KERNFORSCHUNGSZENTRUM

KARLSRUHE

November 1977 KFK 2387/VII

Institut für Neutronenphysik und Reaktortechnik
Projekt Schneller Brüter

The KEDAK Program Compendium
Part VII
CALCUL-Calculation of Composed Cross Sections

I. Langner, R. Meyer

GESELLSCHAFT
FÜR
KERNFORSCHUNG M.B.H.

KARLSRUHE

GESELLSCHAFT FÜR KERNFORSCHUNG M. B. H.

KARLSRUHE

KERNFORSCHUNGSZENTRUM KARLSRUHE

KFK 2387/VII

Institut für Neutronenphysik und Reaktortechnik

Projekt Schneller Brüter

The KEDAK Program Compendium

Part VII

CALCUL-Calculation of composed cross sections

I. Langner, R. Meyer

Gesellschaft für Kernforschung mbH., Karlsruhe

Abstract

CALCUL is a program system to calculate composed cross section quantities and to arrange them according to KEDAK-conventions. A special command language has been generated to simplify the input and to provide interactive communication under time-sharing-option. In CALCUL any amount of data can be processed. The output of the program is a data set which can be directly used as an input to program system KEMA which modifies the KEDAK-library.

Das KEDAK Programm Compendium
Teil VII
CALCUL - Berechnung von zusammengesetzten Wirkungsquerschnitten

Zusammenfassung

Das Programmsystem CALCUL berechnet zusammengesetzte Wirkungsquerschnitte und besorgt deren Aufbereitung entsprechend den KEDAK-Konventionen. Zur Vereinfachung der Eingabe wurde eine eigene Kommandosprache entwickelt, die es auch ermöglicht, interaktiv im Time-sharing-Betrieb zu arbeiten. Mit CALCUL können beliebige Datenmengen bearbeitet werden. Die Ausgabe des Programms ist ein Datensatz, der als Eingabe für das Programmsystem KEMA dient, welches die KEDAK-Bibliothek verändert.

Contents

Introduction

This report is intended to provide the information needed to use and maintain the program system CALCUL. The program system CALCUL assists the evaluator of neutron cross section data by the computation of composed cross section values processing any amount of data points and interpolating the cross section values linearly, if necessary. The data processed are retrieved from the KEDAK-library and/or from an external source (tape, disk or cards).

The output is a set of neutron cross section data in a format suitable for KEDAK-update by the program system KEMA.

The report is divided in two parts:

1. User's guide to give the information necessary to use the program system CALCUL: purpose, capability, layout, input, output, job control language.

2. Programmer's guide - the detailed description of the particular subroutines, auxiliary data sets, work areas and common blocks used in CALCUL.

Zum Druck eingereicht am: Nov. 1977

1. User's guide

1.1 Purpose and capability of CALCUL

CALCUL is a program system to calculate cross section quantities. In order to insert neutron cross section data evaluated elsewhere (or at Karlsruhe) into the KEDAK-library it is necessary to (re-) calculate and arrange them according to KEDAK-conventions. CALCUL processes data from an external source and/or data from the KEDAK-library. A control input, in form of a command language, controls in which manner the data are processed.

The output is a set of data consisting of ADD-records and DROP-records, that tells the KEDAK-Management program, which data are to be added to the library, and which data are to be deleted from the KEDAK-library.

1.1.1 The commands and formulae for calculation of composed data

The commands and formulae for computation of composed cross section values are listed below.

The calculation of these values is performed with the aid of the single arithmetic operations described below, but the control in this case is performed by a subprogram, instead of the control input lists for the single commands, i.e. the writing, reading, and processing of these control input lists is spared.

ALPHA1	$\alpha = \sigma_\gamma/\sigma_f$
ALPHA2	$\alpha = (\nu/\eta)-1$
ETA1	$\eta = \nu/(1+\alpha)$
ETA2	$\eta = \nu*\sigma_f/(\sigma_f+\sigma_\gamma)$
NUSF1	$\nu_f = \nu*\sigma_f$
SGA1	$\sigma_a = \sigma_\gamma + \sigma_f + \sigma_p + \sigma_\alpha + \sigma_d$
SGG1	$\sigma_\gamma = \sigma_f*\alpha$
SGG2	$\sigma_\gamma = \sigma_f*((\nu/\eta)-1)$
SGG3	$\sigma_\gamma = \sigma_a - \sigma_f - \sigma_p - \sigma_\alpha - \sigma_d$
SGI1	$\sigma_{n'} = \sigma_x - \sigma_\gamma - \sigma_p - \sigma_{2n} - \sigma_\alpha - \sigma_{3n} - \sigma_f - \sigma_d$

SGN1	$\sigma_n = \sigma_T - \sigma_x$
SGT1	$\sigma_T = \sigma_n + \sigma_x$
SGTR1	$\sigma_{trans} = \sigma_T - \sigma_n \cdot \mu_L$
SGX1	$\sigma_x = \sigma_T - \sigma_n$
SGX2	$\sigma_x = \sigma_{n'} + \sigma_\gamma + \sigma_f + \sigma_p + \sigma_\alpha + \sigma_{2n} + \sigma_{3n} + \sigma_d$

1.1.2 The arithmetic operations

CALCUL assists the user in the general problem of exercising the basic mathematical operations upon functions depending on one variable (energy). Thus CALCUL simulates a desk calculator operating on functions instead of single numerical values, with the necessary interpolation being performed.

1. Operation "+" $R = R+y$	ADD-command R - the current result, is equal to zero at the start of the calculation R,y are data arrays
2. Operation "-" $R = R-y$	SUBTRACT-command
3. operation "." $R = R\cdot y$	MULTIPLY-command
4. operation "/" $R = R/y$	DIVIDE-command
5. composed arithmetic operation $R = 1/(1+R)$	ETA calculation

1.1.3 The data organization commands

Reset the current result to zero	-	INIT-command
Delete the previous operation	-	DELETE-command
Store the current result for later use in CALCUL	-	NAME-command
Store the current result for the output data set	-	SAVE-command
Read the data from external source, (i.e. data not stored in the KEDAK-library)	-	INPUT-command

1.1.4 The program control commands

Restart the program at beginning	-	RESTART-command
Create the output data set from the data saved during the calculation	-	STOP-command

1.2 Structure of CALCUL

CALCUL is written in modular form with many subroutines to facilitate the addition of new options and to permit the generation of a compact overlay structure.

The subroutines of CALCUL are comprised in the following modules:

1. Control module
2. Operation code (command) definition package - OPDEF
3. Control input processing package - PROCINP
4. Calculation package - CALCPAC
5. Cross section formula calculation package - CROSSEC
6. Data management of the auxiliary data sets - DATAMAN.
7. Output edition package (KEMA-input-format data set) - OUTPUT

1.2.1 The control module

The program flow of CALCUL is mainly controlled by the control input. The main program and the subroutines described in 2.1 "The controle module" perform the functions required to control the program execution. The command language for the control input is defined in the module OPDEF before the calculations once at program start of CALCUL.

The control input processing module is executed for each entered command.

First command entered should be INIT - to initialize parameter values. Then the operation mode: GO or WAIT is ascertained, a check for DD-cards is performed, the operation code is tested, the data to be processed are transferred into the work area, and calculations are performed, as required by the operation code entered.

1.2.2 The operation code definition package - OPDEF

The operation code definition package defines the operation codes (commands) and the corresponding positional and keyword parameters of the control input for CALCUL.

The operation codes, keywords and their parameters are stored in internal tables, the G-array, H-array and the common /OPAR/.

The lengths of the G-array and H-array depend on the number of defined commands, and keywords, and on the number of their parameter values. The area for the arrays G and H is provided by the main program and fixed in OPDEF, at present 500 REAL*4 words for each array.

The G-array (see Table 1) retains the data for the operation code definition: the operation code names (commands = alphameric text as listed above), for each command: the length in characters, the abbreviation length, the number of keywords defined for each command, the number of positional parameters, the addresses of the definition data for each keyword in the H-array, the addresses of the locations in the common /OPAR/ where the values of the positional parameters are stored and the types of the positional parameters.

The H-array (see Table 2) retains the data for the keywords defined for each command: the keyword names (alphameric text), for each keyword: the length of its name, the abbreviation length of its name, address of a flag byte in the common /OPAR/ retaining a flag, which indicates if a parameter value was found in the central input list, the address of the list of the keywords exclusive with this key, the default control parameter, the number of parameters, addresses of the locations in the common /OPAR/ where the parameter values for the keys are to be stored and the parameter type.

The common /OPAR/ retains the names and the definition data of the keywords and positional operands for the package. The keywords are defined before the commands. If the key is to be assigned to a command, the definition data are stored from /OPAR/ into the H-array. (For /OPAR/ see Table 3.)

1.2.3 The control input processing package PROCINP

The control input processing package decodes the character string of a control input list, processes the control input and prepares the positional and the keyword parameter values from input in the common /OPAR/.

The processed control input list is printed for checking.

If the control input for CALCUL is produced in the WAIT-mode in a foreground job on a terminal in TSO, the input entered is completed by the control input processing package and the control input list is printed at the terminal for checking.

The user may then decide, whether this input list should be added to the control input in card image format created on the data set with the reference number 9 or not. The control input list is added by a call to SIMUL for KTOUT = 9.

1.2.4 The calculation package CALCPAC

The basic arithmetic operations (with interpolation if necessary) on tabulated functions are performed in CALCPAC. Also the preparation of data to be processed is done:

1. Check, if data to be processed are already retrieved or calculated and stored in the temporary direct access data set DADS2.

2. Find out, which data are to be read from the KEDAK-library.

3. Find out, if they are available on the KEDAK-library.

4. The requested data are transferred from DADS2, and from KEDAK, respectively, into the working area of CALCPAC by the aid of the proper retrieval routines.

The working area is subdivided in three arrays:

1. (x,y)
   (x1,y1) { The data from previous operation and the current result. A flag ADR in the common /CALCOM/ determines which is the current result:
   ADR = .TRUE. = (x,y) $\hat{=}$ current result
   ADR = .FALSE. = (x1,y1) $\hat{=}$ current result

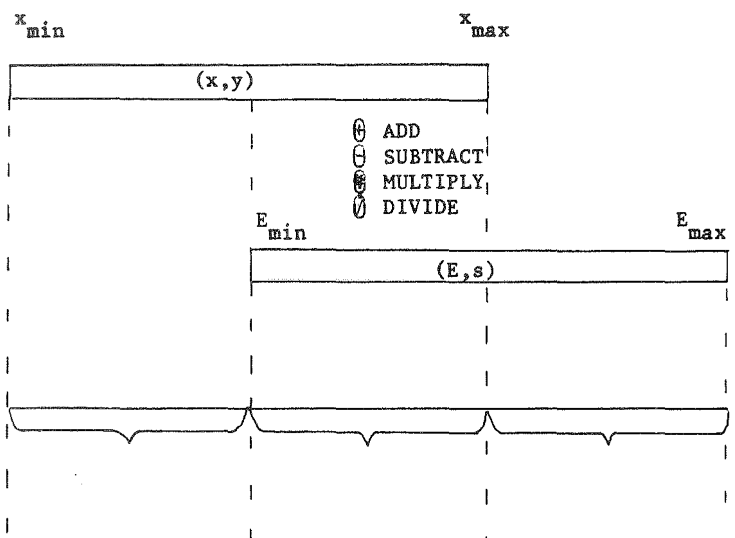
(E,s) - The data from DADS2 or KEDAK.

If the work area in main storage is not sufficient to store the processed data (i.e. the number of data is greater than the array length), the data are stored on two auxiliary sequential data sets with the data set reference numbers 3 and 4 (DCB =(RECFM = VBS, BLKSIZE = 2008)).

The following picture assists in deciding upon the result obtained in cases where the energy ranges of the current result and the data from KEDAK or DADS2 do not coincide.

In our sample the current result extends below the data from KEDAK, while the latter extends to energies above, with an overlapping range between E_{min} and X_{max}.

Then the results of the different types of arithmetic operations would be as follows:



1. ADD

X1 = X
Y1 = Y
	X1 = XUE
	Y1 = Y+S
		X1 = E
		Y1 = S

i.e. $Y = 0$ for $(E < X_{min})$ $E > X_{max}$

* $S = 0$ for $X < E_{min}$ $(X > E_{max})$

2. SUBTRACT

X1 = X
Y1 = Y
	X1 = XUE
	Y1 = Y-S
		X1 = E
		Y1 = -S

i.e. $Y = 0$ for $(E < X_{min})$ $E > X_{max}$

* $S = 0$ for $(X > E_{max})$ $X < E_{min}$

* Outside the energy range [EMIN,EMAX] the (X,Y)-data to be processed are not changed.

Outside the range of available X-values the (E,S)-data are stored unchanged for ADD.

For SUBTRACT the S-values change to negative.

3. MULTIPLY

X1 = X
Y1 = 0
or Y1 = Y
	X1 = XUE
	Y1 = Y*S
		X1 = E
		Y1 = 0

i.e. for $X < E_{min}$ $<$ ZERO = .TRUE. $\Rightarrow$ S = 0; ZERO = .FALSE $\Rightarrow$ S = 1; ZERO $\hat{=}$ ZLOW from input

for $X > E_{max}$ $<$ ZERO1 = T = S $\Rightarrow$ 0; ZERO1 = F = S $\Rightarrow$ 1; ZERO1 $\hat{=}$ ZUP from input

for $E \notin (X_{min}, X_{max})$: S = 0

4. DIVIDE

X1 = X
Y1 = 0
or Y1 = Y
	X1 = XUE
	Y1 = Y/S
		X1 = E
		Y1 = 0

i.e. for $X < E_{min}$ $<$ ZERO = .TRUE. $\Rightarrow$ S = 0; ZERO = .FALSE $\Rightarrow$ S = 1

for $X > E_{max}$ $<$ ZERO1 = .TRUE. $\Rightarrow$ S = 0; ZERO1 = .FALSE. $\Rightarrow$ S = 1

for $E \notin (X_{min}, X_{max})$: S = 0

if S = 0 $\Rightarrow$ Y = 0

1.2.5 Cross section formula calculation package CROSSEC

The module CROSSEC provides a set of often used calculation formulae that will be satisfactory for most applications. The user is released from the job of writing the complex input list for each operation and data type included in the formula, but he must know which cross sections are involved in the formula, and whether data for these types are available on the KEDAK-library (or as external source). If the data for a type are not available, the calculation is performed without these data and a warning is printed.

For each formula a command is provided in the command list. (See the commands in 1.1.1). The commands are defined in OPDEF (see 2.2.1).

The subroutines for the formulae for calculation of composed cross section values are comprised in the cross section calculation package. The subroutines of the calculation package CALCPAC are used by CROSSEC to perform the basic operations. The subroutine EDIT is called to store calculated data intermediate into the temporary direct access data set DADS2.

1.2.6 DATAMAN - The auxiliary data sets of CALCUL: DADS2 - direct access data set and work area - sequential data sets

In order to provide increased flexibility, a special direct access data set (DADS2), with the dataset reference number 2, is created (DCB = (BLKSIZE = 2000, RECFM = F)).

The data from the external source (read by the INPUT command) and the data calculated in CALCUL are stored in DADS2 for later use in calculation and/or to edit them in the KEMA-input-format.

The data to be edited are specified to the program by the SAVE-command, data for later use in CALCUL only by the NAME-command.

The data are stored in records of 2000 bytes = 500 words.

The common /DA1/ retains information about the layout and the status of DADS2, the common /DA2/ the entry tables for the reaction types for which data are stored.

The DELETE-command is provided to delete the result of the last arithmetic operation in CALCPAC. The printout of the data currently stored as result (in CALCPAC) or of the data to be named or saved on DADS2 for checking purposes is managed in DATAMAN and controlled by the key PRINT or NO PRINT in the control input of the processed command.

1.2.7 Output editing package (KEMA-input-format data set)

The data calculated in CALCUL are written for output in ADD-records - with a maximum of 2000 words - for the KEDAK-management program KEMA. The records are sorted for material names, reaction type names, energy, and energy range in the same order as in the KEDAK-library.

The KEDAK-library is checked, whether data already exist on the library at energy points for which data are to be added. To delete these data DROPS-records are written on the output data set for each energy. All data for a requested type may be deleted from the library by DROPA-records. DROPA-records are written on the output, only if indicated by the control input.

1.3 The external references

The retrieval packages RETPAC (see reference 6) and LDFPAC (see reference 1) are used to read the data from the KEDAK-library. The following subroutines are external references: XTAREA (see reference 2) and FREESP (see reference 8) - to handle the dynamic storage allocation

CONVY (see reference 7) and STRING - to convert floating point and integer data to alphameric and vice versa.

DEFI and DINF (see reference 4)- to make the DEFINE FILE statement dynamical.

DDTEST (see reference 3) - to test DD-cards.

1.4 The control input for CALCUL

1.4.1 The function of the control input

The control input for CALCUL has the form of a command language and controls the flow of the program and in which manner the data are processed.

The control input is read in from cards or from a card image input data set on type or disk, and processed by the control input processing package, and provided for the program via common /PARM/.

1.4.2 GO-mode and WAIT-mode

CALCUL is designed to operate either in the GO-mode or the WAIT-mode. An interactive facility enables the communication between the user and the control input processing program, in WAIT-mode in a foreground job on a terminal in TSO, in a COMMAND-mode and in a REPLY-mode. In the COMMAND-mode the user may enter control input according to syntax rules, and in the REPLY-mode he replies to a message from the program. The user may inform himself which replies are allowed by entering a question mark. The GO-mode can be used (in a foreground or) in a background job and is restricted to the COMMAND-mode. The syntax rules for the control input list apply to the WAIT-mode and to the GO-mode.

1.4.3 Input coding in the WAIT-mode

A special facility is provided in the WAIT-mode of CALCUL. The input entered at the terminal via keyboard is supplied by the control input processing program, if necessary, to a complete control input list for each command entered, and provided in a "card-image-format" data set for later use in a GO-mode run.

Input coding in the WAIT-mode enables the user to reduce the possible errors in the complex input coding for the cross section evaluation by a programmed check:

1. if the entered parameter types are valid for the positional and keyword operands,

2. if all required positional and keyword operands for the command are entered,

3. if the syntax rules for coding the control input list for the command are violated,

4. and by completion of the control input list for a command with the default values of parameters (default values remain valid until explicitly overridden by a new input value),

5. and by prompting the input for the parameter values without a default value available.

A brief description of the available commands can be obtained by the HELP command. Information about some general rules for the use of some special characters is also available:

1. hyphen (-)	:input expected or will be entered
2. question mark (?)	:requests information from the program
3. exclamation mark (!)	:cancels the last input
4. underscore (_)	:input continuation is indicated
5. slash (/)	:terminates the input (request) for a command.

The special characters are allowed in the REPLY-mode at the terminal only. The input of a question mark is always allowed, also in the COMMAND-mode, that is whenever the message: "ENTER CONTROL INPUT LIST -" is printed. The user may enter the control input list for a command according to syntax rules specified below. Moreover information about syntax rules and background use is available from the program. If HELP'command name' is entered all available information about the command named is printed.

The user may alter the mode at the terminal (e.g. for testing purposes) by the TERMINAL command (TERM, NOTERM). The LOAD-module of CALCUL is required as a TSO-data set to operate CALCUL in the WAIT-mode on a TSO terminal.

The listing of the TSO procedure OPTEST shows an example of TSO commands necessary to work with CALCUL in a foreground job.

