TABLE OF CONTENTS

SLAC–PUB–7
October 1984

EVERYTHING YOU WANTED TO KNOW ABOUT **PHYZZX** BUT DIDN'T KNOW TO ASK

M. WEINSTEIN[⋆]

Stanford Linear Accelerator Center
Stanford University, Stanford, California 94309

⋆ Work supported by the Department of Energy, contract DE–AC03–76SF00515.

1. INTRODUCTION

Hi TEX fans, the time has come to document the new version of the macro package, PHYZZX. Vadim Kaplunovsky has updated it to run under TEX1. Since we envision this to be the penultimate version of PHYZZX, except for minor corrections to remove heretofore undiscovered bugs, it now pays to explain what PHYZZX is and how to use its various features in some detail. For those of you who are familiar with the old version of PHYZZX you can rest easy, this version has been designed to be as upward compatible as possible. Unfortunately, there are a few minor changes in the way some macros work simply because Knuth changed TEX and made it impossible for us to keep some of the old commands. In general the changes make things easier to use and any file written for TEX97 will, with very few changes, run on TEX1. For this reason the adventurous user can begin by ignoring this writeup completely and try to run his old files. If luck is with you, you will not have to change anything; if luck deserts you, then either PHYZZX will help you along (only somewhat likely) or you will have to do a little reading.

Our present incarnation of PHYZZX resides in two places. The FMT file (a term designed to strike terror into the hearts of those who have not been initiated into the high mysteries) is on the T–disk, and this is the version you really want to use. One uses the FMT file because it loads much faster and so you don't have the interminable wait for something to happen while the computer is reading all of the definitions contained in PHYZZX TEX. To use the FMT file all you have to do is make the first command to TEX say `&phyzzx` instead of putting

\input phyzzx

in the file.

For the curious we try to keep an updated version of the file PHYZZX TEX on the theory group disk. People interested in mucking around in the guts of this file in order to see how things were done and what goodies, not described in this writeup (and there are many), are available can access this disk by saying

GIME PUB$TH

If you wish to use this file to convince yourself how much better it is to use an FMT file, you can do this by accessing the PUB$TH disk and starting your file with

\input phyzzx

For cognoscenti: as in earlier incarnations you can tailor much of PHYZZX to your liking by having a file MYPHYX TEX on your A disk. Each time you run PHYZZX it loads this file. Any definitions which are in this file then take precedence over those specified in PHYZZX. So much for generalities, now on to specifics.

2. BASICS

2.1. What is TeX ?

In order to make this as simple as possible for the first time user, we will say a few words about what TeX is, before going on to talk about what PHYZZX does.

TeX is the baby of Donald E. Knuth of Stanford University and it is our choice for the best available text processor for use at SLAC. *Clear as mud!* you say. *What the hell is a text processor ?*, you say. *How is it different from Xedit and Wylbur ?*, you say. Tell you what I'm gonna do, I'm going to tell you the answers to these questions even if you weren't perceptive enough to ask. A text processor is to Xedit and Wylbur what a publishing company (as embodied by editors and printers) is to a technical typist; namely, it is a thingamajig which takes a crudely typed manuscript and turns it into a book. Physicists and engineers all know that the nicer your paper looks before it goes out, the more your colleagues believe what is in it. Therefore we all want to use TeX to prepare our papers for journals, summer schools and conference proceedings. Some of us (hopefully that means some of you reading this introduction) will use TeX to type our own papers, and some of us will rely on others for this task. In any event somewhere in that chain someone has to know how to use the thingamajig or beautiful manuscripts just won't come out the other end. In that case our colleagues won't know all of the wonderful things we have to say. To facilitate this process to some degree we wrote the macro package PHYZZX. In order to make PHYZZX and TeX more accessible to people at the lab, we have generated this writeup.

As I said, to all intents and purposes you and Xedit (or you and Wylbur, if you insist on living in prehistoric times) make a typist. Actually, you make a pretty good typist for text, but you don't do too well as a technical typist because neither Xedit nor Wylbur does a very good job at typing equations. You plus Xedit plus TeX make a stupendous technical typist and a remarkably good printing company. Pursuing this analogy the process for producing a beautiful book quality manuscript starts with somebody typing the first draft. This somebody is presumably you, the reader, and the tools you use are your trusty computer terminal and an editor like Xedit. Obviously I have no intention of explaining how to use Xedit since this is probably unnecessary and anyhow, everything you have to know for the purpose of typing a text file can be easily learned by reading the **IBM Virtual Machine/ System product: CMS Primer**. Hence, this writeup assumes that you know how to log on to the computer and use the editor.

To use the editor to start collecting a file which TeX will turn into a beautiful paper, memo, or letter, you have to follow a simple procedure. First, in your incarnation as typist, you have log on and enter an editor (e.g., Xedit) by telling it you want to create a file whose name is **CRAP** and whose filetype is **TEX**. You accomplish this feat by typing something difficult like
X (or XEDIT) **CRAP TEX A**
whereupon the powers that be will throw you into the editor. At this point, since you are a person of discerning tastes, the very next line you enter is
\input phyzzx
. This command will make sure that when you tell the computer to print a version of the paper it will become an editor and instruct the typesetter to lay out the copy according to the criteria established by the macro package PHYZZX. The very next thing you do is skip a line (i.e., enter a blank line) in order to make your copy easy to read, and then start to enter your copy. So much for you as a typist.

You As An Editor: Correcting a Mistaken Impression

The preceding discussion indicates that you are to take your manuscript and type it into the editor in exactly the same way as you would at a typewriter. While this is a possible way of proceeding it is not really the best way to do things. In order to you to understand why this is true, you have to understand that once you have functioned as a typist, the next thing you have to be is an editor.

An editor's job is to make sure the text looks good and that the printer knows how he is supposed to layout the copy which he receives. For example, he has to know: how much space to put between lines of text, what font to set chapter headings in, how much space to skip before and after a chapter heading, which math symbols are exponents, greek letters etc. Since you are the editor and TEX is going to be the printer, you have to mark up the copy so that TEX and/or PHYZZX knows what to do with it. This feat is accomplished by including editor's comments (or marks) in the text. These comments (or commands to TEX) have to have a special character in front of them so that TEX will recognize them for what they are; this character is the \, and the combination of a **** followed by a string of characters is called a control sequence. One example of a control sequence is **\par** which tells TEX to begin a new paragraph. Note that this is not a trivial thing since TEX has to know to skip some extra space between the text in different paragraphs, indent the first line by some amount, etc. The basic version of TEX has lots of control sequences which tell TEX to do lots of wonderful things. To learn about these things (probably more than you initially want to know about how TEX operates) you have to read *selectively* in the TEXbook written by Knuth. Since this discussion is between friends, let me warn you this book stinks as a reference manual and ain't so hot as an introduction to TEX. Knuth has a philosophy of learning by ever increasing levels of mistakes which I (and many of my acquaintances) find at best distracting and at worst impenetrable. Unfortunately, this book is all that is available as an in depth introduction to the basic version of TEX. If you want to know how to do sophisticated things which lie outside of the purview of PHYZZX you will have to read quite a lot of it.[⋆] A really good thing to read to get an idea of the basic things you can do with TEX and to learn about the important control sequences is a primer entitled **First Grade TEX A Beginner's TEX Manual** by *Arthur L. Samuel*. This is short, readable and contains everything you need to supplement this discussion of PHYZZX. You can request a copy of this manual from the SLAC Library

First Grade Tex – A Beginner's TEX Manual
by
Arthur L. Samuel
STAN–CS–83–985

Since you are going to be both typist and editor it pays for you to combine these jobs and mark up the manuscript at the same time that you type it. To be precise, this means that you want to type in equations so that superscripts,

⋆ To get a feeling for TEX it is advisable to browse through the first three chapters and read about boxes and modes. If you just wish to type papers in the most straightforward way you can defer this reading until you are more experienced so long as you use PHYZZX. However, in any event, you must read the sections on typing math formulas, typing displayed equations and a little bit about making boxes since I have neither the time nor inclination to go into those things in this writeup, and you must have a basic idea of how this stuff works in order to proceed. After all, ostensibly this writeup is about how to use PHYZZX and is not a primer on TEX.

subscripts, etc., are clearly indicated. You also want to type chapter headings, subchapter headings, etc., in such a way that the font to be used, the space to be skipped above and below the heading, etc. are all clearly indicated. In general you want to tell the printer how to lay out the resulting copy as you type it in. This of course requires that you make many formatting decisions and then use the basic TEX control sequences in order to tell it what to do. It will come as no surprise to you that this can get very complicated very quickly. That's where PHYZZX comes in.

2.2. What Is PHYZZX?

PHYZZX is a *macropackage* which is designed to make typing papers destined for Physical Review or Nuclear Physics as simple as possible. In addition it allows you to type letters and produce memos without knowing much about the way TEX works.

The first question which arises at this point is *What is a macropackage ?*, or for that matter *What is a macro?*. Despite the name a *macro* is not something eaten on a macrobiotic diet. It is a name for a simple thing. Unfortunately, this name has been carefully designed by computerniks to strike terror into the heart of occasional users and keep them in their place. (There is nothing more annoying than an uppity occasional user.) Basically a macro is a way of defining an entry in a dictionary which is kept in the guts of the computer. The computer uses this dictionary to find the meaning of words which have been defined using the control sequence **\def**. A macropackage is, as the name implies, a package of macros (or definitions of commands) which we can use to give complicated formatting instructions to TEX without having to go to the trouble of typing everything out each time we want to give the same instruction.

It follows from this discussion that PHYZZX is just a collection of definitions which tell TEX to do a well defined set of things to make the text which follows look just the way you want it to look. The best thing about a macropackage is that once it is written you only have to know what happens when you issue a given instruction; you don't have to know how it is made to happen. Although each of the commands which appear in the macropackage cause apparently simple things to happen, making them happen can get quite involved if you try to use the PLAIN version of TEX.

Why Create A Macropackage like PHYZZX ?

Finally we get down to brass tacks. We already pointed out that you, the person preparing the manuscript, must to play the role of typist and editor. As editor you have the job of telling the printer how large to make chapter titles, what typeface to use for chapter titles, how much space to skip above and below a chapter title, etc. Having to give this set of instructions every time you want to type a chapter title can get to be a pain in the neck. Not only that, but imagine the problem of remembering exactly what format you want for chapter titles, section titles, subsection titles, etc., if – as is the case with PHYZZX – you use a different format for each one. Clearly, if one had to type all of this each time and keep all this information in one's head one would probably choose not to use TEX at all.

This discussion raises another point, namely that if one did not have a predefined set of macros to tell TEX what to do at a certain point then you would have one more job dumped upon your shoulders. In addition to being typist and editor you would have to be a book designer. That means that you would have to make all of the esthetic decisions about the most attractive print to use for a given paper, the spacing to use between lines, the best way to number equations and the rest of that crap. This can be an onerous task which the average user doesn't want to have to deal with. Moreover, everybody has his own esthetic criteria (even though some people have more taste than others), so without a macropackage, to set the formats into predefined molds, there would be little or no uniformity in the layout of papers coming out of SLAC. While this would be no great tragedy, it could get confusing.

Okay, so PHYZZX is a macropackage which provides a dictionary of editor's commands which allow one to easily format a document for the printer. It makes a set of esthetic choices for how these documents should look and, in the interests of simplicity, doesn't make it trivial for the casual user to mess around with these choices. (Actually, this is not really true since there are many ways in which PHYZZX can be tailored to your personal preferences, but we will discuss that as we go along.) The obvious question which comes up at this point is *Who made all of these esthetic decisions?*. In the interests of preserving the lives of the parties involved, anonymity must be preserved. Rest assured, however, it was a consultative process and many people had an input into the final result. Despite the fact that PHYZZX is an animal put together by a committee, we think it does its job fairly well.

So, what does it do?

The main job which PHYZZX is designed to do is to make it easy to format SLAC preprints destined for submission to Physical Review or Nuclear Physics. Since these journals require different ways of handling line spacing, footnotes, references, etc., PHYZZX you need to specify the journal to which the preprint is to be submitted. Actually, you only have to tell if the journal is going to be Physical Review, since the default format is that for Nuclear Physics. This is because we, with our usual impeccable good taste, decided that the Nuclear Physics format was a better general purpose format. If you don't like our decision write your own macropackage.

