ASTRONOMY & ASTROPHYSICS
SUPPLEMENT SERIES

Astron. Astrophys. Suppl. Ser. **73**, 365-372 (1988)

The FITS tables extension

R. H. Harten (1,*), P. Grosbøl (2), E. W. Greisen (3) and D. C. Wells (3)

(1) The Netherlands Foundation for Radio Astronomy, Postbus 2, NL-7990 AA Dwingeloo, the Netherlands
(2) European Southern Observatory, Karl-Schwarzschildstr. 2, D-8046, Garching, F.R.G.
(3) National Radio Astronomy Observatory, Edgemont Road, Charlottesville, VA 22903-2475, U.S.A.

Received July 17, accepted November 22, 1987

Summary. — The rules for generalized extensions for FITS (**) are used to define a FITS extension to transmit catalogs and tables of astronomical data. This « tables extension » is a self-documenting data structure which is able to fully describe the names, formats and units of the columns of a printable ASCII (***) text table. The potential for use of this design to transmit relational database structures is discussed.

Key words : data processing — tape format — data transport.

1. Introduction.

The concept of utilizing a standard flexible format for the transfer of astronomical data has proved to be appealing and designers of software systems for astronomy want to be able to apply it to a variety of data and information structures. For example, several individuals and organizations have advocated that FITS design concepts should be utilized in the formatting of catalogs of astronomical data, such as the star catalogs which are distributed by the astronomical data centers. Commission 5 of the IAU at the Patras meeting in 1982 appointed a « task force » to investigate this concept. At about the same time, an experimental extension of FITS was developed to transmit source position lists and calibration tables in association with image data. Early in 1983 it became apparent that the two efforts should be combined in order to specify a single format designed to transmit arbitrary tabular data. The result of this design process is presented in this paper.

The European FITS Committee and the Working Group for Astronomical Software of the American Astronomical Society agreed, by formal resolutions adopted in 1986, to implement the tables extension design which is described in this paper, effective on 1 January 1987. The two groups will jointly offer a tables extension resolution to IAU Commission 5 for adoption at the IAU General Assembly meeting in Baltimore in 1988. Meanwhile, this « tables extension » is already in production use at several observatories.

2. FITS for catalogs and tables.

There are three main classes of potential applications which have stimulated the development of the tables extension. First, programmers want to transfer standard catalogs or tables such as star or source catalogs with self-documenting column headings. The catalogs are typically in tabular form already and have well defined formats and layout. The second class of application includes the need to transfer observing information such as logs, calibration tables, intermediate tables, etc. which have a relation to observational data. The actual observations can easily be put into a FITS format ; however, the amount of auxiliary information is too large to be included easily as comments and the programmer does not want to give up the tabular form of the information. The final application is the need to transmit tables of results extracted from observational data by data analysis software. For example, a number of programs exist which can automatically detect sources in digital images and write the computed parameters (position, flux, size, spectral index, polarization, etc.) into output files. If these files could be written to tape in a system independent form, astronomers would be able to transmit such tabular data to each other and could utilize software which is designed to manipulate, merge, and intercompare such tables. The extension to the FITS format

(*) Now at RCA Astroelectronics, Princeton, NJ, U.S.A.
(**) FITS : Flexible Image Transport System.
(***) ASCII : American Standard Code for Information Interchange, reference ANSI X3.4-1986 or ISO 646-1983.
Send offprint requests to : P. Grosbøl.

discussed in the following sections is designed to enable all three of these classes of tabular data to be transferred easily from one computing system to another.

When one analyzes the structure of catalogs or tables, one finds that they consist of a number of rows each with a fixed number of elements and a fixed format ; however, the entries do not form a uniform array. What one needs is a means of describing and referring to the contents of each row in the catalog or table. This can be done in the FITS context by treating the table as an array of characters and then defining the location and format of each field within a row of the character array. This solution requires that all catalogs and tables be stored in character format, which causes a significant overhead due to format conversions. However, use of character format rather than binary format is justified by the simplicity of the design concept and by ease of intermixing various data types in tables while avoiding interchange problems due to a multitude of differences in the internal binary data formats of computers. Also, most of the standard catalogs presently available in computer readable form are in character format. For these reasons, the tables extension is based on the conceptual model of a table, containing multiple columns of numbers and « words » with headings at the top of the columns, which is printed on paper using a printer. The printed page is thought of as a *matrix* of ASCII codes, and the tables extension is designed to transmit and document this matrix, with the headings being encoded in the extension header.

