because `_A1` occurs in the first data block encountered; the output from *cif2cif* in this example will be

```
data_A
    loop_     _A1
               a1   aa1
#             ---end-of-data-block---
```

The behaviour of the program differs from *QUASAR* in two other small ways. When the request list forces the output data stream to contain the same data-block header more than once, an error message is posted to the standard error channel and the data-block headers in the output stream are annotated with a comment of the form '`#<---- duplicate data block`'. In this case the output file does *not* conform to the CIF syntax rules.

When a data name is requested but no matching data item appears in the output file, *cif2cif* writes an error message to the standard error channel. However, unlike *QUASAR*, which inserts the requested data name in the output stream with an associated value of '`?`' (for *unknown*), *cif2cif* produces *no* output for the requested data item.

5.3.5.2.1.5. *Other features*

Some additional features are of use in special circumstances.

The user may preserve the layout of the contents of looped lists exactly as in the input file, or may ask the program to adjust the layout to a more visually pleasing tabular form.

The user may enable recognition of data-name aliases in the dictionaries used for validation. When the relevant command-line argument is set to *true*, user-supplied data names will be transformed to the canonical forms in the validating dictionary. This would permit, for example, a small-molecule CIF using the core dictionary definitions to be converted to mmCIF format.

The user may prefix each line of output with an identical character string. A typical reason for so doing would be to include a fragment of CIF listing within the body of an email message or some other document. Such an output would not conform to the syntax rules for CIF.

5.3.5.2.2. *Invocation of the program*

cif2cif is another application of the *CIFtbx* library by the same author, and so has a similar user interface to that of *CYCLOPS* (Section 5.3.4.1.2). Under a Unix-like operating system, the program is typically called with a command such as

`cif2cif -i` *infile* `-o` *outfile* [`-q` *reqfile*]

where *infile* is the name of the input file, *outfile* is the output file and *reqfile* is an optional file containing a request list for a subset of the original contents.

A more complete set of options available in a Unix-like operating environment is

`cif2cif` [`-i` *infile*] [`-o` *outfile*] [`-d` *dictfile*] [`-q` *reqfile*] [`-f` *cmndfile*] [`-c` *catck*] [`-a` *alias*] [`-t` *tab*] [`-e` *sulim*] [`-p` *prefix*]

where the options are as follows:

-i specifies the name of the input file, *infile*.

-o specifies the name of the output file, *outfile*.

-d specifies the name of a dictionary file, *dictfile*, against which the existence, type and category of data names are checked. The dictionary file may be *either* a CIF dictionary or a list of file names. That is, it may contain dictionary definitions in DDL format or (if the file begins with the characters `#DICT`) it may contain a list of dictionary file names to be entered. Thus, multiple dictionaries may be specified to the program.

-q specifies the name of the request file, *reqfile*, containing a list of data names (with associated data-block directives) that should be extracted as a subset of the contents of the original file.

-f specifies the name of a command file *cmndfile* that contains additional directives to the program.

-c is a flag indicating whether an error message should be raised if a data name has been assigned a category different from the leading portion of the data name itself. The Boolean variable *catck* may take the values 't', '1' or 'y' for *true*, 'f', '0' or 'n' for *false*.

-a is a flag indicating whether data-name aliases in the validating dictionary should be used to replace user-supplied names by their canonical forms. The Boolean variable *alias* may take the same values for *true* or *false* as above.

-t is a flag indicating whether the output should be reformatted with tabs to produce a regular table layout within looped lists. The Boolean variable *tab* takes the same values as above. If *true*, text is reformatted; if *false*, the original formatting is retained.

For the flags expecting Boolean values, the default is 'f' (*false*).

-e specifies the precision to retain in rounding standard uncertainty values. The permitted integer values are 9, 19 (the default) and 29.

-p takes a string value which is prefixed to every line of output. Every occurrence of the underscore character '_' in the prefix is changed to a space on output.

If no input or output file names are specified, the program will read from the standard input channel or write to standard output, respectively. The special character hyphen ('-') may also be supplied as an argument in place of a file name to indicate standard input or standard output as appropriate.

