International Atomic Energy Agency
Nuclear Data Services
DOCUMENTATION SERIES OF THE IAEA NUCLEAR DATA SECTION

IAEA-NDS-39
Rev. 12, Nov. 22, 2004

PREPRO 2004
2004 ENDF/B Pre-processing Codes
(ENDF/B-VII Ready)

Owned, Maintained and Distributed
by
The Nuclear Data Section
International Atomic Energy Agency
P.O. Box 100
A-1400, Vienna , Austria

Originally Written
by
Dermott E. Cullen
University of California
Lawrence Livermore National Laboratory
L-159
P.O. Box 808
Livermore, CA 94550, U.S.A.
tele: 925-423-7359
e.mail: cullen1@llnl.gov
website: http://www.llnl.gov/cullen1

Abstract: The codes are named "the Pre-processing" codes, because they are designed to pre-process ENDF/B data, for later, further processing for use in applications. This is a modular set of computer codes, each of which reads and writes evaluated nuclear data in the ENDF/B format. Each code performs one or more independent operations on the data, as described below. These codes are designed to be computer independent, and are presently operational on every type of computer from large mainframe computer to small personal computers, such as IBM-PC and Power MAC. The codes are available on CD-ROM or diskettes from the IAEA Nuclear Data Section, free of charge upon request or can be downloaded from http://www-nds.iaea.org/ndspub/endf/prepro/

Nuclear Data Section
International Atomic Energy Agency
P.O. Box 100
A-1400 Vienna
Austria

e-mail: services@iaeand.iaea.org
fax: (43-1) 26007
cable: INATOM VIENNA
telex: 1-12645
telephone: (43-1) 2600-21710

Online: TELNET or FTP: iaeand.iaea.org
username: IAEANDS for interactive Nuclear Data Information System
username: ANONYMOUS for FTP file transfer;
Web: http://www-nds.iaea.org

Note

The IAEA-NDS-reports should not be considered as formal publications. When a nuclear data library is sent out by the IAEA Nuclear Data Section, it will be accompanied by an IAEA-NDS-report which should give the data user all necessary documentation on contents, format and origin of the data library.

IAEA-NDS-reports are updated whenever there is additional information of relevance to the users of the data library.

Neither the originator of the data libraries nor the IAEA assume any liability for their correctness or for any damages resulting from their use.

Citation guidelines

For citations care should be taken that credit is given to the author of the data library and/or to the data center which issued the data library. The editor of the IAEA-NDS-report is usually not the author of the data library.

This computer code package should be cited as follows: D.E. Cullen, "PREPRO 2004: 2004 ENDF/B Pre-processing Codes", report IAEA-NDS-39, Rev. 12, Nov. 22, 2004

Nuclear Data Section Introduction

ENDF/B is the internationally agreed upon format for dissemination of evaluated nuclear data. The ENDF/B data library has now been through six (VI) versions; the latest identified as ENDF/B-VI. Documentation for the current ENDF format and convention is available in ENDF-102, from the National Nuclear Data Center, Brookhaven National Laboratory
http://www.nndc.bnl.gov/nndcscr/documents/endf/endf102/
or the Nuclear Data Section of the IAEA
http://www-nds.iaea.org/ndspub/endf/prepro/DOCUMENT/DOCUMENT.HTM

The 2004 ENDF/B Pre-processing codes process nuclear data formatted in any version of the ENDF/B formats; ENDF/B-I through ENDF/B-VI, and are even designed to handle new ENDF/B-VII evaluations. These codes can be used on virtually any computer: everything from large mainframe computers, to workstations, to IBM-PC and Power MAC.

These codes are available free of charge on CD_ROM or diskettes upon request from the Nuclear Data Section (see addresses on cover page) or downloaded from the Nuclear Data Section Web page
http://www-nds.iaea.org/ndspub/endf/prepro/

The present documentation (revision 12) completely supersedes all previous documentation of earlier versions of the ENDF Pre-processing data. **It is strongly recommended that you use ONLY the latest 2004 version of the PREPRO codes.** Failure to heed this warning can lead to completely erroneous results.

Conditions for use of the codes

Any comments on the use of the codes, including difficulties encountered or any suggestions should be sent to the IAEA Nuclear Data Section. If any results obtained from using these codes are used or referenced in a publication, a copy of the publication should be sent to the IAEA Nuclear Data Section.

Dedication

Regardless of whose name appears on the cover of this report, most of the work involved in maintaining, testing and distributing the previous and current versions of the PREPRO codes, was done by Kevin McLaughlin, Nuclear Data Section, IAEA, Vienna. For over 20 years Kevin has played an invaluable role in updating and testing the PREPRO codes. After all of these years I am sorry to have to report that Kevin has now retired. I think I can speak for all present and past members of the Nuclear Data Section in saying that Kevin will be greatly miss both as a coworker and as a good friend.

Acknowledgement

I gratefully acknowledge the contribution of Andrej Trkov in continuing to propose interesting and useful extensions to these codes; keep those great ideas coming Andrej. I also acknowledge Jennie Manneschmidt, RSICC, Oak Ridge, for her review of the documentation and testing of the codes prior to their distribution through RSICC.

