



THESIS / THÈSE

MASTER IN COMPUTER SCIENCE

About dynamic interfaces today

Staes, Olivier

Award date:
1985

Awarding institution:
University of Namur

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

INSTITUT D'INFORMATIQUE
FNDP NAMUR

ABOUT DYNAMIC
INTERFACES
TODAY

ANNEE ACADEMIQUE
1984/85

MEMOIRE PRESENTE PAR
OLIVIER STAES
EN VUE DE L'OBTENTION
DU DIPLOME DE
LICENCIE ET MAITRE
EN INFORMATIQUE

Table of contents

Part I. Introduction

Chapter 1. Origin of the work

- 1.1. Assembler way of work
- 1.2. High-level languages way of work
- 1.3. Some remarks
- 1.4. Basis of my decision
- 1.5. Presentation of the structure of the thesis

Chapter 2. A brief historical review

- 2.1. Introduction
- 2.2. Explanations

Chapter 3. Global context of today's software development

- 3.1. Introduction
- 3.2. Software crisis
- 3.3. Origin of the software crisis
- 3.4. Symptoms of the software crisis
 - 3.4.1. Responsivness
 - 3.4.2. Reliability
 - 3.4.3. Cost
 - 3.4.4. Modifiability
 - 3.4.5. Timeless
 - 3.4.6. Transportability
 - 3.4.7. Efficiency
- 3.5. Basis of intended solution
 - 3.5.1. Law of continuing changes
 - 3.5.2. Law of increasing entropy
 - 3.5.3. Law of statistically smooth grow

- 3.6. Some other comments
- 3.7. Software engineering as solution
 - 3.7.1. Definition
 - 3.7.2. Some comments
 - 3.7.3. Modifiability
 - 3.7.4. Efficiency
 - 3.7.5. Reliability
 - 3.7.6. Understandability
 - 3.7.7. Correctness
- 3.8. Some existing solutions
- 3.9. Reactions under deadline pressure
 - 3.9.1. Eliminating the deadline
 - 3.9.2. Not completing all the work
 - 3.9.3. Not completing software engineering purposes
 - 3.9.4. Increasing the software team productivity
- 3.10. Why is the reaction reducing software engineering efforts the most commonly used?
 - 3.11. Some concluding remarks

Part 2. Definition of the problem and the relations between the problem and the software engineering purposes

Chapter I. Definition of the problem

- I.1. An informal definition
- I.2. A formal definition
 - I.2.1. $M = N$
 - I.2.2. $M < N$
 - I.2.3. $M > N$
- I.3. Some remarks

Chapter 2. Relations with testing

- 2.1. Introduction
- 2.2. Definition
- 2.3. In which context testing should be done?

2.4. Different kinds of tests and the errors they tend to find

2.5. Relations between the interface problem and the difficulties in finding bugs today

2.5.1. Introduction

2.5.2. Bad parameter passing

2.5.3. Dynamic problem

2.5.4. Evolution problem

2.5.5. Integration problem

2.5.6. Dummy module problem

2.5.7. Some concluding remarks

Chapter 3. Debugging

3.1. Introduction

3.2. Definition

3.2.1. Theoretical definitions

3.2.2. Activities involved by the debugging process

3.3. In which context should debugging be done?

3.3.1. Debugging environment

3.3.2. When is it complete ?

3.4. Different kinds of debugging level and their consequences

3.4.1. Handling one module written in Assembler

3.4.2. Handling one module written in higher-level language

3.4.3. Handling a set of modules

3.4.3.1. Set of modules in high-level languages

3.4.3.2. Complex modules

3.5. Which part of debugging is the most difficult and their relations with this thesis

3.5.1. Interfaces enforcement

3.5.2. Handling complex software

3.5.3. Handling large software

3.5.4. Integration

3.5.5. Some concluding remarks

Chapter 3. Maintenance

3.1. Introduction

3.2. Definition

3.3. In which context maintenance should be done ?

3.4. Different level of maintenance and their consequences

3.5. What are the most difficulties encountered when maintaining and why ?

3.5.1. Forward compatibility

3.5.2. High-level languages

3.5.3. Tacit assumptions

3.5.4. Concluding remarks

Part 3. Theoretical purposes

Chapter 1. Introduction

1.1. Type checking problem

1.2. Variable parameter list problem

Chapter 2. Analysis of the possibilities of when doing the checks

2.1. At compile time

2.1.1. Ordered compiling

2.1.2. Semi-ordered compiling

2.1.3. Unordered compiling

2.2. At run time

2.3. At link time

2.3.1. What is a linkage editor

2.3.2. Introduction

2.3.3. Needs for type checking at link time

2.3.3.1. What is expected from compilers?

2.3.3.2. How to use these informations ?

2.4. Comparing the different acceptable solutions

Semi-ordered, ordered and link time

2.4.1. Modifications involved

2.4.1.1. At compile time

2.4.1.2. At link time

2.4.2. Constraints involved by the checking

2.4.2.1. At compile time

2.4.2.2. At link time

2.4.3. Advantages of each methods

2.4.3.1. At compile time

2.4.3.2. At link time

2.4.4. Conclusion

Part 4. Implementation of a solution

Chapter 1. Type checking implementation

I.1. Descriptions of type

I.1.1. First possibility

I.1.2. Second possibility

I.1.3. Third possibility

I.1.4. Adopted solution

Chapter 2. EXECUTION OF THE CHECKING

2.1. General algorithm for compilers

2.1.1. Executing a reference

2.1.2. Executing a definition

2.1.3. Creating the ESD cards

2.1.4. Creating a RLD cards

2.2. General algorithm for linking

2.3. Detailed algorithms

Chapter 3. How to handle variable parameter list ?

3.1. Introduction

3.2. At compile time

3.2.1. Ordered or semi-ordered

3.2.2. Partial conclusion

3.3. At link time

3.3.1. Introduction

3.3.2. Trouble of handling full dynamic
interfaces ability

3.3.3. Some partial solutions

3.4. Conclusions

PART 5. Conclusions and further work

ERRATA.

PAGE 3 : the sentence "...more of them gives a practical ..."
should be readed as "...most of them gives no
practical ..." at third paragraph, fourth line.

PAGE 9 : the word "buger" should be readed as "buyer" at
paragraph III.4.5, the last word.

PAGE 12 : the two words "on" should be readed as "or" in
the fourth line from the end of the page.

PAGE 20 : the sentence " $\exists i : 1 \leq i \leq m : T_i = T_i'$ " should be
readed as " $\exists i : 1 \leq i \leq m : T_i \neq T_i'$ " in the third
line.

PAGE 46 : the word "bathone" should be readed as "bottom"
in the sixth line and in the eleventh line.

ACKNOWLEDGEMENTS .

I am gratefull to Mr. Ramaekers for having accepted to conduct this thesis .

I am indebted to M. Linder and M. Schreiber whese discussions and ideas lead me to the domain of this thesis .

I am also indebted to my friend P. Lippens who painstakingly read this manuscript and made suggestions for improvements and who willingly contributed illustration and problem ideas .

Special thanks are owed to M. Van Parijs who have tipeg this work and has to deal with my writings .

To my wife, Isabelle, I owe my gratitude for her understanding, patience and encouragement .

I would also like to thank all other people involved in the elaboration of this thesis .

Part 1 : INTRODUCTION

"Software is a means, not an end."

I . Origine of the work.

My starting point was the fact that I enjoy working at Operating System level and then, most of the time, the language used is the Assembler language . The environment I use to deal with is often a work for large and complex software and not for a software created only for being used during a few days.

It means also that practical ideas are, for me, more valuable than theoretical ones because many times you see very nice theoretical ideas without any practical consequences (at least at the time of the publication). It is not said that you do not have any part of a practical work which will not deal with modelization or underlying theories.

During my practical work, wich was carried out in Assembler language, I saw many things I thought to be practice of the 50's at no more living! (as some as theachers told us). Most of the programs were not documented. Even worse, programs with comments, but these comments were not updated when the code had to be, are very difficult to understand. Another problem was coming from the fact of compatibility.

I . 1 . Assembler way of work.

I had had to write a few new functions for an existing software. In doing this, I had to deal with many problems due to the fact that I was obliged to extract the coding from existing programs in order to insert them in mine . It is a good idea not to rewrite existing coding but not in this way . Moreover, if some other people is working on the program from which you are extracting some coding . This is because of compatibility between different versions of the software to be supported.

Another problem are to me, in Assembler you can change the expression of a macro without having to rename it . Then, whenever you are assembling a program, you only need to have to say with which module libray it needs to be done . And the macro can have been change without having

to change something in the Assembler program.

As the current evolution is to do operating system in a higher level language and as it is stated before, I was interested to see if this mechanism of different macro expansion can or exist in higher level language as C, Pascal or ADA.

I . 2 . High-Level_languages_way_of_work.

I started than to see if higher level languages support facilities liter different number of parameter in a parameter list, passing unbounded arrays, and so on . That can be done in Assembler, but is it possible to handle "DYNAMICAL INTERFACE" in higher level languages ?

As C was a language of an existing operating system, I looked first to it . The astonishment was that there are no checking of the correspondances between formal and actual parameters . Indeed, the caller puts its parameters in a stack; the caller takes with it supposes to find and as return the caller keeps its parameters out of the stack but no checking is done . After this, I looked to Pascal because it was presented as a good language . In this language, there is no possibility to do it because no external compilation is possible (In its standard form at least). In Ada, the type checking between formal and actual parameters is done but once an interface has to be changed we must recompile and relink to whole software . Than, the check is done but without possibility of dynamic interfaces .

I . 3 . Some_remarks.

In the following text, I will often used the term "strongly typed languages" . This means that objects of a given type may take only those values that are appropriate to the type definition and, in addition, the only operations that may be applied to an object are those that are defined for its type . I will refer to this to distinguish languages like Fortran, Cobol, PL/1... where there are no type checking and to those like Pascal and Ada where check is done.

I . 4 . Basis of my decision.

From now, it is from that point that I started to think that the dynamic interface would be an interesting point to study . During my practice, I met Mister Linder who told me this problem and explained me that if this was resolved, then will be easier to work and easier to handle different version of software.

Another basic fact is that most bugs are localized in very few places in the code, than if this code would be replaced easily by a new version even if the interface has been changed . It was the problem when it is a faulty module in an operating system . It would be easy if the only thing to do was when maintenance operation replace the module without having to recompile or to relink the whole .

I started then, to read many books or articles about software management, software debugging, software testing and software engineering . Quite all these books mention the interfacing problem but more of them gives a practical solution if they give any . Most of the time you can read sentences like this : "The problem lies in the interface, but this problem is behind the scope of this book." And no more explanations are given.

Once started to deal with the problem, I saw that there was a problem with compile implications as well as operating system ones . For instance, in Ada, the compiler does the check but for doing it, it needs to have a Data Base where all informations about each programs are putted in . What this means is that we must have a Data Base management system associated with it.

My point of view is to find solutions such that these solutions are practical . That means that the solutions do not need to much spaces; loose to much efficiency; be too expensive and should be easy to handle.

I . 5 . Presentation of the structure of this publication.

The next chapter will give an overview of what is the environment in which this work was undertaken and first a brief historical review. After this, a formal definition will be given and from this definition I will see its connection with the three mainly costly activities :

Maintainance, Debugging and Testing . All this is seen as means in order to increase software quality and also in order to see wath parts of them are concerned with the solving of the dynamic interface problem. All these will arrive to the conclusion of wath are the advantages in solving the problem and also to the possible theoretical solution which in turns will arrive to some implemented solution . All this with comments and critics because none of the solutions is ideal in all point of view .

II . A brief historical review .

II . 1 Introduction.

This is done in order to see what is the evolution of the software problems encountered during a few decade from the 50's to the 80's . These sentences give only the symptoms of the most troublesome problems: If a sentence is written more than once, it is because this problem was not solved or because of the evolution of the complexity of software activities the problem is once again there.

II . 2 . Presentation.

In the 1950's : -Programming in machin language

-The inefficient Assembler

-A software tool for hardware designers

-The disappearing programs

in the 1960's : -Fully tried and tested ?

-The impossible error message

-It works most of the time

-If you can't fix the errors, ignore them

-Can software failure bankrupt a company ?

-Theory simplifies practice

in the 1970's : -An integrated collapse in the new world

-An integrated collapse in the old world

-Fully tried and tested ?(2)

-Data base management systems : panacea, placebo or poisons ?

-A system expands until it collapses

-Keep trying until it works

-The software development backlay

-Theory predicts problems in practice

-At the end of the decade, there are still collapsing

-To collapse or not to collapse

in the 1980's : -An integrate collapse in the whole world

-A computer works week is shorter than a human works week

- Can computer system push the button ?
- Keep trying until it works
- Software failure can bankrupt a company
- Software improvements ?
- Better late than never
- Where shall we get the teachers ?

II . 3 . Explanations .