All kidding aside, changing to Physical Review format is easy, all you have to do is start your typing with

\input phyzzx

\PHYSREV

Note, when you are typing a control sequence everything counts; i.e., TEX cares about capital letters, small letters, spaces, etc. Be careful to copy just what you see typed here or it won't work; worse yet, it might work in an unexpected way.

In addition to formatting papers PHYZZX does a great job of producing letters and memos. In each of these cases it produces, by itself, the appropriate letterhead and formats the text in a way appropriate to the document at hand. Since, however, we consider the preprint formatting capabilities of PHYZZX to be of paramount importance, we will discuss them first and turn to letters and memos when we have finished.

3. STRUCTURING PREPRINTS

A preprint is a document which eventually will be submitted to a journal for publication. In setting up PHYZZX we assumed that format of a preprint is essentially fixed. The size type to be used in the body of the preprint is chosen so that when run off on the laser printer and reduced for distribution it will be easily readable. In addition, normally we skip one and a half lines between lines of text. We think this looks most attractive with the size type we are using and makes the preprints a little shorter, saving paper. To satisfy the requirements of journals like Phys.Rev. Letters which specify the number of lines per inch which a manuscript must have, you must use double spacing; the command **\PHYSREV** automatically takes care of this, as well as a number of other things.

Concerning spacing and other parameters you may well ask *How fixed is fixed ?*. The answer varies. For example, since we realize that people like to play around with line spacing to some degree, we have provided commands which we hope will enable you to change the average interline spacing to something closer to your heart's desire. The commands which perform this wondrous feat are
\singlespace
\doublespace
and
\normalspace
Obviously, **\doublespace** means twice **\singlespace**; but *What is the meaning of* **\normalspace** *?*. The answer to this question is that **\normalspace** corresponds to a spacing between single and doublespace.

You might think, having read a little in the TEX book, that you can achieve the same thing by changing **\baselineskip** and **\lineskip**. In a sense you would be correct. However, you should know that the commands **\doublespace**, etc., do more; this is because they do different things depending upon the font and format you are using. If you want to do the same thing you will have to modify a lot of basic TEX parameters or the esthetic impact of the document which emerges just won't be the same.

All of this is by way of a parenthetical comment since it won't do you any good unless you know how to use the simplest version of PHYZZX first.

3.1. Chapters, Sections and Subsections

Chapters

Most longer papers are divided into several parts in order to make them easier to read. These parts usually are preceded by headings or titles which indicate the subject to be covered in discussion which follows. Stylistically, several decisions have to be made about how these headings are to be set in type. You, as editor, have to tell the printer what to do.

We will refer to the major divisions of a paper as chapters. In general each chapter has a heading. Our choice as book designers has been to set this heading in larger type and preface the text of the heading with a number. The number is automatically increased each time you declare a new chapter. In addition, some extra space is skipped both above and below each chapter heading. All of this magic is at your disposal if you simply use the **\chapter** command to indicate that the text you are about to type is the heading of a chapter. To be precise, the way you use this command is to type

\chapter { *text* }

where { *text* } stands for the chapter name which you have to make up and insert. For example, the chapter title for this chapter was generated by typing

\chapter{ *STRUCTURING PREPRINTS* **}**

Note, I chose to type the heading in capitals. That's because I like the way it looks. You certainly don't have to do that. If you don't, but instead type
\chapter{ *Structuring Preprints* **}**
then you obtain

3. Structuring Preprints

One thing to notice is the curly braces surrounding the text. These braces are absolutely necessary. They are grouping symbols which tell TEX how much of the text to follow is to be considered part of the chapter heading and if they are omitted you will get a disastrous error message saying something like TEX's capacity is exhausted. Unfortunately, on the screen of a computer terminal, these symbols are easy to confuse with (and so you will have to be careful. If you are in **XEDIT** you will be able to issue the command **balance**. This is an **XEDIT** command defined by an exec file on the U disk which counts matching pairs of curly braces. It is helpful in that it tells you there is a problem, but unfortunately it doesn't find the problem for you.

In general you won't want to play around with the choice of font inside of a chapter heading and so, by default, you will get a roman font. However, you can play with it if you wish. We have provided a series of control sequences, or editors marks, which allow you to change between different fonts all of which have essentially the same size. Hence, if you type **\rm** you get the current roman font, if you type **\bf** you get the bold face version of the current font, if you type **\sl** you get a slanted version of the current font, if you type **\it** you get an italic font, and if you type **\tt** you usually get a typewriter font. I say that you usually get a typewriter font because we do not have typewriter available in all sizes of print. When we don't PHYZZX makes a substitution of the same font at a different size. Another font for which this happens is **\caps** which is a font which is made up of only capital letters. This font distinguishes between upper and lowercase letters only by size. As an example of switching fonts let us see what happens if we change font inside the command chapter.

If you type
\chapter{\bf *Formatting Text*}
you get

3. **Formatting Text**

If you type
\chapter{\sl *Formatting Text*}
you get

3. *Formatting Text*

If you type
\chapter{\it *Formatting Text*}
you get

3. *Formatting Text*

If you type

\chapter{\caps *Formatting Text*}

you get

3. FORMATTING TEXT

Finally, if you type

\chapter{\tt *Formatting Text* }

you get

3. `Formatting Text`

While all of this is very nice you should note that the font used to set the chapter number does not change with the text. This illustrates something about TEX's grouping symbols. Since we put the control sequences for changing fonts inside the braces, only the text inside the braces is affected. That is, the change only occurs *locally*. In general any change you make inside a pair of curly braces only affects the material inside the braces, unless you force the effect to persist. (If you want to know how to do this look up the **\global** command in the TEX book.) It follows, therefore, that if we want the number to change we have to change the font before we give the **\chapter** command.

If you type

{ \bf \chapter{ *Formatting Text*}}

you get

3. Formatting Text

If you type

{ \sl \chapter{ *Formatting Text* **}}**

you get

3. Formatting Text

If you type

{ \it \chapter{ *Formatting Text* **}}**

you get

3. Formatting Text

If you type

{ \caps \chapter{ *Formatting Text* **}}**

you get

3. FORMATTING TEXT

Finally, if you type

{ \tt \chapter{ *Formatting Text* **}}**

you get

`3. Formatting Text`

Much better, no?

We have already mentioned that PHYZZX put extra space above and below a chapter heading. Since all of the chapter headings for this writeup were generated using the command **\chapter**, you only have to examine them to see how TEX handles these extra spaces.

Another remark worth making is that in the current incarnation of PHYZZX you do not have to worry about chapter titles which are too long. The chapter macro is set up to split your line when necessary and automatically center each line. To accomplish this feat PHYZZX makes use of a control sequence called **\titlestyle**. The command **\titlestyle** is worth knowing about because sometimes you want to be able to type in a longish amount of material and want the various lines to be centered, not left or right adjusted after each break. Typing **\titlestyle{** *text* } will accomplish this feat. In addition, if you don't like the way PHYZZX chooses to break the lines you can force your desires upon the machine by inserting: the command **\break**. This command forces the line to break at the point where the command is given. The command **\nobreak** will stop the line from breaking at an undesirable place, and the symbol ˜ will insert some unbreakable space between words. This stuff is covered in the basic books on TEX and we refer you to there for details. Combining these control sequences with the control sequence **\titlestyle** allows you to *easily* achieve many interesting effects. For example, you can generate an invitation by typing

\titlestyle { **\it** *You are cordially invited* **\break**
to the coming out party for **\break**
the new macro package **\break**
\rm *PHYZZX* **\break** }
and obtain

You are cordially invited
to the coming out party for
the new macro package
PHYZZX

If you are really sharp, and I assume that you are, you will have noticed that the type in which the invitation is set is larger than the normal type. Remember that I already told you that titles are set in larger type than the basic text. I also told you that the commands **\rm**, **\bf**, **\it**, etc., change the style of the font but leave it the same size. What we have not discussed is how you change the size of the font the way **\titlestyle** does with impunity. Well, all things come to he who waits! Now's the time to discuss this point.

Font sizes, as you learn from reading the TEXbook come in sizes measured in a mysterious printer's unit call *points*. You don't have to know what a point is, you only have to know that most of the text in this writeup is twelve point type, the titles are in fourteen point type and the footnotes are in ten point type. It is possible for you to force TEX to use a particular size of type by saying **\tenpoint**,**\twelvepoint** or **\fourteenpoint**; in which case the obvious thing happens. Thus, for example, if you change the typed instructions for generating the invitation to read

\titlestyle { **\twelvepoint** **\it** *You are cordially invited* **\break**
to the coming out party for **\break**
the new macro package **\break**
\rm *PHYZZX* **\break** }
you get

You are cordially invited
to the coming out party for
the new macro package
PHYZZX

and if you type

\titlestyle { **\tenpoint \it** *You are cordially invited* **\break**
to the coming out party for **\break**
the new macro package **\break**
\rm *PHYZZX* **\break** }

the result is

You are cordially invited

to the coming out party for

the new macro package

PHYZZX

Sections

Now that we know everything about generating chapters, we turn to the problem of generating smaller subdivisions; namely, sections and subsections.

The numbering is done as {*chapternumber*}.{*sectionnumber*} . Each time you start a new chapter, the section numbers restart from 1. You have already seen many examples of the results produced by the section macro in this writeup. As in the case of the chapter macro all of the numbering is taken care of automatically. To accomplish the feat of generating a new section heading you type

\section { *section heading* }

For example, if you type

\section { *New Section* }

you get

3.1. NEW SECTION

As in the case of chapter headings you can play with the fonts if you insist; however, remember that the space above and below chapter and section headings have been carefully matched to our choices of fonts. If you play too much, the esthetic balance of the final copy won't be the same.

Subsections

The smallest subdivision for which we have a formal macro defined is a subsection. Our choice has been to make subsections unnumbered and underlined. When you type the command **\subsection** { *subsec.heading* } you get the sort of heading which introduces this paragraph. All of the control sequences, **\chapter**, **\section** and **\subsection** force the start of a new paragraph, so in principle you do not have to skip lines as you type in text. However, from the point of being able to proofread your typed copy in order to make corrections, not skipping lines can be a disaster.

The underlining of the text in the section macro is accomplished using the control sequence
\undertext { *text* }
This is a command worth knowing about, but in general using italics is a better way to emphasize text.

This completes our discussion of the macros which handle the generation of titles for the major subdivisions of the usual paper.

3.2. Other Subdivisions: Appendices and Acknowledgements

In addition to chapters, sections and subsections, papers tend to have auxiliary junk attached. This extra junk is usually called either acknowledgements or appendices. To save you the bother of figuring out which font sizes and spacing choices are necessary to make these appendages consistent with the rest of the paper we also have control sequences to generate headings for them. Typing the command
\ack
generates the line

Acknowledgements: Note the beauty of the typeface and the precisely chosen amounts of extra space which gracefully offset the title generated with this simple command. Once again, for the Philistines among you, you can play with the fonts using the commands **\bf**, **\sl**, etc.; however, let the esthetic violence you do to the manuscript be on your own head.

Generating appendices presents us with a slightly more complex situation and so you get more choices as to how to proceed. The questions which arise at this juncture are *How many appendices will there be ?* and *How do you want them numbered ?*. Clearly, if there is to be only one appendix then the choice is simple. In that event you generate a lovely heading by simply typing
\appendix
which causes TEX to generate the line

APPENDIX

If there are to be many appendices and you want to number them in any way which suits you, you only need type
\Appendix{ *text* }
note the capital letter appearing as the first letter of this control sequence. Remember, TEX cares about upper and lowercase, so this is a different command from **\appendix**.

Typing
\Appendix{ *A* }
generates

APPENDIX A

Typing
\Appendix{ *I* }
generates

APPENDIX I

and typing

\Appendix{ *1* **}**

generates

APPENDIX 1

This completes the discussion of the basic commands which generate division headings. Now we turn to a discussion of the special commands which PHYZZX has included to allow you to tailor the way in which these macros handle numbering conventions.

3..3. Controlling the Way Chapters are Numbered

The discussion to follow is really not necessary if what you want to learn is how to use PHYZZX to type a paper. This section is devoted to making PHYZZX do something out of the ordinary; i.e., something it ordinarily doesn't want to do.