The command:'exec optest' initiates the execution of the program for input coding.

```
OPTEST.CLIST
00010 PROC 0
00015 TERMINAL LINESIZE(130) SCRSIZE(12,80)
00020 FREE F(FT05F001)
00030 FREE F(FT06F001)
00040 ALLOC F(FT06F001) DA(*)
00050 ALLOC F(FT05F001) DA(*)
00055 ALLOC F(FT09F001) DATASET('TS0048.DATA.CNTL')
00060 CALL 'TS0048.OPAC.LOAD(OP1)'
00070 END
READY
exec optest.clist
```

The following list is a protocol of a session at a terminal. The user may enter input whenever a hyphen appears as last character of a message. If "VERIFY -" appears, the user may hit the return key for verification of the input, or enter the exclamation-mark to delete this input. The user may enter a question-mark to obtain information upon the type of input expected.

```
INVALID SCRSIZE OPERAND, USE LINESIZE

ARE YOU FAMILIAR WITH PROGRAM DESCRIPTION AND INPUT-CODING? ENTER YES OR NO. -
yes

DO YOU WISH TO OPERATE IN GO MODE OR IN WAIT MODE? ENTER GO OR WAIT -
wait

NO DD-CARD FOR FT01F001 HAS BEEN SUPPLIED TO DEFINE THE KEDAK LIBRARY.
NO EXTENDED TESTS WILL BE PERFORMED.

YES
GO

YOU ARE USING THE KEYED CONTROL INPUT PROCESSING PROGRAM NOW. ENTER INPUT
IF A BREAK IS THE LAST CHARACTER OF A MESSAGE DISPLAYED. TO OBTAIN INFORMATION
UPON THE TYPE OF INPUT EXPECTED YOU ALWAYS MAY ENTER A QUESTION MARK, IF
THIS IS NOT LITERALLY EXCLUDED. TO OBTAIN GENERAL AND SYNTAX INFORMATION
ENTER HELP OR ? WHEN YOU ARE IN THE COMMAND MODE. USE THE TERMINAL COMMAND
AT A 2260 TERMINAL. YOU ARE IN THE COMMAND MODE NOW. ENTER CONTROL INPUT
LIST -
add
DEFAULT ASSUMED.
VERIFY -

NO MATERIAL NAME AVAILABLE. REENTER NAME ONLY, WITHOUT DELIMITING APOSTROPHES -
u 238
NO TYPE NAME AVAILABLE. REENTER NAME ONLY, WITHOUT DELIMITING APOSTROPHES -
sgt
THE FOLLOWING CONTROL INPUT LIST HAS BEEN PREPARED FOR OUTPUT:
ADD     'U 238'    'SGT'    0.0     FROM=    0.0     TO=_
       -1.00000E+00     NOPRINT   OUTUNIT= 10    CONST=  0.0
VERIFY -

ENTER CONTROL INPUT LIST -
i
DEFAULT ASSUMED.
VERIFY -

THE FOLLOWING CONTROL INPUT LIST HAS BEEN PREPARED FOR OUTPUT:
  INIT      FROM=      0.0      TO=      -1.00000E+00
VERIFY -

ENTER CONTROL INPUT LIST -
su    'u 238'    'sgx'    from=2.e+3     to=12.e+5     print=6
THE FOLLOWING CONTROL INPUT LIST HAS BEEN PREPARED FOR OUTPUT:
 SUBTRACT    'U 238'    'SGX'     0.0      FROM=      2.00000E+03      TO=_
             1.50000E+06     PRINT=  6      CONST=  0.0
VERIFY -
!
READY
```

1.4.4 The commands and their valid abbreviations

A command is one of the listed below operation codes. The bracket denotes the valid minimum abbreviation. It may be extended by an optional number of characters up to the full length of the operation code name:

ADD(A), SUBTRACT(SU), MULTIPLY(M), DIVIDE(D), ETA(E), NAME(N), SAVE(S), DELETE(DE), STOP(ST), RESTART(R), INIT(I), INPUT(INP), ALPHA1(AL), ALPHA2, ETA1, ETA2, SGA1(SG), SGG1(SGG), SGG2, SGG3, SGI1(SGI), SGN1(SGN), SGT1(SGT), SGTR1(SGTR), SGX1(SGX), SGX2.

1.4.5 The control input list

The control input list is a block of control input data which must be entered as a logical unit. A control input list consists at least of one command, in general:

1. of the command

2. of the (max. 3) positional parameters
   a) isotope name
   b) data type name
   c) third name (e.g. excitation level energy)
   (e.g. 'U238' 'SGIZ' 1.4E+3)

3. and of up to seven keyword operands with their parameter lists.

1.4.5.1 The data types of the parameter values of positional and keyword operands

Four types of data are accepted by the control input processing program as input data: real, integer, logical, and text data. The real and integer data are coded as usual in FORTRAN; if the type of data entered is not of the type expected, it is converted. Real data of single precision only can be handled. Logical data are coded as F or T for .TRUE. or .FALSE. respectively. Text data must be enclosed within quotes and must not exceed the size expected by the program (maximum 36 characters). The parameter value is assigned to the keyword by the equal sign, e.g. PRINT = 6. The keywords may appear in any sequence and the input of keywords with default values is optional. For these keywords not specified in the control input list the default value is inserted.

Example of a control input list

```
ADD     'U 238'    'SGT'    FROM = 1.E+3_
        TO = 12.E+4    PRINT = 6
```

<u>1.4.5.2 Table I</u> The defined keywords, their parameter types and syntax (R-real, I-integer, L-logical, T-text)

keyword name	parameter value	parameter type	abbreviation of the name	explanation	default value
FROM =	1.0	R	F	lower energy limit	0.
TO =	10.0	R	T	upper energy limit	-1.
PRINT =	6	I	P	output unit number for the printed output	-
NOPRINT	no parameter		N	Print out is suppressed	NO PRINT
OUTUNIT =	10	I	O	data set reference number of the output data in KEMA-input format	10
CONST =	5.	R	C	constant to be added, subtracted etc. to the current result	0.0
ZUP =	T	L	Z	zero range upper TO	T
ZLOW =	T	L	ZL	zero range lower FROM	T
FORMAT =	'(1X,2E13.6)'	TR	FO	format of the data record on the external source	obligatory
UNIT =	20	I	U	data set reference number for the external source	obligatory
SKIP =	2	I	S	number of data records to be skipped on unit = 20 before transfer of data into DADS2 begins	0
REWIND	no parameter				REWIND

Note: The keywords PRINT and NOPRINT are mutually exclusive. IF CONST is entered, no positional operands are allowed in the input.

The input of the keywords FORMAT and UNIT defined for the INPUT command is obligatory,no default values are available.

The single keyword operands with their parameter values are separated in the control input list by a comma or a blank. The positional parameters are also separated by a comma or by blanks. If only the separating comma is coded, the preceeding positional operand is bypassed and its current value is taken as default. Trailing positional operands to be bypassed need not be indicated by commas, but they simply may be omitted from the list. If the number of positional operands entered in the input list exceeds the number permitted by the current command, an error condition will be raised and the excessive values will be skipped by default. If a control input list does not fit in one line, an underscore after the last input item indicates a continuation line for the control input list. A control input list may consist of as many lines as necessary, but its size must not exceed 360 characters.

A slash (/) may be used to indicate the end of a control input list.

The use of the slash is optional, but in case of numbered input cards it may be used to inhibit reading of sequence numbers.

1.4.5.3 The commands and their positional and keyword operands

The three positional parameters (isotope name, reaction type name, and third name) are defined for the following commands: ADD, SUBTRACT, DIVIDE, MULTIPLY, NAME, SAVE, INPUT and all 15 commands for composed cross sections: ALPHA1, ALPHA2, ETA1, ETA2, NUSF1, SGA1, SGG1, SGG2, SGG3, SGI1, SGN1, SGT1, SGTR1, SGX1, SGX2.

Commands with no positional and keyword parameters are: DELETE, RESTART, STOP. The control input list for these commands consists of the single command name. The command HELP without operands provides the print out of information available from the program.

HELP command name (e.g. HELP ADD) provides the information for the command.

Two keywords are defined for the command INIT: FROM and TO . The command INIT must be entered before each command to calculate the composed cross section values (e.g. ALPHA1, etc.) and whenever a new calculation for a type is started with the aid of single commands. The current result is set equal to zero by INIT.

Table II The commands and their positional and keyword operands

command	number of positional parameters	number of keywords	keyword names
INIT	0	2	FROM, TO
SUBTRACT	3	5	FROM, TO, NOPRINT, (PRINT),CONST
MULTIPLY	3	7	FROM, TO, NOPRINT, (PRINT), CONST, ZLOW, ZUP
DIVIDE	3	7	FROM, TO, NOPRINT, (PRINT), CONST, ZLOW, ZUP
ETA	0	2	NOPRINT, (PRINT)
INPUT	3	6	FROM, TO, FORMAT, UNIT, SKIP, REWIND
NAME	3	4	FROM, TO, NOPRINT, (PRINT)
SAVE	3	5	FROM, TO, NOPRINT, (PRINT), OUTUNIT
ALPHA1	3	7	FROM, TO, NOPRINT, (PRINT), OUTUNIT, ZLOW, ZUP
ALPHA2	3	7	FROM, TO, NOPRINT, (PRINT), OUTUNIT, ZUP, ZLOW
⋮	⋮	⋮	⋮ ⋮ ⋮ ⋮ ⋮ ⋮ ⋮
SGX2	3	7	FROM, TO, NOPRINT, (PRINT), OUTUNIT, ZUP, ZLOW
DELETE	0	0	
RESTART	0	0	
STOP	0	0	

1.4.5.4 The function of the commands:

INIT	-	resets the current result to zero.
ADD	-	adds a set of cross section data to the current result.
SUBTRACT	-	subtracts a particular set of data from the current result.
MULTIPLY	-	multiplies the current result by a particular set of data.
DIVIDE	-	divides the current result by a particular set of cross section data.
ETA	-	calculates ETA using the current result as ALPHA, i.e. the new result is 1/(1+R), where R is the old result.
NAME	-	assigns a name to the current result and stores this set of data for later use within the current run. The data are not saved beyond the end of the run.
SAVE	-	assigns a name to the current result and stores this data set for later use within the current run. Data will be edited at the end of the current run in a format suitable for KEDAK-updata by KEMA.
DELETE	-	deletes the result of the previous operation (only if the data were not stored in main storage but on an auxiliary data set).
RESTART	-	restarts the program at beginning. All data constructed so far will be lost.
INPUT	-	reads the formatted input from an external source.
STOP	-	causes editing of all saved data and stops program execution.

The function of the particular commands for cross section calculation is described above by the formulae (see 1.1.1).

The user must know which cross section values are used for calculation by the command chosen, and whether the data are to be used from the KEDAK-library or from DADS2. First the dataset DADS2 - with the data calculated in the current run - is checked, if the data needed in the formula are available, otherwise the data are read from the KEDAK-library. The user must consider this priority of calculated data in coding the control input.

1.4.6 An example of a control input for CALCUL

```
//INR048T1 JOB (0048,101,P6M1A),LANGNER,CLASS=A,REGION=310K
/*SETUP DEVICE=2314,ID=GFK029
/*SETUP DEVICE=2314,ID=GFK050
// EXEC FHG,NAME=OP1
//STEPLIB DD UNIT=2314,VOL=SER=GFK029,DSN=INR.CALCUL.LOAD,DISP=SHR
//G.FT01F001 DD UNIT=2314,VOL=SER=GFK050,DISP=SHR,DSN=KEDAK3
//G.FT02F001 DD UNIT=SYSDA,SPACE=(2000,100),DCB=(BLKSIZE=2000,RECFM=F)
//G.FT03F001 DD UNIT=SYSDA,SPACE=(2008,100),
//         DCB=(RECFM=VBS,BLKSIZE=2008)
//G.FT04F001 DD UNIT=SYSDA,SPACE=(2008,100),
//     DCB=(RECFM=VBS,BLKSIZE=2008)
//G.FT09F001 DD SYSOUT=A,DCB=(RECFM=FB,LRECL=133,BLKSIZE=931)
//G.FT10F001 DD UNIT=SYSDA,DISP=(NEW,DELETE),SPACE=(TRK,30)
//G.SYSUDUMP DD SYSOUT=A
//G.SYSIN DD *
 NO
 GO
  I  /
 SGN  'PU239' 'SGN' FROM=1.E-3  TO=1.0  /
  I /
  SGTR  ,'SGTR'  /
  I /
 SGG 'PU239' , 'SGG', FROM=1.E+3 TO=30.E+5  /
  I /
  SGX2 ,'SGX'  /
  I /
  SGA ,'SGA'  /
  I /
 SGN  ,'SGN'  /
  I  /
 SGTR  ,'SGTR'  /
 I  /
  SGTR 'PU238' 'SGTR' FROM=0.8 TO=5.
   INIT   /
 SGX2       'U 238'   'SGX  '                     FROM=4.00000E+03 TO=_
        1.95000E+06         OUTUNIT=  10      ZUP=T     ZLOW=T
  INIT  /
 SGA1       'U 238'   'SGA'                      FRUM=4.00000E+03 TO=_
        1.50000E+07         OUTUNIT=  10      ZUP=T     ZLOW=T
  INIT  /
 SGN1       'U 238'   'SGN '                      FROM=4.00000E+03 TO=_
        1.50000E+07         OUTUNIT=  10      ZUP=T     ZLOW=T
INIT  /
 ALPHA1       'PU239'   'ALPHA'                        FROM=1.00000E-03 TO=_
        6.75E+2        OUTUNIT=  10     ZUP=T     ZLOW=T
 STOP
/*
//
```

1.4.7 The control input to print the information available from the program

```
//INR048GU JOB (0048,101,P6M1A),LANGNER,CLASS=A,REGION=310K
/*SETUP DEVICE=2314,ID=GFK029
/*SETUP DEVICE=2314,ID=GFK016
//  EXEC FHG,NAME=OP1
//STEPLIB DD UNIT=2314,VOL=SER=GFK029,DSN=INR.CALOP.LOAD,DISP=SHR
//G.FT01F001 DD UNIT=2314,VOL=SER=GFK016,DISP=SHR,DSN=KEDAK3
//G.FT02F001 DD UNIT=SYSDA,SPACE=(2000,100),DCB=(BLKSIZE=2000,RECFM=F)
//G.FT03F001 DD UNIT=SYSDA,SPACE=(2008,100),
//        DCB=(RECFM=VBS,BLKSIZE=2008)
//G.FT04F001 DD UNIT=SYSDA,SPACE=(2008,100),
//    DCB=(RECFM=VBS,BLKSIZE=2008)
//G.FT09F001 DD SYSOUT=A,DCB=(RECFM=FB,LRECL=133,BLKSIZE=931)
//G.FT10F001 DD UNIT=SYSDA,DISP=(NEW,DELETE),SPACE=(TRK,30)
//G.SYSUDUMP DD SYSOUT=A
//G.SYSIN DD *
 NC
 GC
 HELP
 HELP ADD
 HELP DIVIDE
 HELP ETA
 HELP NAME
  HELP SAVE
 HELP MULTIPLY
 HELP SUBTRACT
  HELP INIT
 HELP INPUT
 HELP STOP
 HELP ETA1
 HELP ALPHA1
  HELP ALPHA2
 HELP ETA2
 HELP NUSF1
 HELP SGA1
 HELP SGG1
  HELP SGG2
 HELP SGG3
  HELP SGT1
  HELP SGT1
 HELP SGN1
 HELP SGTR1
 HELP SGX1
 HELP SGX2
  STOP
/*
//
```

1.5 The job control statements for a GO-mode job

```
//INR048TA JOB (0048,101,P6M1A),LANGNER,CLASS=A,REGION=310K
/*SETUP DEVICE=2314,ID=GFK029
/*SETUP DEVICE=2314,ID=GFK016
// EXEC FHG,NAME=OP1
//STEPLIB DD UNIT=2314,VOL=SER=GFK029,DSN=INR.CALCUL.LOAD,DISP=SHR
//G.FT01F001 DD UNIT=2314,VOL=SER=GFK016,DISP=SHR,DSN=KNDF
//G.FT02F001 DD UNIT=SYSDA,SPACE=(2000,100),DCB=(BLKSIZE=2000,RECFM=F)
//G.FT03F001 DD UNIT=SYSDA,SPACE=(2008,100),
//       DCB=(RECFM=VBS,BLKSIZE=2008)
//G.FT04F001 DD UNIT=SYSDA,SPACE=(2008,100),
//    DCB=(RECFM=VBS,BLKSIZE=2008)
//G.FT09F001 DD SYSOUT=A,DCB=(RECFM=FB,LRECL=133,BLKSIZE=931)
//G.FT10F001 DD UNIT=SYSDA,DISP=(NEW,DELETE),SPACE=(TRK,30)
//G.SYSUDUMP DD SYSOUT=A
//G.SYSIN DD *
/*
```

```
//INR048TE JOB (0048,101,P6M1A),LANGNER,CLASS=A,REGION=300K
/*SETUP DEVICE=2314,ID=GFK029
/*SETUP DEVICE=2314,ID=GFK016
// EXEC FHG
//LOAD DD UNIT=2314,VOL=SER=GFK029,DSN=INR.CALOP.LOAD(OP1),DISP=SHR
//G.FT01F001 DD UNIT=2314,VOL=SER=GFK016,DISP=SHR,DSN=KNDF
//G.FT02F001 DD UNIT=SYSDA,SPACE=(2000,100),DCB=(BLKSIZE=2000,RECFM=F
//G.FT03F001 DD UNIT=SYSDA,SPACE=(2008,100),
//       DCB=(RECFM=VBS,BLKSIZE=2008)
//G.FT04F001 DD UNIT=SYSDA,SPACE=(2008,100),
//    DCB=(RECFM=VBS,BLKSIZE=2008)
//G.FT09F001 DD SYSOUT=A,DCB=(RECFM=FB,LRECL=133,BLKSIZE=931)
//G.FT10F001 DD UNIT=2314,DSN=INR.LAN.PU239,DISP=(NEW,KEEP),
//       VOL=SER=GFK016,SPACE=(TRK,30)
//G.FT11F001 DD UNIT=2314,VOL=SER=GFK016,DSN=INR.LAN.PLUTO,
//     DISP=(OLD,DELETE)
//G.SYSUDUMP DD SYSOUT=A
//G.SYSIN DD *
```

1.6 The output of CALCUL

There are two different output types of CALCUL:

a) List output for checking purposes and error messages:

   1. The control input list
   2.1 NOPRINT was specified: short output is printed by ARITHO
   2.2 PRINT was specified: 1. and 2.1 is printed and additionally the lists of the current result of each single operation, of the data to be named or saved, and of the data edited for KEMA.

b) The data processed and calculated by CALCUL are written in a dataset edited in the KEMA-input-format for later use to update the KEDAK-library.

1.6.1 List output for checking purposes

The user may produce a listed output for checking purposes if desired. He may exercise the control over this print out by the keywords PRINT or NOPRINT. The standard option is NOPRINT.

The keyword PRINT = 6 must be entered for each operation where the complete control output is requested.

Then a "print out of data currently stored as result and the total number of data points available" is edited after the performed operation. If NOPRINT was entered, the control input list only is printed and the messages from ARITHO: the number of names, the names of the reaction type, and the lower and upper limit of the energy range processed.

