JUNE 1991

ECN RX--91-063

NL92C0291

ECN

energy innovation

SIMPLIFIED MODELING AND CODE USAGE IN THE PASC-3 CODE SYSTEM BY THE INTRODUCTION OF A PROGRAMMING ENVIRONMENT

B.J. PIJLGROMS
J. OPPE
H.L. OUDSHOORN
J. SLOBBEN

The Netherlands Energy Research Foundation ECN is the leading institute in the Netherlands for energy research. ECN carries out pure and applied research in the fields of nuclear energy, fossil fuels renewable energy sources, environmental aspects of energy supply, computer science and the development and application of new materials. Energy studies are also a part of the research programme.

ECN employs more than 900 staff. Contracts are obtained from the government and from national and foreign organizations and industries.

ECN's research results are published in a number of report series, each series serving a different public from contractors to the international scientific world.

This RX-series is used for publishing pre-prints: the preliminary versions of articles that will appear in a journal, or conference or symposium proceedings. Please do not refer to this report but use the reference provided on the title page: 'To appear in ...' or 'Submitted for publication to ...'.

The Netherlands Energy Research Foundation ECN
Service Unit General Services
P.O. Box 1
NL-1755 ZG Petten
The Netherlands
Telephone: +31 2246 43 23
Fax : +31 2246 34 83

This report is available on remittance of Dfl. 20 to:
ECN, SU General Services, Petten,
The Netherlands.
Giro account (postal account) No. 3977703.
Please quote the report number.

JUNE 1991 ECN-RX--91-06

SIMPLIFIED MODELING AND CODE USAGE IN THE PASC-3 CODE SYSTEM BY THE INTRODUCTION OF A PROGRAMMING ENVIRONMENT

B.J. PIJLGROMS
J. OPPE
H.L. OUDSHOORN
J. SLOBBEN

CONTRIBUTION TO
'INTERNATIONAL CONFERENCE ON NUCLEAR CRITICALITY SAFETY - ICNC '91'
OXFORD, UK, 9 - 13 SEPTEMBER 1991

Simplified modeling and code usage in the PASC-3 code system by the introduction of a programming environment

B.J.Pijlgroms, J.Oppe, H.L.Oudshoorn, J.C.Jobben

Netherlands Energy Research Foundation ECN,
P.O. Box 1, 1755 ZG Petten,
The Netherlands

ABSTRACT

A brief description is given of the PASC-3 (Petten-AMPX-SCALE) Reactor Physics code system and its associated UNIPASC work environment. The PASC-3 code system is used for criticality and reactor calculations and consists of a selection from the Oak Ridge National Laboratory AMPX-SCALE-3 code collection complemented with a number of additional codes and nuclear data bases. The original codes have been adapted to run under the UNIX operating system. The recommended nuclear data base is a complete 219 group cross section library derived from JEF-1 of which some benchmark results are presented.
By the addition of the UNIPASC work environment the usage of the code system is greatly simplified. Complex chains of programs can easily be coupled together to form a single job. In addition, the model parameters can be represented by variables instead of literal values which enhances the readability and may improve the integrity of the code inputs.

1. Introduction

The PASC-3 system (Petten - AMPX - SCALE) is a large system of Fortran programs and nuclear data bases for Reactor Physics Computations. Most of the codes originate from the Oak Ridge National Laboratory (ORNL) AMPX-SCALE package, ref.[1], and were adapted at ECN-Petten to run under the UNIX system on a network of SUN workstations and on a CONVEX-C220 supercomputer. Earlier versions of the PASC-system with a partially different content, are PASC-1 for the Cyber-NOS/BE system and PASC-2 for the Cyber-NOS/VE system.

The relation between AMPX and SCALE is the following: Most of the functional Fortran codes originally belonged to the modular AMPX code system. A complete reactor calculation usually requires the execution of a whole sequence of AMPX modules. The required (ascii coded) inputs for these modules consist of a detailed description of the problem to be solved and are presented in the form of arrays of literals. Especially the modeling of large systems is therefore tedious and the tracing of input errors is cumbersome. The advantage of this "low level" description is that it allows the maximum flexibility within the capabilities of the codes.

