MULTI-COLUMN LAYOUT IN TEX80

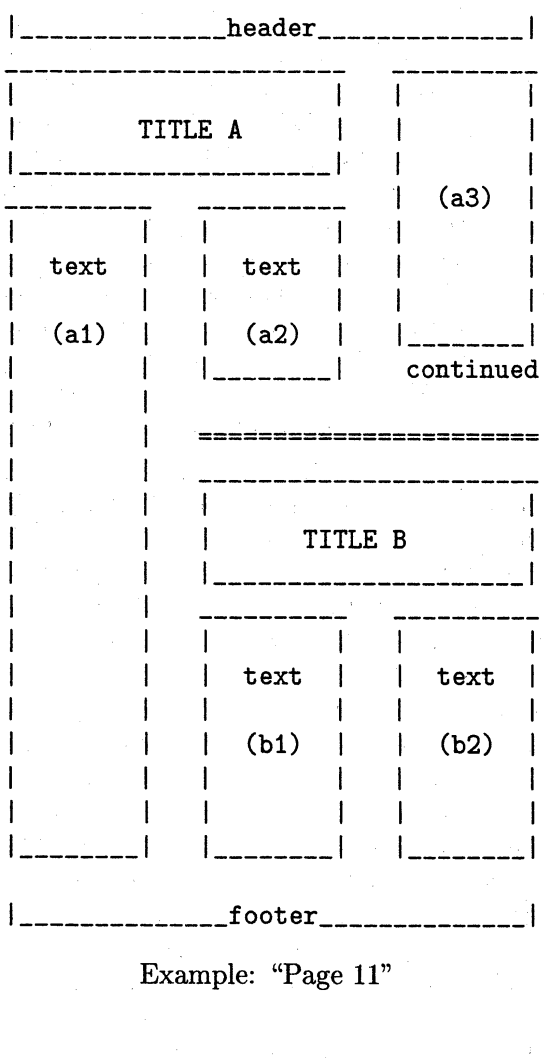
August Mohr

I want to describe how I get TEX to do some of the trickier kinds of 3-column layout as encountered in newsletters and magazines. It's not easy, but it works.

I should start out with the caveat that I program in TEX80 (TEX in C by Unidot/TyX) under UNIX, and not in TEX82. What I describe here would be handled differently in TEX82, I'm sure.

The Problem

The figure below is a sketch of a hypothetical page from the middle of a newsletter. Article "B" is to fit on the page in its entirety, while Article "A" continues onto the next page. In a page like this, pieces *a2* and *a3* must balance, while making allowance for *TITLE A* and the continuation line, and their sizes must also depend on the length of Article B. Pieces *b1* and *b2* must also balance.



Example: "Page 11"

A Solution

In my solution, the \output routine keeps changing the \vsize, and saving the pages in successive \boxes. When the proper number of boxes have been filled, it puts out the completely assembled page. There are two primary pieces in this description: the "run" file, which identifies the text files, page number, and so on; and the "layout" file, which describes the structure into which the text will be fit.

The UNIX file system has been very helpful in keeping track of all the different files I use, and in allowing me to make my TEX programs "modular". I primarily use three directories within my working directory: **tex**, where I keep all the TEX macros and page structures; **text.x**, where I keep the text in its TEX form; and **run.pg**, where I keep the files which TEX initially reads. A subdirectory of **tex** is **tex/layout.pg** where the page layout descriptions are kept.

The Run File

To layout a page I create a "run" file, a numbered file in the directory **run.pg** such as **run.pg/11**, the run file for page 11. More than one page may be included in the run file, and I name the file accordingly (e.g. **run.pg/13.14**). I find it useful to keep the run files small, i.e. one or two pages, to speed processing time during the testing and debugging cycle. It is this run file that TEX initially sees (e.g. from a command line such as **tex run.pg/11**).

File: **run.pg/11**

```
\input tex/newsformat.x
\input tex/layout.pg/11
\beginsheet{11}
\dummynext

\tenpoint
\article{B.filename}
\recyclebox2
\article{A.filename}
\end
```

Within the run file, several files are \input. The main "driver" is **newsformat.x**, which is in the directory **tex**. This file \input-s all the other definition files, defines the \output routine, and defines all the macros used in the run files. The sub-directory **layout.pg** has a layout description for each page of the newsletter. If the run file covered more than one page, the appropriate layout file would also be \input.