For each SAVE command entered, the entry table of the temporary direct access data set DADS2 is listed additionally:

MAT	-	material (isotope) name	
TYP	-	reaction type name	
NNAM	-	the number of names	
EXC	-	third name (e.g. excitation energy)	
EMIN	-	lower	energy limit
EMAX	-	upper	energy limit
IR	-	entry count	

NP	-	the number of data points for each entry
KENN	-	= 0 indicates data to be stored for later use in CALCUL only (NAME-command) = 10 (or > 0) indicates data to be also edited in the KEMA-input-format (SAVE-command) on a data set with the data set reference number equal to KENN.

The current result of the single operations is not printed for commands to calculate composed cross sections. Only the final result is listed completely, if PRINT = 6 was specified for these commands.

1.6.2 The output of the set of data processed in a format suitable for KEMA

The output is edited in the module OUTPUT. The data are sorted by OUTPUT in KEDAK-order. The data to be written in KEMA-input-format on the output data set with the data set reference number 10 (≙ KENN ≙ OUTUNIT) are specified to the program by the SAVE-command. The user may alter the default value = 10 by an input for the keyword OUTUNIT of the SAVE-command. A DD-statement must be supplied at the job control statements for CALCUL, e.g.

```
//G.FT10F001 DD UNIT=2314,VOL=SER=GFK050,SPACE=(TRK,30),
//     DISP=(NEW,KEEP),DSN=INR.GOEL.CALCUL
```

The amount of space in the SPACE parameter depends on the number of calculated data points and the length of the track. If PRINT = 6 was specified in the SAVE-command, the complete output written is listed also. If NOPRINT was specified, the data values are not written, only the type of records and the data type names e.g.

```
DROPS       PU239         SGX
           FROM=1.E+3     TO=3.E+4
```

or

```
ADD         PU239         SGX
           FIRST PAIR:       1.E+3     1.241E+1
           LAST PAIR:        3.E+4     2.445E+0
```

are listed.

2. Programmer's guide - detailed description of subroutines, labeled common blocks, work areas and temporary auxiliary data sets

2.1 The control module

2.1.1 Function of the control module

The control module for CALCUL consists of the main program and the following subroutines:

INIT1	- to initialize data set parameters
OPDEF	- the control routine of the module OPDEF (see 2.2.1)
INQ	- to set the operation mode: GO or WAIT
DDCHK	- to test for DD-cards and attach the KEDAK library
DEFI	- to define the direct access dataset DADS2 (see reference 3)
GETOP	- to call the module PROCINP (see 2.3)
NAMIN	- error correction of the input for the positional operands
TESTOP	- to test the operation to be performed
FILLTP	- to provide from KEDAK library the list of reaction types available for the isotope and serviceable for CALCUL
SPACE2	- to handle the dynamic storage allocation (see 2.7.2)
CROSEC	- to call the formula calculation module CROSSEC (see 2.5.1)
INPUT	- to read data from an external source
ARITHO	- the control routine for CALCPAC (see 2.4.1)
EDIT,CRECT,PRIDAT	- the control routines of the module DATAMAN (see 2.6)
EXIT	- the control routine of the OUTPUT module (see 2.7.1)
XXSIMU,XXOUT,COCARD	- to construct control input in card-image-format

The subroutines INIT1, INQ, DDCHK, TESTOP, FILLTP, INPUT, XXSIMU, XXOUT, COCARD, the common /INOUT/ and /PARM/ are described in the following. The control routines for the modules:

OPDEF	- operation code definition
PROCINP	- control input processing
CROSSEC	- cross section formula calculation
CALCPAC	- basic arithmetic operations
DATAMAN	- data management of the auxiliary data sets
OUTPUT	- edition of the output in KEMA-input-format

are described together with the particular modules.

2.1.2 Initialization subroutines

The subroutine INIT1, entry INIT2

INIT1 serves for initialization of the dataset reference numbers (common /INOUT/) and characteristics (common /DA1/) of the datasets used in CALCUL and of the parameters of processed data (common /PARM/) at program start. INIT2 is for repeated initialization (in case of a new reaction type to be calculated) of parameters in the common /DA1/ and /PARM/

The parameters in the common /INOUT/ and /PARM/ and their initial values are listed below.

The common /INOUT/

```
COMMON/INOUT/KOUT,KIN,KED,KDA,K2W1,K2W2,KTOUT,KTAPE,KINOUT
```

The parameters provided in INOUT are initialized by the subroutine INIT1.

KOUT	= 6-system output unit number
KIN	= 5-system input unit number
KED	= 1-dataset reference number of the KEDAK-library
KDA	= 2-dataset reference number for the temporary direct access dataset (DADS2) for CALCUL
K2W1	= 3 } data set reference numbers of the auxiliary workarea for
K2W2	= 4 } the current result in CALCPAC
KTOUT	= KOUT - dataset reference number for the output on the terminal
KTAPE	= 10 - dataset reference number for the output of CALCUL (evaluated data)
KINOUT	= 9 - dataset reference number for card image input produced in the WAIT-mode at the terminal

The common /PARM/

The common /PARM/ retains input parameters for the processed data type. The parameters in /PARM/ are initialized in INIT1/INIT2, and modified in GETOP/READOP by parameters from the input.

COMMON /PARM/ EXTMS,LGO,LKED,NAMZ,NAMES(4),EA,EB,Z1,Z2,NOP,NEW,P,C,XCON

arguments	initial values	
EXTMS	= .FALSE.	
LGO	= .TRUE.	- GO- or WAIT-mode
LKED	= .FALSE.	- data are to be retrieved from KEDAK, if .TRUE.
NEW	= .TRUE.	- start of the calculation for a reaction type

The logical data above are initialized by INIT1.

NAMZ		- the number of names of the processed data type (from input)
NAMES(1) NAMES(2) NAMES(4)	= BLANK	- the names of the processed data type, initialized in INIT2, modified by the input
NAMES(3)	= 0.	
EA	= 0.	lower energy limits for the processed energy range
EB	= 0.	upper energy limits for the processed energy range
Z1	= .TRUE.	$\hat{=}$ ZLOW,ZERO see description of MULT1 , CALCC
Z2	= .TRUE.	$\hat{=}$ ZUP, ZERO1 see description of MULT1 , CALCC
NOP	= 1	- the number of the operation code (1 $\hat{=}$ADD)
P	= .FALSE	$\hat{=}$ NOPRINT
C	= .FALSE.	- no constant
XCON	= 0.	value of the constant for which the operation is to be performed

The initial values are assigned in INIT2 to the parameters and modified by the input data in READOP. The common /PARM/ is used in the subroutines MAIN,GETOP,INIT1,ARITHO,DROREC,CRECT,LIMPR,TESTOP.

The subroutine INQ

The subroutine INQ sets the operation mode: GO or WAIT and provides input description.

GO-MODE: each operation code entered (command) is executed immediately (foreground or background job).

WAIT-MODE: for each operation code entered output cards are produced, which may serve as input to a background (GO-mode) job.
In both cases syntax checks are performed and if a DD-card describing KEDAK (LKED=.TRUE.) has been supplied, checks on availability of data etc. are performed in the WAIT-mode also. The WAIT-mode is only defined for foreground jobs.

Subroutine DDCHK

The subroutine DDCHK checks with the aid of the subroutine DDTEST (see reference 4) if DD-cards are available for the following datasets:

FT01F001 - The KEDAK-library. The LDFOPN for the KEDAK-file is performed in DDCHK. Missing DD-card for the KEDAK-library causes program stop in the GO-mode. In the WAIT-mode the message is printed: No DD-card for FT01F001 has been supplied to define the KEDAK-library. No extended tests will be performed.

FT09F001 - card-image-format
input generated in XXSIMU (is checked in the WAIT-mode only)
(DCB = (RECFM=FB,BLKSIZE=800,LRECL=80)

FT02F001 - the temporary direct access dataset DADS2.
(DCB=(RECFM=F,BLKSIZE=2000)) or in a foreground job:
(BLOCK(2000) SPACE(100))

$\left.\begin{array}{l}\text{FT03F001}\\ \text{FT04F001}\end{array}\right\}$ sequential data sets for the work area in CALCPAC (DCB=(RECFM=VBS,BLKSIZE=2008))

Missing DD-card causes a printout of a message and stop of the program.

Subroutine NAMIN

The subroutine NAMIN is called, if an error occured during processing of positional operands of the operation code.
The call:

CALL NAMIN(N,NAM)

N	= 1 error in processing material name
	= 2 error in processing reaction type name
	= 3 no material name available
	= 4 no type name available
NAM	a real*8 variable to return the name prompted (WAIT-mode)

In batch processing (background job) an error message is printed.
In the WAIT-mode the erroneous or missing name is prompted at the terminal.

Subroutine TESTOP (NRET)

The subroutine TESTOP checks, whether the operation to be performed allows positional operands; if not, operations with a constant will be performed.

The argument:

NRET	-	returncode
	=	0 no error
	=	1 error: no ETA calculation performed since no result from previous operation available

For operation codes with positional parameters the data type names are read in by NAMIN, if necessary. No positional operands are allowed for all commands, if an operation with a constant is performed (see 2.4.2 - subroutine OPERCC). Then the data type names from the previous operation are used.

Subroutine FILLTP

The subroutine FILLTP selects from the data types available on the KEDAK-file the single valued energy dependent types according to the list in TYPS and stores them into the common /TPFILL/.

The call:

```
CALL FILLTP(MAT,NR)
```

MAT	-	name of the isotope for which the data types are to be selected
NR	-	returncode
	= 0	- error
	= 1	- no error

A call to the subroutine LDFITN provides the list of data types available on the KEDAK-file for the requested isotope.

The names of data types listed in TYPS are:

SGT,SGN,SGX,SGI,SGIZC,SGIZ,SG2N,SG3N,SGIA,SGI3A,SG2NA,SG3NA,SGIP,SGNI,SGA, SGF,SGG,SGP,SGD,SGH3,SGALP,SG2HE,SGTR,MUEL,ETA,ALPHA,NUE,NUEP,CHIF, CHIFD, SGHE3.

2.1.3 The subroutines INPUT, INTERP to read the data from an external source and store them into DADS2

The subroutine INPUT

The subroutine INPUT is provided to process data from an external source, i.e. other than the KEDAK-library, for calculation of the cross section data.

These data are assumed as formatted records, each including one energy value and the corresponding cross section value.

The format of these records is obligatory as input to the keyword FORMAT of the INPUT command. The INPUT command initiates the processing of the INPUT subroutine. The data are read by the INPUT routine and stored with the aid of the UPDAT (UPDN) routine into the temporary direct access data-set DADS2 for later use in the calculation package and/or editing them without change for the KEDAK-management program.

If necessary the data are interpolated at the limits of the processed energy range (EMIN,EMAX) by the subroutine INTERP.

The description of the INPUT command:

The INPUT command has three positional parameters and six keyword parameters. The positional parameters are:

the isotope name the reaction type name	$\left.\right\}$ alphameric text, up to 8 characters enclosed in apostrophes
the third name	- REAL*4 - floating point number

The keyword parameters are:

FROM	$\hat{=}$ EMIN lower limit of the processed energy range, REAL*4 - floating point number
TO	$\hat{=}$ EMAX upper limit of the processed energy range, REAL*4 - floating point number
FORMAT	alphatext up to 36 characters enclosed in apostrophes (e.g. '(1X,24A1,2E16.5)')

UNIT	integer number specifying the dataset reference number of the external dataset from where the data are to be read
SKIP	integer number that specifies the number of records of the dataset to be skipped before reading. Default value = 0
REWIND	- no input parameter. A rewind is requested for the dataset before processing.

2.1.4 The subroutines to construct the control input in card-image-format

The subroutines: XXSIMU,XXOUT,COCARD

are provided to create the control input lists in card-image-format.

XXSIMU is called to print the entered input for checking or for construction of a complete control input list, if the control input is to be created interactive in a foreground job in the WAIT-mode and collected on a data set with the reference number 9 for later use in a background job.

The function of the subroutines:

XXSIMU	- to prepare the control input list
XXOUT	- to create card-image-format
COCARD	- to concatenate output lines

Subroutine XXSIMU

The subroutine XXSIMU provides card image output of the current control parameters in form of a control input list readable by the control input processing program.

The call:

CALL XXSIMU(KPUN,MOPT,OP,IADOP,IG,V,KEY,IADKEY,IH)

The arguments:

KPUN	dataset reference number for the card image output data to be punched
MOPT	number of the operation code for which a control input list is produced

OP	≙ G(1) block of the operation code names in the G-array
IADOP	≙ G(2*MAXOP+1) address of the block in the G-array retaining the data for each operation code
IG	≙ G(1) address of the G-array
V	≙ COMMON /OPAR/ retaining the current values of the parameters for the positional and keyword operands
KEY	≙ H(1) block in the H-array retaining the keyword names
IADKEY	≙ H(2*MAXKEY+1) address of the block in the H-array retaining the data describing each keyword
IH	≙ H(1) address of the H-array

The subroutine XXSIMU is called, if the control input is produced in the WAIT-mode at the terminal, or to print the entered input for checking purposes.

The input from the keyboard (WAIT-mode) is completed by XXSIMU with the default values (if any) from the common /OPAR/ and printed for verification. The prepared control input list can be deleted depressing the attention key or accepted depressing the return key. Then the created control input list is added to the card-image-format dataset with the dataset reference number KPUN. This dataset may serve as input for a later background GO-mode job.

Subroutine XXOUT

The subroutine XXOUT prepares the data in the B-array for editing in the card-image-format. XXOUT is called by XXSIMU.

The call:

```
CALL XXOUT(B,LB,NC,LC,LBMAX,V,VAD,TYP, &900)
```

B	- LOGICAL*1-array containing the data to be converted, adjusted and concatenated
LB	- length of the B-array
NC	- number of characters in B
LC	- length of card (≙ 80)

LBMAX	the maximum number of characters in B ($\hat{=}$ 320)
V	address of the common /OPAR/
VAD	address of a storage location in the common /OPAR/ where the value of the positional parameter is stored
TYP	type of the positional parameter (R,I,L,T)
&900	Statement number to continue processing in case of an error return; the error message is printed: input list to long for output and will be truncated.

The subroutine XXOUT calls the subroutine CONVY to convert integer and real data to characters.The subroutine COCARD is called to concatenate output lines.

Subroutine COCARD

The subroutine COCARD concatenates two output lines. An underscore is placed at the end of the first line indicating, that a continuation line is to be expected.

The call:

CALL COCARD(B,LB,NC,LC)

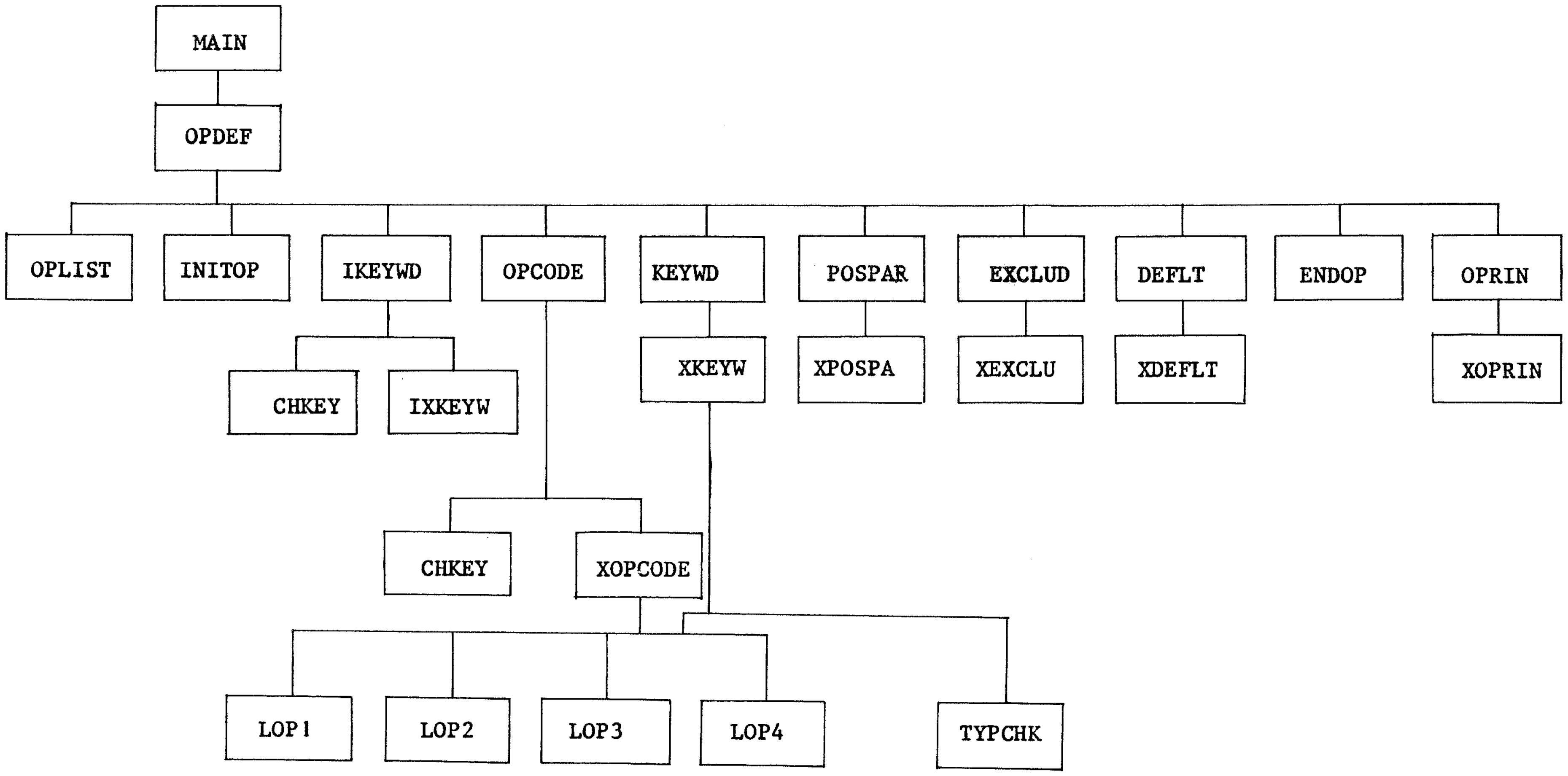
B	array retaining the characters of the input list
LB	current length of the input list
NC	total length of the input list (maximum 320)
LC	length of a card (maximum 80)

The underscore is placed in B(LB+1).

LB is set LB=NC+8

NC is set NC=NC+LC

2.2 Operation code definition package call scheme



2.2 The operation code definition package-function

The operation code definition package consists of the following subroutines:

OPDEF,OPLIST with the entries: INITOP,OPCODE,IKEYWD,POSPAR,EXCLUD,DEFLT, ENDOP,OPRIN,HELP,SIMUL,READOP; CHKEY,XOPCODE,IXKEYW,LOP1,LOP2,LOP3,LOP4, TYPCHK,XPOSPA,XEXCLU,XDEFLT,ENDOP,XOPRIN.

The definition of the different operation codes is performed in the subroutine OPDEF.

2.2.1 The control routine OPDEF

The subroutine OPDEF is used to define the operation code package: the operation codes by a call to OPCODE, the keywords for the operation code by a call to KEYWD, the positional parameters (the isotope and reaction type name) by a call to POSPAR.