To improve the user-friendliness of the AMPX system, the SCALE (Standardized Computer Analysis for Licensing Evaluation) system, ref.[2], was developed. Besides improved versions of the AMPX modules, SCALE contains a number of input-processors (Fortran modules) which generate from a compact and user-friendly model description the input files for a sequence of functional modules which we will call data-processors. Besides that, SCALE contains a driver which controls the job flow of the sequence. This operating system dependent driver obtains some control signals from the input-processors and the data-processors. Various SCALE sequences are available to facilitate reactor analyses.

The SCALE system is certainly an important improvement with respect to the AMPX system, but the way in which the input-processors were implemented has also some disadvantages:

- The flexibility is reduced to a minimum; often this can be considered as an advantage when calculating standard problems but for special studies deviating somewhat from the standard procedure it would require the introduction of a new input-processor.
- The original SCALE input-processors generate the input files in binary form, which has some disadvantages for the user and for the code manager: the generated inputs are unreadable and for each functional module a duplicate version has to be created which accepts binary instead of ascii coded inputs.

The PASC-3 implementation of the AMPX-SCALE system is intended to enhance the portability of the codes at the Fortran level and to provide drivers and facilities (the "UNIPASC" environment) for UNIX systems. The SCALE input-processors are modified to produce ascii coded input files. Therefore, no extra set of functional modules is needed and the generated inputs are readable by the user and by "filters". The filters can be used to intercept and modify the job inputs which are produced by the input-processors. As the filters are programmable stream-mode text editors, this procedure can be automated and makes it possible to regain the complete flexibility of the underlying AMPX system.

The UNIPASC environment contains UNIX drivers to control the SCALE input-processors and/or the AMPX functional modules and contains a number of commands for running and maintaining the system and some filter facilities. UNIPASC allows the user to combine job control and job inputs into a single file and to use C-shell variables in the inputs. As these variables can be given meaningful names, the job inputs can be made readable as a novel making the jobs self-descriptive. Moreover, as the whole C-shell programming language is available, the user can create complex job flows.

2. Overview of the PASC-3 code system

2.1. The layer structure of the software

The code system can be subdivided in different ways. In figure 1 we show a layer diagram of the system in which we can identify from bottom to top the following layers:

- The machine level. This level contains the UNIX operating system and the hardware which at our site can be either a SUN-3, SUN-4 or CONVEX computer.
- The functional modules or data-processors. This level consists of all modules that do the actual work on the data.
- The input-processors or pre-processors. These Fortran modules accept a compact user friendly input specifying a certain class of calculations and generate a set of inputs for a number of data-processors. As some of these input-processors which originate from the SCALE-3 code system also generate a number of job control signals, they are originally called "control modules".
- The UNIPASC control level. This level contains the driver modules which have been written in C-shell language. The actual control of the input- and/or data-processors is done at this level.
- The window system. Optionally, the SUN user can invoke the "SunView" or "OpenWindows" environment that facilitate the multi-tasking usage of the workstations. The UNIPASC drivers test whether one of these window systems is in use.

2.2. The different processing steps

Another way to divide the system is based on the type of physical data that is processed and the level of contraction of information. Figure 2 shows a diagram of the different steps of the data processing. From top to bottom we can identify the following processing steps and associated data types:

- Preparation of the AMPX-Master cross sections. The problem independent evaluated point-wise cross sections in the JEF-1 data files are converted into a group organized AMPX-Master file. This processing is done by the NJOY system which is part of PASC-2 and has not been incorporated yet in the PASC-3 package. The operations involved are the Doppler broadening, resonance parameter treatment for various temperatures and the group averaging. The averaging is done using a standard weighting spectrum for Light Water Reactors. The NJOY system has been extended with the NSLINK interface developed at Delft University, ref.[3]. There are AMPX-Master files for various energy group

structures: the standard SCALE-3 CSRL libraries in 218 and 27 groups based on ENDF/B-4 and the ECNJEF-1 library in 219 groups, ref.[4], based on JEF-1.

