```
                        PROGRAM  CIF_IN
C
C....A.. Define the data variables
         include       'ciftbx.cmn'
         logical       f1,f2,f3
         character*32  name
         character*80  line
         real          cela,celb,celc,siga,sigb,sigc
         real          x,y,z,u,numb,sdev
         data          cela,celb,celc,siga,sigb,sigc /6*0.0/

C....B.. Assign the CIFtbx files
         f1 = init_( 1, 2, 3, 6 )

C....C.. Request dictionary validation check
         if(dict_('cif_core.dic','valid'))  goto 100
         write(6,'(/a/)') ' Requested Core dictionary not present'

C....D.. Open the CIF to be accessed
100      name='test.cif'
         if(ocif_(name))      goto 120
         write(6,'(a///)')  ' >>>>>>>>> CIF cannot be opened'
         stop
C....E.. Assign the data block to be accessed
120      if(.not.data_(' '))       goto 200
         write(6,'(/a,a/)') ' Access items in data block  ',bloc_

C....F.. Extract some cell dimensions; test all is OK
         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 lengths missing!'
         write(6,'(a,6f10.4)') ' Cell ',cela,celb,celc,siga,sigb,sigc

C....G.. Extract space group notation (expected char string)
         f1 = char_('_symmetry_cell_setting', name)
         write(6,'(a,a/)') ' Cell setting   ',name(1:long_)

C....H.. Get the next name in the CIF and print it out
         f1 = name_(name)
         write(6,'(a,a/)') ' Next data name in CIF is   ',name

C....I.. List the audit record (possible text line sequence)
         write(6,'(a)') ' Audit record'
140      f1 = char_('_audit_update_record', line)
         write(6,'(a)') line
         if(text_)  goto 140

C....J.. Extract atom site data in a loop
         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)
         f2 = numb_('_atom_site_fract_z', z, sz)
         f3 = numb_('_atom_site_U_iso_or_equiv', u, su)
         write(6,'(1x,a4,4f8.4)') name,x,y,z,u
         if(loop_)  goto 160
C
         goto 120
200      continue
         end
```

Fig. 5.4.9.1. Sample program *CIF_IN*. See text for explanation.

`'#< not in dictionary'`. This applies to both looped and single data items.

A more complex example of writing a CIF is given in the program *cif2cif* available with the *CIFtbx* release. A similar program that reads a CIF and writes an XML file is *cif2xml*, also available with the *CIFtbx* release.

5.4.11. Error-message glossary

The *CIFtbx* routines will generate explicit error messages in the printout or in the created CIF if requested to do so (*e.g.* during dictionary checks). If data processing cannot continue (*i.e.* fatal errors), an appropriate error message is placed in the printout and execution terminates. However, the default approach is to remain mute and for error detection to be monitored by the application program *via* the *CIFtbx* functions returning `.true.` or `.false.` values that tell the application program whether the command was performed correctly. This places the primary responsibility for error checking on the application software. The importance of this approach is that it enables the local application to respond to run-time problems in a controlled way and to take corrective action if it is possible. However, some types of processing errors, such as exceeding the dimensions of critical *CIFtbx* arrays, do require appropriate messages to be issued and for execution to cease.

```
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
;
_dummy_test     "rubbish to see what dict_ says"

_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_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_measurement_theta_min        25
_cell_measurement_theta_max        31
_symmetry_cell_setting             orthorhombic

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
  s  .20200  .79800  .91667  .030(3)  Uij  ?
  o  .49800  .49800  .66667  .02520  Uiso  ?
  c1  .48800  .09600  .03800  .03170  Uiso  ?

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

Fig. 5.4.9.2. Example CIF read by the sample program *CIF_IN* shown in Fig. 5.4.9.1.

```
 CIFtbx warning: test.cif data_mumbo_jumbo line:     8
 Data name _dummy_test not in dictionary!
 CIFtbx warning: test.cif data_mumbo_jumbo line:     35
 Data name _blat1 not in dictionary!
 CIFtbx warning: test.cif data_mumbo_jumbo line:     35
 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
 Cell setting   orthorhombic

 Next data name in CIF is   _atom_type_symbol

 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

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

Fig. 5.4.9.3. Printout from the example program *CIF_IN* run on the test file of Fig. 5.4.9.2.