The call:

CALL OPDEF(F)

F is the address of an array used to define the G-array (see Table 1), H-array (see Table 2) and the arrays VZ,VI,VFOUND which are structured as the common /OPAR/ (see Table 3)

G(1) $\hat{=}$ F(1) - "G-array" (length: 500 words)
the data for operation code definition are stored in G

H(1) $\hat{=}$ F(501) - "H-array" (length: 500 words)
the data for keyword definition are stored in H

VZ $\hat{=}$ F(1001) - (length: 100 words) array to store the default parameter values (auxiliary storage)

VI $\hat{=}$ F(1101) - (length: 100 words) array to store the initial default parameter values

VFOUND $\hat{=}$ F(1201) - (length: 100 words) array to store the default parameter values from the input

The array addresses and the number of commands and keywords defined for the operation code package are initialized by a call to OPLIST. At present 27 commands (operation codes) and 12 keywords are defined. The operation code package and the package pointers are initialized by a call to INITOP.

The keywords defined are:

FROM,TO,PRINT,NOPRINT,CONST,ZUP,ZLOW,OUTUNIT,SKIP,REWIND,FORMAT,UNIT.

FROM, TO	- the lower and upper limit of the energy range to be processed	
CONST	- a constant to be added, subtracted etc. from the processed data	
ZUP	- zero upper range	if .TRUE. (see also description of CALCC (2.4.2))
ZLOW	- zero lower range	
OUTUNIT	- dataset reference number for the output data of CALCUL. The processed data are written in records which could be read by KEMA (see reference 5)	
REWIND	- are used by the command INPUT to read data from an external source on "UNIT" under "FORMAT"	
SKIP		
FORMAT		
UNIT		

The keywords for the package (name, number of parameters, parameter type, location for the parameter values) are defined by a call to IKEYWD and stored in the common /OPAR/. The default values for a keyword parameter are defined by a call to DEFLT.

Keywords are defined mutually exclusive by a call to EXCLUD. A call to ENDOP provides the utilized length of the G-array and H-array (in the common /OPXX/). A call to OPRIN supplies the print out of the operation code package tables. INITOP,IKEYWD,OPCODE,KEYWD,POSPAR,EXCLUD,DEFLT,ENDOP, OPRIN are entries in the subroutine OPLIST.

The commands defined are: ADD,DIVIDE,ETA,NAME,SAVE,SUBTRACT,MULTIPLY, DELETE,STOP,RESTART,INIT for single operations, INPUT to read external data, ALPHA1,ALPHA2,ETA1,ETA2,NUSF1,SGA1,SGG1,SGG2,SGG3,SGI1,SGN1,SGT1, SGTR1,SGX1,SGX2 - for composed operations. The commands for composed operations are used for calculation of the cross sections for a reaction type with the aid of the operationswhich are controlled by the single commands; but the control is performed in this case by a program that replaces the single commands and the control input for each command (see CROSSEC).

2.2.2 The definition entries comprised in the subroutine OPLIST and the common /OPXX/

The subroutine OPLIST comprises the entries for the operation code package definition.

The call:

```
CALL OPLIST(G,LG,H,LH,VZ,VI,VFOUND,JOP,JOP2,MAXOP,MAXKEY)
```

G	- array to retain the data for the defined operation codes
LG	- length of the G-array
H	- array to retain the data for the defined keywords
LH	- length of the H-array
VZ, VI	- arrays of the same length as the array VAL (common /OPAR/) used to store the default values of the keywords. VZ-auxiliary storage, VI - for initial values
VFOUND	- array to store the keyword parameters supplied by the control input
JOP	- ambiguity control parameter = 0 - abbreviations of the operation code are not allowed in the control input list = 1 - the abbreviation must be unique (e.g. SUBTRACT≡SUBT, STOP≡ST, etc.) =+2 - the length of the operation code is not checked (e.g. SUBTRACT≙SUBT, is also valid) =-1 - the abbreviations are not necessary unique (e.g. SU≡SUB≡SUBT≡SUBTRACT are all valid) but the length of the code is checked and the character string must be clear-cut
JOP2	- default control parameter for positional operands: = 0 the current value of the positional parameter is default < 0 no default values available for the positional parameters, the input of positional parameters is obligatory in any case
MAXOP	- specifies the maximum number of commands for this package
MAXKEY	- maximum number of keywords defined for this package

The entries included in OPLIST are:

INITOP	- to initialize package pointers
IKEYWD	- to define the keyword operands
OPCODE	- to define operation code names
KEYWD	- to assign a keyword name as keyword operand to the operation code defined last
POSPAR	- to define the positional parameters for the operation code defined last
EXCLUD	- to make keywords mutually exclusive
DEFLT	- to define default control parameters and default values for the parameter of a given keyword
ENDOP	- to retrieve the current pointer values for the G-array and H-array: LG and LH and to terminate operation code definition (final pointer setting)
OPRIN	- to print the operation code definition tables
HELP	- to display HELP information
SIMUL	- to produce card image input
READOP	- to read a control input list

Description of the argument lists of the entries in OPLIST.

ENTRY INITOP   no arguments

ENTRY IKEYWD(NAME,IADR,NPAR,PARTYP,PARAD)

NAME	REAL*8 - keyword name to be defined
IADR	address of a location in the common /OPAR/ where a flag is set .TRUE. or .FALSE. for the defined key
NPAR	the number of parameters for the key
PARTYP	array to retain the types of the parameters
PARAD	address of a location in the common /OPAR/ where the values of the keyword parameters are to be stored

ENTRY OPCODE(NAME)

NAME	REAL*8 the name of the operation code to be defined

ENTRY DEFLT(KONTR,NAME,V)

KONTR	default control parameter
NAME	REAL*8 keyword name for which the default value is to be defined
V	default value(s)

ENTRY ENDOP(LG,LH,NR)

LG	length of the used G-array
LH	length of the used H-array
NR	error return code: = 0 no error ≠ 0 the number of errors that occurred at definition of operation codes and keywords

ENTRY KEYWD(NAME)

NAME	REAL*8 - keyword name to be assigned to the operation code defined last

ENTRY POSPAR(TYP,IADR)

TYP	type of the positional parameter for the operation code defined last
IADR	address of a location in the common /OPAR/ where the value for the positional operand is to be stored

ENTRY EXCLUD(N,LIST)

N	number of keywords in LIST
LIST	the list of keywords to be defined mutually exclusive

ENTRY OPRIN - no arguments

ENTRY HELP - no arguments

ENTRY SIMUL(KPUN)

KPUN	unit number e.g. dataset reference number of created card-image-format control input which may be punched on cards

ENTRY READOP(NOPT,NPOS)

NOPT	the number of the operation code read in from the control input list
NPOS	number of positional parameters for this operation code

OPLIST is called to initialize the addresses of the G- and H-array, the ambiguity control parameter, and the default control parameter. A call to OPLIST must be performed before the call to any other entry in OPLIST. After that INITOP is to be called. All keywordsfor the package are defined by a call to IKEYWD before the first call to OPCODE.

OPLIST initializes the length of the arrays: H,G,VZ,VI,VFOUND

The common /OPXX/

The common /OPXX/ is used in the subroutines of the operation code definition package and transfers the parameters used at the definition of operation codes and keywords into the G-array and H-array respectively.

```
COMMON /OPXX/ L1,L2,IOP,MAXOP,MAXKEY,NOP,NKEY,LIG,LIH,EX,IEXCL,IFI,IOP2
```

L1	length of the G-array
L2	length of the H-array
IOP	ambiguity control parameter for abbreviations of the operation code names and keyword names = -1 the abbreviations are not necessary definite, but the length is checked and the character string must be clear-cut (e.g. SU,SUB,SUBT,SUBTR, SUBTRA,SUBTRAC≙SUBTRACT are all valid) = 0 abbreviations are not allowed = +1 the abbreviation must be definite (e.g. SUBTRACT≙SUBT SUBMIT≙SUBM STOP≙ST) = +2 the length of the code is not checked (e.g. SUBMIT≙SUBM is also valid)
MAXOP	the maximum number of operation codes which could be defined by the package (27 at present)
MAXKEY	the maximum number of keywords which could be defined (12 at present)
NOP	the number of the operation code defined now
NKEY	number of keywords to be defined for operation, currently defined
LIG	index indicating the occupied length of the G-array
LIH	index indicating the occupied length of the H-array
EX	indicates if there are keyword parameters to be defined mutually exclusive
IEXCL	number of keyword parameters to be defined mutually exclusive
IOP2	default control parameter for the positional operand parameter $\geq$ 0 the current value of the positional operand parameter is default < 0 no default values available, input for the positional operands is obligatory

L1,L2,IOP2,MAXOP,MAXKEY are set in OPLIST; NOP,NKEY,LIG,LIH,EX,IEXCL are initialized in INITOP and modified in the operation code definition package.

2.2.3 The definition subroutines
Subroutine IXKEYW - keyword definition

The subroutine IXKEYW, entry XKEYW performs the definition of the keywords for the operation code package.

The call:

```
CALL IXKEYW(NAME,IADR,NPAR,PARTYP,PARAD,IADOP,KEY,IADKEY,IH,IG)

CALL XKEYW(NAME,IADOP,KEY,IADKEY,IH,IG)
```

The arguments:

NAME	- literal constant specifying the keyword name
IADR	- location in the common /OPAR/ where a flag is set to indicate, if this key was found in input for this command or not
NPAR	- the number of parameters for the key NAME
PARTYP	- type of the parameters (R,I,L,T) (real,integer,logical,text) (array)
PARAD	- location in the common /OPAR/ where the input value of the keyword parameter is to be stored (array)
IADOP	- address of the block in the G-array (see Table 1) where the pointers to the data field for the operation codes are stored
KEY	- address of the block in the H-array where the keyword names are stored (see Table 2)
IH	- address of the H-array-data for keyword definition
IG	- address of the G-array-data for operation code definition

The entry XKEYW is used to assign the keyword name as a keyword operand to the operation code defined last. The keyword name, the length of the name, the length of the abbreviation, the number of keyword parameters and the type of the parameters are determined and stored into the H-array (see Table 2). The keyword definition data stored into the common /OPAR/.

Subroutine XOPCOD - operation code definition

The subroutine XOPCOD is called by OPCODE to define the operation code NAME and store the data for this code into the G-array (see Table 1).

The call:

```
CALL XOPCOD(NAME,OP,IADOP,IG)
```

NAME	- the name of the operation code to be defined as a character string (literal constant)
OP	- address of the block in the G-array, where the operation code names are stored
IADOP	- address of the block of data for the operation code NAME
IG	- address of the G-array

Subroutine CHKEY (NAME,*)

The subroutine CHKEY is called by IKEYWD and OPCODE to check if the keyword name or operation code name (NAME) to be defined is valid. An error message is printed if a syntax error occured in NAME: the first character is a digit, or the code is too long, or contains invalid characters.

Subroutine LOP1

LOP1 is called to specify the abbreviation length of the code name to be defined, if the ambiguity control parameter is greater than zero, e.g. the abbreviation must be unique. If the code name corresponds to a name already defined, an error return is initiated. LOP1 defines the minimum abbreviation length of the code just processed, and alters the abbreviation length of any similar code defined before with an equal abbreviation.

The call:

```
CALL LOP1(NAME,LNAM,OP,IAD,IG,NK,LK,&80)
```

NAME	- literal constant, the name of the code to be defined
LNAM	- the number of characters in NAME
OP	- array retaining the code names

IAD	- addresses of the data block for each code
IG	- address of the G-array and H-array respectively
NK	- the number of code names already defined
LK	- the number of characters in the valid abbreviation of the code
&80	- statement number to continue processing in case of an error return

Subroutine LOP2

LOP2 is called, if the ambiguity control parameter for the abbreviations of the code names is less than zero, and returns the minimum length of the abbreviation.

The call:

```
CALL LOP2(NAME,LNAM,OP,NK,LK,&80)
```

The arguments:

NAME	- literal constant, the name of the code to be defined
LNAM	- the number of characters in the code name
OP	- array retaining the code names defined
NK	- number of code names already defined
LK	- number of characters in the abbreviation
&80	- statement number to continue processing in case of an error return

Subroutine LOP3

The subroutine LOP3 is called, if the ambiguity parameter for the code name abbreviations is equal zero, e.g. no abbreviation is allowed. LOP3 checks, whether a name similar to the code name to be defined was already defined or not.

The call:

```
CALL LOP3(NAME,OP,NK,LK,&80)
```

NAME	- literal constant, the name of the code to be defined

OP	- array retaining the code names defined
NK	- the number of defined code names
LK	- (in case of an error return) the number of the code similar to NAME
&80	- statement number to continue processing in case of an error return

Subroutine LOP4(NAME,LNAM)

The subroutine LOP4 is called to evaluate the number of characters (LNAM) in the literal variable NAME which specifies the code name to be defined.

Subroutine TYPCHK(TYP,IA,&80)

TYPCHK is called to check, whether a valid type identifier is available for the keyword parameter of the key to be defined.

The arguments:

TYP	- parameter type to be checked
IA	- address of the parameter value which type is to be checked
&80	- statement number to continue processing in case of an error return

Subroutine XPOSPA - definition of positional parameters

The subroutine XPOSPA is called by POSPAR to define the positional parameters for the operation code defined last.

The call:

CALL XPOSPA(TYP,IADR,IADOP,IG)

TYP	- type of the positional parameter (R,TR)
IADR	- pointer to the storage location in the common /OPAR/ where the value of the positional parameter is to be stored
IADOP	- address of the data block of the last defined operation code in the G-array
IG	- address of the G-array

IADR and TYP are stored into the G-array, IADR in ADPOS and TYP in TYPOS.

Subroutine XEXCLU

The subroutine XEXCLU is called by EXCLUD to make keywords in LIST mutually exclusive.

The call:

```
CALL XEXCLU(N,LIST,KEY,IADKEY,IH,V,VI)
```

The arguments:

N	- the number of keywords to be mutually exclusive
LIST	- array retaining the keyword names to be exclusive
KEY	- address of the block in the H-array where the keyword names of the operation package are stored
IADKEY	- address of the block in the H-array where the data for keyword definition are stored
IH	- address of the H-array
V	- address of the common /OPAR/
VI	- address of the array (structured as /OPAR/) retaining the default parameters initialized at the program start

Subroutine XDEFLT

The subroutine XDEFLT is called by DEFLT(KONTR,NAME,V) to define the default control parameter (KONTR) and the default values (V) for the keyword operand NAME.

The call:

```
CALL XDEFLT(KONTR,NAME,VALI,VAL,V,KEY,IADKEY,IH)
```

KONTR - default control parameter:

= 1	- the default parameter is initialized by the program, but the current value is default
= 2	- the initial value set up at command definition time is default
= 0	- no defaults available
< 0	- no default values initialized, input is obligatory

NAME	- name of the keyword for which the default parameter value is to be defined
VALI	- array to store the default parameters initialized at program start
VAL	- address of the common /OPAR/ where the parameter values from input are stored
V	- array retaining the default values for the initialization
KEY	- address of the block in the H-array retaining the keyword names
IADKEY	- address of the block in the H-array retaining the addresses of the definition data for each key
IH	- address of the H-array

Subroutine XOPRIN

The subroutine XOPRIN is called by OPRIN to print the operation code package tables.

The call:

```
CALL XOPRIN(OP,IADOP,IG,KEY,IADKEY,IH)
```

The arguments:

OP	- address of the block in the G-array where the operation code names are stored
IADOP	- array retaining the addresses of the data blocks for each operation code
IG	- address of the G-array
KEY	- address of the block with the keyword names in the H-array
IADKEY	- array retaining the addresses of the data block for each key
IH	- address of the H-array

2.2.4 Table 1: Structure of the G-array
(retains the data for operation code definition)

L ≙ logical, R ≙ real, I ≙ integer, T ≙ text

address	contents	comment
1	OPCODE 1	operation code name =
3	OPCODE 2	= command (real*8 word)
2*NOP-1	: OPCODE NOP	
2*MAXOP	:	
2*MAXOP+1	IADOP(1)	address of the array
	IADOP(2)	retaining the data for
3*MAXOP	:	OPCODE (1,2...)
3*MAXOP+1 (=IADOP(1))	LMAX	length of the operation code name
IADOP(1)+1	LMIN	length of the valid (minimum) abbreviation
IADOP(1)+2	NKEY	number of keywordsof OPCODE 1
IADOP(1)+3	NPOS	number of positional parameters for OPCODE 1
IADOP(1)+4	IADKEY(1)	address of an array in H
IADOP(1)+5	IADKEY(2)	where the data for key number 1 (2,...NKEY)
IADOP(1)+3+NKEY	: IADKEY(NKEY)	are stored
IADOP(1)+NKEY+4	IADPOS(1)	Pointer to a storage location in the common /OPAR/ where the value of the positional parameter is to be stored
	TYPOS(1)	Type of the positional parameter (R,I,L,T)
+(2*NPOS)-1 IADOP(1)+NKEY+3	: IADPOS(NPOS)	see IADPOS (1)
+2*NPOS IADOP(1)+NKEY+3	TYPOS(NPOS)	see TYPOS (1)
IADOP(2)	LMAX	see above
IADOP(2)+1	LMIN	see above

Table 2: Structure of the H-array
(retains the data for definition of keyword parameters)

L ≙ logical, R ≙ real, I ≙ integer, T ≙ text

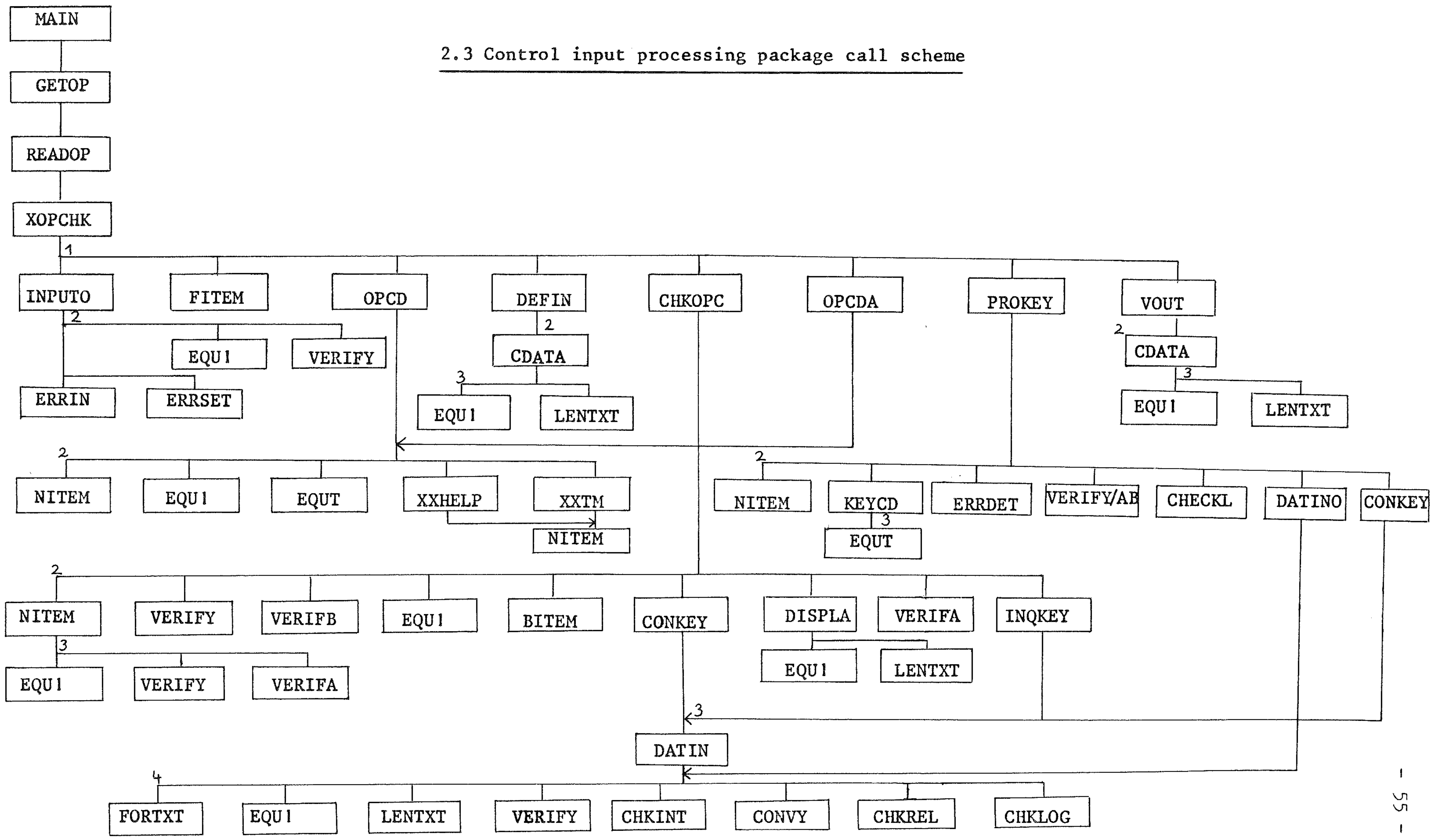
address	contents	comment
1	KEYNAM(1)	name of the keyword
2	KEYNAM(2)	(real*8 variable)
2*MAXKEY	⋮	
2*MAXKEY+1	IAKEYTAB IADKEY(1)	address of the array of data
2*MAXKEY+NKEY	⋮ IADKEY(NKEY)	for each key
IADKEY(1)	LMAXKEY1	maximum length of the keyword name
IADKEY(1)+1	LMINKEY1	minimum length of abbreviation
IADKEY(1)+2	AKEY(IADR)	location in common /OPAR/ to be set .TRUE.(key in input)or .FALSE. respectively
IADKEY(1)+3	IEXCL	Exclusive list address
IADKEY(1)+4	KONTR	default control parameter <0, 0, 1, 2
IADKEY(1)+5	NPAR	number of keyword parameters
IADKEY(1)+6	ADPAR(1)	location in common /OPAR/ where the parameter value is to be stored
IADKEY(1)+7	TYPPAR(1)	parameter type (R,I,L,T)
IADKEY(1)+8	ADPAR(2)	see above ADPAR(1)
+ 2*NPAR 2*MAXKEY+NKEY+6	⋮	
IADKEY(2)	LMAXKEY2	see above LMAXKEY1