These sentences are not given here in order to explicit all the development process of software evolution . It is also not an attempt to describe the evolution of software development process but there you can find the evolution of where the problem was and where is the main difference between these periods .

It also will serve me as reference on the reasons why I undertook this study and why I undertook it with the point of view I defined earlier .

III . Global context of today 's software development.

"Careful programming is not a trivial task, even with smart environment ."

III . 1 . Introduction.

This section is intended to give an overview of the global environment that is available today . It will also try to define why it is important or which problem is or are the most important one(s) . Then when the global context in which all software have to be developed is described, a more precise description of this work will be given.

In order to be able to give a precise description of the software state today, it seems to me that the next important things are to be kept in mind .

First, the fact that programming is a social activity, requiring negotiation and communication . That is important because it can explain why there are many difficulties in dealing with programs . Indeed, if it was not the case you are only dealing with people of the same formation . Then the problem lies on the fact that when dealing with large software you are dealing with a multidisciplinary activity and no more only just with a computer activity .

Second, and related fact is the fact that any programmer needs to communicate in one language with the computer, and in another one with the user . And the problem is that the user uses its technical vocabulary . The problem is, than, the needs of a clear, concise and precise language . There is a problem of communication.

III . 2 . Software crisis .

There is a few years ago that the software crisis was recognized but it was only in the years 1968 in a NATO Conference that the software crisis was for the first time officially recognized. But this crisis was seen long before but was not analysed and no means were given in order to salve it . We must be honest and must say that the crisis is still alive .

In order to see some solution, the best way is to see where come this crisis from and than, to see which were the actions or what were the constitutive elements of this crisis .

III . 3 . Origine of the software crisis .

From most readings, we can find five causes of the crisis :

1. There were (are ?) no cooperation between hardware designer teams and the people who are using it, the software designer teams . But, there is also no cooperation (at least not enough) between software designer teams and the users of the software .

2. As pointed out by Dijkstra, there is a problem with the compatibility that you must preserved even if these compatibility forces you to write bad quality software . It is not possible to write good software if you have to be compatible with bad one .

3. The software crisis is the result of an human limitations in dealing with complexity .

4. Most people who are working on software, do neither understand fully what they are doing, nor the means they use nor the user's needs . For instance, many people I met were not educated in order to be able to write good software, many of them were mathematicians, biologist, and so on . I do not want to say that none of them are good but that there is a problem to find good trained people . For more details about it see the book of "Software Reflected".

5. There is a very large distance (too large !) between the state of art and the state of practice . That is a very crucial point. Indeed, how will you that software quality will be improved if there exist no theory or no theory experimented on a large scale practical projects . I think that there is no mather where this occured but it is not enough recognized . I do not know any other domain where pratician are so negative in front of the theoretist . One could think that the reason why this occurs is the fact that many people believe that the cristicism of a theorician is the cristicism of this own person or work . And here is the place to advocate the need of egaless programming at any level of software development .

Let us seen what are the symptoms of the crisis.

III . 4 . Symptoms of software crisis .

One can hope that if one was able to see the symptoms, may be, can he (she) be able to see the real cause of this bad effect . The next points are perhaps not all symptoms but the most of them .

III . 4 . 1 . Responsiveness .

Too often, computer based systems do not meet user's needs . This should come from the fact that the user was not able to define precisely what were this needs or this should come from the fact that the communication between users and designers was not correct enough . The consequences is that most systems meet not exactly what the user expected it to be .

III . 4 . 2 . Reliability.

Too often, software fails . This do not say that the software was not done but not done completely or not in time . For more details see the book of F. Brooks : "The Mythical Man-Month".

III . 4 . 3 Cost .

Too often, the established cost of the software is not respected . The software costs are seldom predicable and are perceived as excessive. This is a well known fact . This include direct and indirect costs .

III . 4 . 4 . Modifiability.

It is well known that maintainance is a very heavy activity which cost quite 80% of all programmers activities . Maintainance is also complex, costly and error prone . Some time, if you need to modify you are better in rewritting it entirely . For more details, see the corresponding chapter in this publication .

III . 4 . 5 . Timeless .

Software is often delivered with unexpected delays and with less than promised capability . This is very strange that this is quite the only discipline where this is done without reaction from the part of the buger .

III . 4 . 6 . Transportability .

Software written for one system is seldom usable in another one without having to change a lot . Even some time, it is better to rewrite the whole because not all systems have all programming languages.

III . 4 . 7 . Efficiency .

Software development efforts do not make optimal use of the resources involved .

III . 5 . Basis of intended solution .

Once it was seen that both old and new software both suffer from the same symptoms . From this constatation, some people began to work on this problem . Few people were working on theoretical point of view while others were working on practical issues . One of these results where the establishment of the "Laws of program evolution". These laws were established for large and medium sized software . This means that the software will be in use during a period of at least several month and also in use for several people .

III . 5 . 1 . Law of continuing changes .

This law says that a system that undergoes continuing changes until it is judged more cost effective to freeze and recreate it . Indeed, change can be required by many things among them we can mention: environment changes, user's requirements change, error correction,

III . 5 . 2 . Law of increasing entropy .

The entropy of a system (its unstructurness) increases with time, unless specific work is executed to maintain or reduce it . In fact, whenever there is an error correction, there are more chances of introducing other errors . That is because more often the people correcting the software is not the people who has designed it and because there is no documentation precise enough (and updated !) in order to let you understand all interactions between all parts of software (I should tell you one of my experiences . I had to add some new functions in a software . In order to do it as best I could, I asked for documentation of the whole software (and its history) . What was my surprise when I heard the following response : "There was soo much documents, and none knows yet what are the papers in it . Then no one use it . So, as we have no much space and as we need more space, we put all the documentation in the dustbin .) This is an exemple but I think that this kind of story are not so rare even if it occurs for another reason

III . 5 . 3 . Law of statistically smoothgrow .

Growth trends measure of global system attributes may appear to be stochastic locally but statistically they are cyclically self regulating with well defined ranges .

III . 6 . Some other comments .

Before going further, I would like to put here other sentences quite like proverbs about software . The reason why I put them here is that they are important in order to introduce a definition of a new discipline (some people do not agree with the fact that this is a discipline) . This new discipline is commonly named software engineering . The creation of this discipline can be seen as a response to the software crisis . This response should provide solutions to the software crisis or at least to reduce the bad effects of the crisis . These proverbs refer to the invisible part of software :

- There is NO one right set of statements to realize a software
- Software is abstract and therefore difficult to deal with (no exemples are possible)
- Large software cannot be completely error free, even after years of use
- Systems requirements cannot even be fully stated in advances, not even in principle because the user don't know them in advance not even in principle .

Some people say it another way : "A bug free program is an abstract theoretical concept."

What has been said until now seems to drive you to the conclusion that there is nothing to do again the software crisis, but that is not the goal of this . The goal of these lines about is to prevent us to talk too easily of correct programs without knowing that that is a theoretical view . It is time now, I will try to give a definition of what is software engineering, on what is software engineering based and what are today results of this discipline . First of all, we have to see that this discipline has the aim of finding a response to the crisis .

III . 7 . Software engineering as solution .

As there is no established definition, I will keep the next one as the accepted one.

III . 7 . 1 . Definition .

Software engineering is a set of structured ways of performing traditional software development activities and a set of structured activities that are not part of the traditional software development process .

III . 7 . 2 . Some comments .

What is named traditional activities is the activities implied by the fact that some problem was to be solved by machines . But this was done without all the present ways of doing it . There was nothing as structured programming, top down programming, and so .

From this definition, we can conclude that software engineers will be able to give anyone tools and principles in order to increase quality software production . Yet, we have introduced a new concept : quality software . It is a beautiful word but how to establish what quality is when speaking about software . For this purpose, I have chosen the five most important (for me, there are no other) criteria in evaluating software quality which are modifiability - efficiency - reliability - understandability and correctness . These criterias are not easy to realize . Indeed, many times, if you choose to increase the modifiability as the most principle component of your software quality, you will find that understandability and/on efficiency criteria are than no more respected . Let us examine more closely these five criteria .

III . 7 . 3 . Modifiability .

Referring to the law of continuing change, we can say that a software (except short-lived software) is always changing . These changes are due to a number kinds of facts . Among these kinds of facts, one should mention the following . The users change their requirements on the modeled computer world changes on the software environment changes . We know that all these changes occur many times during the complete life-cycle of a software . Many of these changes are not known when the software were designed and constructed . Then, in order

to create software with great ability of changes, we can see that the idea of modularization of software . This is done in order to have to modify only one module instead of a few statements within all the software . But this has also led to the problem of how to cut it in module and worse how to deal with interfaces changes? That is not solved even with language of the 80's as ADA .

III . 7 . 4 . Efficiency .

As software development becomes a more common practice, (that is the fact that software begins to cover many disciplines) real-time software has become needed and also were interactive ones . This has as consequences a much more complex activity than having to deal with the old batch processing . Efficiency has changed from saving spaces because the first series of machines had no many space enough in order to deal with large software, to saving time because of the new systems(networks, teleprocessing, ...) are more time oriented the space saving . The problem now is to increase understandability of program without losing good response time . All this should be done without having to write in low level languages (as Assembler) and with preserving portability, readability and of course the facility of maintaining, testing and debugging this software . Some people say that efficiency is not a problem, I do not agree with that but it is right that efficiency does not have as consequences to loose other criteria .

III . 7 . 5 . Reliability .

This has two meanings: - The first one is the collection of all the techniques which can be used to design and test the software so that it is relatively error free . Some people call these the techniques which apply to the design of reliable software .

- The second one is probability that a given software system operates for some time period without software errors, on the machine for which it was designed, given that it is in used within design limits (see for instance the IBM DOS/MUS, which normally does not crashed) Therefore we must define what is a successful operation of the software what is in practice a non trivial task .

It is not possible to do verifications with traditional program construction . Now, when you construct your software at the same time you think the way which are possible to take in order to verify it .

But how to do it when none of the languages accept to have dynamic interfaces and few of them even accept static interface checks (the only one is ADA, because most Pascal compilers are standard in the sense that they accept external compilation but without any check !)

In order to ensure software reliability, you must ensure software debugging, testing and maintaining . And once more, you must deal with interface where the worse bugs are .

III . 7 . 6 . Understandability .

The language are use as vehicule for expressing a solution must be easy to read and understand for both users and the concepters . In order to realize this some people has proposed to define a formal language in such away that both users and concepters can communicate together in a proper way without confusion or misunderstanding . This implies that one has to know a lot of these languages ...

Some other people see specifications as a means in order to meet this goal . Another way to realize this, is the use of comments and documentation . We define comments as not just paraphrasing what the code does but as explaining what it is intended to do .

In my experience (a short one but ...) this means that it is not good enough as solution if there is no pressure or not heavily enough in order to update them . Whenever I have to deal with assembly coding, I read first the comments in order to see first what it is attended to do before trying to see how it is done . But what was my surprise to see that the coding did not match the comments . When I saw this, I wanted to know why this occured . The response was that there was a lack of time . As we will see later, the lack of time is often a great ennemy of all software engineering effort . As mentioned before, this also occurs with documentation . These are the reasons why I see specification as this means .

III . 7 . 7 . Correctnes .

The starting point, here, is the fact that a software without bugs is a pure theoretical concept . In all practical example, you will never be able to establish that there are no more problem . Once again I can take the interfaces example, in the main program the variable was declared as real and in the subprogram they were declared as integer . Before execution time, there were no problem but at

execution time the results were not correct . During the testing period, this was not seen and how to ensure that can not occur ?

At testing period, you can never prove the absence of bugs, the only thing that you can prove is their presence . This of course is not a reason to stop trying means or stop researching a solution to decrease their numbers ! It is important to see how you can avoid some bad mistakes, how you can write software in such a way that it will easier to handle .

Another problem is to define precisely what correctness means . That also is not a trivial task . A most commonly accepted definition is that a software is said correct if it meets its specification . This as the next consequences :

First, this implies that you have to define what specification is, there is no commonly accepted definition . But supposing that this can be done .

Second, you should be able to prove that the software does exactly (no more but no less) what the specification tells you the software should do . Suppose that this can be done .

Third, you have to prove that the specifications describe exactly the expectation of the user . Because otherwise it is not usable and loses its interest . That, of course, is quite impossible because often the users do not know themselves what they need !

III . 8 . Some existing solutions .

In order to give solutions, some people have given ways to increase software quality but too often these solutions are partial solutions but are seen as Solutions . These methods are mainly : Structured programming - Top down design - Step wise refinement - Systematic testing - Inspection of code - Programmer team and a few others . These methods are be coming to be used in practice . But often they are not used or not completely used because they are time consuming and involves the training of people . In the next section, I will explain one of the most heavy pressure against their uses .

III . 9 . Reaction under Deadline Pressure .

The problem, here, is to deal with time . In all methods I know, this fact is never handled and even never mentioned . But deadline pressure is a fact of life in the real world of software development. It can never be eliminated so that methods without speaking of time are usefull only for theoretical purposes. But if we want to give an answer to it we must learn to live with it . If people see that they will not have finish in time what will they do ? There are four answers and I will briefly explain each one .