- Preparation of the AMPX-Working cross sections. For a specific problem the resonance self shielding effects are taken into account for the resolved as well as the unresolved resonance regions. If needed, the energy group structure can be further condensed. Preferably, this condensing is done in a transport cell calculation which results in an AMPX-Weighted file (mostly compatible with an AMPX-Working file). Some of the reactor codes use the ANISN file format instead of the AMPX format. Working files no longer contain resonance parameters and can therefore be used by the reactor codes.
- The reactor calculations. The full size problem is calculated using the AMPX-Working cross sections in a 1D, 2D or 3D geometry. The calculated quantities can be either the neutron multiplication k_{eff} or k_{∞}, the flux spectrum as distributed over space and energy or nuclide concentrations in depletion calculations or activations or dose rates in shielding calculations.

2.3. The functionality of the modules

An overview of the PASC-3 modules is displayed in figure 3 which also shows the most usual sequential relation between the codes. The present functional modules can be divided into different categories. Some of these codes can be put in more than one category:

- System interfaces. These codes convert ascii coded data files into binary files and/or vice-versa, and are mainly needed at installation time. Some of them can also be useful as communication interfaces when a computation is transferred to systems of different architecture. To this class belong the codes AIM, ANL, AWL, COMPOZ, GIP, MAL, WTS.
- Internal interfaces. These codes change data library formats and provide the interfacing between the functional modules that originate from different code systems or provide a means for simple data base manipulations like merging of libraries or data selection from libraries. To this class belong the codes AJAX, CONTAC, COUPLE, GIP, JAWS, LAVA, OCTAGN.
- Editing codes. These codes make printouts, checks or plots of cross-section data sets. To this class belong the codes DIAL, RADE, VASELINE.
- Cross-section treatment codes. These codes perform the group-condensing, resonance self-shielding and weighting manipulations. To this class belong the codes BONAMI, ICE, MALOCS, NITAWL, XSDRN(PM).
- Reactor calculation codes. These codes perform 1D, 2D or 3D static neutron transport calculations systems that may vary in complexity from a single reactor cell to a complete reactor core. The numerical methods employed are either the discrete ordinates method solving the Boltzmann transport equation or the diffusion equation or the Monte-Carlo technique for a 3D simulation. To this class belong the codes CITATION, DOT, DORT, TORT and XSDRN using the discrete ordinates method and KENEUR and KENO(5A) using the Monte-Carlo method.
- Burnup depletion codes. To this class belong the codes CINDER (ref.[5]), CITATION, ORIGEN.
- Shielding or dose rate analysis code XSDOSE.

2.4. The input-processors

At present only two of the SCALE-3 input-processors have been installed in PASC-3, i.e.:

- CSAS1, the Criticality Safety Analysis Sequence no. 1. This input-processor was designed to automate 1D-cell calculations and produces the input files for BONAMI, NITAWL and XSDRN.
- SAS2, the Shielding Analysis Sequence no. 2. This input-processor automates the shielding analysis of 1D shipping casks containing spent-fuel assemblies. The spent-fuel concentrations are obtained by a replay of the burnup history of the fuel inside the reactor. This processor produces the inputs for the modules BONAMI, NITAWL, XSDRN, COUPLE, ORIGEN and XSDOSE.

3. The UNIPASC environment

3.1. Introduction

Most of the calculations usually require a more or less complex chain of module calls whereby the information is interfaced from step to step by intermediate data files. In particular in parameter studies and calculations of large reactor systems, the amount of work, programs and data is rather involved. Another problem with these calculations is the preparation of the program inputs which contains the steering of the programs. It is important that the inputs of different steps stay consistent with each other. Therefore it was felt desirable to find a way of coupling the input parameters in a chain automatically. As a result of these considerations, the UNIPASC system (or UNIPASC work environment) was developed, with the aim to ease the preparation of the jobs, to enhance the consistency between the different calculational steps and to reduce the problem to manage many programs and files. UNIPASC was embedded in the UNIX operating system which explains its name. UNIX contains a large number of commands, utility programs, facilities like the C-shell control language, ref.[6-7].