The tables FITS extension uses the standard FITS rules (see Wells *et al.*, 1981), plus the new standard for generalized FITS extensions (Grosbøl *et al.*, 1988). The catalog is written in an extension to the main FITS header and is preceded by an extension header which describes the contents of the catalog. The basic concept is as follows. The catalog or table is stored as a large character array. Each row of the catalog or table has the same number of characters. Each row consists of a sequence of fields, and this sequence is the same for each row. The formats of the different fields need not be the same, but the format of a given field must be the same for all rows. Blanks are used to fill out unused space within and between fields. When printed out, the character array should be easily readable and it is recommended that there be a blank between each field within a row. The number of characters in a row and the number of entries or rows in the table or catalog defines the size of the character array.

Each field in a row is described by a series of keywords which describe the name, format, character location within a row, length and units of the information. Using this information, a program could search via variable name, extract the propriate characters and convert them to desired format and units. Because each field in the table is separately defined and because the length of characters in each row is fixed, it is possible to transfer catalogs which contain a large amount of comment information which an automatic decoding program would skip over, yet which can be read by merely printing the entire row. This is especially useful for observation logs and catalogs where a fixed region for comments can be provided. The tables format is quite flexible, allowing one to describe the contents of any standard table or catalog. Extra blank characters may need to be appended to rows of tables in order to insure that each row contains the same number of characters ; this inefficiency is justified by the requirement of regularity in data structures which must be interchanged between diverse systems. This is not a serious problem because the format is primarily intended to be used for the transfer of information rather than the storage of the information.

3. The tables and catalog extension header format.

The table is written in an extension to the basic FITS image, with XTENSION='TABLE'. In the case of many catalogs or tables, there will not be an image, only an extension. But even in these cases, the basic FITS header will still appear in order to preserve compatibility with the older format and to describe the basic characteristics of the FITS file. The extension begins with an extension header which will contain information about the size and contents of the table. This information is provided in the form of keywords, including some of the same keywords as those used in the main header. The rules for the tables extension are in conformance with the rules given in the generalized extension paper (Grosbøl *et al.*, 1988).

A tables extension header begins at the first byte of a new record and will appear in the form shown below. The first eight keywords (XTENSION through TFIELDS) *must* appear, and in the order shown here. For this extension the parameter and group count keywords must be PCOUNT=0 and GCOUNT=1. The keywords TBCOLnnn and TFORMnnn *must* appear somewhere in the header, up to the value of TFIELDS=kkk, in order to properly define the fields of the table. The other keywords, EXTLEVEL, EXTNAME, EXTVER, TTYPEnnn, TUNITnnn, TSCALnnn, TZEROnnn, and TNULLnnn, are all optional. If they are missing default values will be assumed by a reading program :

```
0........1.........2.........3.........4.........5.........6.........7.....
12345678901234567890123456789012345678901234567890123456789012345678901 2...

XTENSION= 'TABLE   '           / Tells the type of extension
BITPIX  =                    8 / Printable ASCII codes (8 is required)
NAXIS   =                    2 / The table is a matrix (2 is required)
NAXIS1  =                 mmmm / Width of table in characters
```

```
0........1.........2.........3.........4.........5.........6.........7.....
123456789012345678901234567890123456789012345678901234567890123456789012...
NAXIS2   =                    nnnn / Number of entries in table (1 is legal)
PCOUNT   =                       0 / Random parameter count must be 0
GCOUNT   =                       1 / Group count must be 1
TFIELDS  =                     kkk / Number of fields in each row
                                   / (i.e., the number of separate pieces of
                                   / information in a row, maximum value of
                                   / nnn in TYPEnnn.)

EXTNAME  = 'name    '              / The name of the table
EXTVER   =                      vv / Version number of table "name" (integer)
EXTLEVEL=                       hh / Hierarchical level (1 is recommended)

TBCOLnnn=                      ccc / Starting char. pos. of field nnn
TFORMnnn= 'qww.dd  '               / Fortran format of field nnn (I,A,F,E,D)
                                   / (NOTE: ww is width of field nnn)

TTYPEnnn= 'type    '               / Type (heading) of field nnn
TUNITnnn= 'unit    '               / Physical units of field nnn
TSCALnnn=                   sss.ss / Scale factor for field nnn
TZEROnnn=                   zzz.zz / Zero point for field nnn
TNULLnnn= 'bbbbbbbb'               / Null (blank) value for field nnn
                                   / (NOTE: exact match left-justified to)
                                   / (the width specified by TFORMnnn.)

END
```

The END card must appear and the remainder of the header record which contains END should be padded with ASCII blanks. In addition to the keywords shown above the extension header may contain additional keywords which describe the table, contain comments, etc. We now give a more extended discussion of the rules associated with the tables keywords.