III . 9 . 1 . Eliminating the deadline .

This is the first answer, but I think that is never used because once an organisation has done a planification of its activities it is not easy to change them . Moreover, how to do it without giving the impression that you have done no work enough or that you are not "usefull" . Then, nor the programmer nor the manager will accept this solution .

III . 9 . 2 . Not Completing all the work .

This is not to realize all promised functions . But the problem is to deal with the reactions of the user . If you are a society, you can not be able to do it without loosing bugers.

III . 9 . 3 . Not completing software engineering purposes .

That is the easiest solution because no one can see if all software engineering steps were accomplished . The only time when this should be seen is when the software is yet in use and at this time it is no more possible to do anything else than to maintain it even this increasing cost of maintainance is greater than if another solution was choosen .

III . 9 . 4 . Increasing the software team productivity .

There are no many ways in doing it and it is not possible to do it many time . The first idea is to increase the number of people working on the project . But as explained in the book of F. Brooks, "The Mythical Man-Month", it is often worse to do this action than to accept the delay .

The second idea is to convince the staff to work harder but if they are working yet at their maximum it is not possible .

The third idea is to force the staff to work much more hours than usually done . But this leads to two consequences: first there are extra hours that cost much more than a normal hour and that after short time they are tired and do not work more efficiently .

The last one is to acquire or develop tools that genuilly increase productivity but this involves a loose of time (which is yet too short) if developping it and a loose of money and time if acquiring it (+ training).

III . 10 . Why is the reaction, reducing software engineering efforts the most commonly used ?

I think that there are mainly four reasons in doing so. First we should mention the fact that the benefits of software engineering efforts are delayed and diffused . There are no means for mesuring it precisely . Second, the decision can be made without knowing the consequences of this decision . If one has to inspect the coding but is late in his (her) own work, it is a normal reaction to first work on his (her) work before working for someone else . But this person should of course never say that he (she) has no readeed it . Then it is suppose to be done with all consequences that are involved. Third, the software engineering efforts is not seen until the software is used for a certain period of time . It is only when maintaining or debugging that these efforts can be appreciated . And fourth, the system fails to reward people for this software engineering efforts . Too often what is heard is keep on schedule. but quite never keep on quality is heard .

III . 11 . Some concluding remarks .

As seen, we can establish that there are three principal ways of increasing software quality : Testing it systematicly, debugging it whenever test has shown a bug but doing it carefully and maintaining which is much more error prone than creating new software . These of course are not independant each one from the others . They will be explained in the next part and their relation with my study will be explained .

Part 2 : Definition of the Problem and the Relations between the

Problem and the Software Engineering Purposes

I . Definition of the problem .

I . 1 . An informal definition .

The problem under study is to find a mechanism which should permits to handle dynamic interface . The reasons of this need is that once operating system are created they evolue . For instance, if one model in an old version (but yet supported) has many bugs and if in a new version these where corrected, it should be great if one would be able to change the faulty one without having to recompile and relink the whole .

What has to be done ? First, it has to be able to do the type checking between two modules which are externally compiled (not compiled in the same compile unit) . This is now quite never done . Second, it has to be able to handle a permitted difference between two parameter list . Of course, it must be able to distinguish between an erroneous call and a correct call with a different number of parameter (or different types) .

I . 2 . A formal definition .

Formally the problem can be describe as follow (using a Pascal-like syntax)

```
proc Y (P1 : T1; ..... Pn : Tn) ;
    .
    .
    .
end Y;
```

and the new procedure Y' which is intended to replace Y in the software :

```
proc Y' (P'1 : T'1; ..... P'm : T'm);
    .
    .
    .
end Y';
```

It must be pointed out at this point that the new types $T'_1 \dots T'_m$ have to satisfy definite constraints if we want old calls throughout the software remain valid. For example, if we replace `Add (X1 : real; X2 : real)` by `Add (X1 : int; X2 : int)` without changing or recompiling the module containing the old call. The new module will add without any care the two machine words (implementation of 32 bits machine word for both types real and int is assumed) adding both mantissa and exponent leading obviously to an invalid result.

There are three possibilities.

I . 2 . 1 . m = n

In this case the type checking will return an O.K. if all $T_i = T'_i$
 $\forall i : 1 \leq i \leq n$

This involves that a subrange is not seen as correct even if logically it would be possible to do it, but this adds the problem of seeing a subrange of a type as the type itself for the checking purpose.

The T_i and the types associated with the P_i which are the parameters. The type defines the correct value than the parameter can take.

I . 2 . 2 . m > n

1) $\forall i : 1 \leq i \leq n : T_i = T'_i$

This constraint is added in order to avoid the problem stated in the example. For illustration let consider: `Add (X1 real; X2 : real)` to be replaced by `Add (X1 : real; X2 : real; X3 : int)`. It is obvious to see that it is not difficult to be sure that old call will continue to be valid.

2) $\exists i : 1 \leq i \leq n : T_i \neq T'_i$

This is more complex to handle. Indeed, let take the example above: `Add (X1 : real; X2 : real)` to be replaced by `Add (X1 : real; X2 : int; X3 : int)`. It is of course not so easy to see how to handle this with care in order to error that old call will remain valid. It is of course the responsibility of the people who replace the software to be sure that they are compatible. It is of course not possible ever at operating system level to see if they are compatible or not.

I . 2 . 3 . m < n

1) $\forall i : 1 \leq i \leq m : T_i = T'_i$

This is quite easy to handle when the $m + 1, \dots, n$ parameters need no more to be used because, for example, the hardware have changed.

Of course, when it is not the case, the responsibility of validating the old call is of the changer .

2) $\exists i \ 1 \leq i \leq m : T_i = T'_i$

This is not easy to handle, and , in all cases, the responsibility of the people who introduces the new module is complete . No systems can be responsible of a problem of this kind of compatibility .

I . 3 . Some remarks .

Here the way to handle this problem is not important, but it is important to see exactly in what the problem consist and from this to see what are the necessary responses in order to find solutions .

As seen, all is not done only by the computer, there is quite always part of responsibility of the user to respect the conditions in which this can be done .

Let us see now what are the relations between the solution of the study and the three main activities : Testing, Debugging and Maintaining .

II . Relation with Testing .

II . 1 . Introduction .

Why do I begin with testing ? Because, once you have finish to write something like specification, coding, ... you want to see if what you have done is correct before doing other thing or before you want to enter in deeper detail . For doing it, one of the more useful technic is testing it . But testing ? It is not an old technic ? Whenever you read a book about software, you will have the illusion of the no-necessity of program testing . In fact, in more or less all books, you will read that when you are written your system you have to write in the same time the correctness proof . Then if you are able to write the proof of the correctness of what you have done... Why should you test it ? And then you are reading written by some practician and you will discover that it is never possible to prove the total correctness of a system even an easy one . For more details see the book of Myer G. J. : "the Art of Software Testing" . In this book Myer writes : "Testing is the process of executing program(s) with the interest of finding errors starting with the Assemptor that the program(s) CONTAINS ERROR !" Said in another way : a bug free program is an abstract theoretical concept. From this point of view, a test case is successfull when finding a notyet discover error and not as in almost every day testing, the goal of test is to prove that than is no more bug . In fact, testing will NEVER prove the absence of bugs only can it prove their presence .

II . 2 . Definition .

Testing is one way (the other one is verification) of validation of software . Testing explores a large number of the possible exception historics of the system in order to fine as many not yet discovered error as possible . Therefore, one can define a good test case as one having high probability of detecting an as yet undiscovered error and a successfull test case as one detecting an as yet undiscovered error . Too often, when one is testing some part of software it is doing it with the idea to prove that the software tested is correct, instead to start with the idea that it is not .

II . 3 . In which context testing should be done ?

The first part of this is to see what is a test case or a good test case . A necessary part of a test case is a definition of the expected output a result of this test case . Another part is the fact that a test case must be written for invalid and unexpected, as well as valid and expected, input conditions because examining a program to see if it does not do what it is supposed to do is only half of the battle . The other half is seeing whether the program does what it is not supposed to do . In other words, a test case has to establish that a program does what it is supposed to do but nothing more, if a program does more it is also an error . For these reasons, a program should never be tested by its programmer because too often, the programmer knows too much about his program and will not be able to create an effective test case because if he had already know what he test, he had already written the corresponding code . As modification of a program is error prove (more than writing a new one), we should never throw away test cases unless the total program have to be thrown away . Because after each modification we will have to retest all the software . An other principle should guide the decision of stopping testing the software estimating that no more error will be found with a acceptable economic price . This principle is the following; the probability of the existence of more error in a section of a program is proportional to the number of errors already found in that section . This can guide the decision of what test next . This principle is based upon a common experience of existing software when more or less 80% of bugs are in only 10% of code . Testing involves finding that an error exist not located it nor correct it, that is the role of debugging . Testing is also the only method for checking not the correctness of the program but checking that the specification corresponds to the requirements .

II . 4 . Different kind of test and the error they tend to find .

It seems to me that there are three level . The easiest one is the level where we are dealing only with one single module on program . The middle level is the level where we are dealing with

a complete software (this software can be seen as a logical unity from user point of view) . The last one is the level when we are dealing with the specification but this level is quite not a "computer job".

Of course, all these level are complementary and are based on different degree of abstraction . Indeed, at the first level, we are only dealing with the statements or set of statements . But there are no problem as synchronization or as mutual exclusion or as static interface (and of course as dynamic interface) . There are only three kind of actions : covering all statements, covering all possible decisions and of course all possible conditions (even when they are multiple conditions) . This is the basic test . This has to be done but it is not so difficult to prepare test cases for all possible inputs (correct or not) and compare the expected output with the real one . Once this level is estimated as correct, we go further and put together a set of these tested module in order to realize a useful software . This level is, from the point of view of this study, considered as the most important because it is the level where the problem of interfaces are. Indeed, you can consider that there are three kind of interfaces : one kind of interfaces is the user interfaces, another one is the interface between internal module of the software and the last one the interface between the software and its environment like operating systems available, tools available,

Here static test is no more enough . Indeed, here the time is playing a great part of the whole . We must than add the dynamic . The test cases are no mare playing with a module but with more than one . This is much more difficult because the way they are executed and the time at which they are executed and also the order in which they can be combined have also to be tested . That is once more impossible to test all possibilities but it seems that if the interfaces could be more precisely defined and checked much of the error will no more occur .

The last level is also not concerned by this study but is mentioned in order to give a complete overview of what testing involves . This level must deal with specifications . The problem is how to be sure

that they are correct . It is not enough to prove that they are consistent and complete because this does not ensure that they meet the users need what can be different of what the user says that he needs .

II . 5 . Relation between the interface problem and the difficulties in finding bugs today .

II . 5 . 1 .

As pointed before, most of bugs remain due to the interface problem (lack of checking) . The reasons or most reason I see to this, is that most of time the programs on modules are not written by the same developers (or developers team) . This involves communication problems . As my short experience has yet showed to me, if the interfaces are only explained by Assembler DSECT, with a few explanation about what is the reason for checking this function and what is the return code (or result) if something is not correct, this is not easy to communicate and of course this leads to ambiguities . This is often because there is no tool (or no easy tool) for insuring that the interface is seen the same way from user as designer point of view . Moreover, as the normal evolution is to go from Assembler language to higher order languages, it seems to me that describing interface as DSECTS is not a natural way of doing it (no many people will know Assembler) . Worse, due to the fact that a team has a deadline too short (lack of time!), what is quite always done (and in the practice, it is what I have seen!) is to offer the lesser that you can and if the users want more they have to tell it to you . This is not a reaction that will insure a good communication more insure that things are seen the same way . At the interface, you are also dealing with the worst practice : the shared data (common Fortran; ...) which are manipulated by both the user (caller) and the module (callee) and this can lead to many problems . An other bad practice is the possibility of using defaults and once again at the interface you must be sure that defaults are handled properly . This is of course an important point when dealing with modules written in different languages for any reason .

The aim of interface testing is the validation of the fact that the basic algorithms and routines are tried together correctly .

In order to achieve this aim, let see what the advantages that we will have when the dynamic interface problem is solved .

II . 5 . 2 . Bad parameter passing .

If we except strongly happed language as ADA and as PASCAL (are there any other ?), there are no check done, in order to see that what is expected by the caller is well what the caller sends . In most languages, no checks are done at this level nor at compilation, nor at like-time, nor at run-time . This has as consequences that the way of checking the parameter passing is correctly done does not exist or if it exist it must handled by human means what is so costly .

In response to this, we will search further how to do these checks .

II . 5 . 3 . Dynamic problem .