Table of Contents

Features of 2004 Version

Features of All Versions

Implementing the Codes

Use of Codes

Details of Compiling and Loading Codes

Comments from Codes

History and Terminology

Originally the Evaluated Nuclear Data File (ENDF) was divided into two different **formats**: ENDF/A which was designed to contain partial evaluations that might later be incorporated into complete evaluations, and ENDF/B which was designed to contain complete evaluations for use in applications. Although the ENDF/A format is no longer used I feel it is important to distinguish the format that we now use, namely the **ENDF/B format**, by including the **/B** in its definition, and that is what I will consistently try to do throughout this document.

There is a difference between the **ENDF/B format**, and **the data** that is coded into this format. The ENDF/B format is now used universally to store evaluated nuclear data: in the ENDF/B library in the United States, in JEF in Western Europe, in JENDL in Japan, in CENDL in China, in BROND in Russia, etc. Here I will not be concerned with the differences between the contents of all of these data libraries. My only concern will be with the common **ENDF/B format** that all of these data libraries use. The PREPRO codes are designed to process evaluated data in any version of the ENDF/B format. The **ENDF/B format** has now been through six major versions, with the current format defined as ENDF/B-6; ENDF/B-VII is planned to be released on the end of calendar year 2005.

Some people feel that it is necessary to make a distinction between **ENDF/B formats** and **ENDF/B data** by using Roman numerals to describe one (ENDF/B-VI) and Arabic numerals to describe the other (ENDF/B-6). I'm sorry but I don't have too much sympathy for this usage. When we speak there is no such confusion and we make no such distinction, e.g., I've often heard people speak of ENDF/B-6 formats or data, but I've never heard anybody speak of ENDF/B "V" "I" formats or data. Therefore in an attempt to stick to common sense and to use the same rules for the spoken and written word, I will consistently use Arabic numbers to refer to ENDF/B formats or data, e.g., ENDF/B-6, and I will try to always make the distinction of whether I am discussing **ENDF/B formats** or **ENDF/B data**.

Features of 2004 Version

Compared to earlier versions of these codes the 2004 version has the following features,

Running Time

It wasn't too many years ago that in order to process major ENDF/B evaluations we needed super, million dollar computers, and even then it could take days to process a large evaluation, such as U-238.

Need I say it: those days are gone forever. Today even small personal computers can be used to quickly process any ENDF evaluations. For example, on my IBM-PC, Pentium IV, 3.6 GHz computer [not a million dollar computer, a $ 1,000 computer (year 2004)], I can process U-238 in the time it takes me to go and get a cup of coffee - and with the next generation, it will require even less time.

So I am not going to list typical running times for the codes, for two reasons: 1) the running times have now become trivial, and 2) by the time you get a copy of this report any times I quote here will be outdated by the availability of newer, faster, and cheaper computers.

Bottom line: running time is no longer a major concern in processing ENDF/B data, and even small personal computers are now powerful enough to be used to process all ENDF/B evaluations.

All ENDF/B Formats and Procedures

These codes can automatically determine the ENDF/B format version that each evaluation is coded in and use the appropriate procedures. It should be particularly noted, that these codes now handle all ENDF/B formats and procedures through ENDF/B-6, Release 8, and they are even designed to handle newer ENDF/B-VII evaluations, including all new ENDF/B formats and conventions adopted as of the November 2004 CSEWG meeting.

WARNING: The 2004 codes include extensions to handle all current ENDF/B formats and procedures, and corrections to problems that existed in earlier versions of these codes. As such the 2004 codes **completely supersede all earlier versions and it is strongly recommended that all users of these codes only use the 2004 version of these codes**. Failure to heed this warning can lead to completely erroneous results.

Consistent Handling of All ENDF/B Formatted Data

All of the codes now use exactly the same routines to handle all ENDF/B formatted input and output. This has resulted in a completely consistent interpretation of all ENDF/B formatted data by all of the codes, and has also allowed the precision of the ENDF/B output to be consistently extended in all codes. For 2004 the ENDF/B output is completely consistent for input into C and C++ codes, while still maintaining the accuracy of the data.

Optional Input Parameters

All of the codes now allow input parameter files and ALL input parameters to be optional; all input parameters now have built-in default values. Of particular note is that allowable uncertainties are now optional input. This allows us to select what we consider the best choices, based on the most recent advances in the speed and size of computer.

Computer Independence

The only computer dependence in the 2004 codes is to define running time. Routines to define running time are supplied for most types of computers, and instructions are provided in this report to help you define a timing routine for any other type of computer.

Bigger, Faster, Improved Accuracy

In line with the enormous increase in computer sizes during the last few years, the 2004 versions are **bigger**, allowing more complicated problems to be run much more efficiently, and in general allowing each problem to be run much **faster**.

All of the codes now use double precision throughout, resulting in **improved accuracy**. Compared to the earlier versions that used a mixture of single and double precision, with modern compilers and hardware, using double precision throughout has also contributed to making the codes **faster**.