— TTYPEnnn = 'name' / The name of the nth field in a row.
(Optional, but strongly recommended, default : ' ') It is recommended that only letters (preferably upper case), digits, and underscore (hexadecimal code 5F) be used in the name. The use of identical names for different fields should be avoided.

— TBCOLnnn = value / The beginning column of the field.
(Required) The value is the number of the starting column of the field. The first column of a row is numbered 1.

— TFORMnnn = 'format' / A single value Fortran-77 format code.
(Required) This may use *only* the Fortran formats Iww, Aww, Fww.dd, Eww.dd, and Dww.dd (i.e., integers, characters, and real numbers). Numbers decoded with the I-format may exceed the 16 bit integer range. The F-format and E-format imply single precision (21 bit mantissa accuracy, 6 decimal digits) and the D-format implies double precision (53 bit mantissa accuracy, 16 decimal digits). Note that numbers coded in the F-format style are processed correctly in the E and D-formats and so we do not actually need the F-format, whereas we *do* need to distinguish the floating point precision. The F-format is provided for convenience. Once again : only I, A, F, E and D formats are allowed. Formats such as 2I2 are not allowed ; they should be I2 and I2 (separate fields) instead. A-format fields should be encoded as plain text, without being enclosed in string quotes.

— TUNITnnn = 'unit' / The units of the variable.
(Default : ' ') Physical unit of field e.g., 'K' for degrees Kelvin (see BUNIT in Wells *et al.*, 1981).

— TSCALnnn = value / Scale factor applicable to the value.
(Default : 1.0) Note that this keyword is not relevant for A-format fields.

— TZEROnnn = value / Zero offset to be applied to the value.
(Default : 0.0) Note that this keyword is not relevant for A-format fields. The true value of field nnn is computed as :

$$(\text{value of field nnn in the table}) * \text{TSCALnnn} + \text{TZEROnnn}$$

— TNULLnnn = 'null string' / Character string to indicate a null field.
(Optional, no default) This allows the program to distinguish between a zero value and a nonexistent one. The string should be left justified and is implicitly blank filled to the width of the field (standard Fortran-77 convention). If TNULLnnn is not specified the reading

program should not compare field nnn with the null string ; thus a default value is not needed. It is recommended that the writing program only specifies TNULLnnn if null values actually exist in the fields. This will improve the efficiency of readers which then only need to perform the string comparison for fields with TNULLnnn. Programmers should consider what action their table reading programs should take when they encounter a value which is illegal. For example, suppose the value '* * *' is present in an I3 field but TNULLnnn has not been specified. Probably the reading program should report the error and default to supplying its internal null value.

— AUTHOR = 'name' / Name(s) of original author of catalog.
(Optional, default : ' ') The original author(s) of the catalog should be given.

— REFERENC = 'reference' / Reference of table or catalog.
(Optional, default : ' ') For published catalogs the bibliographic reference should be used.

The default values are assumed if the keywords are not provided. The keywords TBCOLnnn and TFORMnnn are required for any fields which are to be defined in the table or catalog. If these keywords are not specified an automatic decoding routine cannot decode the table.

Note that the width of each field is specified by the width ww given in its format TFORMnnn. Field nnn begins in character position TBCOLnnn and includes ww characters. The sum of the ww widths is not required to equal the true width of the lines of the table, NAXIS1. There is no prohibition against overlapping fields, although the authors are unable to think of a useful example of such usage. Reading programs should report an error in cases where a field is specified to extend beyond the true width NAXIS1.