CIFtbx error messages are in four parts: 'warning' or 'error' header line, the name of the file being processed, the current data block or save frame, and the line number. Another line contains the text of the message.

5.4.11.1. Fatal errors: array bounds

The following fatal messages are issued if the *CIFtbx* array bounds are exceeded. Operation terminates immediately. Array bounds can be adjusted by changing the PARAMETER values in ciftbx.sys. If the value of MAXBUF needs to be changed, the file ciftbx.cmv must also be updated.

```
Input line_ value > MAXBUF
Number of categories > NUMBLOCK
Number of data names > NUMBLOCK
Cifdic names > NUMDICT
Dictionary category names > NUMDICT
Items per loop packet > NUMITEM
Number of loop_s > NUMLOOP
```

```
C....... Open a new CIF
400      if(pfile_('test.new'))  goto 450
         write(6,'(//a/)') ' Output CIF by this name exists already!'
         goto 500
C....... Request dictionary validation check
450      if(dict_('cif_core.dic','valid'))  goto 460
         write(6,'(/a/)') ' Requested Core dictionary not present'
C....... Insert a data block code
460      f1 = pdata_('whoops_a_daisy')
C....... Enter various single data items to show how
         f1 = pchar_('_audit_creation_method','using CIFtbx')
         f1 = pchar_('_audit_creation_extra2',"Terry O'Connell")
         f1 = pchar_('_audit_creation_extra3','Terry O"Connell')
         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.')
         f1 = pnumb_('_cell_measurement_temperature', 293., 0.)
         f1 = pnumb_('_cell_volume', 1759.0, 13.)
         f1 = pnumb_('_cell_length_b', 8.75353553524313,0.)
         f1 = pnumb_('_cell_length_c', 19.737, .003)
C....... Enter some looped data
         f1 = ploop_('_atom_type_symbol')
         f1 = ploop_('_atom_type_oxidation_number')
         f1 = ploop_('_atom_type_number_in_cell')
         do 470 i=1,3
         f1 = pchar_(' ',alpha(1:i))
         f1 = pnumb_(' ',float(i),float(i)*0.1)
470      f1 = pnumb_(' ',float(i)*8.64523,0.)
C....... Do it again but as contiguous data with text data
         f1 = ploop_('_atom_site_label')
         f1 = ploop_('_atom_site_occupancy')
         f1 = ploop_('_some_silly_text')
         do 480 i=1,2
         f1 = pchar_(' ',alpha(1:i))
         f1 = pnumb_(' ',float(i),float(i)*0.1)
480      f1 = ptext_(' ',' Hi Ho the diddly oh!')
500      call close_
```

Fig. 5.4.10.1. Sample program to create a CIF.

```
data_whoops_a_daisy
_audit_creation_method      'using CIFtbx'
_audit_creation_extra2      'Terry O'Connell'    #< not in dictionary
_audit_creation_extra3      'Terry O"Connell'    #< not in dictionary
_audit_creation_record
;Text data may be
 entered like this
 or in a loop.
;
_cell_measurement_temperature   293
_cell_volume                    1759(13)
_cell_length_b                  8.75354
_cell_length_c                  19.737(3)
loop_
    _atom_type_symbol
    _atom_type_oxidation_number
    _atom_type_number_in_cell
        a               1.00(10)        8.64523
        ab              2.0(2)          17.2905
        abc             3.0(3)          25.9357
loop_
    _atom_site_label
    _atom_site_occupancy
    _some_silly_text                    #< not in dictionary
        a               1.00(10)
;Hi Ho the diddly oh!
;
        ab              2.0(2)
;Hi Ho the diddly oh!
;
```

Fig. 5.4.10.2. Sample CIF created by the example program of Fig. 5.4.10.1.

However, the message

`More than MAXBOOK bookmarks requested`

is not 'fatal', in the sense that the function `bkmrk_` returns `.false.` to permit appropriate action before termination. This is effectively a fatal error for which recompilation with a larger value of `MAXBOOK` is necessary. However, this is usually the result of a logic error in the application, and the error has been made non-fatal to allow the programmer to insert debugging code, if desired. The application should clean up and exit promptly.