<u>Playing With Numbers</u>

There will be times when you want to print out just a piece of a paper either because it is long, or because (as I understand is the case for that strange group of people called *experimental physicists*) many people are working on different chapters (perhaps even sections) of the same paper. In this event you probably want the chapter and section numbers to print correctly when you are proof reading so that you don't get confused. Rest easy, this can be done. PHYZZX defines two quantities which it updates and refers to when it needs chapter and section numbers. These quantities are called **\chapternumber** and **\sectionnumber** (big surprise!). You, as the editor, can tell PHYZZX to set these numbers to a specific value, say 7, by typing

\chapternumber=7

or

\sectionnumber=7

Note that for the first time you have encountered an = sign occurring in a statement to TEX. You will encounter this sort of syntax many times, in general when it occurs it will be because it is the natural way to write something.

Once you get involved in fooling with chapter and section numbers there are a few things you have to know. Both the **\chapter** and **\section** macros increase the respective chapter or section number by 1 before they print anything. Thus, if you want to get a chapternumber like 7 you set
\chapternumber=6
The same is true for setting the section number. If you are so perverse as to start in the middle of a chapter, then when you set the section number you must also set the chapter number, since it appears as part of the section heading. Actually, the quantity which appears in the section heading is called \bf \chapterlabel and to set it you have to say
\let\chapterlabel=*6*
or whatever label you want instead of 6. This extra step is required because we really don't think its such a great idea to start in the middle of a chapter and so didn't bother to make it too trivial to do so. If you are starting a chapter, then you don't have to worry about setting the section number since the command **\chapter** automatically resets the section number to 0.

Changing Styles

Another way you might want to modify the way PHYZZX handles divisions is to have it number chapters using letters, or roman numerals, instead of arabic numerals. Although this defeats the purpose of establishing some sort of uniformity of style for preprints, etc., it is possible to do. This feature was added to PHYZZX because we recognize that some folks, particularly experimentalists, just can't leave well enough alone. The mood of people like this begins to verge upon hysteria if they don't have knobs to twiddle and buttons to push. For this small number of benighted souls there exists the special command called **\chapterstyle**. You change the way in which PHYZZX numbers chapters by saying

\chapterstyle={\alphabetic}	to get the l.c. letters a,b,c,...
\chapterstyle={\Alphabetic}	to get the u.c. letters A,B,C,...
\chapterstyle={\roman}	to get the l.c. roman numerals i,ii,iii,...
\chapterstyle={\Roman}	to get the u.c. roman numerals I,II,III,...

There is one more option open to you, that is to have all chapters, sections and subsections unnumbered.* To select this option you begin your file with the command
\unnumberedchapters
This is not too difficult, but it is long to type.

One final remark. We have not made it possible, other than shutting it off completely, for you to change the style used when numbering subsections. All you get is *chapterlabel.sectionnumber* no matter what you do. Moreover, the section number is always an arabic numeral. If you don't like this, write your own section macro. (This isn't as bad as it sounds. We'll discuss doing this sort of thing at the end of this writeup.)

4. DISPLAYED EQUATIONS: NUMBERING AND NAMING THEM

We have now discussed the macros available for defining the basic parts of a paper and appropriately titling them. Now let us turn to the most important stuff – other than text – which appears in each section; namely equations. If you've done your homework and read Knuth or **First Grade TEX** you know that equations are typed in *math mode* and that you let TEX know that you are in math mode by typing either $ or $$. Enclosing an equation inside a pair of single $ signs puts it in ordinary mode; enclosing the same equation between matching pairs of $$ signs puts it in display mode. Regular math mode is used for inserting equations into a line of text. Since you have done your homework we don't have to say anything

⋆ Note that this implies that all equations will automatically be sequentially numbered

about this, since PHYZZX doesn't play with ordinary math mode at all. Display mode is used for generating equations which are to stand out from the ordinary text. It allows you to type
`$$ (x+y)(x-y) = x^2 - y^2 $$`
in order to get

$$(x+y)(x-y) = x^2 - y^2$$

This is called a displayed equation.

Displayed equations are used in all journals and are usually numbered. In PLAIN TeX there is a built in capability for allowing you to generate equation numbers; it is the macro **\eqno**. If you type
`$$ (x+y)(x-y) = x^2 - y^2 \eqno(13) $$` you get

$$(x+y)(x-y) = x^2 - y^2 \qquad (13)$$

This example shows what the command **\eqno** does. It makes the material appearing to the right of this command into a label for the displayed equation and sticks it in the right hand side of the line. There is also the PLAIN TeX command **\leqno** which does the same thing but puts the equation number on the left. For example, typing
`$$ 7x+12 = 24 \leqno (13) $$`
produces

$$(13) \qquad 7x + 12 = 24$$

So, you ask, *what does PHYZZX have to do with this?* Well the answer to that question depends upon who you are. If you are typing someone else's handwritten manuscript which is in final proofread form, then these macros for putting in equation numbers are all you need. If, however, you are the author of the paper, typing it in yourself, either from a rough copy or composing it directly at the terminal, then these two commands leave a great deal to be desired.

To begin with, if you are anything like me, when composing at the typewriter you usually forget the number of the last equation you have typed. Of course you can always go back and look it up, but this is time consuming and sometimes difficult to do. In addition if, as is usually the case, you expect the manuscript to be changed after it is typed, then it is highly likely that equations will be added, deleted and moved around. In that case, if you have used **\eqno** or **\leqno** you will have to go through the text and change all of the equation numbers and all references to those numbers each time there is a revision. Clearly this can get to be a big nuisance. Wouldn't it be nice if TEX would just take care of numbering all of the equations properly each time it prints the manuscript taking into account all of the revisions automatically.

Joy of joys it can be done!

4.1. A Simple Solution to Numbering Equations

PHYZZX has a macro **\eq** which can take care of this chore for you automatically. All you have to do is type

$$ *x + y =* **{** *2* **\over** *(x - y)* **} \eqno\eq $$**
to get

$$x + y = \frac{2}{(x-y)} \tag{4.1}$$

and you will have automatically generated an equation number. The question is, *What sort of equation number will you generate ?*. Once again, this is a question of style. If you are in the Phys.Rev. format, or in the default Nuclear Physics format then the equation label is of the form *chapterlabel.equation number*. Equations are numbered sequentially within each chapter, but when you change chapters the chapter number is increased by 1 and the equation number is reset to 1. If you have chosen the format with unnumbered chapters and sections, then \eq will automatically generate sequential equation numbers. As far as we are concerned this is the most satisfactory way of numbering things. For short papers, i.e. those with only one chapter, choose the unnumbered chapter format and all of your equations will be numbered sequentially; for longer papers, i.e. those which call for dividing them into chapters, numbering the equations by their chapter and number within the chapter makes it easier to refer to them. Sometimes, however, either out of perversity or because a paper has many chapters but very few equations, an author wishes to choose the format with numbered chapters but sequential equation numbers. This will happen if, before entering the first equation in your paper you type the command **\sequentialequations** Hence, in the default mode, typing

```
$$ 7x + 11x^ 2 = 50 \eqno\eq $$
$$ 3x + x^ 3 = 85 \eqno\eq $$
$$ 4x + 8x^ 7 = 12 \eqno\eq $$
$$ 7x + 4x^ 2 = 0 \eqno\eq $$
$$ x + 21x^ 5 = -5 \eqno\eq $$
```

yields

$$7x + 11x^2 = 50 \tag{4.1}$$

$$3x + x^3 = 85 \tag{4.2}$$

$$4x + 8x^7 = 12 \tag{4.3}$$

$$7x + 4x^2 = 0 \tag{4.4}$$

$$x + 21x^5 = -5 \tag{4.5}$$

If, however, you first type **\sequentialequations**; then typing

```
$$ 7x + 11x^ 2 = 50 \eqno\eq $$
$$ 3x + x^ 3 = 85 \eqno\eq $$
$$ 4x + 8x^ 7 = 12 \eqno\eq $$
```

`$$ 7x + 4x^ 2 = 0 \eqno\eq $$`
`$$ x + 21x^ 5 = -5 \eqno\eq $$`
yields

$$7x + 11x^2 = 50 \tag{6}$$

$$3x + x^3 = 85 \tag{7}$$

$$4x + 8x^7 = 12 \tag{8}$$

$$7x + 4x^2 = 0 \tag{9}$$

$$x + 21x^5 = -5 \tag{10}$$

Since you are an astute reader you have no doubt noticed that what I have told you to this point only solves half of the problem. While **\eq** automatically generates equation numbers which, each time you run the paper through TEX , are automatically updated to conform to the order in which they appear in the text, we have no a priori way of knowing what these numbers are. The question is, *How do we get hold of these numbers so that we can refer to them in the text?* Well, **\eq** provides a partial solution to this problem by definiing the control sequence **\?** a synonym for that number each time it is invoked. Hence, typing **\?** at any point in the text causes TEX to print the number of the last equation in which you used the command **\eq**. For example by typing *equation* **\?** we cause TEX to print equation (10) , which is the number of the last equation appearing in our examples. The only thing wrong with this solution is that the meaning of the symbol **\?** changes each time you invoke **\eq**, so what happens if you want to refer back to an equation at several different points in the text?

4.2. Naming Equations: A Powerful Tool

In order to make it possible to refer back to a specific equation without knowing its number, PHYZZX has a way of giving the equation a symbolic name, like **\?**, which will always stand for the number of that equation. The macro which accomplishes this feat does this at the same time it generates (and prints) the number associated with this equation. Since this command does just one more thing than the command **\eq** does, we have given it a name which is just one letter longer; the name is **\eqn**. To use **\eqn** you type

\eqn*equation_name*

Note carefully the \ appearing before the equation name inside the braces. This backslash must always precede the name itself because you are using **\eqn** to define a new control sequence and, as we have already noted, control sequences always start with a \ . Also note the grouping symbols. They tell TeX to include everything in the braces as the equation name and they are necessary. To use this command you type something like

the value of **$ \pi $** *is*

$$ \pi \approx *3.14159* **\eqn\\***valueofpi* **$$**

to obtain

the value of π is

$$\pi \approx 3.14159 \tag{4.7}$$

If you now type

the circumference of a circle is given by

$$** *2* **\pi** *r* ^ *2* **\eqn\\***circumference* **$$

you define a new equation number and a new name. The result of this is the circumference of a circle is given by

$$2\pi r^2 \tag{4.8}$$

If we now type **\valueofpi** we get (4.7), if we type **\circumference** we get (4.8).

A little thought will convince you that this ability to name equations is a very powerful time saving tool. By giving names to equations you are able to refer to them at will without ever giving a thought to the number they will be assigned when the paper is printed. If you use the command **\eqn***name* to deal with the numbering problem, then the number assigned to the equation and all references to the equation will always be correct. It won't matter if you delete equations, add equations, move equations around, or decide to switch from the normal conventions for numbering equations to sequential numbering, etc.

<u>Restrictions on Equation Names</u>

There are only a few simple rules you should observe when giving names to equations using the command **\eq**

1. First, the name has to be proceeded by a \ as we already noted

2. Second, the rest of the name has to be all letters. *No numbers are allowed*. You may however use upper and lowercase letters, and names can be arbitrarily long. Everyone seems to have his own favorite way of naming equations in order to facilitate referring back to them later. I invite you to play around with this until you find a scheme which makes you comfortable.

3. Third, try to remember to always use a different name for each equation. Neither TEX nor PHYZZX will warn you if you give two equations the same name. They will quite happily change the meaning of the previously defined name to mean the number of the last equation to which you have given that name. If you wish this to happen, o.k., however be sure that is what you want or you will be in for some surprises later.

4. Finally, *never use the name of any macro for an equation name*, this road surely leads to disaster. For example, if you accidentally type **\eqn\bf** you not only change the previous meaning of **\bf** but you also mess up any macro which uses that control sequence.

As an example of this I note that the names **\herman**, **\Suzanne**, **\duMMy** are all good names; however, **\jerk3**, **\don't** and **\sl** are not. There is one exception to this rule, you can give the name **\?** to an equation, which is what the macro **\eq** does. This is useful only if you intend to use it as a temporary name for an equation that will be referred to once or twice before the name is redefined. If you really want to give use numbers for you equation names roman numerals are all right. You can name equations **\eqi**,**\eqii**,. . ., **\eqix** , etc.