Even with a language like Pascal (as representing the calass of strongly typed languages), it is not possible to deal with subprogram call like I/O routines or easy arrays handling. For example, if we want to write a program for matrix inversion, we want of course to write one module to do it for different sized arrays . In Pascal, it is not possible because variables are to be declared before they are used and this involves that the arrays boundary are fixed . It should be very interesting if parametric typed arrays will be available (what is lesser than the dynamic interface problem) with a high degree of confidence . As standard Pascal does not permit the use of separated compiled programs and that is too restrictive, most Pascal compilers available permit it but at price that the interfaces are no more checked and than we can no more speak about strongly typed languages .

The response to this is to find a way for handling dynamic interface but without loosing the checking of parameters .

II . 5 . 4 . Evolution problem .

Whenever you want to test a new version of some modules, it should be an easier task if we would be able to put it in previsions version of the software and see how it runs (when we pass from on release to another there are no so many things that are changing and at least the basic function remain to be handled even if it is not handled the same way) . Of course, if for doing it, you must deal with the necessity of recompiling and relinking the whole software, it is not

possible to do it . Now, because of the problem of the lack of testing mechanism, there is no other safe way to do it . When the problem here under the study is overcome, it should permit to test a new module in two steps .

First step, the checking of the way the new module handle the common functions of the new and the old software . This should permit to check more or less 80% of the coding of the new module .

Second step, as it is done now, to check only the new functions but that is only testing 20% of the coding what is much more easy .

II . 5 . 5 . Integration problem .

Whenever a great software has to be changed, several people are working each one on one module or set of module . As said above, one should test its module in putting it in the old software, but it remains then the last part which is to put all modified modules together . Whenever a mistake is found in the interface it should be changed without having to recompile all other modules . Even when using Ada it is the case now .

As response, the solution will provide means to enforce interface security and interface flexibility .

II . 5 . 6 .

In order to test some module or set of modules, one must often write a lot of short programs to verify if the module(s) under test is (are) running correctly . Indeed, in order to simulate the different calls, we put a few typical input and compare when returning with the expected result . But often, the errors seems are done in this caller program and not in the tested one . If we were able to handle variable interface, it should be easier to handle this because these errors would be seen before running them and not at runtime .

II . 5 . 7 . Some concluding remarks .

As end of this chapter, I hope that I was able to convince the reader that testing is a mandatory activity and as it is needed and thus, realized, all solution that facilitate this activity should be encouraged .

As explained in this chapter, the most difficult bugs to find are bugs presence in the interfaces . And that is precisely the aims of this study : a help for handling interfaces in both ways flexibility and strictness .

III . Debugging .

III . 1 . Introduction .

Once a program (or a set of programs) is in testing phase and once some errors have been found, we have to locate them and afterwards to correct them . The problem of localizing bugs is not a trivial task . Indeed, once some mistakes are seen, the origin of the mistakes can be situated long before the error is signalized . When we have some luck, we have a crash (and a dump) . It is a lucky case because we can see the value of the variables and so on . In the case, where the results are incorrect . This is a more subtil situation because you have no traces of when is this incorrection come from you have no more values in the variables . Once the faulty statement(s) are detected they must be corrected . But doing this correction, we must be sure that we are not introducing new faulty situations. This is not easy because, most of the time, the "corrector" is not the developer and often any of the members of the development team is no more available and than if the documentation is not up to date (which is quite always the case), some decisions were taken without explaining why . As results from this fact, sometimes the debugging process has no termination . This fact was due to the "fantom" bugs that are bugs which appair from time to time . Another situation is when the bugs are localized but as their correction is so costly and as they do not appair too often, no correction is done .

After completion of the debugging process (with some corrections done), the software is once again given to the test team . Normally at the same time all documents and comments were also updated so that they are ready for eventually further use . Finally, the corrected software is given to the user (or only the correction to be done).

III . 2 . Definition .

In order to define what debugging consist in, I will first begin to give two ways of seeing the debugging process . For being sure that we agree with what I consider when speaking of debugging, I will give what actions constitute the different parts of the debugging process .

III . 2 . 1 . Theoretical definition .

The first way of seeing the debugging process is to see debugging as :

Debugging is a process of creating models of actual behaviour from the activity of a system and comparing these models of expected behaviour hold by implementers and users of the system .

This way of seeing involves at least that users' models was correctly understand by the implementers and also was correctly implemented (that is not easy when the system does not handle fully functions) . In practice, the users have no model and also too often their model are not understand and as consequences the implementers have not the same model and sometimes they implement another one . This is what often occurs because there is no tool to handle this . For explaining this, I will give only one point . When a user defines a model, to define it in his own technical jargon and the implementations are doing the same using compile jargon and this can lead as some words can used in several sense, to ambiguities .

These facts have convinced me that I should try to find an alternative definition . The following definition seems to me to have a more practical issue .

Debugging a program is performing queries and updates on a data base the contains program source informations as well as the state of the executing program .

As stated in this definition, this involves the fact that the debuggers need to a data base where all program informations are . This has as effect that debugging is not an activity without data base. What of course is not the case . Debugging process is always present whenever a bug is .

As none definition seems to be complete or precise enough, I will give all activities that the debugging process must perform . And in the following part, I will always threat debugging in term of these activities .

III . 2 . 2 . Activities involved by the debugging process .

There are three subtasks involved by the debugging process :
fault detection, localization and repair .

- 1) Fault detection : before testing, refers to the discovery of discrepancies between expected and observed program behavior. Indeed, when we write a program, we try to execute it . And there is only when it seems to run correctly that the program is passed to test team in order to test it widely and intensively .
- 2) Fault localization : That is the process of identifying when in the software (at which statement of which module) is the cause of the anomalous behaviour of the software . This process is often the most difficult one in case of large software or in case of intensively used module .
- 3) Fault repair : That involves the editing of the statements to eliminate discrepancies . That seems to be the easiest part, but in case of a bug in a very often used module, it is not so easy because the correction involves consequences in very different software . This has as consequences the birth of new bugs . Fault repair has to be done very carefully with full explanation of what was the origin of the bugs and how the repairing was done (even may be the name of the corrector) . All these informations will be of interest when maintaining or debugging again the software .

As behavioral research has shown, fault localization is the most difficult of these subtasks, painding imputers for researching methods to increase the tools efficiency .

As it will be shown further, it is also the place where dynamical interface with type checking will have a great influence in facilitate fault localization .

III . 3 . In which context should debugging done ?

III . 3 . 1 . Debugging environment .

Actually, debugging environment is often too poor . One reason of this is the fact that when anyone is starting to write some software, he has a great confidence in its ability to write it without bugs . For instance, during my practice, one has told me this story : "When all things seem to run without problem, there is always no reason for giving some money to people who are writing debugging aids for users as well as for developpers . But once there is a great problem then the team should be able to provide debugging tools in a very short time because of the deadline to be respected . What would be are seen done ? Something very easy, at the beginning of software design the necessity of debugging tools have to be inserted in the design . In summary, when constructing the original large software (moreover when operating systems or compilers), one needs to think about providing facilities that will enable anyone to provide debugging aids more easely . For more details, see in the annexe, the part reserved for (AID/AIDSYS) software . That software provides debugging aids but were not included in the design of the operating system in which these aids have hat to be included and the problems coming from this fact . Another challenge of debugging systems is that they should be so simple that their features should be learned in a few minutes, especially when these few minutes occur when the user is under stress and also should debugging tools be invoked more selectively (because, by instance, an expert does not need so many details as a "novice") . One should note that too often extra code is added for debugging aims but removed when running under common conditions . That will say that the debugged code is not the same as the running one what should be avoid . Once more, the manner in which parameters are passed needs to be considered . And also look at the interfaces if, by instance, there are modules written in different languages .

III . 3 . 2 . When is it complete .

Debugging process is complete because of two facts . The first one is the good case when fault was discovered and corrected and software seems to run none as it has to . The software needs now to return to

the test team . The other one is the bad case when fault was not localized (one tentative is to try and let see what happens, that is not a solution !). And therefore no repairing action was understak . The unlocalization will come from the fact that there is no time enough for doing it or because there is no information enough about how did the bug appear . The other part of this bad case is when the bug was localized but that the rewritten of part of software is so great that it is too costly . During my practice, I have heard of such well known bugs (infinite loop) but they know that the bug appeared not frequently and if one wants to correct it, all the kernel of the operating system should be rewritten ...

All these are the common environment of debugging process; even with new environment as ADA, LISP or so , these seems to be no improvement or so costly !

III. 4 . Different kind of debugging level and their consequences .

III . 4 . 1 . Handling one module written in low level languages .

This level is not often used today's because there are not many programs written in Assembler like language . For debugging purpose, you must deal only with dumps, traces or errors code . I can consider that it is not so difficult because when you are working at Assembler level, it has as consequence that you know quite very well the machine and also you are not concerned with default . It is also may be easy because as you are concerned with dumps and traces, you can see directly the statements assembled both at the screen in the dump and also in your listing . This of course is not the reality when writing in higher level languages . Normally, a module is not doing many tasks and there are no interactions with other program at this level, it is not so difficult .

III . 4 . 2 . Handling one module written in high level language .

This level is a little harder to handle, because if it is easier to write it and understand what it is supposed to do, it is more difficult to deal with dumps, defaults, ... Indeed, too often the dumps and traces do not handle symbols . This will say that when dealing with dumps, you are dealing with the Assembler expansion of what was the statement in a high level language . It is of course

not easy for people without assembler knowledge, to deal with this kind of tools . Many efforts have been done in finding adapted tools for handling this . Another way for handling this, is to do much efforts for preventing more bugs presence .

III . 4 . 3 . Handling a set of modules .

Here the difficulties added on the handling of interfaces and the handling of time .

III . 4 . 3 . 1 . Set of modules in high level language .

The possibility of cutting a large software is one way of having more facilities in solving a problem . At debugging level, as only one module is concerned, it is easy to handle because a module is quite short and the fault localization is easier . In order to see if we can only search in one module, one is concerned with verifying that all things were correct at beginning of the module . This needs the ability to be sure that the call was wright (system call or user call, I take for this the IBM CMS/VM vision which treats always a program as subprogram of at least the system if it is a main one) . But now how can we sure that it is the situation is which we are ? Indeed, most of the languages even some recent (and presented as wonderful!) one as C does not check anything at run time and worse, they they are not able to handle it at compile time . In other words, there are no checks done at any moment (behalve at design or specification time), in order to prove or to be sure that the call is correct . That will say that even if there is no problem found in the calling sequence, as it is never checked, the problem will only be found during the execution of the body of the program (by instance, the fact that there is some division by zero can be the result of not passing the expected parameter at the place where the module thought to find them, in this case the error discovered is the division by zero but the problem has as origin the fact of wrong parameter passing) . When the problem stated in this work will be solved, check will be done at least at compiling level as it is in strongly typed languages . Then even if checks are done, the problem is not totaly suppressed . Indeed, if one gives a correct value but wrong in the sense that not expected . But if this value has to be seen as error at check time, the error will be discovered at interface location and not after .

In this sense, if you will have, guarding the same example, a divide by zero value, you can be sure that at passing parameters time, it was correctly done and you have only to consider one module and not the module and its interaction with the other .

III . 4 . 3 . 2 . Complex module .

When dealing with a very complex module (either because realizing difficult functions either because it is very often used and in a lot of situations) . Traditionnaly, the debugging process is to analyse the module and once something that looks strange is seen one corrects it and try to see if that was really the symptoms of the problem seen when saying that an error was discovered . The more useful sentence in order to illustrate this is TRY and let see . What is of course often worse than analysing it in some systematic way . But the complexity with one has to deal is sometimes so great that no one sees a formal way to solve it .(Keeping on mind that there is always a deadline process) Now what difficulties are added ? I think that this kind of difficulties is the most difficult to handle because interfaces problem or interaction problems are not state problem. That will say that you have to deal with dynamic problem which can be far from the discovered bug . At crash time, you can see the problem,(by instance, dividing by zero) but why is this variable containing this value ? That is an other question very more difficult to establish . Than,at this time we must deal with interactions, here dumps are not very helpfull, traces are more valuable, but traces are slow and once more, we are dealing with machine code and registers . If you will write it in a high level language and must deal with traces at this level, I think that we will not be able to solve anything . Therefore, was established symbolic traces but there are no tools in tracing directly high level language statement (that should involved a trace for every language!) . And then, finally, we come into the higher level debugging process . It is of course the worse handled level . We should write a large software using different language by instance, part in a high level language, part in Assembler and worse part in one high level language and other part in another (even another one is a third one, ...)

III . 4 . 4 . Concluding remarks .

