MAX-PLANCK-INSTITUT FÜR PLASMAPHYSIK

GARCHING BEI MÜNCHEN

G A P

A GALE Precompiler

Joachim Steuerwald

R/36 March 1980

Die nachstehende Arbeit wurde im Rahmen des Vertrages zwischen dem Max-Planck-Institut für Plasmaphysik und der Europäischen Atomgemeinschaft über die Zusammenarbeit auf dem Gebiete der Plasmaphysik durchgeführt.

IPP R-36 J. Steuerwald

G A P
A GALE Precompiler

March 198o

Abstract:

Computer-aided, predetermined control of experiments requires a simple and reliable tool to change the values of parameters by dialog or program. The values set influence the results of tests and must be taken into account in processing experimental data, so that it is easy to access to those values in a FORTRAN program. The GALE data acquisition system is organized by domain descriptors, assembled in the GALE Name-list File, which contain all necessary information about parameter variables. The GALE precompiler GAP was developed to simplify access to elements of this structure without changes in the FORTRAN compiler.
Though intended for easy use of GALE variables, GAP may be applied to all data structured and described in a similar manner. Archivation and documentation of resulting data by EDDAR (Experimental Data Documentation and Retrieval) system were therefore included.

Contents

1. Introduction

1.1 Parameters of the GALE Data Acquisition System

The GALE data acquisition sytem /1/ is used for the control of experiments. Parameters may be changed by dialog and therefore no errors occur in transmission. The input data is attached to the relevant offsets by the GALE Namelist File (Fig.1).
Each entity is uniquely and completely described by a Namelist Control Block (NCB) of the GALE Namelist File, such that it may be reliably and simply found by name. The values of these entities serve to regulate the parameters of the experiment, and so they are also used to process experimental data.
Entities of different data types are assembled in one structure. In FORTRAN language no feature exists to gain access to variables by using the said domain descriptor. The programmer therefore has to be familiar with the organization of data description defining the relevant arrays. Changes in offsets nearly always lead to changes in programs and are therefore often followed by erroneous results.
Misunderstandings and difficulties cannot always be avoided and lead to time-consuming investigations.
These problems can be solved by using the information contained in the NCB's. The names of entities may then be employed to gain access to them in a FORTRAN program.

1.2 Variables in a Program

Variables in FORTRAN language define a storage address. In compiling the program the name of the variable is lost. Referencing is not known in FORTRAN language - unlike in ALGOL68 - and so no simple way exists to transfer names of variables to the object program without changes in the FORTRAN compiler.
The GALE precompiler was therefore developed to allow access to the values of GALE variables described in GALE Namelist Files in a FORTRAN program without knowledge of the organization of the relevant data fields. It should also be ensured that programs work exactly even if offsets have been changed.

1.3 Storing Results of Processing Experimental Data

If no error has occured, some results of processing experimental data ought to be documented for later use. In this case facilities are expected to locate variables by name and to complete automatically the information about data structure. These goals are reached by means of the GALE precompiler, provided the documented results are organised by the EDDAR Namelist File /2/ in the manner of the NCB's of the GALE Namelist File.

2. Basic Principles

2.1 Survey

The name of entities of one or more diagnostics (an exclusive sub-experiment) are made known to the GALE precompiler. The number of the test provides

- the relevant version number of GALE Namelist File and from NCB
- the offset of entity
- the data type of entity and
- the length of the data field.

Appropriate arrays to receive the data from the user parameter section of the GALE Data File are defined.
Finally in all data gained till now, entity names are assigned to elements of the relevant core regions. In the same manner, the GALE precompiler uses the EDDAR Namelist File for organizing the results of experimental data processing.

Because the data of most diagnostics and devices are similarly organized, names of entities must be made unique by additive information. FORTRAN variables are therefore defined according to special GAP rules, shown in the following section.

2.2 Definition of Variables

In addition to the well-known FORTRAN syntax, the GAP descriptor

<gap descr> ::= GAPI <gap list1> | GAPO <gap list2>

is introduced. Furthermore, we have

<gap list i> ::= <gap item i> | <gap list i>,<gap item i>
<gap item 1> ::= <dev item> <param array> <var list> |
<param array> <var list>
<dev item> ::= <exp name> . <diag id> . <dev id> |
<diag id> . <dev id> | <dev id>
<var list> ::= <var name> | <var list> <var name>
<var name> ::= valid GALE variable name
<exp name> ::= three character EBCDIC short name of experiment
<diag id> ::= one to three digit number of diagnostic within experiment <exp name>
<dev id> ::= <dev name> | <dev name> (<dev number>)
<dev name> ::= two character EBCDIC short name of device
<dev number> ::= one or two digit number of device within diagnostic
<param array> ::= name of array the user parameters are read into (valid variable name)
<gap item 2> ::= <diag item> <var list> | <var list>
<diag item> ::= <exp name> . <diag id> | <diag id>