4.3. Other Macros For Naming Equations

We have already noted that TEX allows you to put equations on the left hand side of a line using the macro **\leqno**. For this reason, we too need a version of the naming macros which do the same thing. The obvious name for such a macro would be **\leqn** and it is invoked by typing
\leqn*equation.name*
If you include this in an equation the number comes out on the left.

In addition to this obvious addition to the macro set there are macros meant to be used for the automatic numbering of aligned equations. To learn about aligned equations you should read in the TEXbook or in **First Grade TEX**. Assuming that you have done so, let me remind you how about how aligned equations are generated, in order to set the stage for explaining how the macros **\eqinsert**, **\leqinsert**, **\mideqn** and **\midleqn** work.

Aligned equations are, as the name implies, a set of equations which appear one after the other. In order to make the spacing between these equations look nice and in order to make them line up so as not to look messy PLAIN TEX introduces the macro **\eqalign**. This macro produces a list of equations which are aligned with respect to some common feature of each equation, for example an = sign. As with single displayed equations TEX allows you to number these sets of equations as if they are a single equation by using the commands **\eqno**, **\leqno**. PHYZZX, too, is kind and permits exactly the same thing; to do this you use the commands **\eqn** and **\leqn**. Since **\eqalign** is a macro whose argument is the text of the equations, the way to use **\eqn**, etc. is to type them *after* the right } which closes the **\eqalign** command. For example, typing

$$ \eqalign{ \cos { \pi \over *6* **}\ = & \ \sqrt{***3/4***} \cr**
\cos{\pi\over*4***}\ =&\ \sqrt{***1/2***} \cr**
\cos{\pi\over*3***}\ =&\ \sqrt{***8***} \cr } \eqn\trig $$**

yields

$$\begin{aligned} \cos{\pi \over 6} &= \sqrt{3/4} \\ \cos{\pi\over 4} &= \sqrt{1/2} \\ \cos{\pi\over 3} &= \sqrt{8} \end{aligned} \tag{4.9}$$

<u>Things to Remember About **\eqalign**</u>

1. The entire argument of **\eqalign** must be enclosed in braces. Successive rows of the equation must be separated by the control symbol **\cr** to tell TEX to make a carriage return.

2. Every row contains two parts (or *templates*) separated by the special tabulation symbol &. This symbol tells TEX what character you want to line up in each line of the set of equations. If you want half of some of the lines to be empty that is all right, but the symbol & must always appear. Left and right braces within a *template* must balance separately.

3. PLAIN TEX sometimes puts aligned equations too close together. If you want to force **\eqalign** to spread the lines apart somewhat there is a special control symbol for doing this. All you have to do is type **\cropen{***space***}** instead of **\cr** at the end of each line. For example, **\cropen{***12pt***}** will force

the lines 12pt further apart. The control sequence **\crr** is an abbreviation for **\cropen**{*10pt*}.

Although it is not obvious from the previous example, **\eqalign** won't always put the equation number where you would like it. For example, if one of the equation lines is very long and pushes into the right hand margin, then **\eqalign** will put the equation number on a separate line below the last equation in the set. If, as often happens, there is only one or two long lines and there is room on one of the other lines for an equation number you can force TEX to put it there if you wish. The way to do this is to use the PLAIN TEX control sequence **\eqinsert** immediately following the **\cr** or **\cropen** ending the line you wish to have the equation on. If you type

$$** ... **=&** ... **\cr \eqinsert{** *equation.number*} **$$

then you get the equation number you specify on the line indicated. The corresponding PHYZZX macros which allow you to do this while at the same time automatically generating a name and number for the aligned set of equations are

\mideqn*name*

If you don't want to name the equation at the same time that you automatically generate a number for it you can simply use

\eqinsert\eq

<u>Numbering Aligned Equations Independently</u>

Sometimes you will wish to generate a set of aligned equations but refer to each of them individually. In this case you will wish to be able to number them independently. PLAIN TEX provides the control sequence **\eqaligno** to handle this eventuality. The command **\eqaligno** works like **\eqalign** except that each equation line consists of three parts (or *templates*) separated by the character &. The third template is where you put the appropriate label for that line of the set of equations. If you wish to number each equation line sequentially without generating a name, then all you have to do is type

$$ \eqaligno { *junk* & *more junk* & **\eq \cr** } **$$**

If, however, you wish to name each line separately you *may not* use the macro **\eqn**. It will not work and will give you a peculiar message of the form *you cannot use* **\eqno** *inside* **\eqalign**. To name individual lines of aligned equations the correct macro to use is **\eqnalign** { *name* } as the entry for the third template followed by the command **\cr**. The same command will work in **\leqaligno** which works in exactly the same way as **\eqalign** but puts the contents of the third template to the left of each equation instead of to the right. In general, **\leqaligno** causes more difficulties than **\eqaligno** and you should read about it in the TEXbook before using it.

5. REFERENCES, FIGURE CAPTIONS AND TABLE CAPTIONS: THEIR NAMING AND NUMBERING

5.1. References: Stuff Which Comes at the Back

To this point we have discussed the partitioning of a document into it's major parts, and the automatic numbering and naming of equations. We now come to the question of generating references. In general a reference is signaled by attaching a label or reference mark, i.e. either a superscripted number or a superscripted number enclosed in delimiters (e.g., braces, brackets or parentheses), to a specified word in the text. This number indicates that in the back of the document, on a page (or pages) labelled REFERENCES, one will find a corresponding chunk of text prefaced by the same number. This text will usually be a reference to work published in some journal or some comment which is only marginally relevant to the subject being discussed. Your problem as typist and editor is to generate the appropriate number, attach it to the desired word and then type the desired text into a separate file of text at the same time numbering it with the corresponding number. You therefore are faced with several formatting decisions; namely, how to attach the reference number to the word in the text, and how to format the references when they are typed in the back of the paper. Clearly, there is a real need for a macro to help you handle this dull and repetitive chore.

<u>What features should such a macro possess ?</u>

First it should handle the task of attaching the desired number to the designated word, and then it should file away the pertinent text in a form which will allow it to be printed at the end of the paper. In addition to the mechanical act of attaching the reference mark and sending the text to another page, we would also like it to automatically generate the number for us; so that we don't have to constantly refer back to the number of the last reference inserted. As in the case of equations, this would be nice, since then the numbering will automatically come out correct even if we add, delete or move references around. Finally, since one often cites the same reference in many places in the text, it would be nice for the macro to allow us to name a reference (in the same way we named equations) so that we can cite it without ever knowing exactly what number PHYZZX has assigned to it.

PHYZZX *has the macro capability to do all of this and more.*

Since, however, there are may different ways in which people like to generate and cite references we had to make the commands for carrying out these tasks more flexible than those which we have been discussing to now. For this reason there are more of them to learn about and their syntax is a little more complicated. We will take them one at a time in increasing order of generality.

<u>Single References</u>

Let us deal with the simplest case first. Suppose you wish to generate a reference, mark the appropriate word and save the indicated text for printing in the back of the paper. The command used to accomplish this feat is **\ref**. The syntax of this command is

\ref{*text*}

where the indicated text is the material which you wish to have appear in the back of the paper. For example if you type[1]**\ref{** *This is the place you want to put a reference* }; then PHYZZX will, as you see, generate a reference number, attach it to the word immediately preceding the macro command and store the text in a file on your A disk. Each time you invoke the macro **\ref** it increases the reference number by 1 and adds the next reference to the file on your A disk. Anytime that you wish to print this file all you have to do is type the command **\refout**. Normally, you would wait to do this until the end of the document; however, to see what happens we will now type **\refout** to obtain

REFERENCES

1. This is the place you want to put a reference

Note, that the simple command **\ref** is like the command **\eq**: it automatically numbers the reference and stores it away, however it does not allow you the freedom of naming it. It is like **\eq** in another way; like **\eq** it automatically gives the reference the name **\?**. This allows you to refer to the last reference until you do something to redefine **\?**. Clearly, this command has only limited flexibility and one needs a more sophisticated command to allow you to individually name each reference. The command which does this is **\Ref**; the syntax for this command is
\Ref*name*{*text of reference*}
where *name* is the name you wish to assign to this reference and *text* is the material which you wish to have appear on the page of references at the back of the document. To use the command **\Ref** you simply begin typing the appropriate reference immediately following the word to which the reference number is to be attached. For example if you type
... *this is the result obtained by Banks and Kaplunovsky***\Ref***junkname*{ *T.~ Banks and V.~ Kaplunovsky* } you get
... this is the result obtained by Banks and Kaplnovsky[1]
Note that the word Kaplunovsky has a superscripted version of the current reference number attached to it and the text has disappeared. The text is stored away

in a file called **referenc.texauxil**. This file is automatically stored on your A disk when the TEX file is processed. Once again, to print this stuff we type **\refout**

REFERENCES

1. T. Banks and V. Kaplunovsky

If we wish to refer to this reference we need only type *Ref.* **\junkname** to obtain Ref.1. Note that when you type **\junkname** you do not automatically get a space after the number generated, even if you typed one. This is because TEX eats a space which appears after a control sequence. If you want a space to follow the number you should type **\junkname** ; i.e., **\junkname** followed by a \ *(space)*, which is TEX's control sequence for an extra space.

Gaining Ultimate Control of Referencing

Recapping, we see that the command **\Ref** basically does for references what **\eqn** does for equations. Ordinarily, **\Ref** is the most convenient version of the reference command and will be the one you will use most often. There will be situations, however, where you wish to generate a named reference without generating its reference mark. For example, you may be typing a formula, or a footnote and wish to put the superscripted reference mark inside the footnote or formula (or for that matter, inside some **\hbox** which you are defining). You must be careful because the command **\Ref** cannot be used in these situations. This has to do with the way in which it goes about doing its job. You really don't have to know the reasons why it won't work, you just have to know it won't. This problem is easily solved if you know how to use the most flexible form of the reference command; namely, the command

\REF*junkname*{ *text* }

As you see the syntax of this command is identical to **\Ref** and in fact, it does almost exactly the same thing. It generates a dummy name (**\junkname**) and writes the text material into the file **referenc.texauxil**. The difference is that when **\REF** is finished it does not attach the reference mark to the preceding word. To attach the reference mark to a specific word you type the command **\refend** immediately following that word. Basically, what **\Ref** does is invoke the command **\REF** immediately followed by the command **\refend**. Dividing the reference generating command from the command which attaches the reference mark solves all of the aforementioned problems. If you wish to put a reference in a forbidden place, like a footnote, or hbox, or formula, all you have to do is use **\REF** somewhere before you enter the forbidden territory and then use **\refend** immediately following the place where you want the reference mark.

Generating Your Own Reference Marks

Since the referencing commands are the ones which are most often played with (because so many people have their own styles for referencing things) it is worth spending a few moments giving you a better understanding of the various subcommands which go into their construction. First, you should know that the current value of the number assigned to the last reference is filed away under the name **\referencecount**. To get hold of this number you type

\number\referencecount

and then TEX will type out the current value of this number. PHYZZX attaches this number to a word by invoking the command **\refmark**. The syntax of the control sequence **\refmark** is

\refmark{ *number or name of reference*}

When this command is issued PHYZZX tells TEX to generate a superscripted version of the material appearing inside the braces and attach it to the word immediately preceding the place where you typed this control sequence. It is not advisable to leave spaces between the word and the command **\refmark** (or for that matter **\ref, \Ref** and **\refend**), because if space is left it might creep into the text. The command **\refmark** does one more thing, it changes the appearance of the reference mark depending upon whether the paper is in Nuclear Physics or Physical Review format.

Since I have referred to **\refmark** as a subcommand it is only fair to give an example of how it is used. One place is in the definition of the command **\refend**. PHYZZX defines the command **\refend** to mean

\refmark{\number\referencecount}

Using **\REF** to define a reference outside of a forbidden region and using **\refend** inside this region to generate the reference mark provides only a partial solution to the problem we have been discussing. This is because **\refend** causes the current value of **\referencecount** to be printed. If, for some reason you wish to delay the printing of several reference names; or, if you wish to mix and match various references and stick them in a single reference mark, you can do this by typing **\refmark{ \junkname** ... } In this way you can gain all but stylistic control of your reference marks. It is still true that **\refmark** changes the form of the reference mark depending upon whether the paper is headed for Phys.Rev. or Nucl.Phys.