These reflexions lead me to this conclusion : Nobody really knows what to do with bugs in complex system even if there is a number of ideas of systematic testing and by careful program structuring associated with some systematic view of the segmentation of a composite system into subparts and of the degree to which these subparts interact. Such segmentation would be interesting, we have none . It seems to me that they are the reasons why the idea of the theoretical EL DORADO of debugging (the idea of proving program correctness by formal mathematics methods) is still alive . Why is this only a dream ? First the general problem of proving algorithms equivalent . This problem is in the technical, mathematical sense unsolvable . This is bad enough, but still worse, the algorithm equivalence problem is unsolvable in a very strong sense . There does not exist any finite set of axioms from which proofs can be elaborated to cover all possible cases of algorithm equivalence . And worse in order for such assertion to be relatively decisive, the proofs to which one refers must be checked by a formal algorithmic mechanism and has to enter in an infinite loop . But in order to be not so pessimist, I have to say that all these research has lead to a number of valuable suggestions as structured programming, module and data design ideas and the concurrent processing and synchronization idea (see references for more details) . All these suggestions have lead to increase software quality but more has been done in order to eliminate the software crisis (if possible) .

III . 5 . Which part of debugging is the most difficult ? and their relations with this study .

It seems to me that the most difficult part of a debugging process, as said before, is to handle the debugging process with large and complex software . As often, these software are written with different languages, each language having its own convention of representation of variable, and of the interface communication .

III . 5 . 1 . Interface enforcement .

When the problem will be solved, you should be able first to see if all calls were done correctly . That is the fact that if any caller executes a call in a wrong way this call will be flagged before execution . That is quite easier to handle than the today situation where value are interpreted as correct without check . This, of course, gives you a easier way in detecting errors instead of having to deal with erroneous results .

III . 5 . 2 . Handling complex software .

If we want to replace a complex module (not complexity because of the environment but the coding complexity) with a lower complex module, for example, if we have found another algorithm for doing the same and suppose that the new algorithm does not require so many parameters. Now, how would you do it ? If it is replaced in an existing software, what will be the result at the interface and when called? Thus, you are loosing many advantages of finding a new solution . What I propose to do is to find a way of both checking the parameter list in order to verify the correspondance between parameters and giving a way of handling different number of parameters . Then, solving this kind of problem .

III . 5 . 3 . Handling large software .

If we want to realize a large software, it is a good idea to cut it in pieces (called module) . But the problem is that there is no means in order to see how to cut it correctly . Whenever a module is designed, there is a interface between it and the others . Then there are many interfaces and as seen, the interfaces are now not handle by computers (not enough nor easily) . This study should provide useful mechanism in both assuring that the interfaces are correctly handle by both part (caller and callee) . But a software is always evolving and what is often done is to add new modules or add some new functions to existing one . This actually involves recompiling and relinking of all the software concerned by this module . Then, when the solution is found, it will be possible to deal with this situation without having to recompile the whole .

III . 5 . 4 . Integration .

If we are dealing with integration of a set of modules, now we usually are confronted with the fact that the parameters passing mechanism is not safe enough in order to insure us that there is no problem . I mean by that, that if in PL/I, I pass a parameter declared as real in the caller and that the corresponding parameter is declared as integer in the callee . When executing the call the string of bits will be interpreted as integers representation, this will not lead to an error but to erroneous result . The problem is how to see that the erroneous results are coming from this ?

Once the problem is solved, this could no more occur . Indeed, at the moment the checks are done, this will have been flagged . Thus, whenever an error occurs, we are sure that this is not at interface level . This can help a lot because you can act as all modules are independant because all interfaces are correct or are flagged .

It is quite easier to handle module one after the other than having to handle the whole system and the interactions between modules .

III . 5 . 5 . Some concluding remarks .

1) As debugging process is not easy, it should be possible to add tools for helping the debugger whenever it is a debugging process . As these tools are very time and space consuming, it is not possible to let it always present in the system . It should be possible to convert the production environment into debugging environment . This should be easier if the flexibility of interfaces are great . Indeed, for having more debugging functions, we can have as solution to have different module (from the level of help given) . This of course should needs additionnal parameters for handling the new functions. At this time, it is only a dream but ...

2) The solution will than have as good effects to permit more security when we are using interfaces and also more flexibility . The only problem is to be sure that there exist an economical solution . Indeed, if the solution involves both time consuming and space consuming and moreover, the rewriting of many things, it is not a solution even if computers are becomming each time more powerfull .

IV . Maintainance.

IV . 1 . Introduction .

Maintainance is a part that always exist when some software has a working life of few month or more . Whenever one write some software and of course it is the fact for large and complex software, one has to think of the maintainance problem . Maintainance is the most costly activity in the all informatic activities . It is not easy to see what is exactly maintainance . In this introduction let say that maintainance consist in all activities that a software needs in order to be usable . This is presented at third position because some of the maintainance activities are constituted by debugging process and sometimes also by testing some extension of an existing software . That is also strange that all these three kinds of most software engineering activities are so interrelated . But each is situated in a different context, maintaining is more a managerial process which involves much knowledge of the all software and also knowledge of how dealing with the process of software extension or evaluation. But there is no formal or theoretical view of what should be done in order to improve further facilities for developping or extend the software . Maintainance have to deal with so different kind of knowledge as hardware knowledge because during the life cycle of a program especially when dealing with operating system the evolution of the hardware will also involve changes of software . Then, some people responsable of maintainance should have a multidisciplinary knowledge (economy - gestion- planning - informatic) And there is no such kind of study . Then often the maintainance will lead for a long time (as seeing from now) into problems because there is nobody really educated in order to accomplish this function . Moreover, maintainance is not often (if sometimes) considered at design time when the software is created . Indeed, as for debugging purposes, maintainance has to work on software which are very difficult because of some practice (as patching) . These practices are very bad both at debugging and at maintaining step and that is very strange because both debugging and maintaining are mandatarly activities which come from the fact that a software exist and is not coming from any artificial need .

IV . 2 . Definition .

The definition of maintainance is quite simple but this definition implies many different kind of activities . Maintainance is the set of all activities needed in order to assure the viability of a software .

In order to have a more precise idea of in what consist the activities of maintainance I will define the most usual part of maintainance activities, I will divide maintainance activities in two parts. These part constituing all activities needed in order to adept the software of an evolued environment . The part required by extending the software in order to meet some new user's requirements .

In the first part, we will be concerned with debugging all not yet discovered errors so that the software will be corrected and then runnable . Another part, is adapting the software in the new environment . That is either the hardware has changed and this change needs to be adapted to software because without changing, the old software is no more able to run either the tools have changed (tools as compiler, operating systems,) . These changes have as consequences the modification of some interfaces or, sometimes, the modification of some module .

In the other part, we will be concerned with adding some new functions that were only useful at some privileged user . By instance, for AIDSYS, I was confronted with offering some functions that were not available for "normal" users but which were disponsible for system developpers . This is also seen as a maintainance activity because otherwise users would not like to one or even worse will not use the software anymore . In the same kind of activities, there are some additional tools or some additional functions that should be added to a software because of debugging aid or any other tools like that . This is not the same kind of activities that are implied for developing new software because when you are writing some new software, we can start from nothing that yet exist but here we are restricted in his choice because what is added has to be compatible with the older use of this software . It is sometimes more difficult because some part which need to be modified can be very old and not so easy to

handle . If it not the case if the software was well written, that is easier to add a new part without having to write all parts but only some extra coding .

IV . 3 . In which context maintainance should be done ?

As pointed before, whenever we are concerned with maintaining some software, we are confronted with very many different kind of activities . First, any maintainer should know what was the aim when this software was designed . It is quite important because this kind of information can help in understanding some reason of unexpected coding . Indeed, sometimes after a more or less long time things have changed but the code still remains . After knowing the beginning point of the study of the software, it is very important to have a good documentation in order to see all changes in the aim of the software and why these changes occurred. This is important to understand why the software does what it actually does and also why the software takes this form . By instance, why it is cutted in the way it is . When all this can be obtained, what is often not possible because at some times no traces of changes are guarded for any reason (no space enough, not used,), we can see what was and is the global context of the software and can decide wether there is something to change in order to adapt it for futur use (sometimes code was done for small machines) . More many times space problem was a very heavy constrain. This constrain can have a very important effect on the style of the software . It seems a good practice to remove it when adapting it to some new environment . Once this history has been established, we have to understand very carefully the whole software and more important the interactions between all parts of the software . This is very important because that will enable any one to do any change with all the knowledge of what are the effects of this change . Indeed, in practice too often changes are made without knowing all these interactions . The result is that once the change is made, wrong comportement of the software is its consequences . And as I think, it is the maintainance team responsability to maintain coding when all changes are made at source level instead of doing it at

object level (Patch) . Also all old code would be removed by the team in order to maintain all software parts proper and understandable . It is the reason why all side effects of programmer activity should be verified, at maintainance level, by the team in order to be sure that documentation was updated at the same time as sources was and also as object was if there is no way in practice to avoid patches . I should mention that if maintainance is so difficult to do, it is because all steps before from designing to coding was not done with the goal on mind to do it with all precautions and all features that will be needed further . Too often, maintainance has to be done upon misconceptionnal software at any level this misconception occured . Maintainance should be a easier but very easier if all steps before were done carefully. These step starts with specifying and ends with coding through documenting . If specifications (and their evolution), coding style, comments and the environment (operating system, machine, ...) are clear and understandable, maintainance is no more a software lack .

IV . 4 . Different level of maintainance and their consequences .

There are several levels of maintainance . The lowest is the level of module maintainance . The middle one is the maintainance of one part of the software (by instance, AID - AIDSYS) . The highest is the level of a product maintainance . The lowest level is of course the easiest but it is not as trivial as one should think . In fact, if one needs to maintain a module or few modules, one needs to have somebody from which he can have all informations needed in order to do its work . Of cours, if it is only a repairing action, one can do his work without having to communicate with someone else but often one is not sure that the cause of the error is in its module or in the module of other maintainer . That is not an easy situation . If he needs to implement some more functions or to modify an existing one, of course, he must have a clear paper which describes what are the users's requirements . But as any one knows, even the clearest paper must be explained because there are always some obscur point . For doing it, he must communicate with other people, mostly the user who has asked this change and the responsable of the maintainance of the product . As the easiest level it is not so easy to handle .

The next level deals with maintaining part of the software . As a software is large or very large, it is cutted in several parts each one has its own responsible . At this level, the complexity of the work has increased a lot . Indeed, now a large part of a software has to be known and also the interaction of the modules that constitute part of this piece of software . Then, we have to deal with the interaction of this part and the others but also with the parts of his piece . Many time, also one has to deal his piece with others "under" him . We have then to work with computer but also with men . That is always a difficult thing . One dealing with that work, is confronted with the fact that his men changed and then he has to forme another one . We have also to be sure that the work of the level spoken user is done correctly in such a way that it becomes not heavier to maintain in the futur . That is an incredible more difficult work and as it is at a higher level it is often a level where quite no coding is done by him but much more administrative work . Sometimes also we have to see with the users if they agree with this part of software or if some changes are needed . If some changes are needed it is a responsibility to give this work to one of an crew .

The highest level is nothing more than the one described just above but dealing with a larger software and dealing with all users of the product . It is the reason why I will not enter in more details for this level .

In conclusion of this, I will say that at any level, work should be easier if software and its environment is clearly known and easy to modify in the sense that tools exist which are flexible but also are strict at use point of view .

IV . 5 . What are the most difficulties encountered when maintaining ? and why ?

As part of maintainance activities deals with debugging and testing, these difficulties were treated above in the correspondant section . The only activities treated here are the activities proper to maintainance activities and not refering to testing nor to debugging .

IV . 5 . 1 . Forward compatibility .

One must say start from the fact that the lack of forward compatibility is troublesome . This problem seems to be currently about the worst compared releases of packaged software generally . Here one insists on the fact that more flexibility is needed in order to be able to handle further releases of a software . One reason of this lack of flexibility is the fact that there is no easy means in order to change only one part of a module without having to consider all interactions with others (and if it is an operating system and that the module to be changed is called or should be called directly by instance, by SVC , there is no means in handling this module because it is not possible to see where it is called nor when . When the problem is solved even with this kind of call, flexibility will be increased without loosing precision . Indeed, at this time, there is no means of insuring that call are done in a right way . Only for this reason, if it is possible to find a mechanism to enforce the checks during calling time it should yet be valuable . And here it is tried to find easy mechanism not only to insure this checking but also to insure flexible ways of grouping modules in order to have not a rigid construction .

IV . 5 . 2 . High level language .

With high level language and especially language of the fourth generation, module interfaces will become both explicitly and implicitly complex and extensives in such source code . This is a real point because as anyone who has programmed a lot knows it is quite always in the interfaces that problems often occur . This problem which I try to solve is at this interface level . Of course, the way it is handled here is not at a high theoretical level but at implementation level . It is not because theory is not important, but simple solution, even not complete, but usable in practice seems to be better than doing a good theoretical solution but too heavy in the sense that the means to be used in order to put this theory into practice too expensive . For this reason, the solution should be found, at compiled time or/and at link time or/and at operating system time (run time) .

IV . 5 . 3 . Tacit assumptions .