The run file next identifies the page number and the file name of each article, and then \end-s. Notice that Article B is read before Article A. That is because the sizes of the pieces of Article A will depend upon the size of Article B. More on this later.

The following definitions are all excerpted from the file **newsformat.x**. The macro \article,

```
\def\article#1{\input text.x/#1 \eject}
```

presumes that its argument is a file in the directory **text.x**. The macro \recyclebox2 has a simple definition:

```
\def\recyclebox#1{\unbox#1\vfil\eject}
```

but that will not mean much until you know how \box2 gets filled. I'll get to that when describing the \output routine and the layout file.

Before going on, there are two more macros in the run file that should be defined. \dummynext sets a simple flag, indicating whether the layout of the next page (in our example, page 12) should be used for material that overflows the current page, or whether a simple "dummy" page should be used instead.

```
\def\dummynext{\gdef\nextisdummy{T}}
\def\nextisdummy{F}
```

If \dummynext is not used, the default is automatically defined. This flag makes it possible to debug page 11 before knowing that page 12 works. Otherwise, the run could die due to a fault on page 12, even if page 11 was perfect. I presume that this is peculiar to the C version of TEX80, that does not allow interactive debugging.

The last macro left to define in the run file is \beginsheet, which sets the page number and the initial \vsize. Like the macros above, it is defined in the file **newsformat.x**. Since the initial \vsize must be set according to the page number, some comparison test must be done. TEX's \if allows comparisons of single characters, and \ifdimen can compare dimensions. Rather than convert a page number into inches, an aesthetically unpleasing mixing of number and quantity, I chose to use \ifx to compare the numbers as strings. To do this, some initial definitions are required. I name these macros using Roman numerals to distinguish page numbers. (Getting around TEX's prohibition on numbers in macro names can be a bother. At first I tried letters A-P for pages 1 through 16 and used the simple \if, but Roman has proven easier to read, if a little more convoluted to program.) These are also used in the \output routine to identify page numbers.

```
\def\I    {1}   \def\II   {2}
\def\III  {3}   \def\IV   {4}
\def\V    {5}   \def\VI   {6}
\def\VII  {7}   \def\VIII {8}
\def\IX   {9}   \def\X    {10}
\def\XI   {11}  \def\XII  {12}
\def\XIII{13}   \def\XIV  {14}
\def\XV   {15}  \def\XVI  {16}
```

Only 16 numbers are defined because I only needed 16 pages in my publication. More may be added as needed, limited only by the space available for \def-s. (If more pages are to be allowed for, changes must be made in this list, in \beginsheet, and in the \output routine. More layout files must also be created.) With these definitions established, the definition of \beginsheet should be clearer.

```
\def\beginsheet #1{\xdef\sheet{#1}
          \setcount0 \sheet
          \ifx\sheet\I   {\setIvone}
     \else{\ifx\sheet\II  {\setIIvone}
     \else{\ifx\sheet\III {\setIIIvone}
     \else{\ifx\sheet\IV  {\setIVvone}
     \else{\ifx\sheet\V   {\setVvone}
     \else{\ifx\sheet\VI  {\setVIvone}
     \else{\ifx\sheet\VII {\setVIIvone}
     \else{\ifx\sheet\VIII{\setVIIIvone}
     \else{\ifx\sheet\IX  {\setIXvone}
     \else{\ifx\sheet\X   {\setXvone}
     \else{\ifx\sheet\XI  {\setXIvone}
     \else{\ifx\sheet\XII {\setXIIvone}
     \else{\ifx\sheet\XIII{\setXIIIvone}
     \else{\ifx\sheet\XIV {\setXIVvone}
     \else{\ifx\sheet\XV  {\setXVvone}
     \else{\ifx\sheet\XVI {\setXVIvone}
     \else{\xdef\sheet{Z}\setZvone
          }}}}}}}}}}}}}}}}
     } %end beginsheet
```

In the example page, I use \beginsheet{11}, so the macro \setXIvone is called. This is defined in the file **layout.pg/11**. If the run file is set up properly, only the macro corresponding to the current page (\setNNvone: NN = I, II,...,XVI) will be accessed. The others will remain undefined because the layout files defining them will not have been read. This modularity saves processing time and definition space.