Table 3: The common /OPAR/

L $\hat{=}$ logical, R $\hat{=}$ real, I $\hat{=}$ integer, T $\hat{=}$ text

address	variable	type	comment
1	LF(FROM)	L	is set .TRUE. or .FALSE.
2	LTO	L	if input is available
3	LP(PRINT)	L	for this keyword parameter
4	LNP(NOPRINT)	L	or not
5	LZUP	L	
6	LZLOW	L	
7	LOUT(OUTUNIT)	L	
8	LC (const)	L	
9	FROM	R	keyword parameter values
10	TO	R	from the input or the
11	IPRINT	I	default values are
12	ZUP	L	stored here
13	ZLOW	L	
14	IOUT	I	
15	CONST	R	CONST has no default value initialized
16	E	R	positional parameter values
17	MAT	R	are stored: E-energy
19	TYP	R	Mat-isotopename, TYP-type name
21	LSKIP	L	is set .TRUE. or .FALSE.
22	LFMT(FORMAT)	L	if input is available
23	LREW(REWIND)	L	or not
24	LUN(UNIT)	L	
25	ISKIP	I	keyword parameter values
26	FMT(FORMAT)	TR	from the input or the default values
35	IUN(UNIT)	I	are stored here. FORMAT has
36	IUM	I	no default value

2.3 Control input processing package call scheme



2.3 The control input processing package-function

The subroutine GETOP is called by the main program to read the control input list for each operation code. A call to READOP provides the control input processing subroutine package to read and process the control input: The positional and the keyword parameter values are moved from card input into the common /OPAR/. The subroutine NAMIN is called by GETOP to indicate and correct erroneous input for material or data type names.

The subroutine of the control input processing package and their functions are:

XOPCHK	- the control routine for the input processing package
INPUTO	- to read the next control input list
NITEM	- to provide the next item from the control input list processed
OPCD	- to provide the number of the operation code entered
DEFIN	- to initialize default values for the positional and keyword parameters of the entered operation code in V (current value) transferring data from the common /OPAR/
CHKOPC	- to test, whether any operands are allowed for the entered operation code
PROKEY	- to provide the values of keyword operands from input
DISPLA	- to display or to prompt the default parameter values
XXTM	- to define a character to simulate attention function
KEYCD	- to return the number of the entered key
CONKEY	- to control the keyword operands in the input list
DATIN	- to process input data items
VERIFY	- to enable error correction and interaction with the program in a foreground job
INQKEY	- to request the input data for a operation code

Auxiliary subroutines and functions are CDATA,CHKLOG,CHKREL,CHECKL,CHKINT, EQU1,LENTXT,FORTXT,EQUT.

2.3.1 The input processing subroutines
Subroutine XOPCHK - the control routine

XOPCHK controls the processing of an input list.

The call:

```
CALL XOPCHK(NOPT,NPOS,VA,V,VI,VF,OP,IADOP,IG,KEY,IADKEY,IH,IFI)
```

NOPT	number of the operation code
NPOS	number of positional operands for this code found in the input list
VA	address of the common /OPAR/
V	array of the same length as the common /OPAR/ to store current values of default parameters
VI	array to store initial values of default parameters
VF	array to store parameter values from input
OP	$\hat{=}$ G(1) address of the block in the G-array where the operation code names are stored
IADOP	$\hat{=}$ G(2*MAXOP+1) array containing the addresses of the data block for each operation code
IG	- address of the G-array
KEY	$\hat{=}$ H(1) array retaining the keyword names
IADKEY	$\hat{=}$ H(2*MAXKEY+1) array retaining the addresses of the data blocks for each keyword
IH	$\hat{=}$ H(1) address of the H-array
IFI	serial count of the processed input lists

The subroutine XOPCHK calls the following subroutines and entries respectively:

INPUTO	- to read the next input list
FITEM	- to initialize pointers in the common /INPUTC/ before retrieving the first item from the input list
OPCD	- to provide the number of the operation code entered
DEFIN	- to initialize default values
CHKOPC	- to check the entered operation code
OPCDA	- to identify an operation code entered repeatedly for correction of an erroneous input item

PROKEY	- to provide keyword parameter values
VOUT	- to move the current values of the parameters from V into the common /OPAR/

<u>Subroutine OPCD</u>

The subroutine OPCD provides the operation code from the input list and returns the number of this code.

The call:

```
CALL OPCD(MOPT,NRET,OP,IADOP,IG,KEY,IADKEY,IH,IQO,VI)
```

MOPT	the number of the operation code
NRET	return code: = 0 - no error = 1 - error in the last input list
OP	$\hat{=}$ G(1) address of the block retaining the operation code names in the G-array
IADOP	$\hat{=}$ G(2*MAXOP+1) address of the array containing the addresses of the blocks of data for each operation code
IG	($\hat{=}$ G(1)) address of the G-array
KEY	($\hat{=}$ H(1)) block in the H-array retaining the keyword names
IADKEY	($\hat{=}$ H(2*MAXKEY+1)) array retaining the addresses of the data block for each key
IH	($\hat{=}$ H(1)) address of the H-array
IQO	counts the question marks entered at the terminal
VI	array retaining the initial default values for the keyword operands

The argument list of the entry OPCDA is the same as for OPCD. OPCDA is called to identify an operation code entered repeatedly to correct an erroneous input item or to prompt (interactive) the input of a valid operation code in the WAIT-mode at the terminal.

Subroutine CHKOPC

The subroutine CHKOPC checks whether operands are allowed and available for the chosen operation code.

The call:

```
CALL CHKOPC(MOPT,NPOS,NRET,OP,IADOP,IG,V,VF,KEY,IADKEY,IH)
```

The arguments:

MOPT	number of the operation code
NPOS	number of positional parameters for this code
NRET	returncode = 1,4,5 - VOUT should be called (the current parameter values are default) = 3 - a call to OPCDA provides the needed information about the operation code = 2 - a new input list must be requested
V	($\hat{=}$ VB or VZ) array retaining the current values of the keyword and positional operands
VF	(VFOUND in OPDEF) array receiving the parameter values from the input list
OP	- array retaining the operation code names
IADOP	- array retaining the addresses of the data block for each operation code
IG	- address of the G-array
KEY	- array retaining the keyword names
IADKEY	- array retaining the addresses of the data block for each keyword
IH	- address of the H-array

CHKOPC prompts also input for the operation code to correct erroneous input. The subroutines NITEM,CONKEY,INQKEY,VERIFA,VERIFB,BITEM,DISPLA are called in CHKOPC.

Subroutine CONKEY

The subroutine CONKEY controls whether all required operands have been encountered in processing the input list.

The call:

```
CALL CONKEY(MOPT,NPOSF,NRET,IADOP,IG,V,VG,KEY,IADKEY,IH)
```

The arguments:

MOPT	the number of the tested operation code
NPOSF	the number of the positional operands found in the input list
NRET	return code
	= 0 no error
	= 2 the input list is deleted (too many errors, attention)

For IADOP,IG,V,VF,KEY,IADKEY,IH see the argument description in XOPCHK.

CONKEY prints a warning in batch processing of a background job, if positional operands are expected for the tested operation code and not found in the input. The input for the requested parameter is prompted at the terminal in a foreground job.

The VF-array is checked for each keyword defined for the operation code, whether data were entered for keyword parameters or not. A warning is printed in a background job. The input is prompted in a foreground job.

Subroutine DEFIN

The subroutine DEFIN initializes the default values for keyword parameters dependent on the default control parameter for the key.

The call:

```
CALL DEFIN(NOPT,VA,VB,VI,VF,IADOP,IADKEY,IG,IH)
```

The arguments:

NOP	number of the operation code for which the parameter values are stored in VA,VI,VB respectively	
VA	address of the common /OPAR/ retaining the default values for keyword parameters	
VB	(VB(1) $\hat{=}$ F(1001))	
VI	(VI(1) $\hat{=}$ F(1001))	see also OPDEF (2.2.1)
VF	(VF(1) $\hat{=}$ F(1201))	
VB	auxiliary storage for the parameter values	
VI	array to store the initial values for the keyword parameters	
VF	array to store the values for keyword parameters from the input	
IADOP	array retaining addresses of the data blocks for each operation code in the G-array	
IADKEY	array retaining the addresses of the data for each keyword (H-array)	
IG	(IG(1) $\hat{=}$ G(1)) address of the G-array	
IH	address of the H-array	

The values of the keyword parameters are moved from the common /OPAR/ into the array VB. If the ENTRY VOUT is called, the values in VB are moved to the common /OPAR/.

Subroutine CDATA

The call:

CALL CDATA(LT,IA,VA,VB)

Data of the length of LT are moved from VB to VA beginning with VA(IA). The subroutine CDATA is called by the subroutine DEFIN.

Subroutine PROKEY

The subroutine PROKEY provides the input for the keyword from the input list.

The call:

CALL PROKEY(NOPT,NRET,IADOP,KEY,IADKEY,IG,IH,V,VF,IPOS)

The arguments:

NOPT	number of the operation code for which the keyword parameters are processed
NRET	returncode: = 0 no error = 1 error
IPOS	number of positional operands available in input

For the arguments: IADOP,KEY,IADKEY,IG,IH,V,VF see the description in CHKOPC.

The subroutines NITEM,CHECKL,DATINO,VERIFA,CONKEY and KEYCD are called by the subroutine PROKEY.

Subroutine KEYCD

The subroutine KEYCD returns the number of the entered key. The number is defined by the index of the keyword name in the H-array.

The call:

CALL KEYCD(MOPT,NRET,KEY,IADOP,IADKEY,IG,IH)

The arguments:

MOPT	number of the operation code for which the keyword is processed
NRET	returns the number of the key

For the arguments KEY,IADOP,IADKEY,IG,IH see the description of arguments in XOPCHK.

2.3.2 The subroutines to process interactive input (real time processing) Subroutine INQKEY

The subroutine INQKEY requests the input data for the operation code OP(MOPT). Positional operands and keyword operands are prompted in a foreground job at the terminal, if not available in the input list or if the user is not versed in input coding and chooses prompting. A warning is printed in batch processing if no defaults are available (background job).

The call:

CALL INQKEY(MOPT,MPOS,NRET,OP,IADOP,IG,V,VF,KEY,IADKEY,IH)

MOPT	number of the requested operation code
MPOS	the number of positional parameters entered for this code
NRET	return code = 0 no error = 2 the processed input list is deleted - too many errors

For OP,IADOP,IG,V,VF,KEY,IADKEY,IH see the description of arguments in XOPCHK.

Subroutine VERIFY

The subroutine VERIFY is used to verify the input list entered at the terminal. VERIFY allows error correction and enables the user to interact with the program in a foreground job.

The input accepted by VERIFY is the question mark, blank, attention key, hyphen and undercore.

The call:

CALL VERIFY(IOP)

IOP = 1	question mark was entered
IOP = 2	input is blank $\hat{=}$ return key was hit
IOP = 3	attention key was hit
IOP = 4	hyphen was entered
IOP = 5	underscore was entered

If the question mark is entered for the first time, the answer is:
Hit the return key for verification.

If attention key was hit, the whole input list is to be deleted. An underscore entered indicates, that the input list is to be extended by additional data. A hyphen entered indicates that the user wishes to supply replacement data.

The ENTRY VERIFA(IOP) is called, if no further information is available.

The ENTRY VERIFB is called to verify the input item prompted.

Subroutine XXTM

The subroutine XXTM is called by OPCD. XXTM defines a character to simulate the attention-function.

The terminal commands NOTERM, TERMINAL or ATTN are accepted by XXTM.
All characters except the underscore, dash or question mark are allowed for attention definition.

The call

CALL XXTM(NR)

NR	returncode	
	= 0	no error
	= 1	error message is printed

EQU1 and NITEM are used in XXTM.

Subroutine DISPLA

DISPLA is called by CHKOPC to display the default values for the operation code: positional operands, keyword operands. DISPLA prompts these parameter values, if no default values are available (WAIT-mode).

The call:

```
CALL DISPLA(MOPT,NRET,OP,IADOP,IG,V,KEY,IADKEY,IH)
```

The arguments:

MOPT	number of the operation code for which the parameter values are to be displayed

For NRET,OP,IADOP,IG,V,KEY,IADKEY,IH see the argument description in XOPCHK (2.3.1).

2.3.3 The subroutines to decode the control input list

Subroutine NITEM

The subroutine NITEM is called to provide the next item from the input list. An item is any data surrounded by separators (comma, blank) or functional separators (apostrophe, slash, equal sign). One data item has a maximum of 38 characters and is passed to the calling subroutine via common /ITEMC/ in the array B.

The ENTRY FITEM is called to initialize pointers IP,IPO in the common /INPUTC/ before retrieving first item.

The ENTRY BITEM is called to backspace one item in the input list.

The ENTRY ERRDET is called to print the whole input list and the item where an error appeared.

The call

CALL NITEM(NRET)

NRET	- returncode
	= 0 no error
	= 1 error in the input item, the input is ignored

The entries VERIFA, VERIFY and the function subroutine EQU1 are called by the subroutine NITEM.

Subroutine DATIN

The subroutine DATIN processes input data items.

The call

CALL DATIN(IA,IIT,V,VF,IDEF,N,NRET,IC)

The arguments:

IA	address of a location in VF where .TRUE. or .FALSE. is stored, if input data are available for the respective parameter or not

IIT	type of the parameter
V	array retaining the current value (= previous input for the requested parameter) of positional and keyword parameters
VF	array to accept data from the input for positional and keyword parameters
IDEF	default control parameter
	≤ 0 no default values available, input is obligatory
	= 1 default values are initialized by the program, but the current value is default (last input)
	= 2 the initialized default values are always default
N	number of the positional parameter for which input is requested or to be processed (for printing purpose)
NRET	returncode
	= 1 no input available, take default value
	= 2 ATTENTION, the item was deleted
	= 3 input error, data not recognizable, the item is ignored
	=10 end of the input list
IC	index for the output text
	1 $\hat{=}$ POS.OP.
	2 $\hat{=}$ PARM.

ENTRY DATINO(IA,IIT,V,VF,NRET,KEY,ERRMS)

The arguments:

For IA,IIT,V,VF,NRET see the argument list description for DATIN.

KEY	the name of the keyword to be processed
ERRMS	.TRUE. - the keyword name is to be printed in the error message
	.FALSE. - an error message for any data type is to be printed without the keyword name

The entry DATIN is called to read and process the input items, the entry DATINO for processing only.

The input data are read in as characters and corresponding to their type, converted to internal representation by the subroutine CONVY.

The entry DATINO is called to check the data type: text data or real data by a call to CHKREL, integer data by a call to CHKINT, and logical data by a call to CHKLOG.

The length of the data item is provided by the function subroutine LENTXT.

The positional parameters and keyword parameters (not available in the control input list of the processed operation code) are prompted (interactive) and processed by DATIN.

DATIN is called by the subroutines PROKEY,INQKEY and CONKEY.

Subroutine CHKREL

The subroutine CHKREL checks if the input item is a real number.

The call:

```
CALL CHKREL(B,LB,NR,C1,IS)
```

B	- array retaining the input item to be checked
LB	- length of the B-array
NR	- = 0 the checked data item is not in a floating point number representation ≠ 0 NR returns the number of digits in the data item
C1	- the number of digits of the mantissa of the real number
IS	- = 7 : length of the number of special characters in the floating point representation of the real number

Subroutine CHKINT (B,LB,NR)

The type and the length of the data item in the B array is checked in the subroutine CHKINT.

The arguments:

B	- array retaining the data item to be checked
LB	- length of the B array
NR	- returns the number of digits in B If the data item in B is not an integer number then NR is returned with a minus sign.

CHKINT is called by DATIN.

Subroutine CHKLOG(B,LB,NR,V)

The subroutine CHKLOG checks if the variable B contains a T or a F. LB must be equal 1, otherwise NR = 0 is returned.

If B = F than NR = 1 and V = .FALSE.
if B = T than NR = 1 and V = .TRUE.
is returned.

CHKLOG is called by DATIN.

Subroutine CHECKL

After a call to CHECKL the length of the input data item stored in B is returned in L.

The call:

CALL CHECKL(LM,B,L)

LM	– the maximum length of the B-array
B	– the array retaining the input data item
L	– the returned length of the data item

<u>Logical function EQU1</u>

The call:

```
CALL EQU1(A,B)
```

EQU1 is called to compare the two characters A and B.
EQU1 is .TRUE. if A equal B, otherwise .FALSE.

<u>Function LENTXT(TXT)</u>

The length of the text TXT is returned:
minimum is 2, maximum is 9.