Finally, if the operating system supports the passing of environment variables to a program, the name of the input file may be passed as the value of `$cif2cif_INPUT_CIF`, and likewise the output file, `$cif2cif_OUTPUT_CIF`, dictionary file, `$cif2cif_CHECK_DICTIONARY`, and request file, `$cif2cif_REQUEST_LIST`, may be specified.

5.3.5.3. *ciftex*: translating to a typesetting language

The program *ciftex* (McMahon, 1993) was developed to create files for typesetting the journal *Acta Crystallographica* using the text-formatting language TEX (Knuth, 1986). Details of its use in the journal production process are given in Chapter 5.7. It is discussed here as an example of translating a CIF to some output format where data values are annotated with different text depending on their accompanying data names.

5.3.5.3.1. *Basic operation of ciftex*

The program is designed to act as a filter, typically in a Unix-style environment, reading a CIF on the standard input channel and outputting a modified data stream to standard output. The output is a file of TEX code that is processed by the TEX program to produce a device-independent file describing the content of a formatted typeset document. Further post-processing allows the formatted document to be viewed on the screen or printed.

Each input token (number, character or text string; data name; `loop_` or `data_` keywords) is transformed as it is identified; there is no lookahead and minimal retention of context. The data stream is treated purely syntactically; no transformations are applied on the basis of the supposed meaning of any of the file contents.

```
_cell_formula_units_Z        2
_cell_length_a               8.79(2)
_refine_ls_extinction_coef    .347e4(5)
_chemical_name_common        'copper sulphate'
```

Fig. 5.3.5.4. Sample CIF data input to *ciftex*.

```
\cellz{2}
\nobreak\cella{8.79 (2)}
\extcoeffLarson{0.347 (5) $\times$ $10^{4}$}
\chemcom{copper sulfate}
```

Fig. 5.3.5.5. Output from *ciftex* run on the data of Fig. 5.3.5.4.

5.3.5.3.1.1. *Non-looped data*

For portions of the CIF that are not contained in looped lists, the transformations are trivial. A (*data name*, *data value*) pair is transformed to a TEX macro and its argument. The macro name is determined from an external 'map' file which the program reads at run time; this file associates CIF data names and the corresponding TEX macros through a simple lookup table.

A CIF data value is in most cases passed as the argument to the corresponding TEX macro with few modifications. If the data value is a character string beginning with an integer, full point, hyphen or plus character, it is assumed to be of type 'numb'. A space is introduced ahead of an embedded open parenthesis (to separate a standard uncertainty from its parent value). A leading zero is printed before any bare decimal point. An embedded E is taken to indicate exponential notation and the format of the number is accordingly modified.

If the input data value is of type 'char' (*i.e.* is a single token beginning with characters other than those recognized as the leading characters for numerical data; or contains multiple tokens delimited by quote marks or semicolons), the program will search the map file for key values exactly matching each token, and if found will substitute the token by its replacement word or text. If no replacement is specified in the map file, the token is passed unchanged to the standard output channel. This facility was found to be useful in making global substitutions of individual words during file processing, but must be used with care since the substitutions are unconditional, without any reference to context.

Some small examples of typical non-looped data items are shown in Fig. 5.3.5.4 and the corresponding *ciftex* translation based on a map file used for typesetting *Acta Crystallographica Section C* is shown in Fig. 5.3.5.5.

Note the transformations of the numerical arguments and the translation of 'sulphate' to 'sulfate'.

5.3.5.3.1.2. *Looped data*

If the input token is a `loop_` keyword, the program enters a different mode of operation. Looped data may be represented in print either as repetitive lists or in tabular format. There is no indication in a CIF dictionary of the appropriate representation (nor should there be, for what is essentially a matter of presentation) and the choice is made based on a flag associated with each data name in the map file. For non-tabular lists, the structure

```
loop_
    _dataname_1
    _dataname_2
       value_1    value_2
       value_3    value_4
```

```
\settabs 5 \columns
\+\relax & $x$  &   $y$    &  $z$   &$U_{\rm eq}$&\cr
\+Re &0.222 (1) &0.003 (1) &0.146 (1) &0.042 (1) &\cr
\+Co &0.234 (1) &0.139 (1) &0.299 (1) &0.046 (1) &\cr
\+P1 &0.358 (1) &0.222 (1) &0.197 (1) &0.044 (1) &\cr
\+P2 &0.106 (2) &0.051 (1) &0.289 (1) &0.046 (1) &\cr
\+C1 &0.308 (6) &0.029 (6) &0.034 (4) &0.057 (4) &\cr
\+O1 &0.356 (5) &0.044 (5) &0.030 (3) &0.079 (3) &\cr
\+C2 &0.066 (6) &0.039 (6) &0.111 (4) &0.056 (4) &\cr
```