5.4.11.2. Fatal errors: data sequence, syntax and file construction

`Dict_ must precede ocif_`
Dictionary files must be loaded before an input CIF is opened because some checking occurs during the CIF loading process.

`Illegal tag/value construction`
Data name (*i.e.* a 'tag') and data values are not matched (outside a looped list). This usually means that a data name immediately follows another data name, or a data value was found without a preceding data name. The most likely cause of this error is the failure to provide '**.**' or '**?**' for missing or unknown data values or a failure to declare a **loop_** when one was intended.

`Item miscount in loop`
Within a looped list the total number of data values must be an exact multiple of the number of data names in the **loop_** header.

`Prior save-frame not terminated`
Save-frame terminator found out of context. Save frames must start with **save_*framecode*** and end with **save_**. These messages will be issued if this does not occur.

`Syntax construction error`
Within a data block or save frame the number of data values does not match the number of data names (ignoring loop structures). This message should occur only if there is an internal logic error in the library. Normally the program will terminate on `Item miscount in loop` first.

`Unexpected end of data`
When processing multi-line text the end of the CIF is encountered before the terminal semicolon.

5.4.11.3. Fatal errors: invalid arguments

The following messages are generated by calls with invalid arguments.

`Call to find_ with invalid arguments`
`Internal error in putnum`

5.4.11.4. Warnings: input errors

`Category <cat-code> first implicitly defined in cif`
The category code in the DDL2 data name is not matched by an explicit definition in the dictionary. This may be intentional but usually indicates a typographical error in the CIF or the dictionary.

`Data name <name> not in dictionary!`
The data item name `<name>` was used in the CIF but could not be found in the dictionary.

`Data block header missing`
No **data_** or **global_** was found when expected.

`Duplicate data item <name>`
Two or more identical data names `<name>` have been detected in a data block or save frame.

`Exponent overflow in numeric input`
`Exponent underflow in numeric input`
The numeric value being processed has an exponent that cannot be processed on this machine. If the string involved is not intended as a number, then surrounding it with quotes may resolve the problem.

`Heterogeneous categories in loop <new cat-code>`
`  vs <old cat-code>`
Looped lists should not contain data items belonging to different categories. This error occurs if the category of a new data item fails to match the category of a prior data item. A special category (none) is used to denote item names for which no category has been declared. Warnings are not issued on this level for a loop for which all data items have no declared category.

`Input line length exceeds line_`
Non-blank characters were found beyond the value given by the variable `line_`. The default value for `line_` is 80 (optionally increased to 2048 in *CIFtbx* 2.7 and later for CIF 1.1 compliance). The extra characters in column positions `line_` + 1 through `MAXBUF` will be processed but the input file may need to be reformatted for use with other CIF-handling programs.

`Missing loop_ items set as DUMMY`
A looped list of output items was truncated with an incomplete loop packet (*i.e.* the number of items did not match the number of **loop_** data names). The missing values were set to the *character string* 'DUMMY'.

`Numb type violated <name>`
The data item `<name>` has been processed with an explicit dictionary type numb, but with a non-numeric value. Note that the values '**?**' or '**.**' will *not* generate this message.

`Quoted string not closed`
Character values may be enclosed by bounding quotes. The strict definition of a 'quoted-string' value is that it must start with a `<wq>` digraph and end with a `<qw>` digraph, where `w` is a white-space character blank or tab and `q` is a single or double quote, and the same type of quote mark is used in the terminal digraph as was used in the initial digraph. This message is issued if these conditions are not met.

5.4.11.5. Warnings: output errors

`Converted pchar_ output to text for <string>`
An attempt was made to write a string with `pchar_` instead of `ptext_`, but the string contains a combination of characters for which `ptext_` must be used.

`ESD less than precision of machine`
`Overflow of esd`
`Underflow of esd`
A call to `pnumb_` or `numb_` was made with values of the number and standard uncertainty (e.s.d) which cannot be presented properly on this machine. A bounding value of 0 or 99999 is used for the e.s.d.

`Invalid value of esdlim_ reset to 19`
In processing numeric output, a value of `esdlim_` less than 9 or greater than 99999 was found. `esdlim_` is then set to 19.

`Missing loop_ name set as _DUMMY`

`Missing loop_ items set as DUMMY`

In processing a **`loop_`**, a dummy string has been inserted for a missing header or value.

