                                                    numb 1
       1.0000            0.3000

_blat2                                                    numb 1
       4.0000            0.3000

_blat1                                                    numb 1
       3.0000            0.3000

_blat2                                                    numb 1
       6.0000            0.3000

_blat1                                                    numb 1
       5.0000            0.3000

_blat2                                                    char 1
b

_blat1                                                    char 1
a

_blat2                                                    char 1
d

_blat1                                                    char 1
c

                                                          null 1
```

Table 5 (*cont.* 2)

```
_symmetry_cell_setting                                    char 12
orthorhombic

_symmetry_space_group_name_H-M                            char 10
P 21 21 21

_symmetry_space_group_name_Hall                           char 9
P_2ac_2ab

                                                          null 1

_atom_type_symbol                                         char 1
S

_atom_type_oxidation_number                               numb 1
       0.0000            0.3000

_atom_type_number_in_cell                                 numb 1
       6.0000            0.3000

_atom_type_scat_dispersion_real                           null 1

_atom_type_scat_dispersion_imag                           numb 4
       0.5570            0.3000

_atom_type_scat_source                                    char 32
'Int Tab Vol III p202 Tab. 3.3.1a'

_atom_type_symbol                                         char 1
O

_atom_type_oxidation_number                               numb 1
       0.0000            0.3000

_atom_type_number_in_cell                                 numb 1
       6.0000            0.3000

_atom_type_number_in_cell                                 numb 2
      20.0000            0.3000

_atom_type_oxidation_number                               numb 1
       0.0000            0.3000

_atom_type_scat_dispersion_real                           null 1

                                                          null 1

                                                          null 1

_atom_site_label                                          char 1
S

_atom_site_fract_x                                        numb 6
       0.2020            0.3000

_atom_site_fract_y                                        numb 6
       0.7980            0.3000

_atom_site_fract_z                                        numb 6
       0.9167            0.3000

_atom_site_U_iso_or_equiv                                 numb 7
       0.0300            0.0030

_atom_site_thermal_displace_type                          char 3
Uij

_atom_site_calc_flag                                      null 1

_atom_site_calc_attached_atom                             null 1

_atom_site_type_symbol                                    char 1
s

_atom_site_type_symbol                                    char 1
o
```

Table 5 (*cont.* 3)

```
_atom_site_type_symbol                                    char 1
C

_atom_site_type_symbol                                    char 1
S

_rubbish_here                                             null 1

_atom_site_type_symbol                                    char 1
O

_atom_site_type_symbol                                    char 1
C

_atom_site_type_symbol                                    char 1
S

_symmetry_space_group_name_Hall                           char 9
P_2ac_2ab

_atom_site_type_symbol                                    char 1
S

_atom_site_type_symbol                                    char 1
O

_atom_site_type_symbol                                    char 1
C

_atom_site_type_symbol                                    char 1
S

                                                          null 1

_atom_site_aniso_label                                    char 1
S

_atom_site_aniso_U_11                                     numb 7
        0.0350              0.0040

_atom_site_aniso_U_22                                     numb 7
        0.0250              0.0030

_atom_site_aniso_U_33                                     numb 7
        0.0250              0.0030

_atom_site_aniso_U_12                                     numb 7
        0.0130              0.0010

_atom_site_aniso_U_13                                     numb 6
        0.0000              0.0010

_atom_site_aniso_U_23                                     numb 6
         0.0000             0.0010

_atom_site_aniso_type_symbol                              char 1
S

_atom_site_aniso_U_12                                     numb 7
        0.0130              0.0010

_atom_site_aniso_U_12                                     numb 7
        0.0130              0.0010