Some of the above mentioned motivations are exactly those which have led to the development of the SCALE input-processors. However, the purpose of UNIPASC is not to replace these modules but to provide additional control facilities.

3.2. The module driver "rpasc"

The modules from the PASC package are called through a common driver **rpasc** ("run a pasc module") which acts as an interface between the user and the code. A driver call looks like:

rpasc module_name [parameters]

With the optional parameters one can control the mode of operation of the driver or the module (i.e. normal run or a certain test mode) and connect the desired files to the module. The driver will assign default user file names belonging to each module. For a module with name "X", the (default) file name conventions are as follows:

X.inp	The standard input file
X.out	The standard output file
X.inlib1	The first binary input library
X.inlib2	The second binary input library
X.outlib1	The first binary output library
X.outlib2	The second binary output library
etc.	

In figure 4 we show schematically the procedure around **rpasc**. The input file X.inp is led through a filter **fldpasc** which performs, if necessary, some reformatting of the file. Most PASC modules accept free formatted input but do not read beyond column 72 or 80. Therefore, **fldpasc** will, if needed, fold long lines. Optionally, **rpasc** will create a new screen window in which the program output will appear. For that purpose the output stream is "piped" into **tee** which duplicates the stream and sends it to the screen and the output file X.out. The figure also shows the modifier flags that can be used to overwrite the default file names.

As a simple example we show how AJAX is called to select 6 nuclides from a standard data set while redefining the nuclide identifiers. Assuming that a file ajax_input contains the following data:

```
1$$ 1  T
2$$ 1  6  T
3$$ 1269 127303  1276  1193 1261039 1262039
4$$ 1001  5010  8016 13027  92235  92238  T
```

Then the command:

rpasc ajax -i ajax_input -lin1 $PASCLIBS/csr1218g074n

will do the required work. The -lin1 parameter is followed the full name of the input library overrules the default ajax.inlib1.

3.3. The batch job compiler "bpasc"

The next step to ease the execution of program chains is to combine job control and job inputs into a single file. Although this can in principle be achieved using standard control language, the job inputs may contain data with a special meaning to the control language. One example of that is the double dollar "$$" sign which means that an array of free formatted integers follows. However, within UNIX the double dollars are replaced by the current process identification number. Therefore it was decided to create a batch job compiler **bpasc** which allows the usage of these critical meta-characters and makes provisions to avoid misinterpretations. **bpasc** also expands a number of macro definitions (all starting with a "*") that make the job appear more friendly. Figure 5 shows schematically that **bpasc** will create from the source "job.BAT" a new file "job.CSH" containing pure C-shell language and run that job. By default the job will be run in a freshly created window (on the SUN only) showing all job outputs. In addition to all data files, also a logfile "job.LOG" will be made with time-stamped log messages. Parameters can be added to the **bpasc** command to change these and other options. **bpasc** can also translate the job into NOS-VE control language for the Cyber.
The AJAX job from above can now be transformed into the following "job.BAT":

```
*begin
set library         = csrl218g074n
*nuclides   $library
set libr_list       = (SH_1 SB_10 SO_16 SAl_27 SU_235 SU_238)
*nuclides   scale_general
set user_list       = (SH_1 SB_10 SO_16 SAl_27 SU_235 SU_238)
set nr_files        = 1
set file_index      = 1
*rpasc ajax         -lin1 SPASCLIBS/$library
*data
1$$ $nr_files    T
2$$ $file_index  $#libr_list  T
3$$ $libr_list
4$$ $user_list   T
*eod
*end
```

The literals in the input block (between the *data and *eod marks) have now been replaced by variables. The variables contain in principle only text but limited possibilities exist for numerical manipulations which are mainly intended for loop control purposes. Within certain restrictions the user can define the variable names freely. The macro definitions can be interleaved by any standard C-shell statements which makes it also possible to create complex sequences and loops.
The macro:

***nuclides library_name**

will automatically define all nuclide identifiers that are valid for the particular library and place them in variables named H_1, He_3, He_4, etc. Before doing that all previously defined nuclide definitions are erased. Therefore, the array "libr_list" will contain the nuclide identifiers as they appear on the csrl218g074n library, and the array "user_list" is loaded with a corresponding list of different ("scale_general") identifiers as needed for the subsequent calculations.

