International Tables for Crystallography (2006). Vol. G, Section 5.4.11.5, pp. 534–535.

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.

references