If \beginsheet is given an argument that it does not recognize, then \sheet is set to "Z" and \setZvone is called. \setZvone is defined in a layout file that describes a simple default or "dummy" layout. The default layout file (**tex/layout.pg/Z**) should be \input within **tex/newsformat.x** so that the default is always available, no matter what the run file contains. The construction of the dummy

layout file should be a simple task, once I explain the required contents for layout files.

The Layout File

The layout file for the example page will illustrate all the important points. A layout file has two major parts: the \vsize definitions and the "build" macro, which defines the page structure. The \vsize definitions themselves have two parts, which I will describe separately.

The first part of the \vsize definitions defines a macro, \setXIvdefs, which is used in the \output routine. When this macro is interpreted, a series of definitions are established. Every macro that is defined within \setXIvdefs may used by the \output routine. Every macro that is used in a definition, which will have "XI" as part of its name in this case, will be defined later in the layout file.

tex/layout.pg/11 – part 1.

```
\def\setXIvdefs{
    \gdef\setvone    {\setXIvone}
    \gdef\setvtwo    {\setXIvtwo}
    \gdef\setvthree  {\setXIvthree}
    \gdef\setvfour   {\setXIvfour}
    \gdef\setvfive   {\setXIvfive}
    \gdef\setvsix    {\setXIvsix}
    \gdef\setvseven  {\setXIvseven}
    \gdef\setveight  {\setXIveight}
    \gdef\setvnine   {\setXIvnine}
    \gdef\setvzero   {\setXIvzero}
    \gdef\setvnext   {\setXIvnext}
    \gdef\numberboxes{\XInumberboxes}
    }
\def\setXIvnext {\setXIIvone}
```

This first part also defines the \setXIvnext macro, which will be used to set the \vsize for the first piece of the following page (page 12). This first part of the \vsize definitions is repeated in every layout file, except that the "XI" in each macro name is changed to match the appropriate page number, and the \set...next is changed to match the following page. Note that \setXIIvone need not be defined in order for the page to be processed properly. The flag set by \dummynext will cause \setZvnext to be used instead. \setXIIvone would be used if I were setting pages 12 and 13 together in this run.

The next part of the \vsize definitions is not strictly required. I create my layout files by starting with a standard "template" file. I include this as a reminder of exactly what needs to be defined in the file, and in what format. The template includes all the macros that I must define, with "default" definitions for each.

As I redefine each macro to reflect the actual layout of the page, I comment out the initial definition. Thus, all macros in this section that are commented out will be defined further on, and any not commented should not be accessed by TEX. (Note that \fullcol, used below as a default, is defined as a dimension, the height of a full column of text, e.g. \def\fullcol{56pc}.)

tex/layout.pg/11 – part 2.

```
%\def\setXIvone   {\vsize\fullcol}
%\def\setXIvtwo   {\vsize\fullcol}
%\def\setXIvthree {\vsize\fullcol}
%\def\setXIvfour  {\vsize\fullcol}
%\def\setXIvfive  {\vsize\fullcol}
%\def\setXIvsix   {\vsize\fullcol}
%\def\setXIvseven {\vsize\fullcol}
%\def\setXIveight {\vsize\fullcol}
 \def\setXIvnine  {\vsize\fullcol}
 \def\setXIvzero  {\vsize\fullcol}
%\def\XInumberboxes{3}
```

The last part of the \vsize definitions contains the actual definitions of the macros that were commented out above. These definitions come in two flavors: an explicit statement of a dimension, which may be a macro such as \fullcol, or else a description of how to calculate the \vsize.

tex/layout.pg/11 – part 3.

```
\def\setXIvone{\vsize 99pc
    \hsize\twowide}   %unboxed in set7
\def\setXIvtwo{\vsize 999pc
    \hsize\normalh}   %unboxed in set3
  %this box is recycled in the run file
\def\setXIvthree{\save2
    \vbox{\unbox2}\vsize 0.5ht2}
\def\setXIvfour{\vsize 999pc}
                      %unboxed in set7
```

These four definitions are required to process Article B, the first one read in the run file. What I am doing here is determining the \vsize and \hsize that will be in effect when each piece of the page is read. Since the layout for the page treats the titles as separate pieces of text (not necessary if the title were only one column wide) there needs to be a separate \vsize for each title. (The \hsize is not necessary in "three" and "four" since it does not change.)