To gain total control of your reference marks you need to use the command **\attach**. The syntax of this command is

\attach{ *stuff* }

What **\attach** does is put *stuff* into math mode, make a superscript out of it and attach it to the word that comes before. It also spaces from that word so as to make the attached quantity look most natural. The **\attach** command does nothing at all to the reference mark and it is independent of the format of the paper.

Multiple References

You have now learned all you should know about generating single references and taking complete control of the way in which they generate reference marks. However, you still do not know everything about the way in which PHYZZX can handle references. This is because we have not yet discussed the question of multiple references.

What is a multiple reference? and *Why should I care about multiple references?* you ask. Good question! In principle you don't need to care about them at all. You already have the basic tools for handling any referencing problem should the need arise, all you need to do is use the commands **\REF** and **\refmark**. Nevertheless, since multiple references occur frequently, especially in experimental papers, they merit a special construct to make them simpler to handle.

The problem of generating multiple references comes up when you reach a point in the text where you wish to add a reference to several authors and at the same time you do not wish to have this saved as a single reference. This may happen for several reasons, the principal reason being that you may wish to refer to some of these references individually at a later point in the text. In this case there exists a set of commands which allow you to do this. These commands allow you to tell TeX that the material to follow will define and name several references. They assign a number to each of these references, store them away individually and then attach the reference mark
{ *number of first ref. in set – number of last ref. in set*}
The presence or absence of delimiters (in the form of brackets) depends upon whether or not you have typed **\PHYSREV**. The commands which perform this service are
\REFS*name*{ *text* }
\REFSCON*name*{ *text* }
and
\refsend .

To use these commands to generate multiple references you invoke them sequentially. For example, suppose you wish the next three references to form a single multiple reference. Begin by selecting the word to which you wish to attach the reference mark. After the word to which you wish to attach the appropriate reference mark, you type in the first reference by typing
\REFS*first*{ *This is the first reference of a series.* }
You then type the next reference using the command **\REFSCON** as follows
\REFSCON*second*}{ *The papers just mentioned are unimaginative, stupid, incorrect and besides I did it first.*}

In the same way you continue with **\REFSCON** to generate all succeeding references. Hence, to input the third reference you type

\REFSCON*third*{ *I am running out of references.* }

You finally cause PHYZZX to generate the reference mark by typing[1–3]

\refsend

after the closing brace of the last reference. As you see, this set of macros has kept track of the numbers of the references, filed the text for each reference away and finally combined the number of the first and last reference in the set into a single reference mark and attached it to the word *typing.* To see what has been filed away we type **\refout** and obtain

REFERENCES

1. This is the first reference of a series.

2. The papers just mentioned are unimaginative, stupid, incorrect and besides I did it first.

3. I am running out of references.

If you wish to generate an automatically numbered multiple reference but have no need to name the individual references then we provide the commands **\refs** and **\refscon**. The syntax of these commands are

\refs{ *text* }

and

\refscon{ *text* }

The command **\refs**{ *text* } is equivalent to **\REFS\?**{ *text* } and the command **\refscon**{ *text* } is equivalent to **\REFSCON\?**{ *text* }. The fact that the name **\?** is assigned to each reference in turn makes it less useful than in the command **\ref** and so you would not tend to use this command unless you are typing an already finalized manuscript.

The Problem of Long References

First, you we should observe that there is no problem combining many citations in a single reference. Obviously, this can be done by typing them in as the argument of any one of the reference commands. If, however, you wish to have a series of citations all appear under the same number, but wish to have them appear on individual lines all you have to do is follow each one by the command **\nextline**. What **\nextline** does is generate a neat break and starts TEX off printing the material which follows at the beginning of a new line. The important fact about **\nextline** is that it preserves the indentation of the text established by the reference macro.

This brings us to the second point. *What happens if you wish to have a reference which consists of several paragraphs ?* Here you have to be careful, since if you use the command **\par** or leave a blank line (which amounts to the same thing) you will spoil the indentation of the text material. To avoid this a special macro exists called **\subpar**. To get vertical space between text in a single reference type the command **\subpar** whenever you would ordinarily insert the command **\par** or leave a blank line. It will accomplish the same feat as **\par \noindent** but it will not mess up the formatting of the reference page.

Having discussed the way to properly generate long references, we now come to the problems this will cause. The fact is, TEX has trouble writing long lines to an external file. Since, as far as TEX is concerned the argument of any one of the reference macros is just one long line, there will come the time when TEX will complain in a mysterious way. If you are running interactively it will issue a horrible looking PASCAL error message (sending the unforwarned into a case of cardiac arrest) and then continue with its processing. Since TEX will resume processing after this frightening hiatus you will promptly recover, shake your head and say *Ain't computers peculiar?*. You will also, in all probability, ignore the fact that it happened. This will be a mistake. If you look in **referenc.texauxil** on your A disk, you will find that TEX has found one of the references is too long and has chopped it off, replacing most of the text by **\ETC**. I presume that in general this will not be what you intended.

Have no fear, PHYZZX is here !

There is a control sequence which solves this problem; unfortunately, it is not a really neat solution. The solution is to go back into the reference and type **\splitout** at various places in the text. This command tells TEX to begin sending a new line to the file. As I said, this is a solution but not an elegant one. We are saddened that the elegant solution has escaped us and hope that a TEXpert out there will tell us a better way to proceed.

Questions of Style

Sometimes you will wish to attach a reference mark to a word which is immediately followed by a comma. This presents you with two choices neither of which produce truly satisfactory text. If you type

... *this is the place,***refend**

you get

... this is the place,[3]

and if you type

... *this is the place***refend**,

you get ... this is the place[3] ,

In the first case the reference mark is really attached to the comma, causing an unsightly extra amount of space; in the second case the reference mark separates the word from the comma, which is even worse. The way out of this predicament is to make use of Knuth's command **\rlap** If you type

... *this is the place***rlap**,**refend**

you get

... this is the place,[3]

which is much better. Obviously the same trick applies to the macros **\refsend**,

\refmark, **\ref**, **\Ref** and **\attach**.

Because of the way references work you always get a single space after an attached reference. Basically this means that if you attach the reference mark to the last word in a sentence, the next sentence will start too close. There is a way out of this dilemma. Type

*the end of the world***\rlap.\refend** The next

which will produce

the end of the world.[3] The next

The \ forces more space between the sentences.

Finally, if you need a reference followed by a colon and want to get rid of the extra space that the reference macro puts in simply type **\refend\unskip**. The command **\unskip** is a Knuth construct which eats the last extra space you entered. (Actually, it eats the last bit of glue that was inserted into the text, but that is a matter for TEXperts.)

Typing Journal Entries: A Convenient Macro

Since many of the references which appear in a paper are citations of articles appearing in various physics journals we have a special macro to handle this problem. The reason one uses a macro to do this is that TEXperts are expected to typeset such references with the journal name in as slanted (**\sl**) font, the volume number in boldface (**\bf**) font and the remaining text in roman (**\rm**) font. You can either remember to do all of this or you can use the command **\journal**. The syntax of this command is

\journal*journal name***&***volume number*(*year*)

This command is used inside any one of the reference commands as follows: if you want to obtain the same result as typing

\ref{ *T.˜ Banks and V.˜ Kaplunovsky* **\sl** *Nucl. Phys.* ˜ **\bf** *B211* **\rm** *(1983) 529* **}**

you need only type

\ref{ *T.˜ Banks and V.˜ Kaplunovsky* **\journal** *Nucl.Phys.***&** *B211 (83) 529* **}**

This will produce a reference entry of the form

1.. T. Banks and V. Kaplunovsky, *Nucl.Phys.* **B211** (1983), 529

Note, the blanks before **\journal**, before and after the character **&** and before and after the *(year)* are optional. If everything is working as it should they will have no effect on the final result. On the other hand, be careful about blanks within the parentheses; if you put them in they will appear. Thus, typing (83) produces a different result from typing (83). The first will result in the journal year being typed as 19 83 and not 1983 which is, presumably, the desired result.

5.2. Figures and Tables

There are two other things which tend to appear at the end of a paper and to which one refers in the text. These are Figure Captions and Table Captions. In general these things come up as you are typing in some text and decide *Aha! This is a good place for a figure (or table).* As with equations and references, it is convenient to be able to generate a number for this figure(table), name it and file away the text of the figure caption (or something which will remind you what figure you wanted to put there) in a file called **figures.texauxil** (or **tables.texauxil**) to be printed at the end of the paper. The commands which accomplish this feat are
\FIG*name*{ *text* }
and
\TABLE*name*{ *text* }
As in all other variants of this naming convention, **\name** will be the name of the figure (or table) generated, and *text* is the material to be stored away in a file on your A disk. These commands work like the reference or equation commands in that the numbering is automatic and will always be correct no matter how you subsequently modify the text. The commands, analagous to **\refout**, which print the list of figure or table captions are
\figout

and

\tabout

which should come as no big surprise.

There is a fundamental way in which the figure and table caption macros differ from the reference macros; namely, they do not automatically generate a reference mark. This is because you normally want to say something like

... as you can see from Fig. 7

or something like that. The basic macros, i.e. **\FIG** or **\TABLE** permit you to type this in as you please. For example, one way to have TEX generate the line above is to type

... as you can see from **\FIG\?{** *text* **}** Fig.~ **\?**

which will produce

... as you can see in Fig. 8

Obviously, using this form of the command (either for a figure or a table) you have total control of what follows. Moreover, for simplicity I chose to name the figure **\?** since I only intended to refer to it this one time. It is clear that I could have given it any name.

For those of you who are happy to use the command exactly in this way, i.e. to name the figure **\?**, file away the caption and then type either *Fig.~ \?* or *fig.~ \?* there are two macros, named **\Fig** and **\fig** respectively. The syntax of these commands is somewhat simpler; namely,

\Fig{ *text* **}**

and

\fig{ *text* **}**

They are the analogues of the set of reference macros which automatically generate a reference mark whose name is **\?** after filing away the text of the figure caption. What they do is first execute the command

\FIG\?{ *text* **}**

and then follow this by either Fig.~ or fig.~ . The analagous macro for tables is

\Table{ *text* }
which performs the same task as typing
\TABLE\?{ *text* } Table~ **\?**

This completes our discussion of macros associated with *things which come at the back*.

6. FOOTNOTES: THINGS WHICH COME AT THE BOTTOM

Everybody knows what a footnote is and you have already seen an example of a footnote appearing on page 5. Basically what a footnote macro has to do is attach a footnote mark to a specified word and then file away material to be put at the bottom of the page (in ten point type). The macro has to decide what to do if the footnote is too long, or if many footnotes appear on the same page. In addition, it has to decide what footnote mark to generate. PHYZZX has two macros which carry out this feat.

The first macro, **\foot**, has the syntax
\foot{ *text* }
where the text of the footnote is to be inserted between the braces. When you invoke this command immediately following the word to which you want to attach a footnote mark it does several things. First it generates the footnote mark and attaches it to the preceding word. If you are in the Nucl.Phys. format, this symbol will automatically be chosen from the set of symbols $\star$, $\dagger$, $*$, $\ddagger$, $\natural$, $\diamond$, $\bullet$ and ∇. The macro will continually cycle through this table, according to a specified algorithm, in such a way that the same symbol will not (except in truly extraordinary circumstances) appear twice on a single page. If you are in PHYSREV mode, then the footnote mark will be a superscripted $\sharp$ sign followed by the current value of **\number\footsymbolcount**. This is a truly ugly convention, however it is included only to provide something for people who insist on numbering footnotes.

Actually, Phys.Rev. does not like footnotes and preferes this material be included in with the references as endnotes.

Note, that as with references, the text of a footnote is indented using the TEX command **\hangindent**, hence to insert multi-paragraph footnotes one cannot use the **\par** command but must use instead the **\subpar** command. If you don't do this, but use **\par** (or equivalently insert a blank line) then TEX will, obligingly, turn off the **\hangindent**. This will, of course, mess up the footnote considerably.

Hints and Warnings

Since footnotes use TEX's basic **\insert** command, they are meant to be used in *unrestricted horizontal mode*. In other words, you use them inside paragraphs, but outside of **\hbox**, **\centerline** or similar commands. If you don't understand the meaning of the phrase *unrestricted horizontal mode* you should read about it in the TEXbook, or in **First Grade TEX**. This fact means that one has to be careful in attempting combine certain other control sequences with the **\foot** command.