Default and tacit assumption are troublesome in maintainance and especially when this occurs with the interface. The searched solution will not accept default during calling handling . The solution is to check wether there is a correspondance between what is given by the caller and what is expected by the caller . These checks do not permit to have default but should ask for explicit information and if it is not the case, this will lead to flag the software . Then what it is looked for is a mechanism which gives more flexibility in the way modules can interact but also enforcing the links between the modules or it is decided which one has to be used with others . This flexibility will also enforce the use of short program because it enforces the maniability of their interfaces .

IV . 5 . 4 . Concluding remarks .

In conclusion of the whole chapter, I can say that whenever problems are situated at interface or interaction or information passing between module, the solution of the problem stated here will increase a lot both flexibility and precision . That is the reason why even if not all cases are handled these which are handled are valuable enough to justify all the time what were spint during this work .

Part 3 : Theoretical Purposes .

I . Introduction .

Once the problem is seen and established (as it is !), we have to see what are the different parts that constitutes the problem . I will divide it in two main parts . The type checking problem and the handling of variable parameter list .

I . 1 . Type checking problem .

At present, indeed, no many languages are handling this check . I will deal with ADA and Pascal as representing the languages that are doing the type checking . They are not alone but, for example PL/I, the checking is done only if there are internal procedure . If there are external procedure, then, no checks are done . If we take another language, like CLU, the solution taken by CLU is quite the same than the solution taken by ADA . Indeed, CLU takes a library of description units and a compiler environment . With these tools, it is possible in a certain measure to modify the interfaces because this is handled by the compiler environment. For more details, see the CLU Reference Manual by B. Liskov and all, edited by Springer Verlag (1981) .

What I mean by type checking, is the check of the correspondance between the parameters list defined in the caller program and the called program . This correspondance must be verified to be biunivocal .

I . 2 . Variable parameter list .

Once the check is handled and there are no more problem with it, it remains to find a way of handle the same problem with less constraints. This is the fact that we must find a solution of handling parameter lists that have no more a so strict correspondance without leasing the advantages of doing the check .

II . Analysis of the possibilities of when doing the checks .

In the litterature, there are mainly two periods during which this can be done . Personally, I will suggest another possibility which seems to me a more interessant time to do it . As I will show the problem will always be constituted by the description of types and the handling of these descriptinns . There are then three possibilities : At compile time, at run-time and at link time .

II . 1 . At compile time .

Duting compilation, the compiler has to handle all variables and their types . It is quite always done in a table . The problem is that we can not use this table because this table contains too much informations . It is then quite a n normal thing that the first idea in solving the problem is to do it during compilation .

What that means is that we must find a way of putting the only useful part of these table somewere in order to use it after . This will, then, resolved great part of problem but it still remains that if we have the informations, how will we use these ? Indeed, when we encounter a call, we put all information about the call somewhere but how will we be able to decide if the call is correct or not ? For dealing with that, I will divide this in three part : ordered compiling, semi-ordered compiling and unordered compiling .

II . 1 . 1 . Ordered compiling .

In these case, the compiler will look at the data base in order to see if there exist a solution in the data base, with as key the name of the call . If there exist nothing(no modes) in the data base that meets the call under examination then the compiler can decides that the call is not correct . What actions the compiler takes afterwards is not an aim and is the responsability of the compiler design .

If the compiler finds a mode which handle the description of the type of each parameter of the call, he can decide that the call is resolved . Of course, this means that there is no problem if it is compiled with the same compiler version or same compiler . If it is not the case, we must find a way in which the types descriptions should take a standard form . This form should then be defined as

possibly handled by all different compilers . This, of course, does not mean that they are dealing the same implementations of the types but only that they deal the same representation of the type .

It is called ordered compiling because we must start with compiling program that do not have any call in it but what are called . This force you to work in a bathone up way what is not actually what is usually done . Indeed, todays methods are more top-down . This of course is not a good way of salving the problem to force to realize the software at logical level in a top-down way and then to be forced to realize it in a bathone up manner .

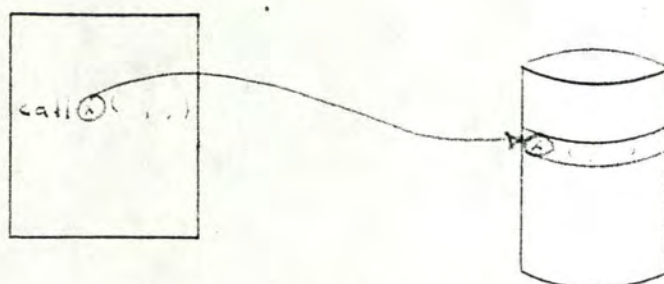
II . 1 . 2 . Semi-ordered compiling .

In response of this critic, we can mention another way of solving the ordering problem . This is the solution taken by ADA . Indeed, in ADA, you must declared and defines all the interfaces . Once these interfaces are compiled, you can use abstractions what are defined at the interface level and the way they are implemented has no importance nor the in order . In fact, when the compiler must compile a body (implementation), it can refer for solving the call, to the definition of the call which is defined in the interface .

For example, in ADA, it is done as follow :

- There are three kind of things :
- 1) the utilisation
 - 2) the description , what
 - 3) the implementation

happens when compiling :



Data base of the description and with more details .

As all interfaces must be know, before that something else can be compiled, whenever it encounters a call, it takes the name of the call and as this name serves also as identifier, it tries to find

the description of this object . Two things can happen . If the description match what it is used in the call, then it is correct and we can continue the compilation . If the description does not match, it looks further in order to be able to handle the overloading . If none matches, then we can sure that an error is detected . The reaction of the compiler is not own purpose .

This has, as effect, that we must not write the modules in a bathon up way . This solved than the type checking problem but with a price, the obligation of having a data base . This data base must be used very often, as we know, it is not easy to manage such kind of data base when many changes are done . There is also, as consequences, a time consuming as a space consumming quite big with this solution .

II . 1 . 3 . Unordered compiling .

If we do not want to have such restriction of ordering the way in which module have to be compiled nor having to writ all interfaces (it is may be not a good practice but you do not deal only with system that contain a way of handling a data base so easily) . But, if you want to do the checks, you can only work with fils but this involves the following problems :

Whenever the compiler has to compile a call, it can not see wether it is a wrong call or a right call . Indeed, if there is nothing that meet the call definition, this can not involve that it is a wrong call . In reality, may be, the called module is only not compiled . We see directly what this means, how or when is it possible to decide wether it is a wrong call or a right call .

One solution to solve that is whenever a program is compiled, to look at it and see if this program under compilation is giving a solution to a call not yet resolved . This has as consequences that all unsolved call sequence must be guarded totally in memory . Suppose that these descriptions of interface not yet resolved are put in a library .

Once a program is compiled, its description must be put in the library and it must be search through the whole library in order to see if this description are meeting an unresolved externa^l of another program . Its description must be guarded because no one

knows if afterwards then will not be other program to be compiled and which calls this one .

Once a call seems to meet both description and name of the call, then and only then the check can be done . But, how to decide what actions must be taken . Because, if we accept the fact that the verifications can be done after compilation, the error will be known only when the caller is compiled . The error is then from caller part of the all calls . And now, which one must be advised that the error is in its coding ?

It seems to me, that as showed, in all cases if we want to execute the checking at compilation time, we must force the users to conforme them selves to an ordering of compiling (ordered or semi-ordered) . The unordered solution needs to exploit to much informations and not useful in practice (see fig 1) .

II . 2 . At run time . .

Here, we must handle the type description but without having any possibilities to refer to symbolic definition . Indeed, at run time we must only interested in strings of bits (0 or 1) . This forces the compilers to find a way of describing the types expressed only with 0's or 1's . And also in a way such that there are no more ambiguities otherwise it is easier to do no checking at all .

As type definition can be very complex and can be combined in a "infinite" way, we can not have such a description . For example, if the parameter type is an array or a record we can have :

```
Type
    T2 = (DO, RE, MI, FA, SOL)
    T1 : set of T2
end ;
    A : array [1 ... 30] of T1
```

The array can be represented as :

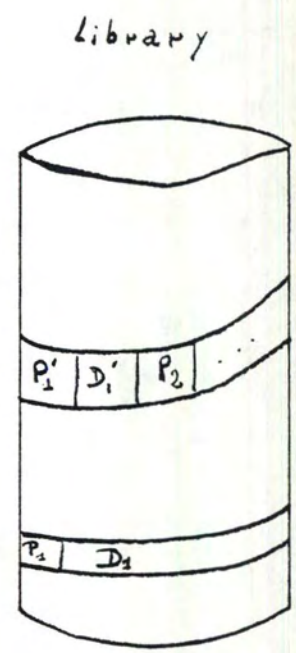


because array do not permit different kind of elements

P_2 :
[
 :
 call P_2 (, ,)
 :
]

P_3 :
[
 :
 call P_1 (, ,)
 :
]

P_1 :
[
 :
 :
 :
]



- We must talk about the problem of consistancies of the library .
Indeed, if P_1 is compiled, its description is set in the library .
If we have a call from P_2 that meets this description and a call
from P_3 that does not meet it . If we say that P_1 description
(P_1, D_1) is the normal way of calling it . Then we can decide
that the call of P_3 is wrong . If we have not yet compiled P_1 ,
we don't have its description . Thus we can not decide wether
 P_2 or P_3 call is wrong . When we compile P_1 , we can check through

whole library the descriptions . In doing it we can see that P1 description described by P2 is metted by the P1 description handling now . But, when can we decide that P3 is wrong ? It is also possible that P1 has no correct interface, than P3 can be right and P2 wrong and P1 also . Once one solution is found, for example, P3 call is wrong we must taken a decision, we must handle different description of the same thing .

- If we accept (and for large system we must accept it!) that some modules have the same name (same entry point name). We can see to what problem we arrive . During the check, we can say that the first encountered is the good one, or if it does not meet the descriptions for the first, we must pass to the second, and so on . And after all, we can not decide the wrong case because may be afterwards, the correct P1 will be compiled . This leads to unmaintainable solution .

- When deleting a module or replacing an old one by a new one , what must be done ? How can we sure that all programs referencing this module are advised ?

For records we can have such description :

```

Type
  T1 = record
    A : array [1 ... 10] of real
    B : Integer
    C : real
    .
    .
    .
  end ;
var
  T1 ;

```

The record can be represented as :



We can directly see how difficult it is to define, in a manner such that this leads never the ambiguities, all possible types only with string of bytes .

In order to show the difference of complexity between compile time and run time, let us see it, with an example (fig 2 & fig 3).

In fig 2, we can see that we must not handle the description of the parameter . Indeed, during the compilation, the compilers must handle the declarations and for doing it, they put the variables declaration in tables . The real way of doing it, is not an purpose but, let say, that the compiler put all variables of the same type in one table (it is not the case, but it simplifies the explanation without changing the nature of the mechanism) . Thus when compiling the subprogram, it checks wether the parameter definition is well of the same type (integer in the example of fig 2) . It can do it because it has the description within a table of one type and can see if it finds the same symbol in the table or not . If not an error is found .

In fig 3, we must not only handle one table of parameter, but two table of parameters and their corresponding type description . Indeed, one was constructed during the compilation of the caller program and the other one was constructed during the compilation

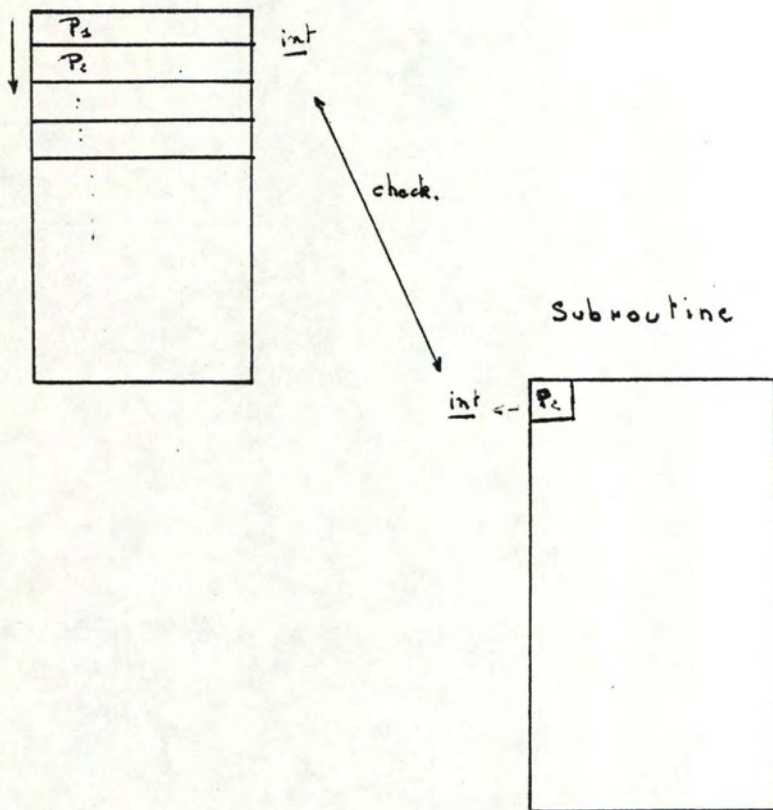
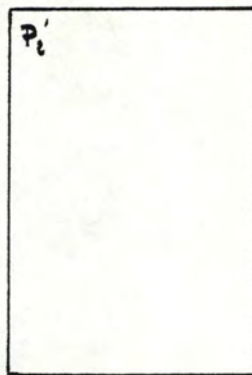


fig. 3.