On-Line Reports

All of the codes now include an on-line report to your screen, and a report to an output file; the on-line report allows users to monitor the progress of each code as it executes. Earlier versions had no on-line report; as far as what the user saw, the code started and ran to conclusion without printing anything on-line. This made it impossible to monitor the progress of each code, and for long running problems often resulted in users terminating the codes before they completed execution, because it appeared that the codes weren't doing anything.

Execution Timing

The codes now include a timer, to print execution time at the end of processing each evaluation (MAT), and at the end of execution.

Features of All Versions

Code Documentation

These codes are designed to be self-documenting, in the sense that the latest documentation for each code is included as comments at the beginning of each code. Printed documentation, such as this report, is periodically published and consists mostly of a copy of the comment lines from the beginning of each code.

The user should be aware that the comment lines within the codes are continually updated to reflect the most recent status of the codes and these comments within the codes should always be considered to be the most recent documentation for the codes and may supersede published documentation, such as this document. Therefore users are advised to always read the documentation within the actual code that is being used.

Data Documentation

It is essential that the pedigree of the evaluated data be documented. This is the purpose of the comment lines at the beginning of each ENDF/B evaluation. These codes are designed to document any operations that they perform on ENDF/B data. If one of these codes produces ENDF/B formatted output which in any way effects the actual evaluated data, what the code did is documented by adding additional comment lines at the end of the comment lines at the beginning of each evaluation, defining the code and input parameters that it used. The sequence of all such

comments completely documents all of the operations that have been performed on the data. Code users are advised that it is very important to leave this documentation directly inside each evaluation, i.e., please do not modify these codes or the evaluations to remove this documentation.

Obtaining the Codes

These codes are available free of charge on CD ROM or diskettes upon request from the Nuclear Data Section (see addresses on cover page) or downloaded from the Nuclear Data Section Web page

http://www-nds.iaea.org/ndspub/endf/prepro/

Your Feedback is IMPORTANT!!!

We are trying to develop a set of codes that are as computer independent as possible. In this effort your feedback is IMPORTANT!!! It is impossible for us to test these codes on all available computer/compiler combinations. Therefore your experience, on your specific computer/compiler can help us to improve the computer independence of these codes. It is also in your best interest to share your experience with us, since it will insure that future versions of these codes are as compatible as possible to meet your needs.

Please send all feedback via e. mail at,

mailto:services@iaeand.iaea.org

Implementing the Codes

What Computers do the codes run on?

The codes are designed to run on virtually any computer. The exceptions to this rule are the interactive graphics codes **complot** and **evalplot**, which are designed to produce on-screen graphics on UNIX workstations, IBM-PC, PowerMAC and VMS, i.e., not mainframe computers. However, even these codes can be used in their non-interactive mode, named **comhard** and **evalhard** (note the names to indicate **hard**copy output), to produce Postscript formatted files that can be printed on any Postscript printer.

For use on IBM-PC running Windows or Linux, and on PowerMAC, the distribution includes executables, ready to use immediately. For use on a variety of UNIX based computers, the distribution includes a batch file for each type of computer, to compile and load all programs, and to then clean up by deleting everything not required to execute the programs. For other types of computers, see the section below on, Details of Compiling and Loading Codes

The Most Up-to-Date Installation Instructions

The most up-to-date installation instructions, documentation, and the codes, can be downloaded from the website,

http://www-nds.iaea.org/ndspub/endf/prepro/

Read the text and then select “Download Codes” or “Download Documentation”

They may also be accessed via FTP as follows:

Open FTP access to iaeand.iaea.org, username ANONYMOUS, password: your e-mail address, change directory as “CD ENDF/PREPRO” and get files for your platform.

Remember to ftp and read the “README” file.

We try to maintain these installation instructions as up-to-date as possible, based on user feedback. So if you have any problems or suggestions regarding installation please e.mail them to the Nuclear Data Section at,

mailto:services@iaeand.iaea.org

Register as a User

We try to maintain these codes and data as up-to-date as possible. So if you are using any of these codes it is important that you tell us about this, so that the Nuclear Data Section can put your name on the distribution list to inform you about the latest updates. This is a FREE!!! service which is provided to users of these codes. We have tried to make this as easy as possible for you - PLEASE take a moment to e.mail to mailto:services@iaeand.iaea.org, and tell what codes you are using, and what type of computer(s) you are using - it’s as simple as that.

Use of Codes

Read the Output Reports

MOST IMPORTANT! You cannot use these codes like a black box and assume that everything is perfect. Don't make the mistake of assuming that all ENDF/B data is perfect, or that these codes are perfect. It's up to you, the code user, to check and be sure that the data output by these codes is accurate and can be used in applications. If you don't, you are wasting your time, and will produce inaccurate results in your applications.

You can do this by reading the output reports produced by each code. These output reports will generally be quite small. They are intended to be used by you to quickly scan through them and look for WARNING or ERROR messages - these indicate problems with the ENDF/B data that you should check before using the data in any applications. You need not read each output report in detail; it is sufficient to merely search for the words WARNING or ERROR – these will always accompany important messages.