As the text is read, TEX builds onto the current \page until an \eject is encountered or the current \vsize is exceeded. To ensure that the pieces are broken properly, I give a somewhat large \vsize initially, and put an explicit \eject into the text between the title and the beginning of the text. By

giving a large, arbitrary \vsize, I ensure that the \eject will control TEX.

In operation, \beginsheet (in the run file) accesses \setXIvone and sets the initial \vsize. When the \eject is encountered, the \output routine places that \page in \box1, and uses \setXIvtwo to set the new \vsize. Each time TEX enters the \output routine, the successive pieces of text are placed in successive boxes and a new \vsize is set.

Remember that in the run file Article B was read before Article A and that after Article B was read, the macro \recyclebox2 was used. Look back at the run file and at the definitions of its macros to see what will happen at the end of Article B.

The macro \article has an explicit \eject at the end, so those 999pc will not be filled. The full text of Article B (less the title that came before the previous \eject) will be saved in \box2. \setXIvthree will be used to set the \vsize for the next piece. \setXIvthree unboxes \box2 and saves it again, effectively relaxing any glue that may have been stretched to fill up that 999pc \vsize. Now \box2 is the "actual" size of that text. The new \vsize is set to be just half of the height of \box2. All this happens at the end of Article B, at the \eject supplied by the \article macro, under the control of the \output routine.

When \recyclebox unboxes \box2 again, it is read in as "text" by TEX. Be careful. It is not really text. It is a list of all the horizontal boxes that were collected in the vertical box that became \box2. Fonts, paragraphing, macros, and so on, may not be changed. The horizontal boxes will be read until an \eject is encountered, or \vsize is exceeded. In this case, it will be the \vsize set by \setXIvthree that controls the end of the piece. The new piece, saved in \box3, will become the left-hand piece of text for Article B.

The text that is being split may contain an odd or an even number of lines, and paragraph spacing may not be an even line-space, so it is unlikely that exactly half of \box2 will fall exactly at the bottom of a line. For this reason, \setXIvfour allows for a larger \vsize than that used in \setXIvthree, and the \eject at the end of the \recyclebox definition will control the end of the next piece. That piece, saved in \box4, will be the right-hand piece of Article B.

Note the comments at the ends of the lines that have the arbitrary sizes (99pc or 999pc). I find it useful to keep track of where such oversize boxes will get taken down to their proper size.

So far I have dealt only with Article B, which is presumed to fit in its entirety within the planned page format. The only constraint upon the sizes of the pieces of Article B is that the two columns balance. Article A, on the other hand, has more complex dependencies for the sizes of its pieces.

Before describing the \vsize definitions for Article A, I need to define a few calculation tools.

Calculation tools

```
\def\add#1to#2into#3{\save0\vbox
     {\vskip#2\vskip#1}\xdef#3{\ht0}}
\def\subtract#1from#2into#3{\save0\vbox
     {\vskip#2\vskip-#1}\xdef#3{\ht0}}
\def\setgreaterof#1#2to#3{\ifdimen #1
                < #2{\xdef #3{#2}}
           \else{\xdef #3{#1}}}
```

In these calculation tools, parameters #1 and #2 are expected to be dimensions, or macros defined as dimensions, while #3 is the name of a macro to be defined to be the result of the calculation.