3.4. The usage of "filters"

Figure 6 shows schematically an application of a filter option which is very useful to generate sequences which slightly deviate from the standard SCALE sequences. The standard sequences are all controlled by a separate driver "resas". The input for *resas is supplied in a *data block in the same way as with *rpasc. With the "-seq" modifier it is possible to execute only a part of the sequence. In the example of figure 6 a first call to resas is used to execute only the CSAS1 input-processor and the data-processors BONAMI and NITAWL. Next, the macro "*importfido" is used to include the XSDRN input created by CSAS1 into the job and to load the input in shell variables. Next the user can modify some of the input

variables and re-use them in the XSDRN input which is supplied as a *data block to the next *resas call which executes the final step of the sequence.

4. Benchmarking.

Validation of code package and nuclear data that are used for criticality calculations are a continuous care of the Nuclear Analysis Group. At present, a number of standard criticality and PWR and BWR reactor benchmarks are carried out to validate our 219 groups ECNJEF-1 library. The reactor benchmarks are partly done in cooperation with other Dutch institutes (IRI at Delft, GKN at Dodewaard and KEMA at Arnhem).

Table I shows our results of the five ORNL-spheres benchmark, ref.[8], calculated with the 1D transport code XSDRN(PM) (used in the CSAS1 sequence) and the 3D Monte-Carlo code KENO-5A. The numbers shown are the k_{eff} from the experiment and the results of the calculations that have been carried out with the 218 groups CSRL library and the 219 groups ECNJEF-1 library. The last row gives the RMS value of the difference between experimental and calculated values. The individual numbers as well as the RMS errors show that the ECNJEF-1 results are better in agreement with the experimental values. A more striking difference has been observed in low enrichment reactor benchmarks where the CSRL library may give results that are 1-2 percent too low. Further experimental benchmark calculations are in progress at the time of writing this report.

5. Conclusions.

- The PASC-3 code system in its present form is a powerful modular system to perform a large class of static reactor calculations. A next step in the development of the code system will be to include modules to facilitate dynamic reactor calculations.
- At ECN, PASC-3 is extensively used for criticality calculations. The benchmark results show that high accuracy calculations are possible with our new 219 group library.
- By the active usage of the UNIPASC work environment, the flexibility of the system is considerably increased. Job inputs are integrated into the job control language whereby literal data may be replaced by variables. This practice may improve the readability of the job and allows further automation. A drawback of our approach is that it relies on the built in programming language of a certain operating system but the basic idea can in principle also be applied in other operating systems.
- Extension of PASC-3 to PASC-4, including the SCALE-4 modules is envisaged.

We like to acknowledge the contributions of N.H.Dekker, H.Koning and M.Bernards for their code conversions and system support. We thank H.Gruppelaar for fruitful discussions.

Table I. Benchmark results for the five ORNL spheres.

		CSRL-218		ECNJEF-1	
Case	exp.	XSDRNPM	KENO-5A	XSDRNPM	KENO-5A
ORNL-1	1.00026	0.99736	0.9950±0.0009	1.00131	1.0035±0.0009
ORNL-2	0.99975	0.99696	0.9977±0.0009	1.00096	1.0012±0.0009
ORNL-3	0.99994	0.99385	0.9952±0.0009	0.99784	0.9991±0.0009
ORNL-4	0.99924	0.99526	0.9936±0.0009	0.99927	1.0017±0.0009
ORNL-10	1.00031	0.99547	0.9969±0.0007	0.99830	1.0007±0.0007
RMS error	-	0.0043	0.0044	0.0015	0.0020

Table I. Benchmark results for the CSRL-218 and ECNJEF-1 libraries using the 1D XSDRN code and the 3D KENO code compared to the experimental values.