Checking these output reports doesn't take very much time, but failing to check them can cause you to waste an awful lot of time and can cause you headaches later, if you try to use data that a code has clearly indicated to be bad. If there are errors in the ENDF/B data, you are clearly in a “garbage in, garbage out” situation as far as the result you calculate in your applications. **Caveat Emptor!**

Standard and Variable Filenames

Currently all input files and input parameters are optional, and have built-in default values.

All of the codes have standard, built-in, filenames, that they will use by default, unless input parameters explicitly define other filenames.

The default filenames have been defined to make it easy for you to remember, and to be compatible with as many operating systems as possible, e.g., DOS, that only allows short filenames, and Unix, that allows longer filenames. The default filenames are all of the form NAME.EXT, where NAME identifies a program name, and EXT identifies the type of file. All default filenames use **ONLY** upper case characters. The basic filenames include,

1) **???.INP** - The **INP**ut parameters for each code, where ??? is the name of the code. For example, the input parameters for **RECENT** are in a file named **RECENT.INP**. This name cannot be changed by input. Currently these input files are optional; if they are not present default values are used for all input parameters.

2) **???.LST** - The output **L**i**ST**ing from each code, where ??? is the name of the code. For example, the output listing from **RECENT** is in a file named **RECENT.LST**. This name cannot be changed by input.

3) **???.IN** - ENDF/B formatted data to be read (**IN**put) be each code, where ??? is the name of the code. For example, the ENDF/B data read by **RECENT** are in a file named **RECENT.IN**. This name can be changed by input.

4) **???.OUT** - ENDF/B formatted data written (**OUT**put) be each code, where ??? is the name of the code. For example, the ENDF/B data written by **RECENT** are in a file named **RECENT.OUT**. This name can be changed by input.

The above simple filename conventions will allow you to easily remember for each code, where the input parameters and output report are located, as well as where the ENDF/B data that is read and written by the code are located.

By input you can change the filenames of the ENDF/B formatted data files; data read and/or written.

If you input blank filenames the codes will use the default names (described above).

If you input anything else, the code will use the filenames you have defined. Variable filenames for each code can be up to 60 characters long. This allows you to specify directory structures, so that you can store your ENDF/B data in some rational way within a directory file structure.

For example if you store all of the ENDF/B-6 data in a directory named ENDFB6, the following input filename used with **linear** will read a file named za092238 on an IBM-PC,

\ENDFB6\ORIGINAL\za092238

or on a Unix workstation,

/ENDFB6/ORIGINAL/za092238

Warning - generally on Unix workstations you will have to include the complete path to files. For example, the path to my files on my workstation may be /home/pd11/cullen, in which case my filename should be,

/home/pd11/cullen/ENDFB6/ORIGINAL/za092238

The ability to directly reference file structures is a very powerful facility that you should not overlook in organizing your ENDF/B data.

Brief Description

Linear - Linearize cross sections
Recent - Reconstruct cross sections from resonance parameters
Sigma1 - Doppler broaden cross sections
Activate - Generate activation cross sections (MF=10) from MF=3 and 9 data
Legend - Calculate/correct angular distributions
Sixpak - Convert double differential data (MF=6) to single differential
Fixup - Correct format and cross sections, define cross sections by summation
Dictin - Create reaction dictionary (MF=1, MT=451)
Merger - Retrieve and/or Merge evaluated data
Groupie - Calculate group averages and multi-band parameters
Complot - Plot comparisons of cross sections (MF=3, 23); Comhard for hardcopy
Evalplot - Plot evaluated data (MF=3, 4, 5, 23, 27); Evalhard for hardcopy
Mixer - Calculate mixtures of cross sections
Virgin - Calculated transmitted uncollided (virgin) flux and reactions
Convert - Convert codes for computer/precision/compiler
Relabel - Relabel and sequence programs

Detailed Description

The codes can be used to: 1) extensively check and correct evaluated data prior to using them in applications, 2) pre-process the data into a form that will make subsequent use of the data much easier.

The normal sequence in which the codes are used is described below. **WARNING** - this is the recommended sequence of codes that you should run to produce **LEGAL** ENDF/B data, that conforms to **ALL** ENDF/B formats and conventions. Note in particular that if you do not run **FIXUP** and **DICTIN** at the end of this sequence the resulting ENDF/B data **WILL NOT** conform to all ENDF/B formats and conventions, and may cause problem if you subsequently try to use the data.

1) **LINEAR** - Linearize cross sections. ENDF/B allows cross sections to be represented as tables of data points using a number of different interpolation laws between tabulated points; in order to obtain accurate results it is important to interpret the data using these interpolation laws. The interpolation laws are very useful during evaluation, but can present problems when they are used in applications. The subsequent use of the data can be greatly simplified and the accuracy of results improved by first linearizing all of the cross sections, i.e., replace the original tabulated data points and interpolation law by a new table where one can use linearly interpolation between tabulated points to within any required accuracy.