Remarks:

1. Valid experiment names are found in AMOS segment EXPP:INFO.EXPN (see ESS, EDDAR system survey /2/ p. 1-3)
2. Valid diagnostic numbers of all experiments are listed in AMOS segment <exp name>:GINF.DIAG (see ESS p. 1-6)
3. Valid abbreviations of devices are assembled in AMOS segment EXPP:INFO.DEV (see ESS p. 1-3)

4. The user may omit the experiment name if he is registered in the user identification list (UIL) in AMOS segment EXPP:INFO.UIL (see ESS p. 1-3).
   As owner of a diagnostic he may also omit the number of his own diagnostic.
   Both simplifications are only allowed in cases of uniqueness.
5. The GAP descriptor must be declared in each program unit containing GAP variables.
6. The device numbers must follow the conventions introduced by the diagnostic owner.
7. The GAP variables are separated by blanks. The end of a GAP item is announced by comma (,) if further items follow, otherwise by blank at the end of the statement.
   Valid device identifications are connected with the keyword GAPI and have the form

ASD.41.PG(2)	ASDEX experiment, diagnostic number 41, second PPG (if only one is attached, (1) may be omitted)
41.PG(2)	the user is identified only in UIL for ASDEX (see remark 4). The variable name is valid in GAP though beginning with a number (replaced later on by alias), implications as above
PG(2)	moreover, the user is owner of the diagnostic number 41 (see remark 4), implications as above.

Valid diagnostic identifications for output are connected with the keyword GAPO. Because the above rules are applied, variable names are mostly unique as long as the user puts the relevant values into the results of his own diagnostic. Otherwise experiment name and diagnostic number are connected with the variable in the same manner as above.

The syntactical rules and the card image of FORTRAN language are valid in GAP. There is only one exception: variable names beginning with a number.

2.3 Substitution of GALE Variables

The extended definition introduced by the foregoing section is necessary for uniqueness of identical variable names within the organization of data of different devices. If it implies a syntactical error in FORTRAN language, the variable name is replaced by an alias name (see Sec. 3.1) during precompilation and entered into the table of variables (TOV, Fig.2).

3. Program Units

3.1 Generating Alias Names

The GAP precompiler looks for a keyword (GAPI, GAPO). If no keyword is found, it finishes performing the actual program unit. Then the GAP precompiler continues with the next program unit or, if there is none, calls the FORTRAN compiler. The GAP precompiler investigates variable names appropriate to GAP rules and announces invalid ones. It continues to perform the program unit, but later on stops compilation.
Variable names invalid according to FORTRAN language rules must be substituted. The extension of a variable name is omitted if the GALE variable name is unique. Otherwise, a string of three characters, not appearing in this program unit, is joined with the content of a counter to build an alias name. If the counter exceeds 999, it is cancelled and GAP looks for another string.
After performing a program unit, the table of variables TOV is listed.

3.2 Describing Data

The raw data of test are assembled in a GALE data file, whose organization is described by the relevant GALE Namelist File. The EDDAR System Survey /2/ affords the possibility of simply using the latter. The experiment name and diagnostic number may be omitted in defining a GAP variable if the user is only listed for one experiment in the UIL (User Identification List), and is owner of the named diagnostic within this experiment, and the variable relates to a device of this diagnostic.

Otherwise

the experiment name destinates the relevant GALE Namelist File,
the diagnostic number and device name define the appropriate NCB list, and
by the variable name, the NCB of the entity is to be found.

Descriptive data such as

offset in user parameter section,
type of data, and
length of array

are entered in the TOV.
Similarly, the organization of resulting data is described in the EDDAR Namelist File. These data are entered in a report as a pseudo-diagnostic or an array of diagnostic results, mostly the diagnostic of which the user is the owner. Properties of variables not specified are introduced into an automatically generated NCB - with the next free offset - in the EDDAR Namelist File.

3.3 Performing the Information Given by the NCB

The offset A (in bytes, beginning with zero) defines with the starting index J (default value: 1) - taking the appropriate addresses of relevant arrays - the index

$$I = J + A/L \qquad (L = \text{length of data item})$$

of the named entity in the parameter array, provided A is a multiple of L.
Nevertheless, the actual organization of the GALE system allows inconsistencies. In this case

- an appropriate LOGICAL*1 array is generated and overlaid by EQUIVALENCE statements,
- the data are transmitted, and
- the above algorithm applied.

Data conversion must be mentioned in these special cases.