P_1	D_1
P_2	D_2
P_3	D_3
⋮	⋮

Subroutine



P_1'	D_1'
P_2'	D_2'
P_3'	D_3'
⋮	⋮

of its subprogram . What must be done here is to prove that the description of the type of each parameter are equal . If we use some formal way of saying it using the notation of fig 3, we can say that what we must prove is :

$$\forall i, i \in [1, n] \quad \begin{array}{l} D_i = D'_i \\ \text{and} \\ n = n' \end{array} \quad \begin{array}{l} \text{when } n \text{ is the number of} \\ \text{parameter of the caller} \\ \text{parameter list and } n' \text{ is} \\ \text{the number of parameter} \\ \text{of the called parameter} \\ \text{list .} \end{array}$$

Saying it by this way, shows the real problem which is to what point we must have a description D_i such that we must be able to prove that $D_i = D'_i$ and at the same time we must be sure that it is not possible to prove that $D_i = D''_i$ when $D'_i \neq D''_i$. This means that we must have an unambiguous representation of type declaration .

Of course, if, for doing it, we must copy the whole description, as it is in the source text, and put it in the parameter list description table, we will have a serious problem of response time . Indeed, let try to see how much overhead this cost . Whenever a call must be performed (calling a system function, a standard module, or other module) what is very often done, if not always, we must check the correspondance of the type . This involves the fact that we must compare pieces to pieces the description of the formal parameter and the actual one . This creates the need of may be lot of space for putting this table in central memory and a lot of time for doing the checks . As the type declaration, for example in Pascal, can be very complex as records of records, this can consume so much time that whenever you want to execute a program, before having a response you can go away to drink some cup of coffee (or tea if wished) . This is a price that none of conceiver will accept to pay .

An other example to show that even if the problem state before is solved, there remain some other problem :

fig 4

<u>caller program</u>	<u>callee program</u>
Type	Type
T1 = Integer	T2 = Integer
end ;	end ;
var	var
A : array [1 ... 10] of T1	B : array [1 ... 10] of T2
.	.
.	.
.	.
call subprogram (A, ...)	B [5] := A [5]
.	.
.	.
.	.
end ;	

I have taken integer in order to simplify the example . It should be user defined type declaration instead of integer . That would complicate even more this example .

How can we find a way of describing the type of A in such a way that we can decide wether the type of A is the same then the type of B ? Of course, as we can see normally, the decision should be yes ; they are equivalent . We must think that we are at compile time, and we have only descriptions and their descriptions are not the same because A is an array of type T1 and B an array of type T2 .

This short and quite simple example shows that the problem of doing the type checking during the routines involves a so costly and complex solution that I will not further continue to speak of it . Because if we are not able (or not in economical way) to handle the type checking, how can we solve the problem much more complex of handling the dynamic interface .

II . 3 . At link time .

Before trying to explain what happens at link time, I will first show what a link edit actually do. This is the IBM implementation, but, all link edit perform the same functions even if they do not accomplish these the same way .

II . 3 . 1 . What is a link editor .

The linkage editor takes the output of the compiler and prepare your program for execution . The output of the linkage editor is executable by the computer . The linkage editor can combine your program with other object and load modules to produce a single load module . It stores your program in a load module library . These load modules can be read into the computer and given control .

The output of the compiler contains these kind of "cards" : the ESD (external symbol dictionary), the TXT (text = statements compiled or Assembled) and the RLD (relocation dictionary) .

In this study, we are only concerned by the external symbol dictionary when all information of the calls and external variable are put. (It describes the control sections and external symbols defined in the module, it helps you find references between modules in a multi modules program .)

For seeing an example of an ESD see fig 10

For seeing an example of a link editor, see fig 11 .

II . 3 . 2 . Introduction .

I have thought that this will be a possible place for doing checks (even if it is never talk about) . Why has this come to my mind ? Well, after link time normally, you do no more compilation . This leads me to the idea that it will simplify the problem of ordering the compilation . Also you do not so often a link operation as a compilation of a program (in theory, this can be discussed but in practice not) . Than if the cost of doing the check at link time is more or less the same than doing it at compilation, it will better to do it at link time . Let us then see what involves the fact that we will do the type checking at link time .

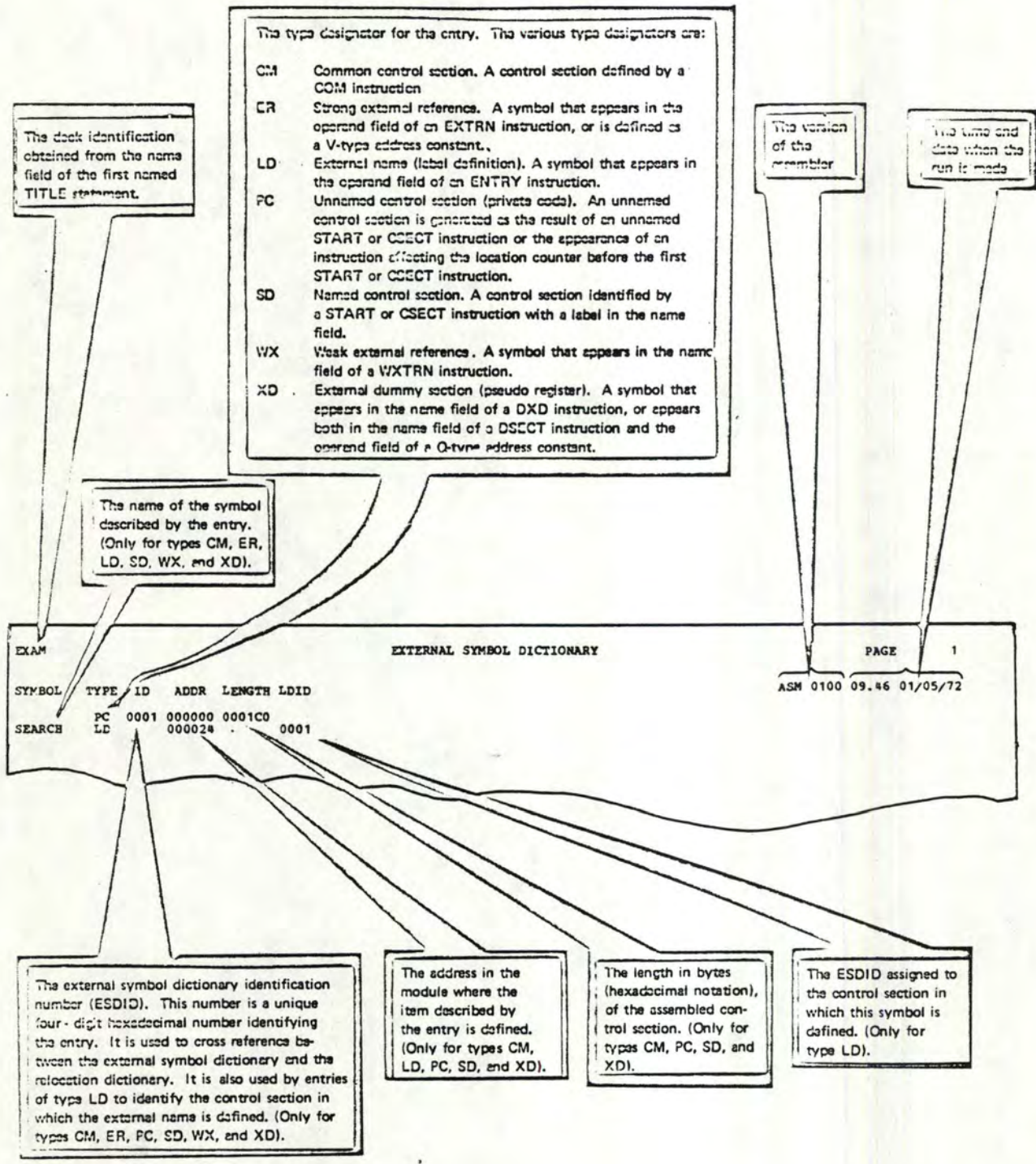


Figure 10. External Symbol Dictionary

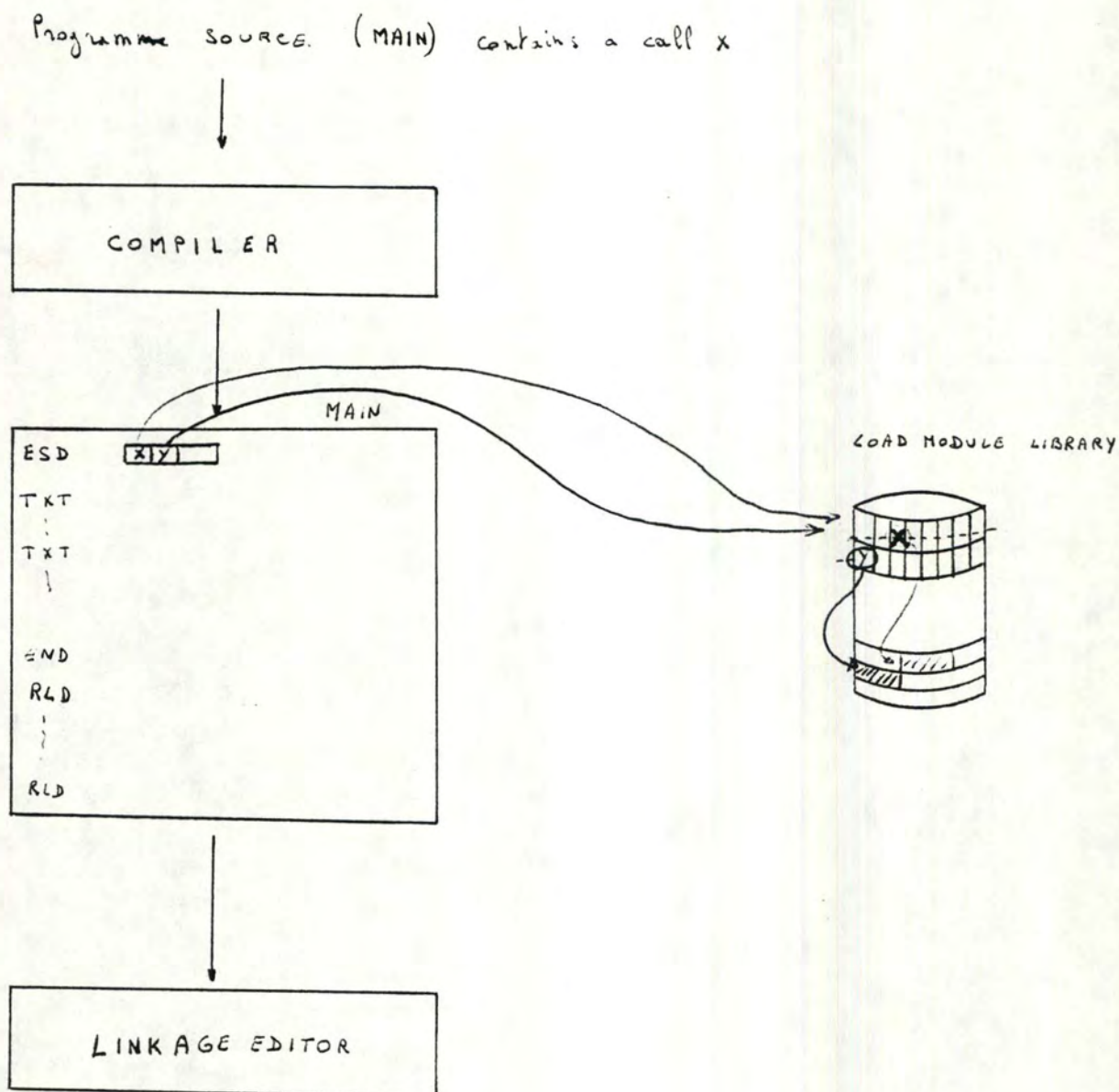


Fig 11.

During the link editor, the linkage editor tries to resolve all unresolved symbol that are in the ESD card(s) . If it can not do it, it gives an error message . But it must also try to resolve all relocatable problem (calculation of adresses, put the lenght of first module to the start adress of the following one,) . After link editing, you have an executable module when all things are defined and adressed .

II . 3 . 3 . Needs for type checking at link time .

II . 3 . 3 . 1 . What is expected from the compiler ?

Of course, as the link time never see what are the TXT cards, it can not see what are the call or when they occur . This involves the fact that the compiler has to give some additional information that it does not yet give today .

The compiler is expected to put the type description of the variable (as it does put it in a table for its own purposes) somewhere . This is the compiler will put the type description both in a table as it does it yet and in a file of unresolved external references .