2) **RECENT** - Add the contribution of resonances to the cross sections. ENDF/B allows cross sections to be represented as a contribution of resonance parameters and tabulated background corrections. This code will add the resonance contribution to the background cross sections in order to define the cross sections as linearly interpolable tables at 0 Kelvin (cold). Therefore subsequent codes need only deal with tabulated, linearly interpolable, 0 Kelvin cross sections.

3) **SIGMA1** - Doppler broaden cross sections to any temperature of interest in applications. As in the case of LINEAR and RECENT all cross sections read and written by this code are tabulated, linearly interpolable. All subsequent codes need not explicitly consider temperature effects and need only deal with tabulated, linearly interpolable cross sections at a given temperature.

4) **ACTIVATE** – Combine neutron interaction cross sections (MF=3) and multipliers (MF=9) to create activation cross sections (MF=10). LINEAR and GROUPIE have been updated to process multipliers (MF=9) and activation cross sections (MF=10). The sequence of codes LINEAR, ACTIVATE, and GROUPIE allow you to produce group averaged activation cross sections.

5) **LEGEND** - Convert tabulated distributions and Legendre coefficients to linearly interpolable tables (similar to what **LINEAR** does for cross sections). Check all angular distributions and Legendre coefficients, in particular check for negative angular distributions and if found, correct the distributions to make them positive. Note, negative angular distributions can lead to numerical instabilities and unreliable results if they are used in applications.

6) **SIXPAK** - ENDF/B-6 introduced double differential data (MF=6) into the ENDF/B system for the first time. If your application codes have not yet been updated to handle double differential data, you can use **SIXPAK** to obtain single differential (MF=4 and 5) approximations to double differential data. Earlier versions of SIXPAK only output results for outgoing (emitted) neutrons and

photons, however currently SIXPAK will output angular distributions for discrete charged particle levels.

7) **FIXUP** - Define all cross sections to be consistently exactly equal to the sum of their parts, make format corrections, and a number of other tests and corrections to the data, BEFORE the data is actually used in applications. It is extremely important for use in applications to guarantee that the cross sections are exactly consistent. For example, the total cross section MUST to defined as equal to the sum of its parts at all energies that appear in one or more of the contributing parts. In addition it should be mentioned that the total will be equal to the sum of its parts at all energies (not just the energies at which the total is tabulated), only if all of the cross sections are linearly interpolable; this illustrates the importance of the steps described above in processing data through each of these codes. Note, if FIXUP's option to output all cross sections on a uniform energy grid is used, the FIXUP output is compatible for use as **NJOY** input.

8) **DICTIN** - Update the section index in MF=1, MT=451. This step need only be run if the subsequent codes that use the data refer to this index. If you are unsure whether or not this is the case, it is always best to include this step, since relative to the other codes described above this step requires very little running time.

After this sequence of codes has been run the results will be evaluated data that has been carefully checked for consistency and has been reduced to a form that can be used more easily and reliably in subsequent applications.

In addition to the codes mentioned above, this package includes a number of useful utility codes including,

1) **MERGER** - Retrieve and/or combine evaluated data. This code can be used to create a single file of data in the ENDF/B format from a number of different files, each of which is in the ENDF/B format. It can also be used to retrieve specific evaluated data from a larger ENDF/B library in order to simplify and optimize the subsequent use of the data in applications, e.g., if you have an entire ENDF/B library, but will only be using five evaluations for your applications, you can first use this code to create a mini-library containing only the five evaluations that you need for your application.

2) **GROUPIE** - Calculates self-shielded, multigroup cross sections and multiband parameters. This code can be used as a simple and very economical means of obtaining multigroup cross sections, in the ENDF/B format, which can be used in many applications where only multigroup cross sections are required, e.g., dosimetry. For comparing data using **COMPLOT** this code can be used to reduce evaluations that have many resonances, to a form in which integral differences through the resonance region can be more easily seen.

3) **COMPLOT** – Plot a comparison of cross sections from two different evaluations. This code can be used to compare cross sections, for each reaction, to define exactly how two evaluations differ. This can be extremely important if one has already used a given evaluation in applications and wishes to quickly and inexpensively determine whether or not a newer evaluation can be expected to produce significantly different

results when used in applications. It is also an excellent and simple means of documenting the differences between two evaluations, e.g., what's the difference between the ENDF/B-6, Release 4 and 5, U-235 cross sections? See the above comments under **GROUPIE** for suggestions concerning comparing evaluations that have many resonances. This code can be used as a simple means of visually checking all of these cross section data types and is often very useful to help understand the results obtained when data is used in applications. In addition, the graphic Postscript output can serve as a part of the documentation for evaluations. Two versions of exactly the same code are provided: **complot** to produce on-screen graphics, and **comhard** to produce Postscript, hardcopy, output.