Now I can go on with the \vsize definitions for this page, that cover Article B.

tex/layout.pg/11 – part 4.

```
\def\setXIvfive{\vsize 99pc
    \hsize\twowide}   %unboxed in set6
\def\setXIvsix{\save5\vbox{\unbox5}\!
    \subtract{\ht5}from{\fullcol
                       }into{\XIvsix}\!
    \vsize\XIvsix\hsize\normalh}
\def\setXIvseven{\save1\vbox{\unbox1}\!
    \save4\vbox{\unbox4}\!
    \subtract{\ht1}from{\ht6
                }into{\stilltoomuch}\!
    \setgreaterof{\ht3}{\ht4
                }to{\prevhigh}\!
    \subtract{\prevhigh
                }from{\stilltoomuch
                }into{\XIvseven}\!
    \vsize\XIvseven}
\def\setXIveight
   {\add{\ht5}to{\ht7}into{\prevcol}\!
    \subtract{\contheight}from{\prevcol
                       }into{\XIveight}\!
    \vsize\XIveight}
\def\XInumberboxes{8}
```

Piece five, the title of Article A, is handled just like piece one. It, too, needs an explicit \eject between the title and the text. To get the \vsize for piece six, I subtract the relaxed height of piece five (\box5) from the height of a full column (\fullcol). Note the temporary macro \XIvsix. Although it is named with Roman numerals, as the "set" macros, it could have been

called anything, e.g. \finalv. The same is true for \XIvseven and \XIveight. All three could even have the same name without affecting the calculations. The macros \prevhigh and \prevcol are also temporary, and could be named anything.

To keep track of the calculation for piece seven, remember that \box2 was thrown away by \recyclebox, so Article B consists of \box1, \box3, and \box4. Piece seven then fits into the space left in a column after taking out \box1 (the title of Article B), \box5 (the title of Article A), and the taller of \box3 and \box4 (the text pieces of Article B which are supposed to balance).

Piece eight is then to be as tall as the combination of \box5 and \box7, after taking out the height of the "continuation line" (\contheight).

That is it! When all eight boxes are filled (OK, seven are filled; \box2 is now null) I can build the final page. The \XInumberboxes macro tells the \output routine how many boxes to fill for this page, before going on to the next page. When that number is reached, the \output routine calls the "build" macro.

The "build" macro describes how the finished "sheet" is assembled from its "pieces". Since it is used inside the output routine, between the header and the footer, it must be a vertical list.

tex/layout.pg/11 – part 5.

```
\def\buildXI{\!
  \hbox to \sheetwidth
    {\hbox to \normalh
      {\vbox to \fullcol
        {\box5\copy6}\!
      \hss}\!
    \hfil\vrule height 1ht6\hfil
    \vbox to \fullcol
      {\hbox to \twowide
        {\save8\vbox{\box8\contline}\!
        \box7\!
        \hfil\vrule height 1ht8 \hfil
        \box8}
      \vfil
      \hrule width \twowide height 4pt
      \vfil
      \vbox
        {\box1
        \hbox to \twowide
          {\ifdimen 1ht3 > 1ht4
            {\copy3\!
            \hfil\vrule height 1ht3\hfil
            \vbox to 1ht3{\unbox4}}
          \else
            {\vbox to 1ht4{\unbox3}\!
            \hfil\vrule height 1ht4\hfi
            \box4}\!
          }%end \hbox to \twowide
        }%end \vbox
      }%end  \vbox to \fullcol
    }\!  %end \hbox to \sheetwidth
  } %end \buildXI
```

Most sheets that I build end up having a horizontal outer level, a horizontal stack of columns so any \buildNN is usually a single \hbox to \sheetwidth. Just about every \buildNN should have a height of \fullcol, the exceptions being pages such as Page 1 that do not get the normal header and footer. Such exceptions will require additional tests in the \output routine to change or leave out the header and footer, which I have not included.

The macro above, \buildXI, has three components within its outer \hbox: an \hbox, a \vrule and a \vbox.

Notice that the \hbox has a width of \normalh the normal \hsize of a text column. Inside this \hbox is a \vbox containing \box5 and \box6, and some \hss (horizontal shrink and stretch). The \hss makes up the difference between the width of the outer \hbox and the width of the enclosed \box5, which had an \hsize of \twowide, as wide as two columns of type.