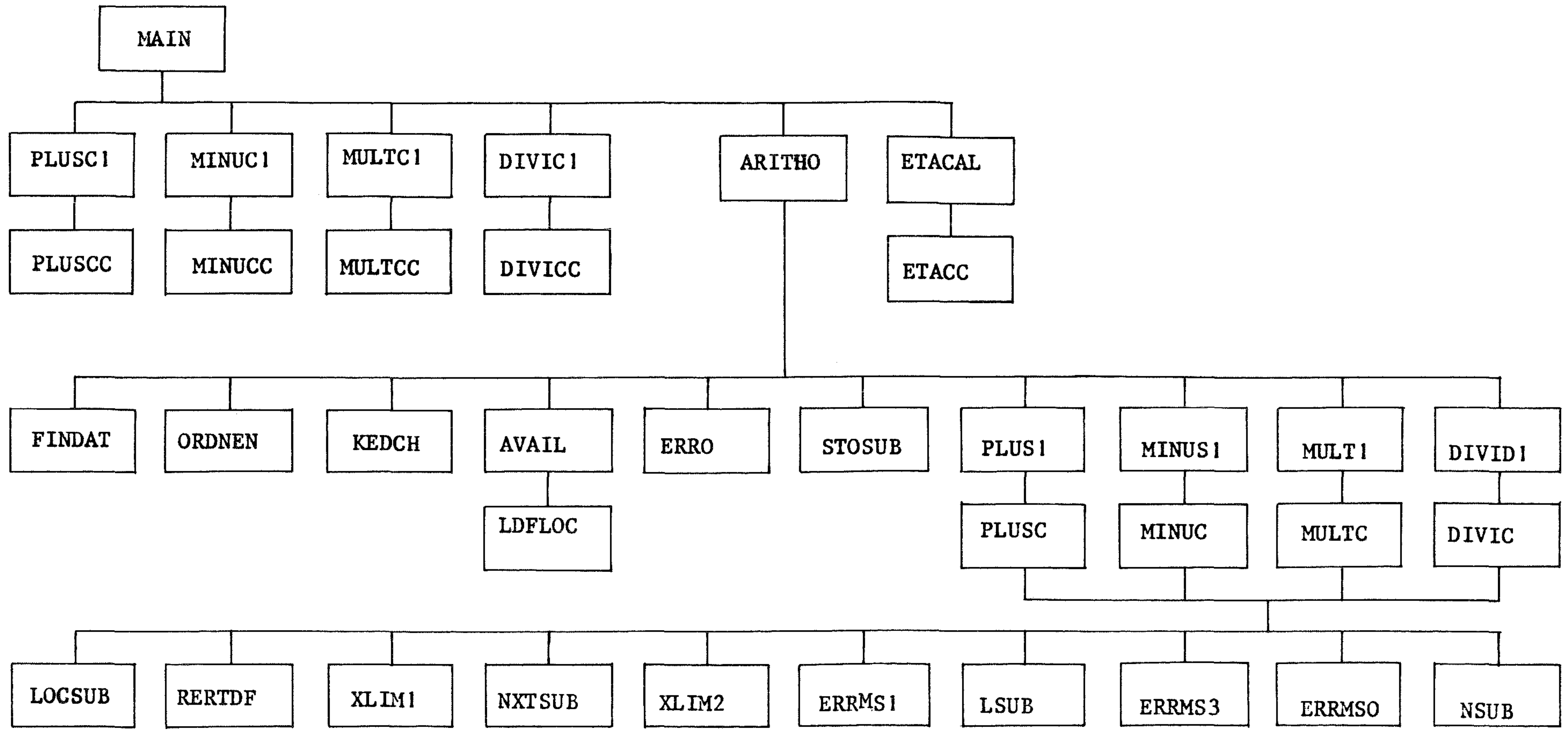
<u>Subroutine FORTXT(LENT,AFT)</u>

The length LENT is returned in AFT as an alphanumeric text.
(LENT $\leq$ 9)

<u>Subroutine EQUT(LA,A,LB,B,LE)</u>

The characters in the A array and B array are compared and the number of equal characters is returned in LE.

2.4 Calculation package CALCPAC - call scheme



2.4 The calculation package CALCPAC-function

CALCUL simulates a desk calculator but operates on functions instead of single numerical values. The basic arithmetic operations: ADD,DIVIDE,MULTIPLY,SUBTRACT are performed in the calculation package CALCPAC.

CALCPAC consists of the following subroutines:

ARITHO	the preparation of the data for the arithmetic operations
CALC1	the interfacing routine to the arithmetic operations in CALCPAC: CALCC,OPERCC,ETACC
CALCC	arithmetic operations on tabulated functions with linear interpolation
OPERCC	to perform the arithmetic operation with a constant
ETACC	to calculate Y = 1/(1+y) for the evaluation of ETA(y=ALPHA)
FINDAT	to check, if requested data are available on the auxiliary dataset DADS2
ORDNEN	to sort the values of a tabulated function in ascending order of the arguments
KEDCH	to find the energy intervals for which the data are to be retrieved from the KEDAK-file
AVAIL	to check, if requested data are available on the KEDAK-file
ERRMSO	to print error messages
LOCXS RETXS	Data retrieval from the KEDAK-file (see reference 6)
LTLOC LTNXT	Data retrieval from DADS2
XLIM1	Interpolation to the energy range limits

The labeled common block /CALCOM/ is used by the CALCPAC subroutines.

2.4.1 The subroutines to prepare the data to be processed

Subroutine ARITHO

The preparation of data for the arithmetic operations performed in CALCPAC is done by the subroutine ARITHO with the aid of the subroutines FINDAT,ORDNEN,KEDCH,ERRO,AVAIL.

The subroutine FINDAT checks which data of the processed type are available for the requested energy range on the temporary direct access dataset with the reference number 2(DADS2).

The arrays EMIN,EMAX are arranged by the subroutine ORDNEN in ascending order of the energy limits.

The subroutine KEDCH checks the energy ranges to find out gaps to be filled with KEDAK data. KEDCH sets for each range a flag in the array IP:
1 - for the data available on DADS2
2 - for the data to be retrieved from the KEDAK-file

The subroutine AVAIL searches, if the requested data type is available on the KEDAK-file.

The names of the proper retrieval routines are transferred to CALCPAC by a call to STOSUB according to the source of data requested. RETXS,REPXS, LOCXS,NXTXS,LDFLOC,LDFNXT are used for the retrieval from the KEDAK-file, and RETXS,REPXS,LOCXS,NXTXS,LTLOC,LTNXT for retrieval from DADS2.

The arithmetic operations are performed corresponding to the command (operation code) from the control input list calling:

PLUS1	for the command ADD
MINUS1	for the command SUBTRACT
MULT1	for the command MULTIPLY
DIVID1	for the command DIVIDE

ERRO is called to initialize the array with the number of errors that occurred for CALCPAC to zero (in common /ERRORC/).

Subroutine FINDAT

The subroutine FINDAT searches the entry table in the common /DA2/, to find out for which intervals of the energy range /FROM,TO/ of the data type specified in NAMES, data are available on the temporary direct access data set DADS2.

The call:

```
CALL FINDAT(NDAT,NDAMAX,NAMZ,NAMES,FROM,TO,E1,E2)
```

NDAT	the total number of intervals on DADS2
NDAMAX	the maximum number of intervals (79)
NAMZ	the number of names for the checked data type
NAMES	the names of the type
FROM,TO	lower and upper limit of the processed energy range
E1,E2	arrays retaining the lower and upper energy limits of the intervals

Subroutine ORDNEN

The subroutine ORDNEN sorts the arrays FELD,WERT, e.g. arguments and function values, in increasing order of the arguments.

The call:

```
CALL ORDNEN(KMAX,FELD,WERT)
```

KMAX	the length of the array to be sorted
FELD	array of the arguments
WERT	array of the function values

Subroutine KEDCH

The subroutine KEDCH states the intervals in /FROM,TO/ which are to be filled with KEDAK-data.

The call:

```
CALL KEDCH(NDAT,NDAMAX,EMIN,EMAX,E1,E2,IP,F,T)
```

The arguments:

NDAT	the number of energy intervals in the energy range (F,T)
NDAMAX	the maximum number of intervals allowed NDAMAX negative indicates an error return = -1 the number of intervals generated is greater than NDAMAX = -2 no data found on KEDAK, and no data available on DADS2
EMIN EMAX	arrays, to transfer the lower and upper energy limits of the intervals to KEDCH and to return the stated new intervals
E1 E2	auxiliary arrays, to retain the energy limits from (EMIN,EMAX), and the additional limits of the intervals, to be filled with KEDAK data. These values are then returned to the calling program in (EMIN,EMAX).
IP	array retaining a flag for each interval: = 2 for KEDAK data = 1 for the data from DADS2, the auxiliary direct access dataset

The subroutine AVAIL

The subroutine AVAIL checks if data of the requested type are available on the KEDAK-file. The common /TPFILL/ is filled in FILLTP. LDFLOC is called to check, whether the requested data type is available on KEDAK.

The call:

```
CALL AVAIL(MAT,TYP,&130)
```

MAT	REAL*8 isotope name of the requested type
TYP	REAL*8 reaction type name of the requested type
&130	error return, if requested data type not available on the KEDAK-file

Subroutine ERRMSO

The subroutine ERRMSO with the entries ERRMS1, ERRMS3 and ERRO provides the error messages for the calculation package.

ERRO initializes the array NERR retaining the error rate for each error. The entries ERRMSO,ERRMS1,ERRMS3 are used to transmit arguments for printout together with error message. For example:

CALL ERRMS1(NR,X)

If ERRMS1 was called with the error number NR = 1 the message is written: warning message: Error 1 occurred when performing requested arithmetic operation. Energy of requested dataset X is below first energy of current result. Action taken: the current result is assumed to be zero at this energy, e.g.

	new result:
for: "+" :	y = s
for: "-" :	y = -s
for: "•" :	y = 0
for: "/" :	y = 0

For NR = 4 the error message is: Energy of current result X is above last energy of requested data set (E,S). Action taken: Depending on ZUP, the values of the requested data type are assumed to be zero at this energy, or current result is left unchanged at this energy, e.g. (after the end of (E,S) data) X = X(L), the next X-value available.

for "+"	: y = y(L)
for "-"	: y = -y(L)
for "•", "/"	: y = 0 if ZUP = .TRUE.

For NR = 6 the message is: Energy EXC/EV/ of requested type is above last energy of current result. Action taken: current result is assumed to be zero at this energy, e.g. X = EXC - the next higher E value available (end of data for (x,y)).

	for "+"	y = s
X = E(I)	for "-"	y = -s
	for "•", "/"	y = 0

Subroutine LTLOC, entry LTNXT

The subroutine LTLOC(LTNXT) handles the data retrieval from DADS2. The retrieval is organised like the retrieval in LDFLOC,LDFNXT (see reference 1), but no OPEN call is required.

The call:

```
CALL LTLOC(NR,NARG,NAMES,Z)
CALL LTNXT(NR,NARG,NAMES,Z)
```

NR	- returncode = 0 - no data found = 1 - requested data are available, stored in Z
NARG(1)	the number of names
NAMES	array retaining the reaction type names
Z	array to receive the data values

The data are read by the aid of the subroutine LTREC (see reference 1)

Subroutine XLIM1, entry XLIM2

The subroutine XLIM1 interpolates the retrieved data to EMIN, the entry XLIM2 to EMAX.

The call:

```
CALL XLIM1(NARG,NAMES,EMIN,EMAX,E,S,NUMS,NR)
CALL XLIM2(NARG,NAMES,EMIN,EMAX,E,S,NUMS,NR,EK,SK)
```

The arguments:

NARG(1)	- the number of names of the processed data type
NAMES	- the names of the processed type
EMIN EMAX }	- energy limits for the processed energy range
(E,S)	- arrays to retain the retrieved data
NUMS	- the number of data in (E,S)

NR	- returncode, from the previous data retrieval routine
EK, SK	- if NR = 2, EK = E(NUMS), SK = S(NUMS)

2.4.2 The subroutines to perform the arithmetic operations

Subroutine CALC1

CALC1 is the interfacing routine to the arithmetic operations in CALCPAC and includes the following entries:

STODSN,INQDSN,STOSUB,INQNUM,STONUM,CAL1IN,ERRSTO,XMGSTO,EQUAL,EQUALC, REMV1,PLUS1,MINUS1,MULT1,DIVID1,PLUSC1,MINUC1,MULTC1,DIVIC1,ETACAL.

CALC1 translates calls to its entries into suitable calls into CALCPAC and sets up the complete argument list.

The call:

CALL CALC1(X,Y,X1,Y1,L1,E,S,L2,NDA,NDB)

X Y X1 Y1	working areas for CALCPAC to store the current and previous result
L1	length of the work arrays
E	array to store the energy values from the KEDAK-file
S	array to store the cross section values from the KEDAK-file
L2	length of the arrays E and S
NDA NDB	dataset reference numbers for the datasets on disk, where the processed data are stored (X,Y,X1,Y1), if the number of data is larger than L1 and data do not fit into main storage

CALC1 provides the addresses of the working area for CALCPAC (i.e. X,Y, X1,Y1,E,S) and is called for new optimization of the storage allocation each time a new cross section is to be calculated.

The entries STODSN and STONUM are used to store data into the common /CALCOM/.

The calls:

CALL STODSN(NDA,NDB)

The dataset reference numbers of the auxiliary datasets are specified and stored in ND1,NDX1

CALL STONUM(NUM)

The number NUM of data points in the current result is stored in NX in the common /CALCOM/.

The entries INQDSN and INQNUM inquire the values of the dataset reference numbers and of the number of data points processed.

The calls:

CALL INQDSN(NDA,NDB)
CALL INQNUM(NUM)

NDA,NDB and NUM respectively are returned by the call. If NUM is greater L1, that is: the number of data points is greater than the length of the working area, the data are stored on anexternal storage on sequential datasets with the reference numbers NDA and NDB.
This storage must be made available to the program by data definition statements for FT03F001,FT04F001 (DCB=(RECFM=VBS,BLKSIZE=2008)).

The entry STOSUB is provided to transfer the names of the data retrieval routines to CALCPAC.

ENTRY STOSUB(LOCSUB,NXTSUB,LSUB,NSUB)

The actual arguments are:

1. (RETXS,REPXS,LTLOC,LTNXT)-to retrieve data from the temporary direct access dataset DADS2.
2. (RETXS,REPX,LDFLOC,LDFNXT)-to retrieve data from the KEDAK-library.

3. (LOCXS,NXTXS,LTLOC,LTNXT)-to retrieve data from DADS2.

4. (LOCXS,NXTXS,LDFLOC,LDFNXT)-to retrieve data from the KEDAK-library

1. and 2. are for the retrieval in a requested energy range /EMIN,EMAX/,
3. and 4. for the retrieval of all data of requested type available on the dataset. STOSUB is called at the beginning of the calculation and the call is repeated whenever new retrieval routines are needed.

The entry CAL1IN initializes flags in the common /CALCOM/ which indicates where the current result is stored (ADR,EXCH,EQC). NOLD - the number of data of the previous result and NX - the number of data from current result are set to zero.

The entry ERRSTO applies to store the error numbers into the NN-array in /CALCOM/.

The call:

CALL ERRSTO(LL)

LL	- array which supplies the error numbers

The function of the entry EQUAL is to store data from the KEDAK-file or the external source unchanged into the array of the current result.

The call:

CALL EQUAL(NAMZ,NAMES,EMIN,EMAX)

NAMZ	- number of names for the data type
NAMES	- the names of the data type
EMIN	- lower } energy limit of the
EMAX	- upper } energy range to be processed

The entry EQUALC allows to enter a constant function value (cross section value) XC for a given energy range /EMIN,EMAX/.

The call:

CALL EQUALC(XC,EMIN,EMAX)

The <u>entry REMV1</u> deletes the values at the energy points 0. and 1. E+10.

<u>The entries to perform arithmetic operations are PLUS1,MINUS1,MULT1, DIVID1.</u>

The calls:

```
CALL PLUS1(NAMZ,NAMES,EMIN,EMAX)
CALL MINUS1(NAMZ,NAMES,EMIN,EMAX)
CALL MULT1(NAMZ,NAMES,EMIN,EMAX,ZERO,ZERO1)
CALL DIVID1(NAMZ,NAMES,EMIN,EMAX,ZERO,ZERO1)
```

For description of arguments see the argument list description for the entry EQUAL. The additional arguments ZERO and ZERO1 are explained here:

ZERO	- indicates that the function values in the interval $X < E$ are to be set equal to zero, if ZERO = .TRUE., otherwise they remain unchanged
ZERO1	- all function values for $X > E$ are to be set equal to zero, if ZERO1 = .TRUE.

<u>Entry PLUS1</u>

In the energy range /EMIN,EMAX/ the operation ADD: "+" is performed

$Y1 = Y+S$	with $Y=0$ for $E < X_{min}$ or $E > X_{max}$
$X1 = X \cup E$	and $S=0$ for $X < E_{min}$ or $X > E_{max}$
$Y \Leftarrow Y1$	(X,Y)-data outside (EMIN,EMAX)
$X \Leftarrow X1$	remain unchanged
	(E,S)-data outside (X_{min},X_{max})
	are received unchanged, X1 is the merged grid:
	$(X \cup E)$ (X1,Y1), (X,Y) - the arrays retaining
	the previous and the current result.
	(E,S)-array retaining the data from KEDAK or
	external source

$Y = S$	if the calculation started
$X = E$	with the call to PLUS1.

Entry MINUS1

The operation performed is SUBTRACT: "-" .

$Y1 = Y - S$	with $Y = 0$ for $E < X_{min}$, $E > X_{max}$
$X1 = X \cup E$	and $S = 0$ for $X > E_{max}$, $X < E_{min}$
$Y \Leftarrow Y1$	
$X \Leftarrow X1$	

$Y = -S$	if the call to MINUS1 was the first
$X = E$	call for the calculation

Entry MULT1

The operation Multiply: "•" is performed:

$Y1 = Y \cdot S$
$X1 = X \cup E$
$Y \Leftarrow Y1$
$X1 \Leftarrow X1$

for $X <$ EMIN and $\begin{cases} \text{ZERO} = T \Rightarrow S = 0 \\ \text{ZERO} = F \Rightarrow S = 1 \end{cases}$

for $X >$ EMAX and $\begin{cases} \text{ZERO1} = T \Rightarrow S = 0 \\ \text{ZERO1} = F \Rightarrow S = 1 \end{cases}$

and $E \notin (X_{min}, X_{max}) \Rightarrow S = 0$

Entry DIVID1

The operation Divide: "/" is performed:

$Y1 = Y/S$
$X1 = X \cup E$
$Y \Leftarrow Y1$
$X \Leftarrow X1$

for $X <$ EMIN and $\begin{cases} \text{ZERO} = T \Rightarrow S = 0 \\ \text{ZERO} = F \Rightarrow S = 1 \end{cases}$

for $X >$ EMAX and $\begin{cases} \text{ZERO1} = T \Rightarrow S = 0 \\ \text{ZERO1} = F \Rightarrow S = 1 \end{cases}$

and $E \notin (X_{min}, X_{max}) \Rightarrow S = 0$
if $S = 0 \Rightarrow Y = 0$

The arithmetic operations between a constant and a cross section value are performed by PLUSC1,MINUC1,MULTC1,DIVIC1. The argument lists for these entries are the same as described for the entry EQUALC. The entry ETACAL is provided for the evaluation of ETA.

Subroutine CALCC

The subroutine CALCC provides the entries PLUSC,MINUC,MULTC,DIVIC to perform the arithmetic operations. In CALCC the data for a requested energy range EMIN,EMAX are retrieved, if necessary interpolated to EMIN,EMAX by the subroutine XLIM1(ENTRY XLIM2), and processed. The arithmetic operations ("+", "-", "/", "•") are performed upon the functions (X,Y) (current result) and (E,S) (retrieved data) with linear interpolation. (X1,Y1) contain the result of the operation on the merged grid $(X \cup E)$.