4) **EVALPLOT** - Plot cross sections, angular distributions, Legendre coefficients and/or energy distributions, for neutron interaction data, neutron induced photon production data, and photon interaction data. This code can be used as a simple means of visually checking all of these data types and is often very useful to help understand the results obtained when data is used in applications. In addition, the graphic Postscript output can serve as a part of the documentation for evaluations. Two versions of exactly the same code are provided: **evalplot** to produce on-screen graphics, and **evalhard** to produce Postscript, hardcopy, output.

5) **MIXER** - Can be used to define the cross sections for a combination of materials, e.g., stainless steel. This code can be used in combination with **COMPLOT** to see which energy ranges are important for each material and each constituent of a material. This code can also be used to define the correct total cross section for use in transmission calculations (see, **VIRGIN**), as well as in self-shielding calculations (see, **GROUPIE**), in order to avoid the approximations normally incoherent in the Bonderenko method of self-shielding. Since ENDF/B-6 has moved in the direction of representing separate isotopes for each element, this code is particularly useful if your applications only requires a natural mixture of isotopes, e.g., use MIXER to combine isotopes into the natural element.

6) **VIRGIN** - Can be used to perform exact uncollided (virgin) transmission calculations (exact, assuming the tabulated, linearly interpolable cross sections are exact - no other approximations are used). By using the data that has been prepared by a combination of **LINEAR, RECENT, SIGMA1, MIXER,** etc., this code can be used to simulate transmission through any given material, or layers of different materials, at any given temperature. The results include both transmitted flux and reaction rates (as measured in self-indication measurements) vs. material thickness. The results can be obtained either on a continuous energy basis, or they can be binned (energy integrated) to simulate any given experimental resolution.

In addition there are two utility codes that operate on the codes, rather than on ENDF/B data.

1) **RELABEL** - Is a file maintenance code used to maintain all of the codes in this package. This code will normally not be used by users, unless they plan to modify these codes.

2) **CONVERT** - Format and optimize codes for use at any given computer installation. This code is no longer required by the PREPRO, since the codes are

now completely computer independent. It is still included in this package only because users have found it useful for other purposes. Generally this code was used only once to format all of the codes prior to their first use on any computer.

Verifying Implementation

This distribution comes with a file named VERIFY, which is designed to run all of the codes, one after another, with the final two steps being to run EVALPLOT and COMPLOT, so that you can see the final results. VERIFY is a simple text file; its contents are shown below,

```
linear
recent
sigma1
activate
legend
fixup
dictin
groupie
mixer
virgin
evalplot
complot
```

When executed as a batch file, this will run the codes in the order indicated. The distributed input parameters have been defined so that each code reads the ENDF/B data file produced by the preceding code, and writes the ENDF/B data file that will be read by the following code.

To verify implementation immediately after you have installed the codes, DO NOT change any input parameters for ANY codes, and execute VERIFY. It will take between 5 minutes and an hour (depending on the speed of your computer), to run all of the codes. When you get to the final two graphics codes, EVALPLOT and COMPLOT, you can be assured that all of the codes have run successfully.

COMPLOT will compare the cross sections calculated by you on your computer to a standard set of results distributed with PREPRO 2004. In both cases cross sections are calculated by each code to within an accuracy of 1 %. Therefore when COMPLOT compares the results you may find differences of about 1 %. This difference is o.k., and merely indicates the differences due to precision to which the cross sections have been calculated. Subsequently, for use in your applications you can feel free to modify the input parameters for each code to meet the precision that you require.

WARNING – for UNIX users - some UNIX systems now include diction as a system command. In order to avoid this conflict, in PREPRO 2004 the code previously named diction has been renamed dictin.

Use of the Codes in Combination

Almost any computer will allow you to submit a batch job, in which case you can perform any number of operations one after the other, as is done in the above verification. These computers can utilize this facility to run any number of these codes in combination, minimize the total amount of disk space used, and most important, optimize the use of your time.

In order to run any number of codes one after the other, all you need is the facility to: 1) start a program, 2) rename a file, 3) delete a file, if you want to minimize disk space.

For example, if I want to run the sequence of codes, **LINEAR, RECENT, SIGMA1, ACTIVATE, LEGEND, FIXUP** and **DICTIN** and only keep the original data read by **LINEAR** and the final results output by **DICTIN**, I can use the standard ENDF/B filenames for the data read and written by each code, and submit the following batch file on an IBM-PC,

```
linear
rename LINEAR.OUT RECENT.IN
recent
delete RECENT.IN
rename RECENT.OUT SIGMA1.IN
sigma1
delete SIGMA1.IN
rename SIGMA1.OUT ACTIVTE.IN
activate
delete ACTIVATE.IN
rename ACTIVATE.OUT LEGEND.IN
legend
delete LEGEND.IN
rename LEGEND.OUT FIXUP.IN
fixup
delete FIXUP.IN
rename FIXUP.OUT DICTIN.IN
dictin
delete DICTIN.IN
```

Note, when each code finishes the above batch deck renames the ENDF/B data output by the code to the filename of the ENDF/B data input to the next code. When the next code finishes, the ENDF/B data input to it is deleted (we no longer need it), and the cycle starts for the next code. More efficiently you could have defined ENDF/B input and output file names in the input parameter files for each code to link them together, e.g., instead of copying LINEAR.OUT to RECENT.IN, you could have defined the input file to RECENT to be named LINEAR.OUT.