Fig. 5.3.5.6. TEX markup for typesetting a table of atomic coordinates.

is translated to a sequence of TEX codes of the form

```
\macro_one(value_1)
\macro_two(value_2)
\macro_one(value_3)
\macro_two(value_4)
```

In the case of tabulated data, the `loop_` header is translated into a set of table headings and typographic codes are introduced to lay out in columnar format the values in the body of the list. The number of different data names in the loop header is counted and the data values are identified by their position in the loop modulo the total number of data names in the header (in effect, by their 'phase' in the loop). In the simplest case, a TEX command is emitted that builds a table with *n* columns, where *n* is the number of different data names. Then the data values are counted as they are processed. After every *n*th data value, a TEX code is emitted indicating 'end of table row' and a further code is emitted before the next value (if there is one) that means 'beginning of new table row'. In all other cases, a code is emitted signifying 'move to next column'.

Fig. 5.3.5.6 is a simplified extract from a table of atomic coordinates derived from the `_atom_site_` loop in a CIF.

5.3.5.3.1.3. *The ancillary map file*

The translation between a CIF data name and its replacement text in the TEX output file is defined in the external map file. The format of the translation is very simple, as illustrated in Fig. 5.3.5.7.

Each line starts with a CIF data name, which is terminated by a space character. The next character is either 'T' or 'N' to indicate whether the output should be tabulated or not. The next character is an arbitrary character from the ASCII character set, and is chosen to collect together data that will appear in the same logical section of the output file. This locator character may be associated, in another ancillary file described below, with additional text for output. The remainder of the line is the replacement text.

In the example supplied, the cell-length parameters map to the TEX macros `\cella`, `\cellb` and `\cellc` (each preceded by a standard TEX macro forbidding a page break immediately before the contents are printed). The details of the publication authors are described by a set of TEX macros that will occur in two different locations in the output file (the authors' names and addresses may be looped together in the location labelled by the character `a`; any explanatory footnotes and email addresses will be printed elsewhere in the paper, at the location labelled `x`). The anisotropic displacement parameters U^{ij} will be printed in a table and the replacement text consists of the TEX codes that will be printed at the head of each column in the table.

The initial text on the line need not be a CIF data name; it may be any other single word. In this case, every occurrence of that word in the input CIF will be replaced by the replacement text.

```
_cell_length_a Ng\nobreak\cella
_cell_length_b Ng\nobreak\cellb
_cell_length_c Ng\nobreak\cellc

_publ_author_name Na\author
_publ_author_address Na\address
_publ_author_footnote NX\aufootnote
_publ_contact_author_email NX\email

_atom_site_aniso_label TU\relax
_atom_site_aniso_U_11 TU{\hfill $U^{11}$ \hfill}
_atom_site_aniso_U_12 TU{\hfill $U^{12}$ \hfill}
_atom_site_aniso_U_13 TU{\hfill $U^{13}$ \hfill}
_atom_site_aniso_U_22 TU{\hfill $U^{22}$ \hfill}
_atom_site_aniso_U_23 TU{\hfill $U^{23}$ \hfill}
_atom_site_aniso_U_33 TU{\hfill $U^{33}$ \hfill}
```

Fig. 5.3.5.7. Example map file for use with *ciftex*.

```
#[:\newif\ifproof \prooftrue
#]:\iftwocol\vfill\enddoublecolumns\fi
#a:\pretolerance1000\parskip0pt\tolerance5000
#a:\vskip10pt
#g:
#g:%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
#g:\iftwocol\enddoublecolumns\twocolfalse\fi
#g:\tenbf Experimental
#g:\noindent\ninebf Compound \datablock\vskip2pt
#g:\noindent\nineit Crystal data\par
#g:\vskip2pt\begindoublecolumns\twocoltrue\defaultfont
#U:%%%%%%%%%%% Table of anisotropic U's %%%%%%
#U:\iftwocol\enddoublecolumns\twocolfalse\fi
#U:\rm Table \tableno. \it Anisotropic displacement
#U:parameters \rm (\AA$^2$) for \datablock
#U:\vskip 6pt
```

Fig. 5.3.5.8. Example format file for *ciftex*.