References.

[1] RSIC computer code collection, Radiation Shielding Information Center, Oak Ridge National Laboratory, Tennessee, USA. Codes made available from the NEA Data Bank, Saclay, France.

[2] Workshop on the SCALE-3 modular system, Saclay, 24-27 June 1986, NEA Data Bank Newsletter No.33, October 1986.

[3] P.F.A.de Leege, NSLINK: NJOY-SCALE-LINK, report IRI-131-091-003, Delft University of Technology, Delft, The Netherlands.

[4] Yu Peihua and H.Gruppelaar, The ECNJEF1 library, A 219 neutron group library from JEF-1 in the AMPX master format. ECN report NFA-LWR-90-03, Petten, The Netherlands.

[5] Argonne Code Center, Argonne National Laboratory, Illinois, U.S.A.

[6] B.W. Kernighan, R. Pike, The Unix Programming Environment, Prentice-Hall Software Series.

[7] G. Anderson, P. Anderson, The Unix C Shell Field Guide, Prentice-Hall.

[8] The Oak Ridge National Laboratory unreflected sphere benchmarks ORNL-1,-2,-3,-4 and -10 taken from: Cross Section Evaluation Working Group benchmark specifications, ENDF-202 / BNL 19302 (November 1974).

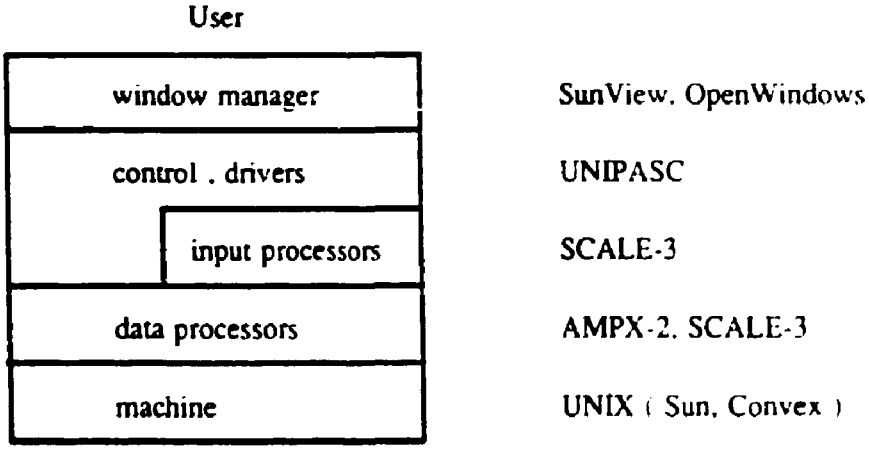

Figure 1. A layer diagram of the PASC-3 software. The data- and input-processor layers consist of Fortran modules; the UNIPASC layer consist of UNIX C-shell programs. The window manager level is optional.