The format keyword, TFORMnnn, is an area where some degree of common sense must be used by the programmers. To keep things manageable and understandable, each field must have a separate format (multiple formats such as 2I2 are not allowed). If a distinction between + 0 and − 0 is required (i.e., declination) then the sign field should be defined separately. This is absolutely necessary since many computers do not know the difference between + 00 and − 00. The sign should be defined as a character field and checked when decoding the associated number field. Thus the declination defined in degrees/minutes/seconds format would require four fields to be defined, each with its own TTYPEnnn, TFORMnnn, etc. But a declination defined as a floating point number in degrees would only require a single field and would conform to standard FITS rules.

It is recommended that the exponents of real numbers consist of a D or E followed by a sign and 2 numeric digits. Character data should be left justified, while integers and reals should be right justified to prevent the problem of how trailing blanks are treated in different computers. To prevent trailing blanks from defaulting to trailing zeroes, the decoding of table extension data should be done as though the Fortran-77 OPEN statement specifier BLANK is set to NULL (i.e., blank characters should be ignored).

For those creating a new catalog or table format, it is recommended that there should be a blank between the different fields. A general rule should be that the character array containing the table should be easy to read in itself. This makes it possible to print out a number of rows of a table (using the header to determine the number of characters per row, etc.). Unfortunately, some existing catalogs do not have the fields separated by blanks. The FITS format is still valid and applicable for these catalogs ; however, the simple printout option is less attractive.

When creating a table, one may need to distinguish between a 'null' (or undefined) value and a zero value. Normally, blanks in a numeric field will be interpreted as zeroes (standard Fortran-77 rules). In those fields where blanks should be considered to be nulls, the keyword TNULLnnn can be used to specify a 'null' value. Null values must be separately specified for all fields for which they are needed (if TNULLnnn is not defined for a field, then all values in that field are defined). Note also that the null value is a character string of the length ww which is specified by TFORMnn. It is *not* required to be decodable by the format specified by TFORMnnn. For example, a null value of '* * *' might be used for an I3 field.

4. Catalog column headings and units.

The values of TTYPEnnn and TUNITnnn which are shown in the sample headers in this paper should be regarded as examples of possible values. By keying on the field names, TTYPEnnn, programmers can create automatic decoding routines which read and selectively decode the desired fields of a catalog while ignoring the remaining information. This is an excellent means of interfacing the information contained in catalogs with differing formats to standard reduction programs which would use the catalog information. This will allow users to be able to access automatically a wide range of astronomical data, without having to write a different program for each catalog. For this to be completely successful it will be useful to agree on a set of standard field names and units for the contents of catalogs. The authors expect that IAU Commission 5 will produce a standard list of column headings and will recommend any units other than the standard SI units which are needed for existing catalogs.

5. Table data records.

The data records are stored as a large character array, NAXIS1 characters across by NAXIS2 characters long and with NAXIS1 varying most rapidly, starting from the upper left corner of the table. All information is stored as 8-bit printable ASCII characters with the eighth bit (the

« parity » bit) set to zero (i.e., hexadecimal codes in the range 20 through 7E). Special characters with codes outside this range should be avoided since their meaning can be computer system dependent. No integer or real data values occur in the data array. Each data record is 2880 8-bit bytes long. The data records treat the character array as one large bit string. The data records are written one after the other and no attempt is made to prevent partial rows occurring in a record. If the user wishes to force the format to provide complete rows in a data record, then the number of characters per row must be chosen as to divide into 2880 evenly. The final record of the data should be padded with ASCII blanks.

6. Pointers and data structures.

It is possible to use the table format to construct more complicated data structures. Suppose that some application requires multiple tables with a directory. The programmer could establish the directory as a table with whatever value fields he needed and include a column which would give the names of the subordinate tables associated with each entry in the directory table. The field would be a file name expressed as an alphanumeric string. Such a field amounts to a *pointer* which is pointing to another data structure (the subordinate table). Other fields in the directory table could give various qualifiers associated with the subordinate tables in order to help in searching for classes of entries. Obviously, fields in subordinate tables could point to yet other tables. In fact, even more complicated constructions are possible. For example, the fields in the first table could give a column type and row number as well as the name of the subordinate table. Such pointers effectively implement a full relational data base. Implementors should note that although the values of pointer fields will be fully portable, the *interpretation* of the data structures which are implied by them will be application dependent.