`Output CIF line longer than line_`

In outputting a line, the data exceed the limit specified in `line_`. This occurs only if a single data name or a value exceeds this limit.

`Out-of-sequence call to end text block`

The termination of a text block has been invoked before a text block has been started. This can only occur with irregular use of the *CIFtbx* routines rather than the standard interface routines.

`Output prefix may force line overflow`

A prefix string placed in `prefx_` exceeds `line_` less the allowed length of tags.

`Prefix string truncated`

A prefix string specified to `prefx_` is longer than the maximum line length allowed. The prefix string is truncated and processing continues.

5.4.11.6. Warnings: dictionary checks

`Aliases and names in different loops; only using first alias`

If a DDL2 dictionary contains a loop of alias declarations, the corresponding data-name declarations are expected to be in the same loop. Only the first alias name is used.

`Attempt to redefine category for item`

`Attempt to redefine type for item`

If a DDL2 dictionary contains a category or type for a data item that conflicts with an earlier declaration, the first is used.

`Categories and names in different loops`

`Types and names in different loops`

If a DDL2 dictionary contains a loop of category or type declarations, the corresponding data-name declarations are expected to be in the same loop. Only the first category name or type is used.

`Category id does not match block name`

In a DDL2 dictionary, the save-frame code is expected to start with the category name. If a category name within the frame is not within a loop, it is checked against that in the frame code and a warning is issued if these do not match.

`Conflicting definition of alias`

A DDL2 dictionary contains a new declaration of a data-name alias which is in conflict with a previous alias definition. The first alias declaration is used.

`Duplicate definition of same alias`

A DDL2 dictionary contains a new declaration of an alias for a data name which duplicates a previously defined alias pair.

`Item name <name> does not match category name`

If category checking is enabled and the category assigned to an item name does not match the intial characters of the item name, this message is issued. This may indicate a typographical error or a deprecated item in the dictionary.

`Item type <type-code> not recognised`

The DDL2 dictionary type codes are translated to the DDL type codes 'numb', 'char' and 'text'. If an unrecognized type code is found no translation occurs.

`Multiple DDL category definitions`

`Multiple DDL name definitions`

`Multiple DDL type definitions`

`Multiple DDL related item definitions`

`Multiple DDL related item functions`

DDL1 and DDL2 declarations for categories, data names, data types and related items are used in the same data block or save frame.

`Multiple categories for one name`

`Multiple types for one name`

A dictionary contains a loop of category or type definitions and an unlooped declaration of a single data name. The first category or type definition is used.

`No category defined in block <name> and name <name> does not match`

A DDL2 dictionary contains no category for the defined data item and it was not possible to derive an implicit category from the block name. This usually indicates a typographical error in the dictionary.

`No category specified for name <name>`

A dictionary contains categories and category checking is enabled but no category is defined for the named data item.

`No name defined in block`

`No name in the block matches the block name`

These messages are issued if a dictionary save frame or data block contains no name definition or if all the names defined fail to match the block name.

`No type specified for name <name>`

A type code is missing from a dictionary and type checking was requested in the `dict_` invocation.

`One alias, looped names, linking to first`

A DDL2 dictionary may contain a list of data names and a single alias outside this loop. In this case, the correct name to which to link the alias must be derived implicitly. If the save-frame code matches the first name in the loop no warning is issued, because the use of the block name was probably the intended result, but if no such match is found this warning is issued.

5.4.12. Internals and programming style

CIFtbx is programmed in a highly portable Fortran programming style. However, on some older systems, some adaptation may be necessary to allow compilation. Implementors should be aware of the extensive use of variables in common blocks to transmit information and control execution (programming by side-effects), the use of the `INCLUDE` statement, the use of the `ENDDO` statement, the names of routines used internally by the package, the use of names longer than six characters and the use of names including the underscore character.

Some aspects of the internal organization of the library to deal with characteristics of CIFs are worth noting. *CIFtbx* copies an input CIF to a direct-access (*i.e.* random-access) file, but writes an output CIF directly. All data names are converted to lower case to deal with the case-insensitive nature of CIF. A hierarchy of parsing routines is used to deal with processing white space.

The major issues of programming style and internals are summarized here. See the *Primer* on the CD-ROM for more information.