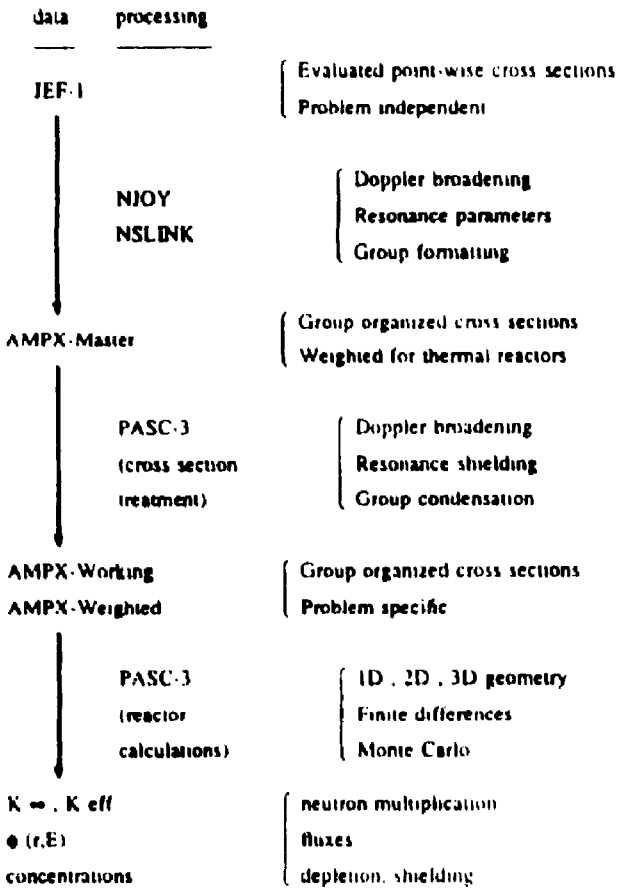

Figure 2. The different processing steps from the Evaluated Nuclear Data File to the final physical quantities. The NJOY-NSLINK system is the first step which produces the AMPX-Master files and is part of PASC-2 which runs on our Cyber. The present PASC-3 system comprises the second and third steps, i.e. the preparation of the cross sections and the reactor calculations.

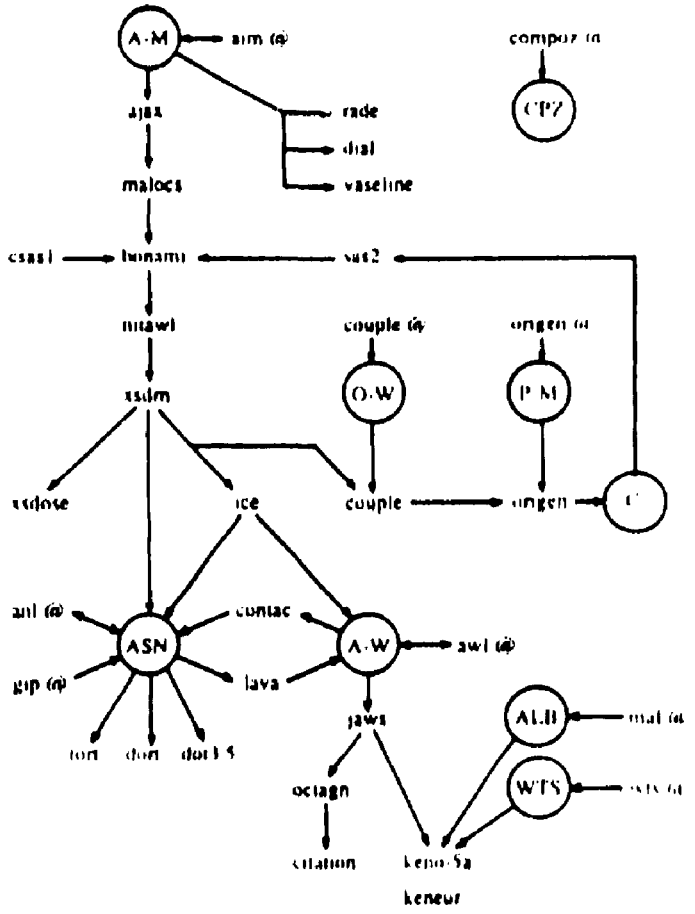

Figure 3. Overview of the data processors of the PASC-3 code system. The codes marked with a "@" are system interfaces or codes that have been used as such. Only the most important library types are shown: AMPX-Master (A-M), Standard Compositions (CPZ), ORIGEN-Working (O-W), Photon Master (P-M), Origen-Concentrations (C), ANISN binary (ASN), AMPX-Working (A-W), Albedos (AL.B), and Weights (WTS).