If the entity contains more than one element, the necessary array is declared.
Formal parameters in subroutines may not be overlaid in FORTRAN language. They are therefore to be omitted in the subroutine call and an appropriate COMMON statement is inserted. Only labeled commons, with labels like alias names in Sec.(3.1), are generated.

3.4 Substitutions of Variable Names

Finally, the variable names are replaced, if necessary, by alias names. In particular, it must be noted that all variable names not valid in FORTRAN language are to be replaced by valid ones. Besides the TOV, all error messages are listed and the FORTRAN compilation is not performed if there are errors. Otherwise, GAP initiates compilation or calls the next program unit.

4. GAP Messages

4.1 General

The output of GAP messages may be controlled by the user by the logical variable OUT:

OUT=false	- short form of output: title only in case of any message, warnings and errors are presented by their number (see Sec.4.2), TOV is suppressed.
OUT=true	- long form of output: the title is always listed, TOV, warnings and errors appear with full text.

The default value of variable OUT is false.

4.2 Messages

The title is of the form

JOB AAANNN OF DD.MM.YY AT HH:MM:SS
GAP - PRECOMPILER
PROGRAM UNIT AAAAAA PRECOMPILED

This is followed by the table of variables (TOV) and then the appropriate messages, if any.

W 1 AAAAAA VARIABLE MULTIPLE DECLARED
The variable name is declared more than once, all unnecessary declarations being dropped.

E 2 AAA INVALID EXPERIMENT NAME
The experiment name does not exist in AMOS segment EXPP:INFO.EXPN, look for the valid name or ask the system manager.

E 3 NNN INVALID DIAGNOSTIC NUMBER FOR EXPERIMENT AAA
The diagnostic number was not found for this experiment in AMOS segment <expname> :GINF.DIAG. Look for a valid number or ask the system manager.

W 4 AA DEVICE NAME NOT KNOWN
The device short name is not known. Look for a valid one in AMOS segment EXPP:INFO.DEV

E 5 NN INVALID INDEX OF DEVICE AA OR TOO MANY DEVICES
The owner of this diagnostic has defined a combination of devices not containing the indexed one.

E 6 EXPERIMENT NAME OMITTED, USER NOT IN UIL
The extension of the variable has no experiment name, though the user is not introduced for any experiment in the UIL

E 7 DIAGNOSTIC NUMBER OMITTED, USER NOT OWNER OF ANY DIAGNOSTIC
The extension of the variable contains no experiment name and diagnostic number, though the user is not owner of any diagnostic.

E 8 Free for later use

E 9 FOR EXPERIMENT AAA EXISTS NO NAMELIST FILE
Though the experiment name is valid, no namelist file was found. Consult the system manager.

E 1o NO NCB LIST FOP DEVICE AA(NN) OF DIAGNOSTIC NN FOUND
For this device, no entry to NCB list within the named diagnostic was found. Examine the existence of this device in the definition of this diagnostic or consult the system manager.

E 11 NO NCB FOR VARIABLE AAAAAA
This variable name does not exist for this device. Look for the valid name in the appropriate NCB list.

E 12 OFFSET OP FIELD LENGTH AAAAAA OUT OF BOUNDS IN ARRAY AAAAAA
The offset of a variable or a field is out of bounds within the declared array. Examine whether offset of field declaration is invalid, or whether the declared array is too short.

E 13 NO CONFIGURATION FILE FOR EXPERIMENT AA FOUND
The relevant configuration file, though necessary for the GALE precompiler, does not exist. Consult the system manager.

E 14 VARIABLE AAAAAA NOT IN GAP-DESCRIPTOR
The variable is not defined by a GAP descriptor in this program urit. The extension of this variable in the FORTRAN program may not be resolved.

W 15 VARIABLE AAAAAA NOT IN THIS PROGRAM UNIT
A variable name declared in a GAP descriptor was not detected in the actual program unit.

E 16 VARIABLE AAAAAA INVALID IN PROGRAM UNIT
The variable name does not follow the syntactical rules of FORTRAN language, though not declared in a GAP descriptor.

W 17 CALL OF SUBROUTINE AAAAAA(aaaaaa,...,aaaaaa)
CHANGED TO (bbbbbb,...,bbbbbb) COMMON CCCCCC INTRODUCED
Formal parameters in the call of a subroutine were omitted and put into an automatically generated common. The user should inspect this action in all its consequences.

Remarks

W Warning, compilation follows

E Error, no compilation

5. Remarks

5.1 A New Feature

To make the programmer more familiar with the substitution of variables we introduced a new feature. The use of programmer-designed variable names is forced in the GAP-Descriptor by

< gap item i > := <gap subitem i > = < for varname >

(s. Section 2.2), where

< gap subitem i > substitutes < gap item i > and

< for varname > is the variable name expected in the FORTRAN program.