1. Do not define references, figure captions, table captions, etc., inside the text of a footnote. If you must put a reference mark inside a footnote use the command **\REF** somewhere before you start the footnote and then use **\refend** inside the text of the footnote.

2. If you choose to invoke a footnote when you are in vertical mode between paragraphs, then the footnote, complete with its footnote mark, will be generated; however, since there is nothing to attach this mark to, no superscripted footnote mark will appear. Since the command will generate a footnote mark, the counter **\footsymbolcount** will be advanced and so, insofar as PHYZZX is concerned, the symbol is waiting around to be used. Hence, if you wish to attach this symbol to a word at a later point this can be accomplished by use of the subcommand **\footattach**. All you have to do to use **\footattach** is type it immediately following the word to which you wish to attach the footnote mark. This command works just like **\attach** or **\refmark** except that

it attaches the current footnote mark and knows whether to use Nucl.Phys. or Phys.Rev. format.

3. We already noted that if you wish to attach a footnote inside of a box (either horizonatal or vertical) the way to do it is to use **\foot** after the previous paragraph has ended, but before you make the box in question. Then use the command **\footattach** inside the box. What happens if you need to attach the footnote mark to something inside a box which appears inside a paragraph? In this case you can use the PLAIN command **\vadjust** to accomplish this feat. Simply type
   \vadjust{ *text* **}\nobreak**
   immediately before the box and then use **\footattach** inside the box.

Controlling Your Footnote Marks

Once again, this section is for those who wish to go beyond the mundane and use unusual symbols for their footnotes. This can be done by using the PHYZZX command
\footnote{ *symbol* **}{** *text* **}**
which will generate a footnote using the *symbol* you have specified as the footnote mark. The symbol can be any math mode symbol, or any outré thing that you can think up. The symbol does have to be in math mode, so if it is something you made up instead of a math mode control sequence, be sure to **\hbox** it and enclose it in $ signs.

7. ITEMIZED LISTS: POINTS, ITEMS AND OTHER STUFF

You now have all of the necessary information to allow you to use PHYZZX to format a paper for the printer: you can tell him to set up chapter, section and subsection headings; number equations; format the reference, figure caption and table caption page; format footnotes, keep track of how many are on a page and carry them over to succeeding pages if it is necessary. From a stylistic point of view there is nothing more for you to learn about papers. Using the basic commands defined in PLAIN TEX and those PHYZZX macros which we have already discussed, you are now capable of generating a paper of the correct style to be the body of a SLAC preprint. What we have not discussed to this point are things like macros for title pages and macros which make typing special constructions simpler than they would be in PLAIN.

In this section we will discuss a set of macros which make the construction of *itemized lists* particularly easy to achieve. Itemized lists appear most often when one is outlining a paper and trying to arrange the material in some systematic fashion. This sort of outline shows up in memos and the introduction to a paper. Itemized lists also appear in the body of the text when one wants to make a series of points and, at the same time, clearly separate them from the format of the general body of the text.

Lists of this sort are conventionally set up as a series of indented paragraphs preceded by some sort of highlighting symbol. This symbol can be a number, letter, roman numberal or some math mode symbol such as a dot, star, diamond, etc. When used in writing outlines, etc., one indicates the major divisions of the outline by indenting the text associated with them some fixed amount and highlighting this text with a specific type of symbol. When itemizing the ideas which fall within each major division, one usually indents the text somewhat more than one did for the major divisions and highlights each of these sub-points with a symbol which differs in type from those used to highlight the major points. One then further subdivides each of these categories into smaller steps and indicates this fact by once again increasing the amount of indentation and once again changing the type of highlighting symbol used. So on ad-infinitum ...

Obviously, at least for those of you who have read the TEXbook, all of this can be done using the commands **\hangindent** and **\hangafter**. However, if one is creating long lists of points, subpoints, subsubpoints, etc., then keeping track of the symbols and appropriate levels of indentation can get to be a problem. For this reason PHYZZX provides a series of macros to make this as simple as possible. As in the previous cases, we will begin by discussing the fixed format versions of the macros, which are the ones which are easiest to use, and go on to discuss the ones which allow you greater flexibility.

7.1. Fixed Format Lists: Points, Subpoints, Subsubpoints

Points

The macros **\point**,**\subpoint** and **\subsubpoint** exist to make it easy to deal with itemized lists having up to three levels of indentation. The first level of indentation is the smallest and is generated by typing the command **\point** followed by the text associated with that point. Note that unlike the reference or footnote macros, the text associated with a **\point** *does not* need to be enclosed in braces. Points, etc. are constructed using the **\hangindent** command and so TEX accepts all of the text up to the next **\par** (or blank line) and formats it as you desire. For this reason, if a point is to consist of more than one paragraph *do not use the* **\par** *command to generate the second paragraph* but, instead, use the special command **\subpar**. In addition to indenting the material to follow, the command **\point** also generates a number for the point and attaches it to the left of the first line. You have seen examples of points in other parts of this writeup; go back and look at them if my sentences are confusing. Each time you type **\point** PHYZZX generates a new paragraph, increases the number of the last point by one and appends this to the left of the first line of the new point. Obviously, this works well until you finish with a given set of points and then decide you want to make a new list of points at a later point in the same paper. Except in unusual circumstances you would like the first item on this list to be labelled 1, and not start up from the number assigned to the last point in the previous list. To avoid this sort of calamity you use a special command to begin a new set of points namely, the command **\pointbegin**. (Clearly this choice of nomenclature is far from sprightly and imaginative, but it is easy to guess what the correct command is even if you have forgotten it.)

As an example suppose you type

\pointbegin

This is the first point I wish to make.

\point

This is the second point.

\point

I am not very imaginative, so this is the third point.

You then obtain

1. This is the first point I wish to make.

2. This is the second point.

3. I am not very imaginative, so this is the third point.

You then return to typing ordinary text by following the text of the last point in your list with a **\par** or blank line.

Subpoints and Subsubpoints

The format of a subpoint is that it is indented further and highlighted by lower case letters enclosed in parentheses. The command for generating a subpoint is **\subpoint** and, as with points, each time you type **\subpoint** you automatically generate the next letter in the alphabet as a highlight. As with points, in order to start the numbering over again you start each set of subpoints by typing **\subpointbegin** instead of **\subpoint**. For example, typing

\pointbegin

This is the first point I wish to make.

\subpointbegin

This is the first subpoint.

\subpoint

This is the second subpoint

\subpoint

Etc.

\point
This is the second point.
will generate

1. This is the first point I wish to make.

    (a) This is the first subpoint.

    (b) This is the second subpoint

    (c) Etc.

2. This is the second point.

Subsubpoints are the last automatically provided subdivision and are generated using the commands **\subsubpointbegin** and **\subsubpoint**. They are indented still further and highlighted by lower case roman numerals enclosed in parentheses. I leave it to you to generate you own examples of subsubpoints, the syntax is completely similar to that of the previous examples.

The command **\spoint** is acceptable as an abbreviation for **\subpoint**. Abbreviations for the other commands are similar; i.e.,

\spointbegin	for	**\subpointbegin**
\sspoint	for	**\subsubpoint**

and

\sspointbegin	for	**\subsubpointbegin**

7.2. For Less Structure Consider Items

There will come the time, especially for those of you who learn to use TEX to generate transparencies, when you want to be able generate itemized lists, but you don't want the highlighting to happen automatically. Instead you might like to generate stuff with a **\bullet** or an **\ast** as a highlight. For those among you who are left brain dominated and feel the need for a macro set which takes care of the boring details of indentation, but allows you to play fast and loose with the highlights, PHYZZX provides the macros

\item{ *symbol* }

\subitem{ *symbol* }

and

\subsubitem{ *symbol* }

(The last two commands can be abbreviated as **\sitem** and **\ssitem** respectively.)

- • For example, this item was generated by typing **\item\bullet**. Note that I did not put **\bullet** in braces since it is a single control sequence. If I had, however, it would have done no harm. Spaces and blank lines before **\item** are irrelevant, however, since **\bullet** is a control sequence and eats one space following it, you have to have at least one blank after *symbol.* Once again I caution you that if you put a blank line or **\par** after **\item** and before the text, or inside a multiparagraph **\item**, then you destroy the level of indentation. *Remember, indentation only holds for a single paragraph in TEX.*

∗ As a second example consider this item which was generated by typing **\item\ast**

— Or this, which was generated by typing **\item\dash**.

At this point I have inserted a blank line in my text and have caused the item to terminate and the text reverts to its ordinary format.

♣ This is item **\item\clubsuit**

    ◇ And this is **\sitem\diamondsuit**

        — Finally, this is a subsubitem;
        it was generated by typing **\ssitem\dash**.

7.3. Having Your Cake and Eating It Too!

Clearly, the **\item**,**\sitem** and **\ssitem** macros provide you with the ultimate in flexibility in the choice of highlighting conventions. With them alone you can choose your own format for numbered lists and never have to accept the format imposed by **\point**, etc. However, the total lack of structure of **\item**, etc. has a drawback; namely, if you use it to define your points, subpoints etc., you are responsible for keeping track of all of the numbering. This is certainly not too formidable a task, but it can get tedious.

Fortunately, there is a middle ground.

Levels of Indentation

If all that you want to do is change the levels of indentation of points, subpoints and subsubpoints you only have to modify the parameter **\itemsize**. This parameter is a dimension stored away in TEX's memory and items are indented **1.0\itemsize**, subitems are indented **1.75\itemsize** and subsubitems are indented **2.5\itemsize**. The current value of **\itemsize** is 30pt (30 points); however, you can change this by simply typing

\itemsize = *40pt*

or whatever else strikes your fancy.

Getting More Serious

Let us suppose, although I cannot imagine that this could possibly be the case, that you have more serious problems with points, subpoints and subsubpoints. Perhaps you would prefer to define your own listing macros so that **\mypoint** generates a point whose highlight is a capital letter followed by a period; **\myspoint** generates a subpoint highlighted by a number square brackets; and **\mysspoint** generates a subsubpoint highlighted by a lowercase letter in parentheses. You have two options, you can write your own macro set or you can use the PHYZZX command **\newlist**.

The command **\newlist** allows you to define your own listing macros which format things in a manner which is closer to your heart's desire. All you have to decide is what sort of delimiters you want around the highlighting number and what you want the numbering convention for that level of point to be. Your available choices are the five defined when we discussed changing chapterstyle; namely, **\Number**, **Alphabetic**, **\alphabetic**, **\Roman** or **\roman**.

The syntax of the newlist command is as follows

\newlist*name= [left delim.][style]& [right delim.] & [dimen];*

(Note that the =, & , and ; all count.) As an example of its application note that the **\point** command is defined by saying

\newlist\point=\Number&.&1.0\itemsize;

This line tells PHYZZX to define a new command called **\point** and to have it generate a highlighting symbol which consists of no left delimiter, followed by a number, followed by a right delimiter which is a period and the indentation of a point is to be **1.0\itemsize**. Similarly, subpoints are defined to be

\newlist\subpoint=(\alphabetic&)&1.75\itemsize;

and subsubpoints are defined by saying

\newlist\subsubpoint=(\roman&)&2.5\itemsize;

The abbreviations for **\subpoint** and **\subsubpoint** are obtained by following their definition with the statements

\let\spoint=\subpoint

\let\sspoint=\subsubpoint

Note, if you have chosen to redefine **\subpoint** and **\subsubpoint** using newlist, the commands **\spoint** and **\sspoint** will *not* be redefined unless you follow your newlist instructions with the two lines printed above.

To define the new set of listing macros mentioned earlier you would type

\newlist\mypoint=\Alphabetic&.&1.0\itemsize;

\newlist\myspoint=[\number &]& 1.75\itemsize;

and

\newlist\mysspoint=(\alphabetic&)& \2.5\itemsize;

Note, while there are only five styles available for numbering things, delimiters are limited only by your imagination.