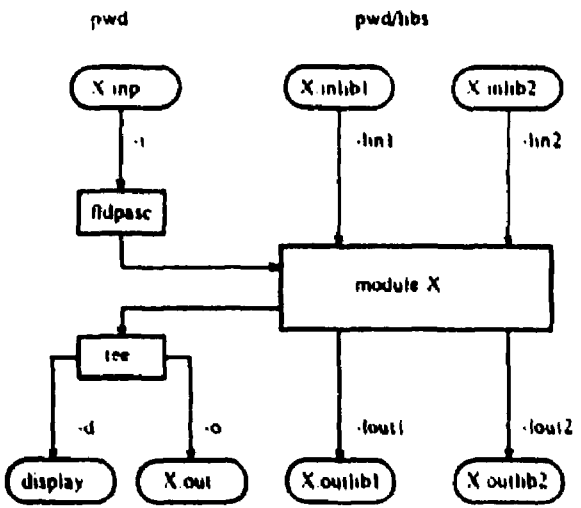

Figure 4. File handling by the module driver "rpasc". The figure shows the default file names for an arbitrary module "X". The input "X.inp" is filtered by "fldpasc" before it is applied to the module. The "tee" command is used to duplicate the standard output to the display and the file "X.out"

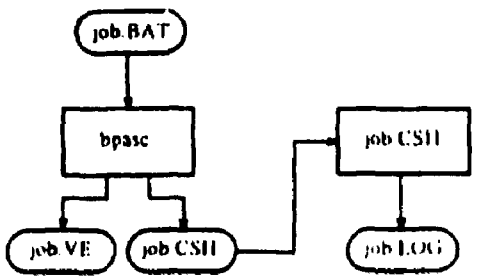

Figure 5. The job compiler "bpasc" is used to expand the macro definitions in "job.BAT" and produces the Unix job "job.CSH" and optionally for the Cyber "job.VE".

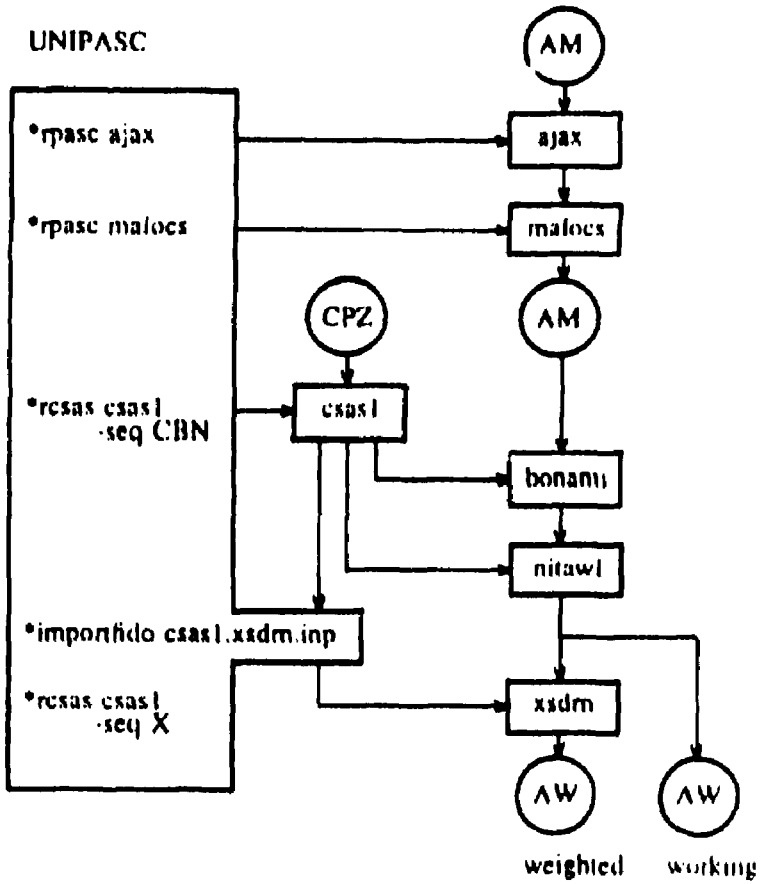

Figure 6. Flow diagram of a job sequence in which the XSDRN input file as produced by CSAS1 is filtered to modify the last part of the procedure.