The interpolation is performed with the function: FUNC(A2,A1,B2,B1,B) = A1+(A2-A1)/(B2-B1)*(B-B1)

The argument lists for the entries MINUC,MULTC,DIVIC are as described for PLUSC, but MULTC and DIVIC have two further arguments:

ZERO	- ($\hat{=}$ ZLOW) = .TRUE. - all function values (current result) in the interval X < E are set equal to zero. ZERO = .FALSE. - the current result is set equal to the previous result. If no data found for requested type on KEDAK and ZERO or ZERO1 is .TRUE., the current result is set equal to zero.
ZERO1	- ($\hat{=}$ ZUP) = .TRUE. - all function values for X > E are set to zero

The call for PLUSC:

```
CALL PLUSC(X,Y,X1,Y1,E,S,NAMZ,NAMES,EMIN,EMAX,LOCSUB,NXTSUB,LSUB,NSUB)
```

The arguments:

X,Y X1,Y1	working areas contain the current result and the previous result by turns
E,S	arrays retaining the retrieved data (from the KEDAK-file or from DADS2)
NAMZ	- the number of names of the processed data type
NAMES	- array retaining the names of the data type
EMIN EMAX	- lower and upper limit of the - energy range to be processed

LOCSUB NXTSUB LSUB NSUB	the external names of the retrieval routines to be transferred to the program (see entry STOSUB in CALC1)

The data to be combined with the current result are retrieved by a call to LOCSUB and NXTSUB:

```
CALL LOCSUB(NARG,NAMES,EMIN,EMAX,E,S,NUMS,LE,NR,LSUB,NSUB)
CALL NXTSUB(NARG,NAMES,EMIN,EMAX,E,S,NUMS,LE,NR,LSUB,NSUB)
```

If EMIN is greater than EMAX, then EMIN, EMAX are ignored by the program, and the retrieval entries without these arguments are used:

```
CALL LOCSUB(NARG,NAMES,E,S,NUM,LZ,NR)
CALL NXTSUB(NARG,NAMES,E,S,NUM,LZ,NR)
```

and all data available for the data type are retrieved.

The arguments:

NARG	NARG(1) = NAMZ: the number of data type names
NAMES	array retaining the names
EMIN,EMAX	if applicable, give the (energy) limits for retrieval. Retrieval starts with the last energy $\leq$ EMIN and will stop with the first energy $\geq$ EMAX. Interpolation to EMIN(EMAX) is done by the subroutine XLIM1(XLIM2).
E,S	are the arrays into which arguments and functional values are stored successively
NUM	is the number of data points transmitted by the current call
LZ	gives the maximum number of data points, that may be stored into X,Y
NR	is a returncode, set by the called routine, the value of which depends on various conditions detailed below
NR=1	last data point for the requested data has been stored in (E,S)
NR=2	LZ data points have been filled into (E,S), without reading the end of the data type. (An entry is provided to continue with retrieval, after a section of LZ data points has been handled.)

NR=3	No data found for the requested type
NR=4	argument of the first data point already greater than EMAX. This data point is transmitted.
NR=5	argument of the last data point for the requested type less than EMIN. This data point is transmitted
NR=10	transmission of data stopped, because upper energy limit was reached.

Returncode 1,2 and 10 indicate normal return, all other returncodes indicate exceptional condition.

The calculation of a cross section value by CALCC is carried out, operating on the current result, and the retrieved data by the proper arithmetic operation, interpolating linearly if necessary. The result of this operation replaces the former current result.

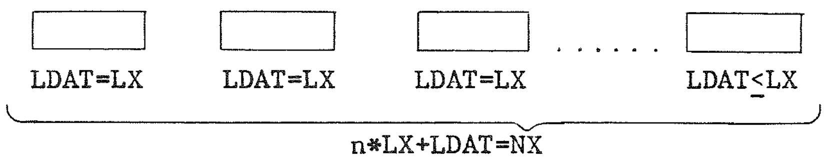
If the number NX of data points processed (X1,Y1) is less than the length LX of the working areas, the data are kept in the main storage. Otherwise an external storage on disk is used, to store the data in datasets with the reference numbers NDX,NDX1.

The data are read from NDX by the statement:

```
READ(NDX)LDAT,(X(I),Y(I),I=1,LDAT)

        LDAT ≤ LX
```

This read statement is repeated, till all NX data are read.



Blocks of data read from NDX.

Each block of data is processed, and written on NDX1:

```
WRITE(NDX1) LX,(X1(I),Y1(I),I=1,LX)
```

Note that the number of data may increase, since energy points may have been inserted.

If a new data type is to be calculated, not using the current result, then a call to CAL1IN (by the main program) has to be performed for initialization of the working areas (see also: ENTRY CAL1IN in CALC1).

The current result is lost, and therefore should be saved first, if required.

The common /CALCOM/

The common /CALCOM/ is used in the calculation package to transmit specifications about the calculated data and datasets used. /CALCOM/ is accessed in the subroutines CALC1,CALCC,ETACC,EQU,OPERCC of CALCPAC, and in EDIT,PRIDAT and CRECT.

```
COMMON /CALCOM/ IRET,LE,LX,NX,NDX,NDX1,EQC,NN(9),XMGS(2),ADR,OLDADR,EXCH,NOLD
```

IRET	returncode from CALCC = 1 data (current result) are kept in main storage in (X,Y) or (X1,Y1), dependent on the value of ADR = 3 the current result is written on NDX or NDX1 respectively = 4 the number of processed data is less or equal to LX (the length of the working area), or the number of data processed is equal to zero
LE	- length of the areas (E,S)
LX	- length of the areas (X,Y), (X1,Y1) LE and LX are set in the SPACE2 subroutine
NX	- the number of data processed. NX is initialized in CAL1IN with 0. As long as $NX \leq LX$ the processed data are kept in main storage, as soon as $NX > LX$ the data are stored on an external storage on disk, in the dataset with the reference number NDX1, and are retrieved from NDX.

NDX NDX1	dataset reference numbers of the auxiliary datasets for the current result
EQC	- is set .TRUE., if the entry EQUALC was entered. EQC is initialized .FALSE. in CAL1IN
NN(9)	- an array to retain the numbers of the error messages. NN is initialized in ERRSTO
XMGS(2)	- not used in CALCUL
ADR	- indicates where the current result is stored: = .TRUE. in (X,Y) = .FALSE. in (X1,Y1)
OLDADR	- indicates where the previous result was stored
EXCH	- indicates, whether the dataset reference number NDX,NDX1 were exchanged, after processing the data type

Subroutine OPERCC

The subroutine OPERCC includes the entries, which perform the arithmetic operations PLUSCC,MINUCC,MULTCC,DIVICC for a constant, and the entries EQUCC and RMVC.

The calls:

```
CALL PLUSCC(X,Y,XC,EMIN,EMAX)
CALL MINUCC(X,Y,XC,EMIN,EMAX)
CALL MULTCC(X,Y,XC,EMIN,EMAX)
CALL DIVICC(X,Y,XC,EMIN,EMAX)
CALL EQUCC(X,Y,XC,EMIN,EMAX)
```

The arguments:

(X,Y)	- the arrays of the data for which the operation has to be performed
XC	- the value of the constant, which is to be combined with the cross section values in the Y-array
EMIN,EMAX	- lower and upper energy limit of the data for which the operation is performed

The entry EQUCC sets Y to a constant either at 0. and 1.E+10 (EMIN > EMAX) or at EMIN,EMAX. The values above EMIN, previously defined, are lost.

The entry RMVC removes the function values (Y) at 0. and 1.E+10 (X), if EQC = .TRUE.

The call:

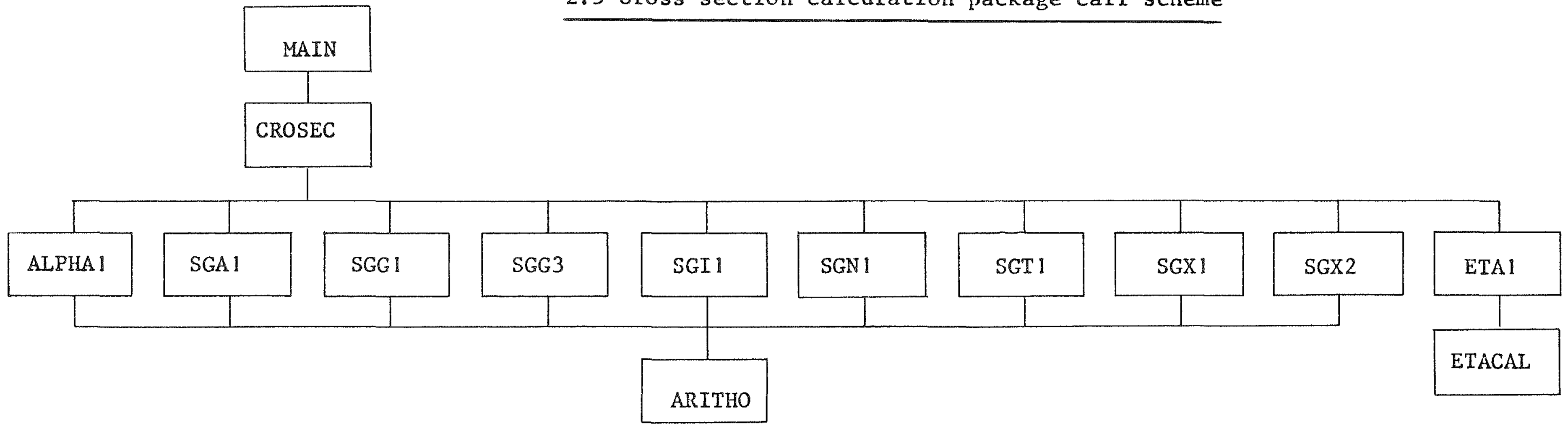
CALL RMVC(X,Y,X1,Y1)

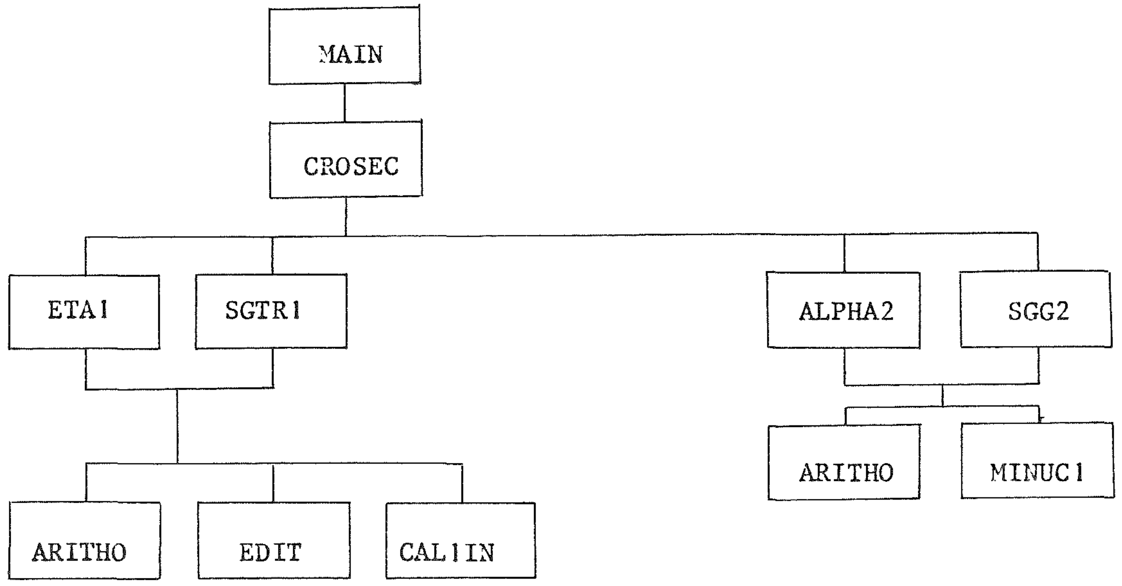
(X,Y)	- arrays of the data before the operation
(X1,Y1)	- arrays of the data after the operation

Subroutine ETACC

The subroutine ETACC performs the operation $Y_* = 1/(1+Y)$ which is needed for the calculation of ETA. Y on the right side of the above formula would stand for ALPHA in this case. For those energies, where Y is equal to zero, no operation is performed, and Y_* remains unchanged.

2.5 Cross section calculation package-call scheme





2.5 The cross section calculation subroutines CROSSEC-function

The module CROSSEC was written to provide the most commonly used formulae for cross section calculation as an integral part of the command language and to relieve the user at input coding. CROSSEC consists of the following subroutines:

CROSEC,TWOOP,THROP,SIXOP,ETA2,SGGSTR,ALPETA

The subroutine CROSEC is the control routine for the particular subroutines:

Subroutine name	ENTRY-name
TWOOP	ALPHA1 NUSF1 SGG1 SGX1 SGT1 SGN1
THROP	SGA1 SGG3
SIXOP	SGX2 SGI1
ETA2	
SGGSTR	SGG2 SGTR1
ALPETA	ALPHA2 ETA1

2.5.1 Subroutine CROSEC - the control routine for the neutron cross section calculation

The subroutine CROSEC is the control routine for the neutron cross section calculation of the following data types:

ALPHA,ETA,SGA,SGG,SGI,SGN,SGT,SGTR,SGX

The explanation of the symbols:

σ_T	-	total cross section
σ_n	-	elastic scattering cross section
σ_{tr}	-	transport cross section
σ_x	-	non-elastic cross section
η	-	effective number of secondary neutrons emitted per neutron absorption
ν	-	average number of secondary neutrons per fission
μ_1	-	average of the cosine of the elastic scattering angle in the laboratory system
σ_γ	-	radiative capture cross section
σ_f	-	fission cross section
σ_p	-	cross section for the (n,p)-process
σ_α	-	cross section for the (n,α)-process
σ_a	-	absorption cross section
$\sigma_{n'}$	-	total inelastic scattering cross section
σ_{2n}	-	cross section for the (n,2n)-process
σ_{3n}	-	cross section for the (n,3n)-process

The call:

```
CALL CROSEC(X,Y,X1,Y1,LX,NR)
```

X	- array of the energy values
Y	- array of the cross section values

X1	–	see X
Y1	–	see Y
		X,Y,X1,Y1 are the work areas of the calculation package CALCPAC
LX	–	the length of the X,Y,X1,Y1 arrays
NR	–	returncode

= 0	no data found for the requested data type, either on the KEDAK-file nor in the auxiliary dataset DADS2
= 1	no more data available on KEDAK for the processed data type
= 2	the index counter for the processed data point is equal MAXNUM i.e. the length of the work area
= 3	no data available on the KEDAK-file for the requested data type
= 4	no data found for the requested energy range
= 5	no further data on the KEDAK-file for the processed data type
=10	data found on the KEDAK-file lie above the requested energy range

The subroutine CROSEC uses the operation code number NOP from the common /OPAR/. The calls for the operation codes are performed based on the sequence of the commands defined in OPDEF. The commands are conformable to the subroutine or entry names respectively: ALPHA1,ALPHA2,ETA1,ETA2, NUSF1,SGA1,SGG1,SGG2,SGI1,SGN1,SGT1,SGTR1,SGX1,SGX2.

The subroutine EDIT is called to store the calculated data type on the auxiliary direct access dataset DADS2 for later use.

2.5.2 The subroutines for the calculation of the particular cross sections: TWOOP,THROP,SIXOP,ETA2,SGGSTR,ALPETA

The following applies to all subroutines listed above:

The names of the data type to be calculated and the operation code number are transmitted to the subroutines via the common /PARM/. The arithmetic operations: plus, minus, divide and multiply are performed by a call to ARITHO. The operation code number specifies the data type (cross section) and the formula for this calculation.

The subroutine EDIT is called in ETA2 and SGGSTR to store the calculated data into the auxiliary direct access dataset DADS2. A call to CAL1IN causes the initialization of the work areas, i.e. the areas are set to zero, when the operation was carried out and the data were stored on DADS2.

The subroutine TWOOP provides six entries which manage the calculation of the following data types:

```
Entry name
ALPHA1: ALPHA = SGG/SGF
NUSF1 : NUSF  = NUE * SGF
SGG1  : SGG   = SGF * ALPHA
SGT1  : SGT   = SGN + SGX
SGX1  : SGX   = SGT - SGN
SGN1  : SGN   = SGT - SGX
```

The subroutine THROP with two entries allows the calculation of the types:

```
Entry name
SGA1: SGA = SGG+SGF+SGP+SGALP+SG2N+SG3N+SGD
SGG3: SGG = SGA-SGF-SGP-SGALP-SG2N-SG3N-SGD
```

The subroutine SIXOP with two entries calculates the types:

```
SGX2  : SGI   = SGX-SGG-SGF-SG2N-SGP-SGALP-SG3N-SGD
SGI1  : SGX   = SGI+SGG+SGF+SG2N+SGP+SGALP+SG3N+SGD
```

The subroutine ETA2 calculates the data type ETA = NUE * SGF/(SGF+SGG)

The subroutine SGGSTR provides two entries for the calculation of the data types:

SGG2	: SGG	= SGF * ((NUE/ETA) - 1)
SGTR1	: SGTR	= SGT - SGN * MUEL

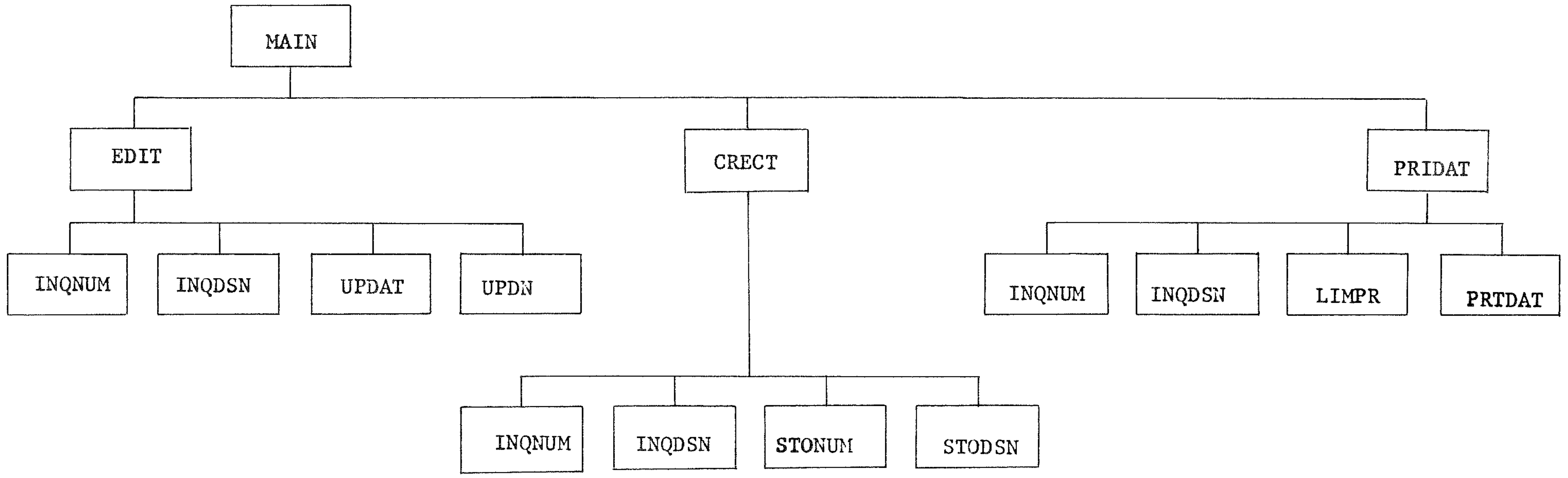
The subroutine ALPETA provides two entries for calculation of:

ALPHA2	: ALPHA	= (NUE/ETA) - 1
ETA1	: ETA	= NUE/(1+ALPHA)

The subroutine ETACC supplies the value Y = 1/(1+ALPHA).