The result will be the original data read by **LINEAR** is still in the file named **LINEAR.IN**, and the final result is in the file named **DICTIN.OUT**. All other intermediate files have been deleted.

On any other system, such as Unix, the names **delete** and **rename** may be different, but the basic idea remains the same.

An alternative to the above approach is to use the facility of the codes to read and write files from any file structure. For example, assume I have a directory named ENDFB6, and within this directory I have three sub-directories: ORIGINAL, TMP, and K300 (data Doppler broadened to 300 Kelvin). What I can do is define **LINEAR** input parameters to read a file from ENDFB6/ORIGINAL, define input parameters to **RECENT, SIGMA1, ACTIVATE, LEGEND** and **FIXUP** to produce ENDF/B output in ENDFB6/TMP, and have each code read the output from the preceding code. Finally I can define **DICTIN** input parameters to write the ENDF/B output into ENDFB6/K300, with its final filename. In this case if I do not worry about deleting the intermediate files, the batch input need only be the names of the codes to run, i.e.,

linear
recent
sigma1
activate
legend
fixup
dictin

Using a batch approach can save you a great deal of your precious time. You don't have to sit there and babysit your terminal in order to start each code as the preceding one finishes. You can use batch jobs to combine code executions, and go off to work (or play) until the sequence of codes finishes. If you then want to be sure that everything ran correctly, you can read the output reports from each code, i.e., see the **???.LST** from each code, e.g., for RECENT see RECENT.LST – **it is HIGHLY Recommended that you always read these files.**

Details of Compiling and Loading Codes

For use on IBM-PC running Windows or Linux, and on PowerMAC, the distribution includes executables, ready to immediately use. For use on a variety of UNIX based computers, the distribution includes a batch file for each type of computer, to compile and load all programs, and to then clean up by deleting everything not required to execute the programs. Only for other types of computers need you be concerned with the details concerning compiling and loading the codes, which are described here.

Parts of the Codes

The codes have now been divided into a number of parts that should be combined when compiling and loading; see, example compile/load instructions below. The parts are,

1) The basic code

2) An include file to define code storage

3) Routines to allow all codes to now uniformly treat all ENDF/B formatted input and output (**endfio.f**)

4) Routines to allow scratch files to be defined either with or without file names,
scratcha.f = with file name
scratchb.f = without file name

Most compilers/computers allow scratch files to be defined without scratch file names, so use either **scratcha.f** or **scratchb.f**. However, some compilers/system combinations get confused when there are multiple scratch files without file names, e.g., Lahey on IBM-PC (use **scratcha.f**), and some compilers do not allow scratch files with file names, e.g., ABSOFT on IBM-PC and Power MAC (use **scratchb.f**).

4) A timer, to define the execution time for each code. The standard timer routine (**timer.f**) distributed with the codes uses the standard Unix routine **ETIME**; on some computers you will have to consult the on-line manual to see how to link to **ETIME**, e.g., HP.

If you are not using a Unix based computer, you will have to supply your own timing routine. It is recommended that you use the distributed version of **timer.f**, and add a function **ETIME**, that defines the execution time on your computer - see, the timing routines included for a variety of UNIX computers

If you do define a non-standard timer, try to define EXECUTION - NOT WALL CLOCK time - on some computers this isn't possible, e.g., IBM-PC running DOS - in which case use whatever you can.

If you can't figure out how to define running time, or you don't want the codes to print running time, instead of using the distributed **timer.f**, define and use the following dummy routine,

```
SUBROUTINE TIMER
RETURN
END
```

If you do define a non-standard timer, PLEASE send us a copy, identifying what computer/compiler you are using - over a period of time we intend to build up a library of timer routines for as many different computers as possible - which we will then distribute with the codes = future versions will be more compatible to meet your needs.

5) A graphics interface, for complot and evalplot.

Compiling/Loading

This section applies to all of the codes, except the graphics codes, **complot** and **evalplot**; see, below under graphics codes. Below is an example of how to compile/load the codes on a Unix based computer. For this example I illustrate how to create executables on a SUN workstation; timing routines are provided for most types of computers. Note,

1) No special libraries are used by these codes, so that compile/load instructions are very simple.
2) How the pieces are combined.
3) Use the HIGHEST LEVEL OPTIMIZATION available on your computer - this can make a BIG difference in running time.
4) SUN.f is the timing routine to use on a SUN workstation. Similar timing routines are provided for most types of computers.

```
f77 -o linear  -O linear.f  endfio.f scratchb.f timer.f SUN.f
f77 -o recent  -O recent.f  endfio.f scratchb.f timer.f SUN.f
f77 -o sigma1  -O sigma1.f  endfio.f scratchb.f timer.f SUN.f
f77 -o fixup   -O fixup.f   endfio.f scratchb.f timer.f SUN.f
f77 -o legend  -O legend.f  endfio.f scratchb.f timer.f SUN.f
f77 -o sixpak  -O sixpak.f  endfio.f scratchb.f timer.f SUN.f
f77 -o mixer   -O mixer.f   endfio.f scratchb.f timer.f SUN.f
f77 -o merger  -O merger.f  endfio.f scratchb.f timer.f SUN.f
f77 -o dictin  -O dictin.f  endfio.f scratchb.f timer.f SUN.f
f77 -o virgin  -O virgin.f  endfio.f scratchb.f timer.f SUN.f
f77 -o groupie -O groupie.f endfio.f scratchb.f timer.f SUN.f
f77 -o relabel -O relabel.f
f77 -o convert -O convert.f
```

