The LatexTableMakerClass Reference

Contents

1 Introduction

The LatexTableMakerClass lets you write Latex formatted tables from a Fortran program. The implementation is pretty bare-bones, but at the same time, it is designed to be as simple to use as possible. At its simplest you can generate a table using a double precision array of numbers with one call:

```
PROGRAM main
  USE LatexTableMakerClass
  IMPLICIT NONE
  TYPE(LatexTableMaker) :: TMaker
  REAL(KIND(1.0d0))     :: tableData(4,5)
  CHARACTER(LEN=5)      :: headers(5)  = ["one  ","two  ","three","four ","five "]
  .
  .
  tableData = 0.1d0
```

```
  CALL TMaker % writeTableAsLatex(tableData = tableData, &
                                  headers   = headers,   &
                                  caption   = ''Table Caption'',  &
                                  label     = ''Tab:TabLab'',     &
                                  fileUnit  = 6)
END PROGRAM main
```

This call will generate the Latex to create the following table:

Table 1: Table Caption

one	two	three	four	five
0.100	0.100	0.100	0.100	0.100
0.100	0.100	0.100	0.100	0.100
0.100	0.100	0.100	0.100	0.100
0.100	0.100	0.100	0.100	0.100

Formatting options exist to choose whether or not to include the vertical and horizontal lines, plus the formatting of the numbers in each entry. An instance of the class can also write out an integer array using the writeIntegerTableAsLatex subroutine.

If you have heterogeneous data, specifically columns with different data types, then you can create a table with an array of instances of the LatexTableColumn class. A table column has a title (the header), the format that you want to output the data in, and an array that contains the column's data. An example that has three columns with CHARACTER, INTEGER and DOUBLE PRECISION values is

```
    TYPE(LatexTableColumn), DIMENSION(3) :: tableColumns
    CALL constructLatexTableColumn(tableColumn   = tableColumns(1),&
                                   title         = "Names",&
                                   columnData    = headers,&
                                   columnFormat  = "(A)")
    CALL constructLatexTableColumn(tableColumn   = tableColumns(2),&
                                   title         = "Integers",&
                                   columnData    = intData,&
                                   columnFormat  = "(i6)")
    CALL constructLatexTableColumn(tableColumn   = tableColumns(3),&
                                   title         = "Doubles",&
                                   columnData    = realData,&
                                   columnFormat  = "(f12.5)")
    CALL TMaker % writeTableAsLatex(tableColumnsArray = tableColumns,&
                                          caption = caption, &
                                            label = label, &
                                         fileUnit = 6)
```

The example creates Latex for Table 2.

Table 2: Table Caption

Names	Integers	Doubles
one	1	0.10000
two	2	0.20000
three	3	0.30000
four	4	0.40000
five	5	0.50000

Table 3: Table Caption

Names	one	two	three	four	five
Integers	1	2	3	4	5
Doubles	0.10000	0.20000	0.30000	0.40000	0.50000

You can also transpose the table to look like Table 3. Just include the optional argument.

```
CALL TMaker % writeTableAsLatex(tableColumnsArray = tableColumns,&
                                          caption = caption, &
                                            label = label, &
                                         fileUnit = 6, &
                                        transpose = .TRUE.)
```

2 The Table's Data

The table data is represented as a two dimensional double precision array, a two dimensional array of integers, or as a one dimensional array of LatexTableColumns.

2.1 Table Data as an Array

Table data can be declared as a two dimensional array of double precision, real, or integer values:

```
REAL(KIND(1.0d0)) :: doubleTableData(4,5)
REAL(KIND(1.0e0)) :: realTableData(4,5)
INTEGER           :: integerTableData(4,5)
```

2.2 Table Data as an Array of LatexTableColumns

For more flexibility, specifically for heterogeneous columns, create a table as an array of LatexTableColumns, e.g. a three column table would be defined by

```
TYPE(LatexTableColumn), DIMENSION(3) :: tableColumns
```

You initialize a table column by calling the constructLatexTableColumn subroutine:

```
CALL constructLatexTableColumn(tableColumn = ,&
                               title = ,&
                               columnData =,&
                               columnFormat = )
```

Here tableColumn is of TYPE(LatexTableColumn), title is a CHARACTER variable that will serve as the header for the column, columnData is a singly dimensioned array of any length of either CHARACTER, INTEGER, or DOUBLE PRECISION type. The column format is a Fortran format string that will specify the format of each column entry. An example would be "(f12.5)" for a floating point column, or "(i6)" for a 6 character width integer entry. For CHARACTER variables, it is sufficient to use "(A)".