If the initial *character* of the line is a hash mark `#`, the line is treated as a comment and discarded.

5.3.5.3.1.4. *The ancillary format file*

Because a printed paper may be more verbose than its parent CIF data file, it is necessary to add text to the output from *ciftex* to represent section headings, line spaces or other formatting instructions. The program reads an ancillary file, known as the format file, for such additional text.

Each line in the format file begins with a hash mark `#`, a single ASCII character and a colon. The second character is chosen to match the corresponding locator character associated with data names in the map file. The rest of the line is text to be output. When the locator character associated with the data name currently being processed differs from the previous one, the output text from all lines in the format file with the new locator character are output.

The special strings `#[:` and `#]:` indicate text to be emitted at the beginning and end of the output stream, respectively.

Fig. 5.3.5.8 is an example of a simplified format file. The first line is printed at the start of the output TEX file; the second line at the end. The next line will be printed on the first occurrence of a data name flagged with the locator code `a` in the map file. In this example, that will be the name or address of an author of the paper; some typographic directives are emitted immediately before the authors' names and addresses, including the introduction of a blank line ('vertical skip', or 'vskip') of height 10 typographic points.

The lines beginning `#g:` are emitted immediately before the first data name in the group that is associated with locator code `g`. In this example, the effect is to output a heading and subheading before printing the cell-length parameters and to switch to double-column format. The line containing *only* the characters `#g:` provides for the introduction of a blank line into the TEX file, with the sole purpose of making the file more readable by human editors.

The lines beginning `#U:` are emitted at the beginning of the table of anisotropic U values.

The mechanism looks complicated at first sight, but addresses the need to generate headings at standard locations in a printed paper when the exact content of the paper is not known in advance.

The different format for directives in the map and format files means that the same file can be used for both purposes, if required. In practice it is often easier to maintain different files: the same mapping between CIF data names and TEX macros might be common to different journals, while each journal uses its own format file.

5.3.5.3.2. *Invocation of the program*

The program reads a CIF on the standard input channel and outputs TEX code on standard output. There is no provision to specify file names. It is therefore invoked within a Unix-style operating system by a command such as

`ciftex < infile > outfile`

where *infile* and *outfile* are the input and output files respectively; or it may be called as part of a pipeline of procedures:

`program 1 < infile | ciftex | program 2 ...`

A number of command-line options may be supplied to modify the operation of the program. Other than the specification of the map and format files, they are largely relevant to differing house styles for IUCr journals.

The options -map *mapfile* and -format *formatfile* specify the names of the ancillary map and format files. If not specified, they are sought in default locations on the user's file system (different values may be defined when the program is compiled) or as specified in the environment variables `$CIFTEX_MAP` and `$CIFTEX_FORMAT`, respectively.

The options *-H* and *-N* specify, respectively, whether or not hydrogen atoms in coordinate tables should be printed. The hydrogen-atom lines in the table are in fact always emitted on standard output, but in the case of the *-N* option are prefixed by a `%` (TEX comment) character and so ignored by TEX.

Options *-c* and *-F* specify the printing of centred decimal points or commas for decimal points, respectively. Finally, the option *-d* modifies certain assumptions that *ciftex* makes when typesetting CIF dictionaries. The details are of interest only to a specialist.

5.3.5.3.3. *Some general comments*

Although *ciftex* is available for public use and redistribution within the academic community, it is clearly of most interest to users who need to generate typeset representations of the contents of CIFs. Nevertheless, some elements of its design are relevant to other applications that perform on-the-fly file transformations on a strictly syntactic basis.

First, the functionality is very simple, essentially tokenizing the input data stream and exchanging tokens for replacement text as directed. An immediate consequence of this is the need for additional utilities to manipulate the input file if, for example, the data need to be presented in a particular order. In the journals production process, *QUASAR* is used to reorder an input file before passing it to *ciftex*.