The \vrule that is part of the outer \hbox and all the other rules on the page, are optional design elements. I like to put them in because they make it much easier to read narrow tight columns of text, and because it is so easy to do compared to hand paste-up.

The last component of the all-encompassing \hbox to \sheetwidth is a \vbox consisting of three parts itself. The top half is an \hbox containing \box7 and \box8, next there is a separating \hrule (thicker than the hairline rule separating the columns), and the bottom half is a \vbox containing \box1, \box3, and \box4 (\box2 got recycled).

In the upper half, \box8 gets the pre-defined continuation line (\contline) added to the end of it. This makes it easy to get the \vrule between \box7 and \box8 to be the right height. The \vbox in the lower half contains \box1 followed by an \hbox containing \box3 and \box4. The \if ... \else ... construction is used to allow for not knowing whether \box3 is taller than \box4. I could have used \setgreaterof here, and then \unbox-ed both into \vbox-es of the same size, but I chose this method for variety.

This is essentially all that has to be done for each page: creating the run file and the layout file. (This includes, of course, debugging the layout and discovering where you run up against the system's limits).

As described above, the text files only need to allow for this structure in one way: if the the title and the text are to be treated as two separate pieces, such as when the \hsize changes from two columns wide to one column wide, the article-file must contain an explicit \eject between the parts. The new \hsize may then be set either within the text file, or else in the layout file, as part of the \vsize definitions (viz. \def\XIvtwo{999pc\hsize\normalh}).

So far I have only described the run files and the layout files, which are all that, in practice, need to be changed between pages or between editions. Now on to the \output routine, which does not change at all.

The Output Routine

The \output routine is responsible for changing the \vsize, and saving the completed \page-s in successive \box-es. When the proper number of boxes have been filled, it puts out the completely assembled page. Because the \output routine is a single, fairly long block, I will not interrupt my description of its workings to show segments of it, but instead will save the complete code until the end. There are many comments in the code, which should help make its workings clearer.

The \vsize-s for our sample page are defined in the layout file by the macros \setXIvone, \setXIvtwo, etc. As I said before, the \output routine uses macros \setvone, \setvtwo, etc. It is \setXIvdefs which connects those two series and links our definitions to the macros used in the output routine. Thus, the first thing that happens in the \output routine is to determine which sheet we are in, and read in the appropriate definitions. This only happens if this is the first time that the \output routine has been entered, i.e. when \pieces is "0", meaning that no pieces have previously been saved for this sheet. This process is similar to that in \beginsheet.

The second thing that happens is the saving of the \page in the appropriate \box and the incrementing of \pieces. The \vsize of the the next \page is also set.

The \output routine cannot set the initial \vsize, that of the first piece on the first sheet. In the run file, the macro \beginsheet does that by using the macro \setXIvone. The \output routine can, however, set the \vsize for the first of the next sheet, through the macro \setvnext which has been defined by \setXIvdefs to be \setXIvnext which calls \setXIIvone. If \setXIIvnext is not defined, i.e. because the file **tex/layout.pg/12** has not been \input, the macro \dummynext must be used to override \setvnext and use the dummy layout defined in **tex/layout.pg/Z**. This is what I did in the sample run file.

After saving the current \page, the \output routine tests to see if \pieces, the number of boxes filled so far, is equal to \numberboxes, the number of boxes used in the page layout. If not, TEX leaves the \output routine and continues reading text. Although the \output routine allows for all ten boxes to be filled, ending with \box0, in practice this is not a good idea, since the calculation tools, defined above, use \box0, and could inadvertently wipe out the stored text.

If there are now enough boxes to fill the sheet, the \output routine prepares for the next sheet under the control of the flag set by \dummynext, and defines the \buildit macro to be \buildXI.

So far I have not mentioned where the header and footer are defined. I recommend creating a file of all the "edition-dependent" macros and having that \input within **tex/newsformat.x**.

At this point, all that is left to do is to set the footer according to the page number in \count0, and output the finished sheet, with a \header, the \buildit macro describing the body, and the \footer. Then I clean up the \box-es so there is not a lot of stuff hanging around using up memory (\def\null{\hbox{}}), and it is all done.