To write the table, you call the LatexTableMaker subroutine writeTableColumnArrayAsLatex.

```
CALL TMaker % writeTableAsLatex(tableColumnsArray = , &
                                        caption = , &
                                          label = , &
                                       fileUnit = , &
                                      transpose = )
```

where tableColumnsArray is the array of table columns, caption is a CHARACTER variable that creates the caption, label is the table's label and fileUnit is the unit of the file where you want the table written. The transpose argument is optional.

3 Table Formatting

By default, the table is formatted to have horizontal and vertical lines. You set whether you want them or not by calling the setHasLines subroutine, as in

```
CALL TMaker % setHasLines(set = ''NO'')
```

(You can enquire whether it will have lines by calling the LOGICAL function hasLines, e.g.,

```
IF( TMaker % hasLines() )  Do something...
```

)

You can also control and enquire the format of the entries a homogeneous table defined by an array by the methods setTableEntryFormat and tableEntryFormat. For example,

```
CALL TMaker % setTableEntryFormat(formatString = ``(f12.6)'')
PRINT *, ``Table entries formatted as '', TRIM(  TMaker % tableEntryFormat() )
```

These formats are ignored when using defining the table by columns. Instead, you set the entry format column by column as described above.

4 Functions and Subroutines

Here we list the Functions and Subroutines associated with the LatexTableColumn and LatexTableMakerClass classes.

4.1 LatexTableColumn

```
SUBROUTINE constructLatexTableColumn(tableColumn,title,columnData,columnFormat)
   TYPE(LatexTableColumn) :: tableColumn
   CHARACTER(LEN=*)       :: title, columnFormat
   REAL(KIND={KIND(1.0d0) /OR/ KIND(1.0e0) /OR/ INTEGER /OR/ CHARACTER(256)}) :: columnData(:)

FUNCTION numberOfRows(tableColumn) RESULT(N)
   IMPLICIT NONE
   TYPE(LatexTableColumn) :: tableColumn
   INTEGER                :: N
```

4.2 LatexTableMakerClass

All are type bound procedures, so that you call them from an instance of the class, e.g. TMaker % hasLines().

```
SUBROUTINE setHasLines(set)
   IMPLICIT NONE
   CLASS(LatexTableMaker) :: self
   LOGICAL                :: set

LOGICAL FUNCTION hasLines()
   IMPLICIT NONE
   CLASS(LatexTableMaker) :: self

SUBROUTINE setTableEntryFormat(self,formatString)
   IMPLICIT NONE
   CLASS(LatexTableMaker)  :: self
   CHARACTER(LEN=*)        :: formatString

CHARACTER(LEN=256) FUNCTION tableEntryFormat()
   IMPLICIT NONE
```

```
   CLASS(LatexTableMaker)  :: self

SUBROUTINE writeTableAsLatex(tableData, headers, caption, label, fileUnit)
   IMPLICIT NONE
   CLASS(LatexTableMaker) :: self
   INTEGER /OR/   REAL(KIND(1.0d0)) /OR/ REAL(KIND(1.0e0)) :: tableData(:,:)
   CHARACTER(LEN=*)       :: headers(:)
   CHARACTER(LEN=*)       :: caption, label
   INTEGER                :: fileUnit

SUBROUTINE writeTableAsLatex(tableColumnsArray, caption, label, fileUnit,transpose)
   IMPLICIT NONE
   CLASS(LatexTableMaker)  :: self
   TYPE(LatexTableColumn)  :: tableColumnsArray(:)
   CHARACTER(LEN=*)        :: caption, label
   INTEGER                 :: fileUnit
   LOGICAL, OPTIONAL       :: transpose
```