8. MISCELLANEOUS STUFF ABOUT PAPERS

8.1. Page Numbers

Page numbering is a totally automatic procedure and, in general, requires none of your attention. Normally, if you are not doing anything funny PHYZZX will produce a document whose pages are numbered consecutively starting from 1. In general the first page you get will be numbered, unless you have told PHYZZX not to do so. The macros **\FRONTPAGE**, **\MEMO**, **\letter** and **\titlepage** do issue such a command, and when invoked produce a document where all pages but the first page are numbered consecutively. If you type **\FRONTPAGE** then PHYZZX will complete the previous page (if there is any), set the current value of the page number to 1 and then tells TEX not to number this page. Visible numbers appear at the bottom of all the pages which follow and the first visible number is 2. The command **\Frontpage** is a synonym for **\FRONTPAGE**. The **\letter** and **\MEMO** macros, which will be discussed later, and the **\titlepage** macro, which is invoked when you are about to type the title page of your latest magnum opus, all suppress the printing of the pagenumber at the bottom of the first page.

If you wish to start a document in the middle and set the pagenumber all you have to do is reset the counter called, of all things, **\pagenumber** by typing
\pagenumber= *number*
This will make the first page TEX prints have this number. To make sure that the printing of this number is not suppressed you should also type **\frontpagefalse**. If you wish to have PHYZZX number the pages using roman numerals you can accomplish this by saying **\pagenumber=-1** (or **\pagenumber=-n** if you wish to start with the roman numeral equivalent to the number n).

— If you wish no pagenumbers type **\nopagenumbers** (be careful, its not easy to make them come back afterwards).

⋆ If you want to suppress the page number on a particular page you can type **\frontpagetrue**, which will make PHYZZX think that this is the front page.

You'll sometimes have to play to make this happen where you want.

8.2. Spacing

We have already talked about changing font sizes and selecting spacing, and have pointed out that three families of fonts are available. These are fonts which are fourteen points high, twelve points high and tenpoints high. The twelvepoint font is the font PHYZZX uses for normal text because it looks best when reduced. The fourteenpoint fonts are used for chapter titles, etc., and the tenpoint font sets, which are kind of small when reduced, are generally reserved for footnotes. To change font size at will you type either **\tenpoint**, or **\twelvepoint**, or **\fourteenpoint**. To change spacing conventions you use the commands **\singlespace**, or **\normalspace** or **\doublespace**. Note that all of these commands can be used inside the paper to temporarily change what is going on; however, you should only invoke the commands which change spacing when you are between paragraphs. If you violate this rule, eventually, you will be sorry.

8.3. Commands Which Break Lines and Pages

TEX has its own way of breaking lines and pages and you can learn about them by reading the TEXbook. However, we have added three commands which are convenient. The first command **\nextline** allows you to generate and incomplete line. For example, typing *This is the way the world ends* **\nextline** produces
This is the way the world ends
Notice that TEX has not tried to space the words out to fill the line the way it would have if you had typed **\break**. The command **\nextline** is essentially the same as typing **\hfil\break**; it differs slightly in that it also does some things about glue, but you don't have to know about that.

The command which forces TEX to break the page at will is **\endpage**. Typing this command forces TEX to start a new page and at the same time it insures that even if the previous page is not completely full of text the material will be spaced correctly.

The command with which you finish a paper, memo, letter, etc., is **\bye**. This command tells TeX to complete the last page, print it and then turn itself off, returning you to CMS.

8.4. Some Remarks About Penalties

There are a few commands, which belong to PLAIN and not PHYZZX which you should know about. The basic command which allows you to tell TeX that you would advise it that this is a good or bad place to break a page or line of text is **\penalty**. Penalties can be either positive or negative. If a penalty is positive it tells TeX that this is not a good place to put in a break. The bigger the penalty is, the worse TeX thinks it is to cause a break at this point. The biggest allowable penalty is **\penalty 10000**. If you type **\penalty 10000** at a specific point in the text than TeX, as the printer, will assume that no matter what his judgement tells him, you insist that no break can occur at this point. TeX defines a control sequence **\nobreak** which equivalent to typing **\penalty 10000**. If the penalty is negative, then TeX is advised that you believe this point to be a good place to cause a break. The more negative the penalty, the better you think this place is. If the penalty is negative and greater than -10000, then TeX will use its own judgement to balance off your suggestion against its own esthetic criteria. If the penalty is -10000 (or less), then you are telling TeX that your judgement must carry the day. The abbreviation of **\penalty -10000** is **\break**.

Note that using the same commands inside paragraphs and between paragraphs tell TeX to do different things. Inside of a paragraph **\break** tells TeX to break the current line at this point, between paragraphs it tells TeX to break the page. If you wish to force TeX to break the page inside a paragraph, without terminating the paragraph, you have to type
\vadjust\break

8.5. The Titlepage

We have now finished with our discussion of macros which PHYZZX defines for simplifying the formatting of the body of a paper. To complete our discussion of the paper generating capabilities of PHYZZX we now have to turn to the question of generating the title page. A SLAC title page has several parts.

1. First, in the upper right hand corner of the title page one has the publication number, date of submission and a line telling whether this paper is to be distributed as T, E or T/E . This chunk of text has to be set up as a block with the appropriate typeface chosen for each line and then the entire block is right adjusted. This block is referred to as the pub-block

2. Next comes the title of the paper, which is set in fourteen point type and this is followed by the list of authors and the institutions they come from.

3. After the title and author list you have a choice. If you have a short abstract it is included at this point and printed on the title page. If, however, the abstract is too long, it will not fit on the title page. In this event you skip it and put it on the next page of the paper.

4. After the abstract, or after the title and author list if the abstract is too long, we have a line which tells the journal to which the preprint has been submitted.

5. The next line ends the main body of the title page. After this line the remaining material, which are contract acknowledgements, etc., are put at the bottom page in footnote format.

As you can see from this discussion the problem of formatting the title page can get to be quite difficult if we don't have special macros to help out. For this reason a special set of macros for formatting title pages has been included in PHYZZX.

The Publication Block or Pub-block

The first commands which you use relate to setting up the block of type which

gives the date, publication number, etc. If you do not want such a block, you type
\nopubblock
If, as will usually be the case you want a pub-block, then you have to specify what goes into it. To specify the publication number you type
\pubnum{*number* }
This causes PHYZZX to put the quantity **\pubnum** into the publication number. When you start the quantity **\pubnum** is **0000**.

The next item which appears in the macro is the date. If you do nothing, then each time you run TEX to produce a new copy of the file it will put the current date in this place. If you wish to set the date to printed by yourself, then you type
\date = { *month day, year* }
Note the braces around the entry because they have to be there.

Finally, the last thing that has to specified for the pub-block is the pubtype, which is either T, E, or T/E. To specify this quantity you type
\pubtype={ *publication type* }

Specifying the Titlepage

The next macro which has to be invoked is
\titlepage
This is a single control sequence which you type in just before the titlepage. This command must follow the setting of the **\pubnum**, **\date** and **\pubtype** or you will only get the default values for these quantities. What this macro does is tell TEX that this is the titlepage of the document and it should not be numbered.

Typesetting the Title

The next item to be typed is the title of the paper. Since this is set in different size type and requires a certain amount of spacing above and below the text, you should use the macro **\title**. The syntax of this command is
\title{ *title* }

You are now ready to type in the name and institution of the author (or authors) of the paper.

Getting the Author(s) Right

If there is a single author you type

\author{ *author's name* }

This is, of course to be followed by his address. PHYZZX will set the name in upper and lower case capital letters and center it on the page. Then, if you use the macro **\address** PHYZZX will typeset the address in italics and center it directly below the author's name. The syntax of this macro is

\address{ *address* }

In general PHYZZX will let TEX decide how to break the address into several lines if it is too long. If you wish to force these breaks use the command **\break**. Note, this is a place where you do not want to use the command **\nextline** or you will force the address to come out without having been properly centered. For those lucky persons who live at SLAC, there is a special macro to save them typing. This macro has the name **\SLAC** and if you wish to tell PHYZZX that your address is SLAC all you have to type

\SLAC

If the author of the paper is a visitor, or is here on detached service, then you might want to give two addresses. To have TEX print the second address below the first with the word *and* in between, all you have to do is type

\andaddress{ *second address* }

If the paper has more than one author you have two options. If the number of authors is small and they come from different institutions, then you use the macro **\andauthor** whose syntax is

\andauthor{ *author's name* }

This does what the **\author** command does but it also precedes this name with a line which has the word *and* centered above the author's name. This command is then to be followed by **\address** in order to give the address of the second author. This can go on as long as you like.

All of this works well for theorists, however it does not work well for experimental papers. Experimantal collaborations tend to involve many authors from several different institutions. All we can do in this situation is to type the command **\authors** and then follow this with a list of authors and institutions. All this command does for you is put the correct amount of space between the list of authors and the material preceding it.

Setting up the Abstract

At this point you have to decide whether or not you want to include the abstract on the titlepage. If you wish to do so this is the time to type **\abstract**. This will cause TEX to typeset the word ABSTRACT in the center of the next line of text, skip the appropriate amount of space and set you up in a new paragraph. The next thing you do is type in the body of the abstract. If you don't put the abstract on the titlepage then you carry out this sequence of commands after you have finished the titlepage with the command **\endpage**. After you type in the abstract on the second page you should force TEX to print this as a separate page by once again typing **\endpage**. This will produce an abstract on its own page with white space above and below the text.

If you are in the situation of having an abstract which is too long for the title page but too short to fill the entire second page you will get ugly output if you type it in and follow it with the command **\endpage**. TEX will simply shove everything to the top of the page because **\endpage** is the same as **\vfil\break**. The way to force TEX to center the title and material is to type

\vfil
\abstract
text of abstract
\endpage

Where's The Paper Going?

In order to specify the journal to which you have submitted the paper you type
\submit{ *journal* **}**
For example, if it is going to Phys.Rev. D you would type
\submit{ *Physical Review* **\bf** *D* **}**
which will produce the line

Submitted to Physical Review **D**

Acknowledgements

It is customary to acknowledge the support of various grants or contracts somewhere on the titlepage. Generally this is done in the form of a footnote. The easiest way to get TEX to typeset such acknowledgements is to use the macro **\foot**. You insert this footnote just after the last word of the title or author's name, depending upon the situation, and then continue setting the title page. The **\title**, **\author** and **\address** macros have been written in such a way that the **\foot** command can be used with these macros without any trouble. If you fool around with the title page and put things inside boxes using the PLAIN TEX commands **\line** or **\centerline** this is not true. In this case the text will be put inside a box in restricted horizontal mode and you must use the commands **\footnote** and **\footmark** as explained in our discussion of footnotes.

For theory group users there is a special macro which generates the acknowledgement to the Department of Energy. the command is **\doeack**. To generate this acknowledgement you simply type **\doeack** immediately after the last word in the title. If other groups want to have their own **\doeack** macro they need generate a file called MYPHYX TEX and make sure it contains the lines
\def\doeack{\foot{ Work supported by the Department of Energy, contract **$\caps** DE-... **$ }}**

Finishing the Title Page

Close the title page with command **\endpage**. If you don't do this TEX will make some stupid decision about adding extra material to this page. You won't be happy with the results.

8.6. A Sample Titlepage

Let us conclude this section by including a sample title page. If you type

```
\pubnum={ 6666}
\date{September 1984}
\pubtype{CRAP}
\titlepage
\title{THE JOY OF TYPING PAPERS USING PHYZZX \doeack}
\author{Marvin Weinstein \footnote{\dag}{another acknowledgement } }
\SLAC
\andauthor{V.~ K.~ Kaplunovsky }
\address{ Princeton }
\abstract
\centerline{ This paper is a crock. }
\submit{ Physical Review D }
\endpage
```

then you get

SLAC–PUB–6666
September 1984
CRAP

THE JOY OF TYPING PAPERS USING PHYZZX[⋆]

MARVIN WEINSTEIN

Stanford Linear Accelerator Center
Stanford University, Stanford, California 94309

and

V. K. KAPLUNOVSKY

Princeton

ABSTRACT

This paper is a crock

Submitted to *Physical Review D*

⋆ Work supported by the Department of Energy, contract DE–AC03–76SF00515.
† another acknowledgement

9. MEMOS

We have now finished discussing all of the macros which have to do with the formatting of papers. Now as promised we will turn to a discussion of the two other formats which PHYZZX supports. The first format is that of a SLAC memorandum. The second format is that of a letter.