Summary

With the \output routine taking care of most of the processing, creating a complete page is mostly a matter of creating the two files which describe it. The run file names the page number and the article-files, and the layout file defines the \vsize-s and how the finished page is built. The only other consideration is that the article-files (the text) must have an explicit \eject whenever you want the parts treated as separate pieces in the layout, such as when \hsize changes.

Well, it's not as simple as "\input basic \input text \end", but this is pretty complex page layout we're doing. Simplifications from here might include (barring a simpler design in the first place) having a "library" of a couple dozen parameterized "standard" pages, and calling them up as needed.

Appendix: The Output Routine Code

```
%----------------------------------------------------------------------
% \pieces contains the number of pieces already stored, as a character,
% not a count.    Start assuming that nothing has been put into the sheet,
% and the first text goes in piece 1.

\def\pieces{0}

%----------------------------------------------------------------------
% Since this is read in as part of the initialization in newsformat.x,
% both of these will be overridden if \beginsheet is used.
% Being able to use \setZvone presumes that layout.pg/Z has already
% been input.

\def\sheet{Z}   % sheet is the number of the sheet we're working on.
                % If nothing else is stated, use "Z" as dummy/default.

\setZvone       % Before setting any type, we need the vsize
                % for the first piece of the sheet.

%======================================================================
%
%               ***** THE OUTPUT ROUTINE *****
%

\output{

%----------------------------------------------------------------------
% When we have a full page-piece, if this is the first piece of the
% sheet, then set the vsizes for the sheet that we're in.
% The initial vsize must be set elsewhere.
% The default \setZvone or \beginsheet does that.

\if 0\pieces
     {\ifx \sheet\I     {\setIvdefs}
\else{\ifx \sheet\II    {\setIIvdefs}
\else{\ifx \sheet\III   {\setIIIvdefs}
\else{\ifx \sheet\IV    {\setIVvdefs}
\else{\ifx \sheet\V     {\setVvdefs}
\else{\ifx \sheet\VI    {\setVIvdefs}
\else{\ifx \sheet\VII   {\setVIIvdefs}
\else{\ifx \sheet\VIII  {\setVIIIvdefs}
\else{\ifx \sheet\IX    {\setIXvdefs}
\else{\ifx \sheet\X     {\setXvdefs}
\else{\ifx \sheet\XI    {\setXIvdefs}
\else{\ifx \sheet\XII   {\setXIIvdefs}
\else{\ifx \sheet\XIII  {\setXIIIvdefs}
\else{\ifx \sheet\XIV   {\setXIVvdefs}
\else{\ifx \sheet\XV    {\setXVvdefs}
\else{\ifx \sheet\XVI   {\setXVIvdefs}
\else{\gdef\sheet{Z}\setZvdefs
     }}}}}}}}}}}}}}}}}}}
\else{
}       %-------------------------------------------------------------
        % Don't do anything here if this isn't the first piece.
```

```
%---------------------------------------------------------------------
% Now that vdefs are set, save the piece and set \vsize for the next.
% \numberboxes is set inside \setNNvdefs; where NN is I, II,..., XVI, Z.
% If there are already N pieces, then set pieces=N+1 and save the
% page in a new box.    Call the setv...  macro defined in \setNNvdefs.

     \if 0\pieces{\gdef\pieces{1}\save1\page\setvtwo}
\else{\if 1\pieces{\gdef\pieces{2}\save2\page\setvthree}
\else{\if 2\pieces{\gdef\pieces{3}\save3\page\setvfour}
\else{\if 3\pieces{\gdef\pieces{4}\save4\page\setvfive}
\else{\if 4\pieces{\gdef\pieces{5}\save5\page\setvsix}
\else{\if 5\pieces{\gdef\pieces{6}\save6\page\setvseven}
\else{\if 6\pieces{\gdef\pieces{7}\save7\page\setveight}
\else{\if 7\pieces{\gdef\pieces{8}\save8\page\setvnine}
\else{\if 8\pieces{\gdef\pieces{9}\save9\page\setvzero}
\else{\if 9\pieces{\gdef\pieces{0}\save0\page\setvnext}
\else{          %-------------------------------------------------------