_atom_site_aniso_U_12                                     numb 7
        0.0130               0.0010
```

to the CIF logical structure. If any of these conditions are not met, the function will be returned as *false*. For looped data, the data name is entered as blank and the order and number of invocations must match that of the `ploop_` invocations. There are two arguments:

data name Requested name of output character string
character variable Character string to be output.

`ptext_`

A logical function for putting a data name and text line into the output CIF. This function is invoked repeatedly until the text is finished. Only the first invocation will insert a data name. The function is returned as *true* if the name is unique; *and* if `dict_` is invoked, the name is defined in the dictionary; *and* if the invocation conforms to the CIF logical structure. If any of these conditions are not met, the function will be returned as *false*. For looped data, the data name is entered as blank and the order and number of invocations must match that of the `ploop_` invocations. There are two arguments:

data name Requested name of the output text string
character variable Text line of up to 80 chars to be output.

`close_`

Subroutine for closing the output CIF. This routine must be called if `pfile_` is used.

Control variables

`text_`

A logical variable that signals if a text line is the next data item in the input CIF. This variable is *true* if the next line in the CIF is part of the same text sequence being accessed with a `char_` function. It is used as a branching variable to access all lines in a text sequence (see examples in Table 1).

`loop_`

A logical variable that signals if another 'loop packet' is present in a data sequence being accessed with `numb_` or `char_` functions. A loop packet is a set of data items that match a set of data names at the head of the loop structure. The variable will be set to *true* if another packet exists in the current loop structure. Not all data items in a loop packet need be accessed to reset the value of `loop_`. Each access of the same data item will cause the packet counter to advance. Note that if a data item outside the current loop structure is accessed it will cause the value of `loop_` to be set to *false*. This variable is used as a branching variable (see examples in Table 1).

`type_`

A character*4 variable containing the data type of the data item identified in a `test_` command.

`numb` for number data
`char` for character data
`text` for text data
`null` if data missing or value '?'

`list_`

An integer variable containing the sequential number of the loop block (in the current data block) of the data item identified in a `test_` command. If the data item is not in a loop structure, this will be zero.

bloc_

A character*27 variable containing the name of the data block currently set *via* the last data_ invocation.

strg_

A character*80 variable containing the current data-item string.

long_

An integer variable containing the length of the data string in strg_.

file_

A character*80 variable containing the file name of the current input CIF.

longf_

An integer variable containing the length of the file name in file_.

align_

A logical variable that is set by the programmer to specify the alignment of loop data output *via* the pchar_, pnumb_ and ptext_ commands. If the variable is set to *true*, each packet of data items starts at a new line. If it is set to *false*, data items will be output continuously (*i.e.* independent of the packet boundary). The default is *true*.

Implementation

The procedure for adding the *CIFtbx* routines to a Fortran application is straightforward. The tool box comes in three parts: the source file (labelled ciftbx.f), the system common definition file (labelled ciftbx.sys) and the application common definition file (labelled ciftbx.cmn). The implementation steps are:

1. Every program or subroutine that employs *CIFtbx* commands **must** contain the following statement in the data definition area: include 'ciftbx.cmn' **or** an equivalent compiler statement for inserting the file ciftbx.cmn.

2. The *CIFtbx* routines may be added to an application in one of two ways.

(i) Insert the statement include 'ciftbx.f' at the start of the application source (before the first non-comment line) or at the end of the application source (after the last END line). If include is not recognized by a compiler, the editor may be used to add the source code directly. Note that when the application is compiled with the included source ciftbx.f, the file ciftbx.sys will be automatically included. This latter file must be available in the current directory.

(ii) Compile the source file ciftbx.f separately from the application. The ciftbx.sys common file will be included automatically. Add the object file ciftbx.o when linking the application object files. This is the most efficient approach when developing an application, as the *CIFtbx* source need only be compiled once.

3. During the compilation of an application using *CIFtbx* functions, warning messages may be issued about unused variables. These are prevented by removing the unused data declarations from the ciftbx.cmn file with an editor.

4. Programmers should be aware of the following *CIFtbx* requirements.

(i) The pfile_ command will not overide an existing CIF of the same file name. This is a protection facility.

(ii) Make sure that the correct directory path information is included with file names declared in the open_, dict_ and pfile_ commands.

(iii) Always apply the *CIFtbx* commands in such a way as to allow for 'missing' or incorrectly 'typed' data. Never assume that the requested data are present in a CIF.

(iv) Take care when reading character data items that can be either type *char* or type *text* (*e.g.* _chemical_name_systematic). In such cases, always use char_ within a loop controlled by the logical variable text_.

Error messages

The *CIFtbx* commands are designed so that most run-time errors are signaled by a returned *false* value of a logical function. However, some types of errors will cause data processing to halt. If these occur, *CIFtbx* issues an error message with the line number of the CIF (or dictionary file) and then stops. Here is a summary of error messages and brief description of the likely cause of failure.

dict_ must precede open_

The dictionary files must be loaded before an input CIF is opened because some of the data-name checking occurs during the CIF loading process.

cifdic names > 1000

The number of data names loaded from the dictionary or dictionaries exceeds 1000. These limits may be changed by increasing the array sizes of DICNAM() and DICTYP() in ciftbx.sys.

_type line is missing

The DDL definition of _type is missing from the specified dictionary definition.

Item miscount in loop

The input CIF contains a loop structure in which the number of data items does not match an integer multiple of the number of items in a single loop packet.

Number of loop_s > 50

CIFtbx only allows for 50 separate loop structures in a data block. This may be changed by increasing the array size of LOOPNI() and LOOPNP() in ciftbx.sys.

Number of data names > 500

CIFtbx only allows for 500 data names in an input CIF data block. This may be changed by increasing the array size of seven array variables in ciftbx.sys.

`Items per loop_ packet > 20`

CIFtbx only allows for 20 items per input loop packet. This may be changed by increasing the array sizes of LOOPCH(), LOOPLN() and LOOPRD() in the routine GETITM.

`Syntax construction error`

An illegal construction has been detected in the input CIF.

`Unexpected end of data`

An unexpected end to text data encountered. This is probably due to a missing semicolon at the end of a text string.

Distribution

CIFtbx is distributed as the file `ciftbx` containing the Fortran source, the two common files, a test application file and test files. The standard CIF dictionary file `cifdic.C91` may be needed for use with the command `dict_`. The files `ciftbx` and `cifdic.C91` may be obtained free of charge in several different ways. The simplest and fastest approach is to use anonymous FTP to *get* the file from the directory cif on the host 130.95.232.12. Alternatively, send an email containing the lines `send ciftbx` and `send cifdic.C91` to sendcif@crystal.uwa.edu.au or sendcif@iucr.ac.uk. As a last resort, airmail a floppy disk to the author stating the mode of copy required.

References

HALL, S. R. (1993*a*). *J. Appl. Cryst.* **26**, 474–479.
HALL, S. R. (1993*b*). *J. Appl. Cryst.* **26**, 480–481.
HALL, S. R., ALLEN, F. H. & BROWN, I. D. (1991). *Acta Cryst.* A**47**, 655–685.
HALL, S. R. & SIEVERS, R. (1993). *J. Appl. Cryst.* **26**, 469–473.