A memorandum, as we have defined it, is a document meant for internal distribution, addressed to a specific group of people and designed to transmit information on a specific subject. This sort of document tends to have a heading which identifies it as a memorandum, gives the date, says to whom the memorandum is being sent, from whom the memorandum comes and briefly summarizes the topic being discussed. This heading is then separated from the body of the memorandum by some sort of ruled line and then one types in the text of the memorandum.

To type a memorandum you begin by typing the formatting instruction
\MEMO
which tells PHYZZX the document to follow is a memorandum. It also forces the document to be single spaced and generates the heading which identifies the document as a SLAC memo and gives the date at which the file was processed; *i.e.*, the current date. If you wish to type a heading which has the date of your choice then use the command **\memodate** instead of **\MEMO**. The syntax of this command is

\memodate{ *date of your choice* }

Now you have to generate the remainder of the heading; i.e., the to, from and topic (or subject) entries. The macros which do this are
\to , **\from** , and **\topic** or **\subject**
The commands **\to**, **\from** and **\topic** or **\subject** work like points or items. Hence, they do not need braces.

Finally, after typing in these commands followed by the relevant text you need to generate a line to separate the heading from the body of the memorandum. This is done with the command **\rule**. Having done this you are now ready to type in the text of the memorandum.

Note, all of the itemized list macros which we have discussed in earlier sections work in the MEMO format too. Hence, you can use them with impunity. The same is true for footnotes, etc.

Finally, when you complete the memo, if you wish to indicate to whom copies are to be sent you type

\copies

This command works like **\point** or **\item** and so you would type

\copies

Mr. A. **\nextline**

Mr. B. **\nextline**

etc.

in order to get what you want

Don't be surprised when you generate multipage memoranda. The way pages are numbered is going to be different from the numbering of papers. For a memorandum all but the first page are numbered with the numbers appearing at the top of the page.

A sample memorandum is generated as follows;

\MEMO

\to

Whom it may concern

\from

Marvin Weinstein

\subject

Generating memos without tears.

\rule

Look how easy it is to generate a sample memo. Even a baby can type something like this. If you use this macro to type your memos people will probably take what you have to say more seriously. (They shouldn't but they will!) You can even type points, for example:

\pointbegin

Any fool would recognize that if we proceed in this way we can only wind up with a disaster.

\point

Who cares if the DOE doesn't want us to use the construction funds to take a Hawaiian vacation. They're just a bunch of old stick in the muds.

\copies

Burton Richter **\nextline**

Richard Taylor

\endpage

Which generates something which looks like this

SLAC MEMORANDUM September 1984

TO: Whom it may concern

FROM: Marvin Weinstein

SUBJECT: Generating memos without tears.

Look how easy it is to generate a sample memo. Even a baby can type something like this. If you use this macro to type your memos people will probably take what you have to say more seriously. (They shouldn't but they will!) You can even type points, for example:

1. Any fool would recognize that if we proceed in this way we can only wind up with a disaster.
2. Who cares if the DOE doesn't want us to use the construction funds to take a Hawaiian vacation. They're just a bunch of old stick in the muds.

cc: Burton Richter
Richard Taylor

If we use the commands **\memodate** and we would obtain

\memodate{ *the fifth of octember* }

\to

Whom it may concern

\from

Marvin Weinstein

\topic

Generating memos without tears.

\rule

Look how easy it is to generate a sample memo. Even a baby can type something like this. If you use this macro to type your memos people will probably take what you have to say more seriously. (They shouldn't but they will!) You can even type points, for example:

\pointbegin

Any fool would recognize that if we proceed in this way we can only wind up with a disaster.

\point

Who cares if the DOE doesn't want us to use the construction funds to take a Hawaiian vacation. They're just a bunch of old stick in the muds.

\copies

Burton Richter **\nextline**

Richard Taylor

SLAC MEMORANDUM fifth of octember

TO: Whom it may concern

FROM: Marvin Weinstein

TOPIC: Generating memos without tears.

Look how easy it is to generate a sample memo. Even a baby can type something like this. If you use this macro to type your memos people will probably take what you have to say more seriously. (They shouldn't but they will!) You can even type points, for example:

1. Any fool would recognize that if we proceed in this way we can only wind up with a disaster.
2. Who cares if the DOE doesn't want us to use the construction funds to take a Hawaiian vacation. They're just a bunch of old stick in the muds.

cc: Burton Richter
Richard Taylor

10. TYPING LETTERS AUTOMATICALLY

In addition to typing papers and memos PHYZZX supports a format for typing letters. To choose this format all you need to type the command

\letters

immediately following

\input phyzzx

This command will cause PHYZZX to switch over to a different page size, 8 inches long by 6.5 inches wide (this is a little wider than a preprint) and it automatically changes the linespacing to singlespace. In addition, the same command tells PHYZZX to put a SLAC letterhead at the top of the first page, and to change the way in which pages are are numbered.

Having switched over to this mode the very next thing you need to do is type in the date, followed by the name and address of the person to whom the letter is being sent. Getting the alignment of these various elements just right takes some doing and so we have a macro to do the job. The name of this macro is

\letter

Do not confuse this command with the initialization command **\letters**. Both statements must appear in your file and the command **\letters** must come first. I apologize for this construction but it is a leftover from previous incarnations of PHYZZX. Since we wanted this version of PHYZZX to be upward compatible we didn't change it. What the command **\letters** does is to tell PHYZZX that the text to follow is to be set in **\letterstyle** and then it makes PHYZZX print **STANFORD UNIVERSITY** across the top of the first page. The command **\letter** tells PHYZZX that this is the first page of a letter, prints the rest of the SLAC letterhead, suppresses the printing of the pagenumber at the bottom of this page and then allows you to type in the name and address of the person to whom the letter is being sent. The syntax of this command is

\letter{ *Name* **\cr**

first line of address **\cr**

... **\cr**

last line of address **\cr }**

Note that any of the **\cr**'s can be replaced by the command

\cropen{ *extra space* **}**

which will cause PHYZZX to leave the extra amount of space requested between the lines; or you can replace the **\cr** by **\crr** if what you want is 10 points of extra space.

The reason for typing the commands **\letters** and **\letter** separately is that **\letters** initializes the format for typing a letter and then **\letter** allows you to start several new letters in a single file. So long as you begin each letter with the **\letter** macro each letter will start on a new page, and the letters will not know about one another.

As with the titlepage macro, PHYZZX assumes that the date which is to appear above the name and address is the current date. If, however, you wish to fix the date appearing on the letter you can do so by typing

\date{ *your date* **}**

The **\date** macro will then make *your date* appear on all of the letters in this file.

Once you have generated the letterhead, date and filled in the name and address of the person to whom the letter is being sent, you are ready to type in the body of the letter. This is to be done in the same manner as for a paper or memo, and all of the itemized list, etc., macros will work in this format too.

Typing the Salutation

Finally, having typed in the body of the letter we come to the problem of generating the signature. Once again, since placing this upon the space and leaving the correct space between the *Very truly yours,* and the typed version of your name can get to be hairy, we provide the macro **\signed**. The syntax of this command is

\signed{ *Very truly yours (or whatever)* **\cr**

Your name **\cr** }

If copies of the letter are to go to several people you type this at the bottom of the letter using the **\copies** command discussed in the section on memos.

You finish a letter with the command **\endletter**. This command insures that if there are several letters in the same file each one will be processed independently.

A sample letter is generated as follows

\letters

\letter{ *Dr. Boris Kayser* **\cr**

Program Director for Theoretical Physics **\cr**

National Science Foundation **\cr**

Washington, D. C. 20550 **\cr** }

Dear Boris,

Please send more money.

\signed{ *Sincerely,* **\cr**

Michael E. Peskin **\cr** }

\endletter

\end

which produces

STANFORD UNIVERSITY

STANFORD LINEAR ACCELERATOR CENTER

SLAC, Bin 81
P. O. Box 4349
Stanford, CA 94309
(415) 926–2266

fifth of octember

Dr. Boris Kayser
Program Director for Theoretical Physics
National Science Foundation
Washington, D. C. 20550

Dear Boris,

Please send more money.

Sincerely,

Michael E. Peskin

1. Multiple Letters

Very often you wish to send the same letter to many different people; hence, the text of each letter is the same but the salutations will be different. There are two ways to go about generating such a series of letters. The only difference between them is that the first is somewhat more conservative of computer time, but it can only be used for relatively short letters; the second is a bit more demanding of computer time, but it doen't care how long the letter is. By a short letter I mean one which is no more than one or two pages in length.

Short Letters

To type in short letters you begin by typing

\letters\multiletter

This command is to be followed by the text of your letter. What **\multiletter** does is tell PHYZZX to file away the material to follow in the depths of its memory, to be used later. You let PHYZZX know to stop stuffing stuff away by typing

\letterend

when you have finished. Having now saved the body of the letter for future use you generate your letters by typing

\letter{ *Addressee* }

followed by

Dear Mr. Soandso **\par**

followed by the commands

\lettertext\endletter

As you will have guessed the command **\lettertext** is the name PHYZZX has assigned to the body of the letter.

For Longer Letters

The second method for generating many letters is slightly more wasteful of computer time, but to my mind it is simpler to use. I reccommend it over the command **\multiletter** and have included a brief discussion of the previous case only to keep the macro set upward compatible. Do us both a favor and forget I mentioned it.

This method takes a little longer because it requires TEX to input the same file many times. It is simpler because it requires no change in syntax from the format used for a single letter. To use this method you create a file YOUR FILE A (the filetype doesn't have to be TEX here), which contains the body of your letter, including your signature. You then create a driver file (filename TEX A) of the form

\input phyzzx

\letters

optional **\date{** *your date* **}**

\letter{ *Addressee* **}**

Dear ... **\par**

\input your.file

\endletter

...

\letter{ *Addressee* **}**

Dear ... **\par**

\input your.file

\endletter

\end

Note, the period between the filename and filetype is crucial. What is being done in this method is that we are using TEX's ability to input an external file as often as you wish. This technique has the added advantage that you can even have the letters vary somewhat. To perform this trick you use the **\def** command. To begin you use the names of some undefined control sequences in the file YOUR FILE A, and then redefine these sequences in the main driver file after each **\endletter** and before the start of the next letter.

11. MISCELLANEOUS MACROS WHICH PHYZZX DEFINES

We have now finished our discussion of all of the macros which pertain to the formatting of special documents. This section simply lists some special macros which we have included in PHYZZX simply because they simplify the task of typing things which occur frequently.

The commands will be listed giving the syntax of each command and an example of what it does.

1.

\ie generates *i.e.*

2.

\eg generates *e.g.*

3.

\dash generates —

4.

\\\\ generates \\

In math mode we have the following useful commands

5.

\coeff{ a }{ b } generates

$$\tfrac{a}{b}$$

6.

\partder{ *f(x)* }{ *x* } generates

$$\frac{\partial f(x)}{\partial x}$$

7.

\bra{ *\Psi* } generates

$$\langle \Psi |$$

8.

\ket{ *phi* } generates

$$|\phi\rangle$$

9.

\VEV{ *H* } generates

$$\langle H \rangle$$

10.

\Tr{ *G* } generates

$$\mathrm{Tr}\, G$$

11.

\int generates

$$\int$$

12.

\prop generates $\propto$

13.

\lsim generates $\lesssim$

14.

\gsim generates $\gtrsim$

12. GOODBYE

This completes the basic writeup of the capabilities of PHYZZX. As I indicated in the introduction, this is not the final version of the macro set, but is in fact the last but one version that I will have something to do with. The final version will appear one of these days and will include in the body of the FMT file several macros for creating wrap around figure inserts and simple tables. The macros already exist but have not yet been collected together and finalized in format. In addition, it will probably include a label making capability to generate mailing labels for letters. Don't hold your breath for this version, however, first I have to get over having typed this documentation.

12.1. Using Your MYPHYX File

I promised you a short discussion of tailoring PHYZZX into your own inimitable macro package. The simplest way to do this is to have a complicated file called

MYPHYX TEX on your A disk. If there are a few features of PHYZZX which do almost what you want, but not quite, you can look at the file PHYZZX TEX. By reading Knuth and studying this file you will see how everything PHYZZX does is made to happen. The way to change something is to steal the relevant definition from this file and copy it to your MYPHYX file. Then, make the changes which you wish to make. Since PHYZZX inputs this file after it makes its own definitions, you desires will override the PHYZZX definitions.

GOOD LUCK !!!!!