7. An example of the tables extension format.

This section contains an example of how one could put part of the AGK3 Star Cat. of Positions and Proper Motions (Dieckvoss, 1975) into FITS format. Each row of the catalog contains sixteen items, which are described in sixteen fields. Two of the fields contain information in character format and the remaining fields contain numerical data. The FITS header describing the catalog and data records for three rows in the catalog are shown in the example below.

The formatting of the value fields in the example follows the rules of basic FITS. In particular, the required keywords obey the required fixed format. The optional keywords in this example also use a fixed format, and this is a recommended practice. Note that string values are always written with at least *eight* characters, beginning in column 11.

The basic FITS header for this catalog would have the following form :

```
0........1.........2.........3.........4.........5.........6.........7.....
12345678901234567890123456789012345678901234567890123456789012345678901 2...
SIMPLE  =                    T / Standard FITS format
BITPIX  =                    8 / character information
NAXIS   =                    0 / No image data array present
EXTEND  =                    T / There may be standard extensions
ORIGIN  = 'CDS     '           / Site which wrote the tape.
DATE    = '23/09/83'           / Date tape was written

COMMENT AGK3 Astrometric catalog, formatted in FITS Tables Format.
COMMENT see: W. Dieckvoss, Hamburg-Bergedorf 1975.
END
```

The extension header begins in a new record :

```
0........1.........2.........3.........4.........5.........6.........7.....
12345678901234567890123456789012345678901234567890123456789012345678901 2...
XTENSION= 'TABLE   '           / Table extension
BITPIX  =                    8 / 8-bits per "pixel"
NAXIS   =                    2 / simple 2-D matrix
NAXIS1  =                   74 / No. of characters per row (=74)
NAXIS2  =                    3 / The number of rows (=3)
PCOUNT  =                    0 / No "random" parameters
GCOUNT  =                    1 / Only one group.
TFIELDS =                   16 / there are 16 fields per row
EXTNAME = 'AGK3    '           / Name of the catalog

TTYPE1  = 'NO      '           / The star number
TBCOL1  =                    1 / start in column 1
TFORM1  = 'A7      '           / 7 character field
```

```
0........1.........2.........3.........4.........5.........6.........7.....
123456789012345678901234567890123456789012345678901234567890123456789012...
TTYPE2  = 'MG      '           / stellar magnitudes
TBCOL2  =                    8 / start in column 8
TFORM2  = 'E4.1    '           / xx.x SP floating point
TUNIT2  = 'MAG     '           / units are magnitudes

TTYPE3  = 'SP      '           / spectral type
TBCOL3  =                   13 / start in column 13
TFORM3  = 'A2      '           / 2 character field
TNULL3  = '        '           / blanck is indefinite value

TTYPE4  = 'RAH     '           / right ascension hours
TBCOL4  =                   16 / start in column 16
TFORM4  = 'I2      '           / 2 digit integer
TUNIT4  = 'HR      '           / units are hours
TNULL4  = '99      '           / null value

TTYPE5  = 'RAM     '           / right ascension minutes
TBCOL5  =                   19 / start in column 19
TFORM5  = 'I2      '           / 2 digit integer
TUNIT5  = 'MIN     '           / minutes of time
TNULL5  = '99      '           / null value

TTYPE6  = 'RAS     '           / right ascension seconds
TBCOL6  =                   22 / start in column 22
TFORM6  = 'E6.3    '           / xx.xxx SP floating point
TUNIT6  = 'S       '           / seconds of time
TNULL6  = '99.999  '           / null value

TTYPE7  = 'DECDSIGN'           / declination sign
TBCOL7  =                   29 / start in column 29
TFORM7  = 'A1      '           / character field

TTYPE8  = 'DECD    '           / declination degrees
TBCOL8  =                   30 / start in column 30
TFORM8  = 'I2      '           / 2 digit integer
TUNIT8  = 'DEG     '           / degrees
TNULL8  = '99      '           / null value

TTYPE9  = 'DECM    '           / declination minutes
TBCOL9  =                   33 / start in column 33
TFORM9  = 'I2      '           / 2 digit integer
TUNIT9  = 'ARCMIN  '           / minutes (angle)
TNULL9  = '99      '           / null value

TTYPE10 = 'DECS    '           / declination seconds
TBCOL10 =                   36 / start in column 36
TFORM10 = 'E5.2    '           / xx.xxx SP floating point
TUNIT10 = 'ARCSEC  '           / seconds (angle)
TNULL10 = '99.99   '           / null value

TTYPE11 = 'EPOCH   '           / epoch of positions
TBCOL11 =                   42 / start in column 42
TFORM11 = 'E7.2    '           / xxxx.xx SP floating point
TUNIT11 = 'YR      '           / units are years

TTYPE12 = 'N       '           / no. photo. obs.
TBCOL12 =                   50 / start in column 50
TFORM12 = 'I1      '           / one digit integer
```

```
0........1.........2.........3.........4.........5.........6.........7....
12345678901234567890123456789012345678901234567890123456789012345678901234
TTYPE13 = 'RAPM    '           / proper motion in r.a.
TBCOL13 =                   52 / start in column 52
TFORM13 = 'E4.3    '           / .xxx SP floating point
TUNIT13 = 'ARCSEC.YR-1'        / units are arc-seconds/yr
TNULL13 = '9999    '           / null value

TTYPE14 = 'DECPM   '           / proper motion in dec.
TBCOL14 =                   57 / start in column 57
TFORM14 = 'E4.0    '           / xxx. SP floating point
TUNIT14 = 'ARCSEC.YR-1'        / units are arc-seconds/yr
TSCAL14 =                0.001 / scale factor = 0.001
                              / (Note use of scale factor!)
TNULL14 = '9999    '           / null value

TTYPE15 = 'DEPOCH  '           / difference in epoch AGK3-AGK2
TBCOL15 =                   62 / start in column 62
TFORM15 = 'E5.2    '           / xx.xx SP floating point
TUNIT15 = 'YR      '           / unit is years

TTYPE16 = 'BD      '           / Bonner Durch. star number
TBCOL16 =                   68 / start in column 68
TFORM16 = 'A7      '           / 7 character field
TNULL16 = '        '           / blanks indicate null

AUTHOR  = 'W. Dieckvoss'
REFERENC= 'AGK3 Astrometric catalog, Hamburg-Bergedorf, 1975'
DATE    = '14/07/82'           / date file was generated

END
```