    }}}}}}}}}} % This last \else should never be encountered, since
               % \pieces will only ever be 0-9.

%---------------------------------------------------------------------
% If we have all the pieces for the sheet, reset the number of pieces.
% Use the \nextisdummy flag to override \setvnext by calling \setZvdefs.
% Reset the \sheet id to the next one, and define the \buildit routine.
%
% Then proceed to build the full sheet.
%
% The "\if \pieces\numberboxes{...}" block continues to the end of the
% \output routine.    The matching \else, accessed each time a piece is
% saved but the sheet is not complete, is at the very end.

\if \pieces\numberboxes
          {\gdef\pieces{0}
           \if T\nextisdummy{\setZvdefs}\else{}
           \setvnext
           \ifx \sheet\I   {\xdef\sheet{\II}  \gdef\buildit{\buildI}}
     \else{\ifx \sheet\II  {\xdef\sheet{\III} \gdef\buildit{\buildII}}
     \else{\ifx \sheet\III {\xdef\sheet{\IV}  \gdef\buildit{\buildIII}}
     \else{\ifx \sheet\IV  {\xdef\sheet{\V}   \gdef\buildit{\buildIV}}
     \else{\ifx \sheet\V   {\xdef\sheet{\VI}  \gdef\buildit{\buildV}}
     \else{\ifx \sheet\VI  {\xdef\sheet{\VII} \gdef\buildit{\buildVI}}
     \else{\ifx \sheet\VII {\xdef\sheet{\VIII}\gdef\buildit{\buildVII}}
     \else{\ifx \sheet\VIII{\xdef\sheet{\IX}  \gdef\buildit{\buildVIII}}
     \else{\ifx \sheet\IX  {\xdef\sheet{\X}   \gdef\buildit{\buildIX}}
     \else{\ifx \sheet\X   {\xdef\sheet{\XI}  \gdef\buildit{\buildX}}
     \else{\ifx \sheet\XI  {\xdef\sheet{\XII} \gdef\buildit{\buildXI}}
     \else{\ifx \sheet\XII {\xdef\sheet{\XIII}\gdef\buildit{\buildXII}}
     \else{\ifx \sheet\XIII{\xdef\sheet{\XIV} \gdef\buildit{\buildXIII}}
     \else{\ifx \sheet\XIV {\xdef\sheet{\XV}  \gdef\buildit{\buildXIV}}
     \else{\ifx \sheet\XV  {\xdef\sheet{\XVI} \gdef\buildit{\buildXV}}
     \else{\ifx \sheet\XVI {\xdef\sheet{Z}    \gdef\buildit{\buildXVI}}
     \else{\gdef\sheet{Z}\gdef\buildit{\buildZ}
          }}}}}}}}}}}}}}}}
           \if T\nextisdummy{\gdef\sheet{Z}}\else{}
                % Z is used for dummy pages and for sheets beyond \XVI.
```

```
        %-----------------------------------------------------------------
        % Having set up for the next sheet and defined \buildit,
        % proceed to build a complete output sheet:

        \ifeven0{\gdef\footer{\evenfoot}}
        \else   {\gdef\footer{\oddfoot}}

        \vbox to \sheeth
                {\header
                \buildit
                \footer
                \advcount0
                }

        %-----------------------------------------------------------------
        % Clean up by emptying all the boxes.

        \save1\null\save2\null\save3\null\save4\null\save5\null\save6\null
        \save7\null\save8\null\save9\null\save0\null

        %-----------------------------------------------------------------
        % This is the end of ``\if \pieces\numberboxes {..."
        % sheet-building block.

        }
\else{} % This else is accessed each time the sheet is not complete.

} % end of output block.

%=======================================================================

%=======================================================================
%
% (c) Copyright 1981, 1982, 1983, 1985 by August Mohr.    The described
% macros and TeX code may be used by TUG members for internal company
% and personal purposes only.    This TeX code may not be distributed, in
% any form, to non-members without written permission from the author.
% Any commercial use of this code, such as but not limited to producing
% a product, or part of a product, for sale, may only be with written
% permission of the author.    Any distribution of these macros and/or
% other code, must contain this copyright notice.
%
%=======================================================================
```