MINUC1 is called to subtract the constant XC = 1 from the processed data type in the energy range /FROM,TO/. The values FROM,TO are obtained from the common /OPAR/.

2.6 DATAMAN - call scheme



2.6 Data Management of the temporary direct access dataset DADS2 and of the two auxiliary datasets of CALCPAC - DATAMAN

The module DATAMAN manages the auxiliary datasets of CALCUL: the direct access dataset DADS2 and the two auxiliary datasets (working area) of CALCPAC.

The data to be processed in CALCPAC are transferred into the working area. The data calculated by CALCPAC, or the data read from the external source are stored into DADS2 for later use in CALCPAC and/or in other modules of CALCUL (e.g. CROSSEC, OUTPUT,DATAMAN). DATAMAN consists of the following parts:

1. The subroutines to create and update DADS2: EDIT,UPDAT,UPDN. The labeled common blocks/DA1/,/DA2/,/DA3/,/DA4/,/DA5/ are used at the definition and organization of DADS2.

2. The subroutine CRECT to delete data from the auxiliary datasets of CALCPAC.

3. The subroutines to print data for checking purposes: PRTDAT,LIMPR, PRIDAT.

2.6.1 The subroutines EDIT,UPDATE,UPDN to create and update the temporary dataset DADS2

The subroutine EDIT is the data management routine for the temporary direct access dataset on FT02F001 (DADS2), where the processed data are stored for later use in CALCPAC, CROSSEC and for editing by the OUTPUT module.

The call:

```
CALL EDIT(X,Y,X1,Y1,LX,NERR)
```

The arguments:

X	– energy values	arrays retaining the previous and current result alternatively
Y	– cross section values	
X1	– energy values	
Y1	– cross section values	
LX	– length of the arrays	
NERR	– error returncode	
	= 0 – no error	
	= 1 – error appeared, no data saved	

Data to be stored in DADS2 are specified to the program by the SAVE or NAME command. Data "named" are for internal use only; the output unit number for these data is set to zero in the directory (common /DA2/).

The subroutine EDIT inquires the number of processed data points (NUM) by a call to INQNUM. If the number is greater than LX (the length of the incore work area) a call to INQDSN provides the dataset reference number (NDA) of the auxiliary dataset where the data are written by CALCPAC.

EDIT reads the data with the read statement

```
READ (NDA) LDAT,(X(I),Y(I),I=1,LDAT)
           [LDAT = NDAT = NUM]
```

and writes them with the aid of UPDAT,UPDN on DADS2.

If no data were processed (NUM=0) for the data type to be named or saved, the error message is printed: SAVE/NAME REQUESTED. NO DATA FOUND, NO OPERATION PERFORMED.

If no data are found for the requested energy range, the error message is printed: SAVE/NAME REQUESTED. NO DATA FOUND IN SPECIFIED ENERGY INTERVAL.

Subroutine UPDAT, entry UPDN

The subroutine UPDAT creates the temporary direct access dataset DADS2 and updates the directory in /DA2/.

The call:

```
CALL UPDAT(NR,IKENN,NAM,NAMES,NX,X)
```

NR	returncode = 0 no error = 1 error message is printed
IKENN	= 0 for data to be "named" only = unit number of the output dataset for data to be saved
NAM	number of names
NAMES	the names of the processed data type
NX	number of data in X
X	array retaining data to be stored on DADS2

The data on DADS2 are stored in records of 2000 bytes = 500 words. The record length is initialized in the common /DA1/.

UPDAT is called for the first record to be written for a processed data type. For each subsequent record of the same type UPDN is called.

The common blocks /DA1/ and /DA2/ are used to maintain the direct access dataset DADS2.

/DA1/ retains information about the layout and the status of DADS2.
/DA2/ retains the entry table of the data types stored in DADS2.

Data for a maximum of 79 different reaction types may be stored on DADS2.

<u>The labeled common block /DA1/</u>

```
COMMON /DA1/ LREC,NREC,MAXENT,KENNA,NSREC,NAVREC,NENT
```

/DA1/ provides information about the specification and the status of DADS2. The parameters are initialized in INIT1,INIT2.

LREC	is the record length in the temporary direct access dataset (DADS2) with the dataset reference number 2. LREC=BLOCKSIZE/4=2000/4=500 at present
NREC	maximum number of records available in the direct access dataset. NREC is retrieved by the subroutine DINF (reference 3) from the space parameter in the DD-statement for FT02F001. NREC is initialized in INIT1 with 100.
MAXENT	maximum number of entries that may be retained in the entry table (directory) for DADS2 (= 79 at present)
KENNA(3)	is anarray that contains the identifier "TEMPSTORAGE"
NSREC	number of records not used (NSREC=0 at present)
NAVREC	number of the next available record
NENT	the current number of entries in the entry table = the actual length of the entry table

/DA1/ is used in UPDAT,MAIN,EDIT,LTLOC,LTREC,INPUT, FINDAT.

The labeled common block /DA2/

COMMON /DA2/ MAT,TYP,EXC,EMIN,EMAX,NNAM,IR,NP,KENN

/DA2/ contains the entry table of the data types stored in DADS2.

MAT	REAL*8	is the array to retain the isotope names
TYP	REAL*8	is the array to retain the type names
EXC	REAL*4	is the array to retain the third names (see reference 5, KEDAK conventions)
EMIN, EMAX	REAL*4	Energy boundaries of the energy range for the data type specified by "MAT,TYP,EXC" stored in the DADS2 dataset
NNAM	INTEGER*2	number of names for the specified data type
IR	INTEGER*2	number of the record, where data are stored on the DADS2 dataset (associated variable)
NP	INTEGER*2	number of data points stored for this type
KENN	INTEGER*2	a flag that indicates, whether the data stored in DADS2 are to be edited for output in KEMA-input-format. KENN=dataset reference number of the output dataset, which may be used to update the KEDAK-library. KENN=0 indicates that the data are used only for calculation in the current job. The default value for KENN is 10, it may be changed via input for the keyword OUTUNIT.

/DA2/ is used in UPDAT,LTLOC,INPUT.

The labeled common block /DA3/

COMMON /DA3/ LR, X(500)

LR	record number (associated variable) of the record read from the temporary direct access dataset by the subroutine LTREC into X
X	the array to retain the data of one record from DADS2

/DA3/ is used in EDIT,LTLOC,LTREC.

The labeled common block /DA4/

COMMON /DA4/ NZ,NI,NX

NZ	current number of the processed (located) data point NZ $\leq$ NX
NI	current number of the data point in the record (NI $\leq$ LREC)
NX	the number of data points stored for the required data type

/DA4/ is used in LTLOC.

The labeled common block /DA5/

COMMON /DA5/ NRS,NLRF

NRS	a flag that indicates whether updating the temporary direct access dataset with the subroutine UPDAT was successful: NRS=0 or not: NRS=1
NLRF	a flag that indicates whether the last written record is complete: NLRF=0 or not: NLRF=1 Error message: data truncated

/DA5/ is used in UPDAT.

2.6.2 The subroutine CRECT(LX)-Command: DELETE

The subroutine CRECT is called by the main program for deletion of the result of the last arithmetic operation if the DELETE command was entered in the control input. LX is the length of the work area for CALCPAC in the main storage.

INQNUM is called to ascertain the number NUM of processed data. If NUM is greater than LX, i.e. data are stored on the auxiliary data set, INQDSN is called to inquire the dataset reference numbers, STODSN to reset the dataset reference numbers, and STONUM to reset the number of processed data.

If LX is greater than NUM, i.e. all processed data are kept in main storage, then the erroneous result of the last operation is not deleted.

2.6.3 The subroutines PRIDAT,LIMPR,PRTDAT - to print an output list for checking purposes

Subroutine PRIDAT

PRIDAT manages the printout of data currently stored as result or data to be named or saved on DADS2 for checking purposes.

PRIDAT is called by the main program to print the data processed in CALCPAC if the key PRINT=6 is specified in the control input list.

The call:

CALL PRIDAT(X,Y,X1,Y1,LX)

X	array retaining energy values	
Y	array retaining cross section values to be printed X,Y data are stored on the auxiliary dataset i.e. NUM>LX	
X1	energy values	for LX < NUM data are kept in main storage
Y1	cross section values to be printed	
LX	length of the arrays X,Y,X1,Y1	

The following subroutines are called by PRIDAT:

INQNUM	to inquire the number of data processed for the data type to be printed
INQDSN	to provide the dataset reference number of the auxiliary dataset, where data are stored, if the number of data NX is greater than the length LX of the working area in CALCPAC
LIMPR	to establish the energy limits for the data to be printed according to the requested energy range FROM,TO
PRTDAT	to print a block of LDAT data

Subroutine PRTDAT

The subroutine PRTDAT is called by PRIDAT to print the total number of data points and the data processed for a data type.

The call:

```
CALL PRTDAT(X,Y,LDAT,LL,NP,FIRST,LAST,TX,LTX,NX)
```

X	array of the energies to be printed
Y	array of the cross section values to be printed
LDAT	number of data to be printed from the X, Y arrays
LL	number of the printed line
NP	the number of the first point printed in the line LL
FIRST	= .TRUE. or .FALSE., indicates whether the line is the first line or not
LAST	= .TRUE. or .FALSE., indicating whether the line is the last one or not
TX	array containing the text for the heading line
NX	the total number of data points for the printed data type

Subroutine LIMPR

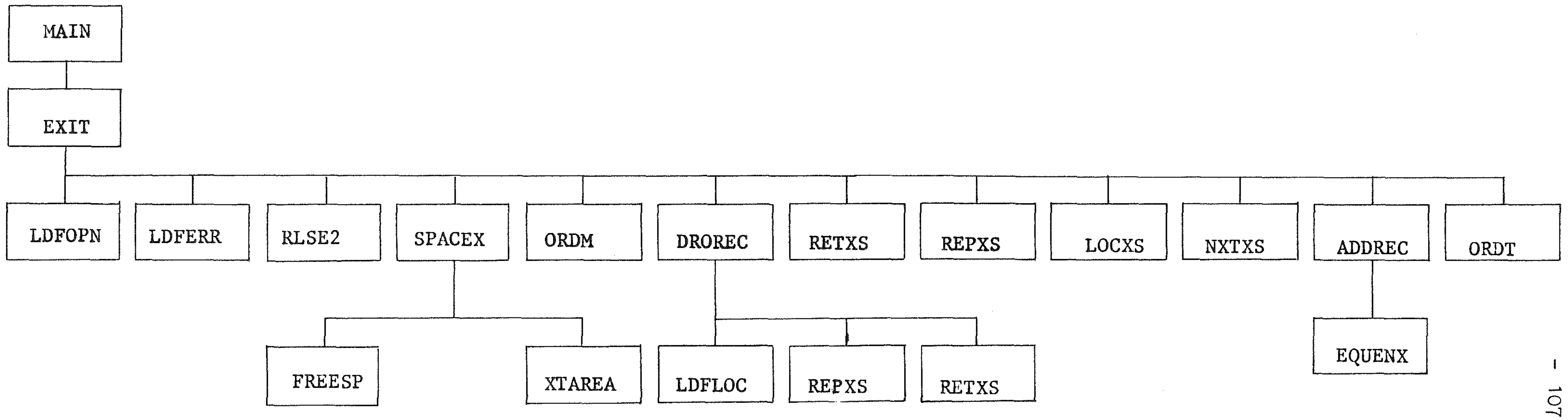
The subroutine LIMPR is called by PRIDAT to ascertain the first value of the data to be printed.

The call:

```
CALL LIMPR(X,LDAT,IANF,LAST)
```

X	array to retain the energy values of the data to be printed
LDAT	the number of data in X
IANF	index of the first value of the data to be printed from X
LAST	= .TRUE. or .FALSE. indicates whether the data in X are the last for the processed data type or not

2.7 OUTPUT edition package - call scheme



2.7 Output editing package (KEMA-input format dataset) - OUTPUT-function

The output of CALCUL is written in a dataset of ADD- and DROP-records for the KEDAK-Management program. The organization of this output dataset is performed in the module OUTPUT with the aid of the following subroutines:

1. EXIT	- the control routine for OUTPUT
2. ORDM	- to sort the isotope names in KEDAK-order
3. ORDT	- to sort data for reaction types, energies, and energy ranges
4. SPACEX,SPACE2	- to allocate dynamically work area for OUTPUT
5. ADDREC	- to write the ADD-records
6. EQUENX	- to remove data for multidefined points
7. DROREC	- to write DROP-records

The retrieval routines RETXS,REPXS,LOCXS,NXTXS are used to retrieve the data from the KEDAK-library and from DADS2.

2.7.1 The subroutine EXIT-control routine for OUTPUT

The subroutine EXIT performs the editing function for the output of CALCUL in KEMA-input-format. The output is written in ADD and DROP-records which could be processed by the KEDAK-Management program (reference 5).

The call: CALL EXIT

A call to LDFOPN (see reference 1) provides the KEDAK-file on which data are to be changed, added or deleted.

RLSE2 and SPACEX are called to provide the working areas for EXIT.
The subroutine ORDM sorts the material names in the directory (common /DA2/) of the auxiliary direct access dataset in KEDAK-order. The subroutine ORDT sorts these data for type, energy and energy range in KEDAK-order.

The subroutine DROREC is called to write the DROP-records (reference 5).
The ADD-records (reference 5) are written by the subroutine ADDREC.
RETXS and REPXS (reference 6) are called to retrieve data from the KEDAK-file, LOCXS and NXTXS to retrieve data from the auxiliary direct access dataset.

2.7.2 SPACE2,SPACEX - handling of dynamic storage allocation

Subroutine SPACE2, entry RLSE2

The subroutine SPACE2 provides the space of the work areas for CALCPAC in main storage.

The call:

CALL SPACE2(F,L1,LX,LE,NP)

F	- address of the area
L1	- length of the area F
LX	- the length of the areas retaining the current and previous result
LE	- the length of the area to retain the data from the KEDAK library
NP	- the maximum number of data points for the reaction types of the processed isotope

FREESP (reference 8) is called to provide the number of bytes available for CALCUL in main storage.

XTAREA (reference 2) is called to establish the address of the work area.

The ENTRY RLSE2 is called (in the main program before a repeated call to SPACE2, and in EXIT before the call to SPACEX) to release storage with the aid of the REXTAR routine (reference 2).

Subroutine SPACEX

SPACEX is called to provide the work areas for the subroutine EXIT.

The call:

CALL SPACEX(X,LA,LX)

X	- address of the work area provided for EXIT
LA	- displacement
LX	- length of X

The available space is provided by a call to FREESP. The address of the work area is ascertained in the XTAREA subroutine (reference 2).

2.7.3 ADDREC,DROREC - subroutines to write the output records

Subroutine DROREC

The subroutine DROREC writes the DROP-records (reference 5) to inform the KEDAK-Management program which data on the processed KEDAK-file are to be deleted.

The call:

```
CALL DROREC(MAT,TYP,EXC,NAMZ,EMIN,EMAX,X,Y,Z)
```

MAT	is a real*8 variable to retain the name of the isotope for which data are to be deleted
TYP	is a real*8 variable to retain the reaction type name
EXC	excitation energy for the inelastic excitation cross section
EMIN EMAX	lower and upper energy limits of the energy range where data are to be deleted
X	array to retain the energy values of data points to be deleted
Y	array to retain the cross section values
LX	the number of data points to be deleted

If EMIN is greater than EMAX,all data are deleted for the processed reaction type, i.e. a DROPA-record (reference 5) is written on a dataset with the reference number KTAPE.

To delete the data for a given energy range /EMIN,EMAX/ the data are read with the aid of the subroutines RETXS,REPXS from the KEDAK-library and DROPS records are written for each energy available in the processed energy range on KEDAK and in the ADD-records. If the key PRINT=6 was specified in the control input list of the SAVE command,a list is printed, in order to check the output.

Subroutine ADDREC

The subroutine ADDREC is called by the EXIT routine. ADDREC writes the data processed in CALCUL as ADD-records for the KEDAK-Management program on a dataset with the reference number NOUT=10.

The call:

```
CALL ADDREC(MAT,TYP,EX,NAMZ,MX,X,Y)
```

MAT	- a real*8 variable retaining the isotope name
TYP	- a real*8 variable retaining the reaction type name
EX	- the third name (i.e. excitation energy)
MX	- number of data points processed
X	- array retaining the energy values
Y	- array retaining the cross section values

A list output is printed for checking the results, if the key PRINT=6 was specified in the control input list of the SAVE command.

The subroutine EQUENX is called to handle double defined energy points.

2.7.4 ORDM,ORDT - to sort data in KEDAK order, EQUENX - to remove multidefined points

Subroutine ORDM

The subroutine ORDM sorts the array M1 according to the order of M2.

The call:

```
CALL ORDM(N,M1,K,M2)
```

The arguments:

N	- the number of isotope names to be sorted (= the number of isotopes to be edited for output)
M1	- a real*8 array retaining the isotope names to be sorted
K	- the number of isotopes available on the KEDAK-file
M2	- a real*8 array retaining the names of the isotopes available on the KEDAK-file

Subroutine ORDT

The subroutine ORDT sorts the arrays TYP,ES,EMIN with the priority:
all types for ES,EMIN

The call:

```
CALL ORDT(NT,TYP,ES,EMIN,EMAX,NN)
```

NT	- the number of reaction types
TYP	- a real*8 array to retain the reaction type names
ES	- an array to retain the third names (e.g. excitation energy)
EMIN } EMAX }	- energy limits of the processed energy range
NN	- array to retain the number of names for each reaction type

The subroutine EQUENX

The call:

CALL EQUENX(N,E,S,NAMZ,NAMES)

The arguments:

N	- the number of data points
E	- array retaining the energy values
S	- array retaining the cross section values
NAMZ	- number of names
NAMES	- the names of the reaction type

EQUENX is called by the subroutine EXIT to test the array E for equal energies and to remove them from the arrays E,S resetting N. If the cross sections at such multidefined points do not agree, a warning message is printed in addition.

E,S is supposed to be ordered according increasing E. Two energies are considered to be equal, if they differ less than 0.0001 %.

In the absence of the authors the publication of this report was prepared by B. Goel and R. Moser.

3. References

1. I. Langner, R. Meyer
IDFPAC/LDFPAC - two retrieval packages for the Karlsruhe Evaluated Nuclear Data Library, KFK 2387/III, Section 2, April 77

2. W. Höbel
XTAREA, REXTAR - dynamische Dimensionierung von FORTRAN-Feldern, KFK, to be published

3. G. Arnecke, H. Bachmann
DEFI, DINF dynamisches DEFINE FILE, KFK, to be published

4. G. Arnecke
DDTEST - Benutzte Dateien, KFK, to be published

5. B. Krieg
The KEDAK Program Compendium Part II
KEDAK Basic Management, KFK 2387/II, 77

6. R. Meyer
RETPAC - A user oriented retrieval package for use with the Evaluated Nuclear Data Library KEDAK, KFK 2387/III, Section 4

7. H. Blesene
CONVY - FORTRAN-Unterprogramm für die IBM/360 zur Umwandlung von in maschineninterner bzw. in alphanumerischer Darstellung vorliegenden Test- und Gleitkommazahlen in alphanumerische bzw. maschineninterne Darstellung, unpublished

8. G.H. Hinze
FREESP - Subroutine zur Bestimmung des noch freien Kernspeichers für FORTRAN-Benutzer an der IBM/360-65, unpublished