The extension header shown above has 102 lines and therefore will be written in 3 logical records of 2880 bytes. (The third record will be padded with 6 blank lines). The actual character data of the catalog would begin at the start of the next record. The three lines of 74 characters each (taken from page 46 of Dieckness, 1975) will be in the first 222 bytes of the record.

```
0........1.........2.........3.........4.........5.........6.........7....
12345678901234567890123456789012345678901234567890123456789012345678901234
+82457 11.4 G5 15 30 57.480 +82 15 06.18 1960.37 2 −005 +006 29.99 +82 459
+82458 11.4 F5 15 32 41.150 +82 10 17.17 1958.36 2 −010 +004 27.97 +82 460
+82459 12.1    15 32 42.107 +82 40 28.83 1960.37 2 −018 +004 29.99 +82 461
```

Note that the spectral type field of the third line is blank, which is a null (see keyword TNULL3 above). The remaining 2658 bytes of the record should contain ASCII blanks and a tapemark with follow. The FITS file will contain a total of five records : the basic header in the first record, then three extension header records, and finally one table data record.

8. Conclusions.

The tables extension to the FITS format provides an easy-to-use and convenient means of transferring catalog and tabular information between different computing facilities. The format treats the contents of the tables as a character array. The keywords define the different fields and provide information on the format, units and scale factors. These features facilitate the automatic retrieval and processing of information from astronomical catalogs and from tables created during the automated analysis of observational data.

Acknowledgements.

The authors would like to thank F. Ochsenbein, W. Warren and K. Tritton for their detailed comments on early versions of this paper and on the general problem of encoding and distributing tables and catalogs.

References

DIECKVOSS, W. : 1975, *AGK3 - Star Catalog of Positions and Proper Motions North of – 2.5 Declination* **Vol. 1,** Hamburger Sternwarte, Hamburg.

GREISEN, E. W., HARTEN, R. H. : 1981, *Astron. Astrophys. Suppl. Ser.* **44,** 371.

GROSBØL, P., HARTEN, R. H., GREISEN, E. W., WELLS, D. C. : 1988, *Astron. Astrophys. Suppl. Ser.* (this issue). *IAU Inf. Bull.* **No. 49,** 14, 1983.

WELLS, D. C., GREISEN, E. W., HARTEN, R. H. : 1981, *Astron. Astrophys. Suppl. Ser.* **44,** 363.