5.2 Acknowledgement

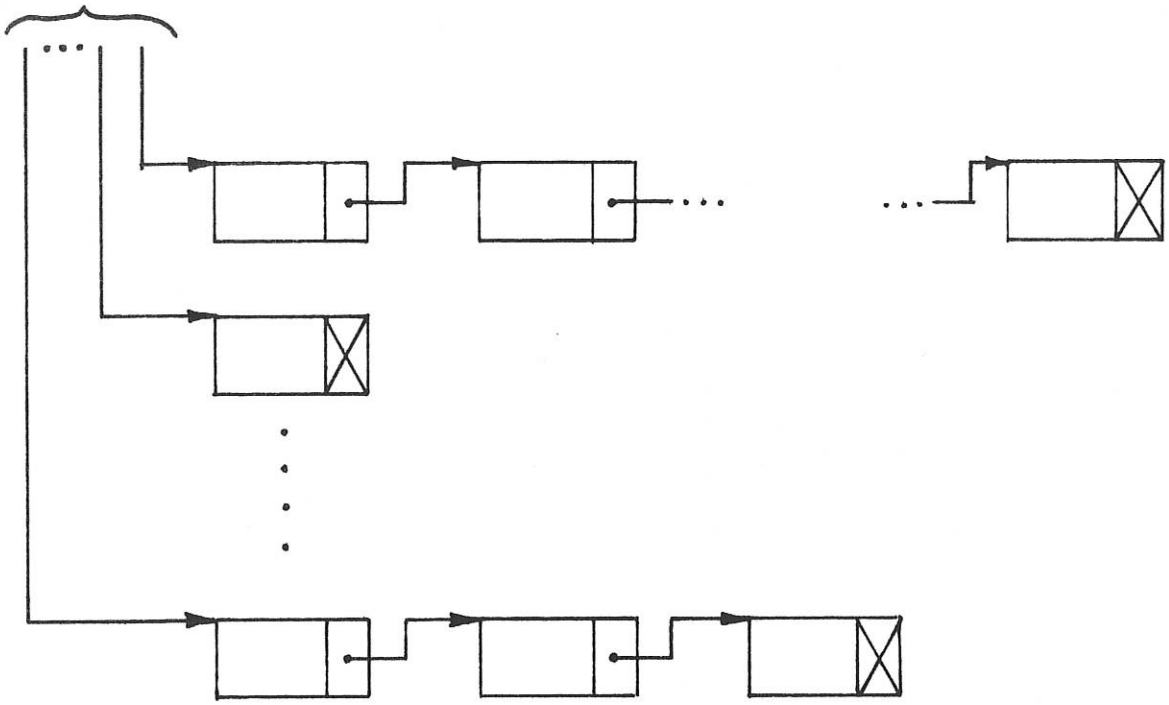
The author wishes to thank Mrs. Tichmann and Mr. H. Fisser for many helpful discussions.

Literature

/1/ Lathe R., Müller E.: GALE Programmer's Handbook. Max-Planck-Institut für Plasmaphysik, Garching. Bericht R/27 vom Februar 1978

/2/ Steuerwald J., Tichmann Ch.: EDDAR System Survey. Max-Planck-Institut für Plasmaphysik, Garching. Bericht R/34 vom November 1979

FROM CONFIGURATION FILE



OFFSET	LENGTH	NAME	CONTENTS
0	2	N.PRV	ACCESS PRIVILEGED FLAG
2	2	N.NXT	Pointer to next NCB
4	2	N.HLP	Pointer to Help Record
6	4	N.NAM	Name of Variable (RAD 50)
10	1	N.FLG	Flag (Data REMOTE OR LOCAL)
11	1	N.TYP	TYPE OF VARIABLE
12	1	N.LEN	ARRAY LENGTH OF DATA
13	1	N.MSK	IF TYPE IS BINARY, N.MSK CONTAINS THE BIT-POSITION
14	1	N.OFS	POINTER TO THE ASSOCIATED DATA REGION

FIG 1: GALE NAMELIST FILE

JOB JKS123 OF 22.04.80 AT 12:31:25

GAP - PRECOMPILER

PROGRAM UNIT MAIN PRECOMPILED

TABLE OF VARIABLES

ALIAS NAME	EXTENSION	VARIABLE NAME	OFFSET	TYPE	LENGTH BYTES
XYZ0001	ASD.41.PPG(1).	TIME	0	REAL*4	132
XYZ0002	ASD.41.PPG(1).	PRMD	132	INTEGER*2	2
		FRQ	0	REAL*4	4
		PRE	4	REAL*4	4
		POS	4	REAL*4	4

FIG. 2 : TABLE OF VARIABLES