Graphics Codes

The graphics codes - **complot** and **evalplot** - can be used to produce either,

1) Postscript output files for printed hardcopy, using executables named **comhard** and **evalhard**.
2) On screen graphics, using executables named **complot** and **evalplot**.

The two executables, **complot** and **comhard**, are exactly the same code, loaded with different graphics interfaces; both executables use the same input and output files, **COMPLOT.INP** and **COMPLOT.LST**. Similarly, the two executables, **evalplot** and **evalhard**, are exactly the same code, loaded with different graphics interfaces; both executables use the same input and output files, **EVALPLOT.INP** and **EVALPLOT.LST**.

Postscript Output Files

The Postscript graphics interface should be completely computer independent, and as such should run on any computer.

It will create a series of output files - none of which are sent to your printer during execution of the code.

Output for each plot is saved on disk, so when the code ends all of the plot files will still be on disk, and you can then send them to your printer, and/or, save them for later use.

WARNING - the codes always use the same file names, **PLOT0001.ps, PLOT0002.ps**, etc. So that running a code again will overwrite any files that you previously created. If you want to save files, moved them or rename them before running a code again.

To use this method to create these Postscript files use **hardsave.f** with the codes.

For Postscript graphics, no special libraries are used, and an example of how to compile/load the codes on a Unix based computer is shown below - this is very similar to the compile instructions shown above, with the addition of hardsave.f,

```
f77 -o comhard  -O complot.f  endfio.f scratchb.f timer.f hardsave.f
SUN.f
f77 -o evalhard -O evalplot.f endfio.f scratchb.f timer.f hardsave.f
SUN.f
```

Note, that here the executables are given the names for the hardcopy versions of the codes, **comhard** and **evalhard**.

On Screen Graphics

For on screen graphics the codes are loaded with **screen.f**, in contrast to the hardcopy version of the codes, described above, for Postscript graphics that are loaded with **hardsave.f**.

Example Makefiles are included for a variety of Unix systems.

On screen graphics is VERY computer dependent, so on Unix computers you may have to modify the Unix Makefile - this should only involve finding out where the X11 graphics library is on your computer, and setting the correct path in the Makefile.

If you do have to modify the Makefile, please send me a copy of the modified file, identifying your computer, so that we can build up a library of Makefiles to be distributed with the codes then future versions will be compatible with your needs.

The codes are distributed with graphics interfaces for,

1) Unix and openVMS systems, using the X11 graphics library (**screen.f**, **nodash.c, dash.c**)

2) If you are using any other system, you will have to supply your own graphics interface - see, **screen.f** for a description of the simple interface used by these codes.

Interacting with Graphics

When you are using **evalplot** there is no true on-screen interaction with the plots. If you wish to view different data over different energy ranges your only option is to change your input parameters in the file **EVALPLOT.INP**.

When you are using **complot** you can interact with the on-screen plots. Once a plot is displayed on your screen if you would like to see a portion of the energy range of

the plot in greater detail, you can do this by using your mouse to zoom in by indicating the lower and upper energy limits of the energy range you would like to see. As soon as you select the range the next zoomed plot will appear on your screen, with the same data as on the previous plot, but only over the energy range that you have selected. WARNING – **complot** only generates plots when the two evaluations differ by more than the allowable uncertainty you define by input in the file **COMPLOT.INP**. This also applies when you interact with the plots. Therefore, if you use your mouse to select an energy range over which the two evaluations do not differ by more than your allowable uncertainty a zoomed plot will not be produced, but the results of the comparison will be reported in the output file COMPLOT.LST, and **complot** will proceed to its next comparison.

Comments from Codes

These codes are designed to be self-documenting, in the sense that the most up-to-date documentation is included as comments at the beginning of each code. Periodically documentation, such as this report, is published. But the user is warned that the comments in the codes are continuously updated and it is these comments within the codes that should be considered to be the most up-to-date documentation, and the user should read these comments before, and while, using these codes.

The following section contains a listing of the comments from the codes as of the publication date of this report. The comments are listed for each code alphabetically according to the name of the code, including,

ACTIVATE
CONVERT
COMPLOT
DICTIN
EVALPLOT
FIXUP
GROUPIE
LEGEND
LINEAR
MERGER
MIXER
RECENT
RELABEL
SIGMA1
SIXPAK
VIRGIN