5.4.12.1. Programming style

A traditional Fortran style of programming is used in *CIFtbx*. Common blocks are declared to report and control the state of the processing. This allows argument lists to be kept short and avoids the need to create complex data structure types, but introduces extensive 'programming by side-effects'. In order to reduce the impact of this approach on users, two different views of the common blocks are provided. The declarations in ciftbx.cmn are needed by all users. The more extensive declarations in ciftbx.sys, which include the same common declarations as are found in ciftbx.cmn and additional declarations used internally within *CIFtbx*, are provided for use in maintaining the library. Caution is needed in making internal modifications to the library to maintain the desired relationships among the actions of various routines and the states of variables declared in the common blocks.

Statements are written in the first 72 columns of a line, reserving columns one through five for statement labels and using column six for continuation. Approaches that would require the use of C libraries or non-portable Fortran extensions are avoided. For this reason, all the internal service routines are written in Fortran, all memory needed is preallocated with DIMENSION statements and a direct-access file is used to hold the working copy of a CIF.

5.4.12.2. Memory management

Since *CIFtbx* does static memory allocation with DIMENSION statements, it is sometimes necessary to adjust the array dimensions chosen to suit a particular application. It may also be necessary to increase the storage allocated for individual tags to allow for unusually long ones.

The sizes of most arrays and strings used in *CIFtbx* that might require adjustment are controlled by PARAMETER statements in the files ciftbx.sys and ciftbx.cmv (the variable-declaration portion of ciftbx.cmn). The parameters are shown in Table 5.4.12.1.

These values can result in *CIFtbx* requiring more than a megabyte of memory. On smaller machines working with a small dictionary and simple CIFs, considerable space can be saved by reducing the values of NUMDICT and NUMBLOCK.

On the other hand, an application working with several layered dictionaries and large and complex CIFs with many data items and many loops in a data block might require a version of *CIFtbx* with larger values of NUMDICT, NUMBLOCK and, perhaps, of NUMLOOP.

The variables NUMPAGE and NUMCPP control the amount of memory to be used to buffer the direct-access file and the size of the data transfers to and from that file. Smaller values will reduce the demand for memory at the expense of slower execution.

5.4.12.3. Use of INCLUDE

The INCLUDE statement allows the statements in the specified file to be treated as if they were being included in a program in place of the INCLUDE statement itself. This simplifies the maintenance of common-block declarations and is an important tool in keeping code well organized. In *CIFtbx*, the INCLUDE statement is used to bring the statements in the files ciftbx.cmn and ciftbx.sys into programs where they are needed, and to simplify ciftbx.cmn and ciftbx.sys by using INCLUDEs of the files ciftbx.cmv and ciftbx.cmf. The file ciftbx.cmv contains the definitions of the essential *CIFtbx* data structures as common blocks, for inclusion in both ciftbx.cmn for user applications and in ciftbx.sys for the *CIFtbx* library routines themselves. Most compilers handle the INCLUDE statement, but, if necessary, a user may replace any or all of the INCLUDE statements with the contents of the indicated file. For example, the only non-comments in ciftbx.cmn are

```
include  'ciftbx.cmv'
include  'ciftbx.cmf'
```

This means that the file ciftbx.cmn could be replaced by a concatenation of the two files ciftbx.cmv and ciftbx.cmf.

Table 5.4.12.1. *Parameter statements in CIFtbx*

NUMCHAR	Maximum number of characters in data names (default 48)
MAXBUF	Maximum number of characters in a line (default 200, increased to 4096 in *CIFtbx* 2.7 and later)
NUMPAGE	Number of memory resident pages (default 10)
NUMCPP	Number of characters per page (default 16 384)
NUMDICT	Number of entries in dictionary tables (default 3200)
NUMHASH	Number of hash table entries (a modest prime, default 53)
NUMBLOCK	Number of entries in data-block tables (default 500)
NUMLOOP	Number of loops in a data block (default 50)
NUMITEM	Number of items in a loop (default 50)
MAXTAB	Maximum number of tabs in output CIF line (default 10)
MAXBOOK	Maximum number of simultaneous bookmarks (default 1000)

5.4.12.4. Use of ENDDO

CIFtbx makes some use of the ENDDO statement (as well as nested IF, THEN, ELSE, ENDIF constructs) to improve readability of the source code. Most compilers accept the ENDDO statement, but if conversion is needed then constructs of the form

```
      do index = istart, iend, incr
        ...
      enddo
```

should be changed to

```
      do nnn index = istart, iend, incr
        ...
nnn   continue
```

where *nnn* is a unique statement number, not used elsewhere in the same routine.

5.4.12.5. Names of internal routines

The following routines are used internally by later versions of *CIFtbx*. If these names are needed for other routines, then changes in the library will be needed to avoid conflicts.

(*a*) Variable initialization:

```
block data
```

Critical *CIFtbx* variables are initialized with data statements in a block data routine.

(*b*) Control of floating-point exceptions:

```
subroutine clearfp
```

If a system requires special handling of floating-point exceptions, the necessary calls should be added to this subroutine.

(*c*) Message processing:

```
subroutine err (mess)
  character mess*(*)
subroutine warn (mess)
  character mess*(*)
subroutine cifmsg (flag, mess)
  character mess*(*), flag*(*)
```