That means that whenever the compilers finds a call, it put the call description in this place . The call description can be sure as name (program name or entry point), and as attribute (parameter list with their type description) . These informations are the only ones to be kept . Once the compiler has done its work, the compiler store this table in a file which will be used by the linkage editor when all parts of necessary information for type checking purposes and for number of parameter checking purposes are . This way of working can permit overloading as the compiler does (of course, if it does not, it is not possible) .

II . 3 . 3 . 2 . How to use these informations ?

When the linkage editor is invoked, it look for the ESD of the main program and try to search through the file in order to see if there is an entry with the name of the ESD under execution . There are now three situations . First one and easiest one, it can not find any matching description within the file with its entry point name . Then the result is saying that an error has occurred (we will see in part III, how this can be done in practice but also what precision can the error message have) . Second one, it find in the file the description associated to this entry point name and the problem is now to see if its associated description match the description of the ESD under checking . (This is to see if all parameters have the same type and if their number are equal) . If the correspondance between the description can be established then we can go further to see if another call has to be resolved . The third is the same of the previous one but here we can not establish the

correspondance between the descriptions . If we permit overloading we go further in order to see if we find any that match and we are in the situation described before or we can not find any and we have an error . Of course, if we do not permit overloading, if we find a distinct description associated with the same entry point name, we can yet decide that we have found an error .

Of course, as we are only dealing with theoretical purposes, I do not give any solution because this will lead me to give implementation detail .

II . 4 . Comparing the different acceptable solution : at semi-ordered or ordered compile time and at link time .

II . 4 . 1 . Modifications involved .

II . 4 . 1 . 1 . At compile time .

We must modify the compiler but we must not change any thing from logical point of view . I can say that the only thing to be change is the fact that the compiler can not delete the table of description of the variable type, but it must put the externals one somewhere in a file . This should not require to modify too much the existing coding .

We must of course add the check ability to the compiler . This involves to change the handling of calls, when we must check what is found and what is expected to found . This means that the compiler knows when these interfaces descriptions are and that it can access them this ability is the most heavy modification .

We must also have a possibility of defining the interfaces in such a way that the compiler is able to use it .

II . 4 . 1 . 2 . At link time .

We must modify the compiler in such a way that the information, the compiler put in a table must be put in a file in such a way that the checking can be done . This does not involve to modify the logic of compiler .

We must modify the linkage editor in such a way that it can do the type checking . This does also not involve the modification

of the logic of the linkage editor but only to add a new module which will accomplish these functions .

And we will, may be, have to do other changes (see implementation) at the operating system level .

II . 4 . 2 . Constraints involved by the type checking .

II . 4 . 2 . 1 . At compile time .

As said before, for doing it at compile time we must introduce an order in which the module must be compiled . For example, ADA forces the user to compile all interfaces before to begin to compile a body . Worse, it is not said precisely, at least in the ADA reference book, what a modification of an interface involves . In fact, it seems that for handling it for packages we are in a semi-ordered solution and if not we must recompile the whole . This is because whenever we modify the interface we will not be able to determine precisely what was their interfaces on the whole software .

II . 4 . 2 . 2 . At link time .

It seems that here we have no constraints at user point of view . Only the system must handle file for access and consistancy but it is done yet now .

II . 4 . 3 . Advantages of each method.

II . 4 . 3 . 1 . At compile time .

We must say that this solution involves no too much modification and this modification is situated only in the compilers .

We can see the obligation of defining the interfaces first as an advantage . Indeed, this will forces the designers to begin with defining the interfaces so precisely that they would not evoluce .

We can see that the handling of the solution is simple . Indeed, the interfaces are defined, when a call is encounter, it takes this description and look to the corresponding interface definition to see if the definition of the interface and its instantiation is the same .

II . 4 . 3 . 2 . At link time .

We must say that doing it at link time involves only the cost of the check only when the programmers have no more error at compile time . This means that the cost of the check is only to be paid once during the link and neither when compiling nor when executing

the software .

We can see that there is no obligation of when compiling the different part of the software . This involves a greater flexibility .

We can see that the error message here will be very precise because the linkage editor does not yet handle so many error message and because we know what type of error we can deal with .

This will not force the system to have a great management of files (or, as ADA, the handling of a data base with all problem of consistency .)

II . 4 . 4 Conclusion .

I prefer the link solution because it is not the practice, in most cases today, to define precisely all the interfaces before going to begin to code . Indeed, whenever some modules are defined, we can begin to design them . At compile time, this will no more be possible, because it is expected to have all the interfaces defined . Moreover, whenever an interface have to change we must recompile a lot of modules .

Another advantage is the ability of having a more flexible reaction . Indeed, if the checking at compile time can be sure as good when using the modern methodologies of programming development, it is no good if these methodologies are not used what is often the case today leading to loose the practical interest of this solution .

As compiling is done more often than linking, the cost of doing the check at compile time is more costly from time consuming point of view . Moreover, if the type checking must be done for different languages we must modify all compilers for the whole modification when there often exist only a linkage editor .

The precision of the error message can be greater at link time because there are not so much possibility of different kind of errors comparing with the great number of possibilities when compiling . This should have a major advantage for maintainance and debugging .

When seeing the implementation, we will that the modifications involved by doing the type checking at link time instead of doing it at compile time, are not great and often can reused what is yet used without modification .

III . How to handle variable parameter list .

III . 1 . Introduction .

As for type checking, these are three moments during which this can be handled . The problem here is to find a good way for doing it without having to use switches or other means like that . In the same time, this handling should not be visible by the users . From logical point of view, this can be handled at the same time when dealing with type checking . It seems obvious that in production environment this can not be used often, but in development environment well . I will no more consider the run time nor the unordered compile time for handling this because yet for handling type checking they are not useful .

III . 2 . At compile time .

III . 2 . 1 . Ordered and semi-ordered .

We must do it in such a way that the type checking can be done and such a way that it would be possible to handle variable parameter list . We must then find an "extended type" such that $D_i = D_i' \forall i$ but when some D_i have a special type .

On logical means of doing this is to say somewhere that the old interface is no more valid but that the new one has to be used . This means that the compiler can do the checks with the old interface definition and description but will execute the program with the new one .

This of course can not assure the user of the module that the old call and the new one is compatible . Indeed, there is no way nor mechanism for doing it without man responsibility . Thus, I have to recognize that no full automated solution is found . I will let to the conceptor of the module that he (she) replaces responsibility of the compatibility between the old and the new call . It is not possible to let to the compiler or to the system the responsibility of checking all possible changes that have to be done in order to insure this compatibility, there are too many possibilities . But, the compiler or the system have to provide him (her) with a mechanism that will give him (her) the possibility of checking which call

it is confronted to .

I will treat this as a special case of overloading but in a transparent manner . I mean that the user must not see any of this kind of changes .

The problem is how to do for maintaining the consistence of the data base . Indeed, how can we deal with such situations that some records in the data base are flagged but can not be deleted . Moreover, the logical description will not cover what is actually done . Whenever an error is discovered at run time, for example, how can the user deal with such a situation where the code executed is not the code that the user knows . What at debugging time ?

Personnaly I do not like this solution, because this involves the introduction of such illogicality . This is not safe and will perhaps conduce to a worst situation then not handling this .

Of course, it is possible to check only the corresponding parameters and not checking the other types .

$$\begin{array}{ll} \text{if } n < m & \text{then } \forall i, 1 \leq i \leq n \quad D_i = D'_i \\ \text{if } n > m & \text{then } \forall i, 1 \leq i \leq m \quad D_i = D'_i \end{array}$$

$$\left. \begin{array}{l} \text{for the } i > n \text{ in the first case} \\ \quad \quad \quad i > m \text{ in the second case} \end{array} \right\} \text{the other descriptions are not checked .}$$

III . 2 . 2 . Partial conclusion .

There is no mean of handling this without giving some responsibility to the conception of the replacing module .

There is also a problem of preserving the data base integrity .

III . 3 . At link time .

III . 3 . 1 . Introduction .

As mentioned at the formal definition of the problem, we have restricted us with the situation where all common part between formal and actual parameter must have the same description .

The reason why I have taken such restriction is because we must first have to accomplish the type checking (and it is yet no easy)

and after we must relax the constraints . This means that if we want to introduce fully the ability of dynamic interfaces, we must have to do no more checking . Indeed, how can a system decide in next situation :

```

call P(A, B, C)
.
.
.
call P(A', D', E')
Proc P(A'', B'')

```

If we say that we can link the modules together, how can we decide that the first call is correct and the second one contains an error . The only way, of knowing it, is when the programmer have interchange the old module with the new one . he (she) is responsable of the compatibility between the interfaces .

After all, what I have done is to force the interfaces to be checked precisely and if we want to relax too much constraints, we must no more be able to perform the checks .

III . 3 . 2 . Trouble of handling full dynamic interface ability .

For debugging purposes, it is also not so good that one can be able to replace an old module without advise . Indeed, if there is a dump, how can the user manage such situation .

There is also problem for preventing the use of parameter that does not exist when performing the old call . For example,

- 1)if the new program module is such that it needs give parameters, and if we must deal with such program :

```
Proc P (P1, P2, P3, P4, P5)
```

$$Y = f(a,b) + (\dots) \times P_5$$

```
end .
```

If we must perform the old call, what can happen? The P_5 has not receive any value. It will then have the value contained by the following bits of P_4 . This of course can lead to many trouble. We must remember that the old call performed does not know that the module have changed. He can thus have strings effect due to the change. I do not see any solution for this if we want the system to take account of this.

- 2) if the new program is only adding one function in a case, we have no problem in fact we will never perform the case of the new function.

<u>old</u>	<u>new</u>
Proc P($P_1, \dots P_4$)	Proc P($P_1, \dots P_5$)
case of P_1	case of P_1
case of P_2	case of P_2
case of P_3	case of P_3
case of P_4	case of P_4
	case of P_5
end case;	end case;
end P;	end P;

This of course is the good case but one reflexion comes directly to mu mind : Why adding this new function to the module instead of doing a new module with this function ?

III . 4 . Some partial conclusion .

I have decide that such situation can not fully be accomplish by the system at least not easily nor economically. For instance, the PLIOPT which in the PL/I compiler optimiser which can handle a part of such situation but at cost of 64 passes (not the whole program is examined but some part of coding).

But, I must provide at least some mechanism for helping the designers or programmers for handling such situation. This mechanism

should provide a way of testing whether the parameter is present or not .
 If it is present, it is tested . But, it is the responsibility of
 the programmer to use it or to write his (her) coding keeping in mind
 the fact that it can be called with an old interface .

This will lead to such situation :

1) I use once again the example 1 in the previous section .

Proc P(P₁, P₅)

Proc P(P₁, P₅)

Y = f(a,b) + (...) * P₅ becomes Y P₅ then z = (...) * P₅

else z = 0

Y = f(a,b) + z

end;

end;

2) Another kind of solution is :

Proc P (P₁, P₅)

if is present (P₅) then

else

end;

The practical way of doing this will be mentioned briefly in
 the next part . But we can see that we are able to provide some
 help but we do not have a full practical mean to solve the dynamic
 interface fully .

III . 5 . Conclusion .

After many readings, after many meetings with people who are
 working in different society, I become to doubt of the necessity
 or the interest of solving such problem . In all cases, the problem
 of testing the interfaces is solved . We are now able to verify the
 correspondance of the definition between formal and actual parameters .

Another thing is that, if we permit the use of such ability, who
 are the people that can use such facility . If we do not force only

the people who needs it to use it, we have loose our aim . We must then once again give the responsibility of doing this to some one . Indeed, if untrained people use this, this will involve a worse situation than it is today where no checking is done for external procedure .

On advantage of not solving the whole is the fact that when a software is designed the coherence between modules will be enforced without human intervention .

I will then say that the partial solution we give is the best "compromis" between having a reliable system, a system usefull in practice (not too slow nor too space consuming) and a flexible system .

Part 4 : Implementation of a solution .

I . Type-checking implementation .

I . 1 . Description of type .

I . 1 . 1 . First possibility .

We can decide that we do not offer the handling of predefined type . This will say that we are only confronted with types like Integer, real, array and so on .

This solution is too restrictive because most languages offer more ability for type handling . We will not want to restrict the type definition ability .

Moreover, this will not solve the problem of handling records within which different name occur .

For instance,

$t_1 = \text{array } [1 \dots 10] \text{ of integer}$ $t_2 = \text{integer}$ $r_1 = \text{record } \begin{cases} i_1 : t_1 \\ i_2 : t_2 \end{cases}$	$t_2 = \text{array } [1 \dots 10] \text{ of integer}$ $t_3 = \text{integer}$ $r_3 = \text{record } \begin{cases} \text{info 1} : t_2 \\ \text{info 2} : t_3 \end{cases}$
end ;	end ;

Are these record types equal ?

I . 1 . 2 . Second possibility .

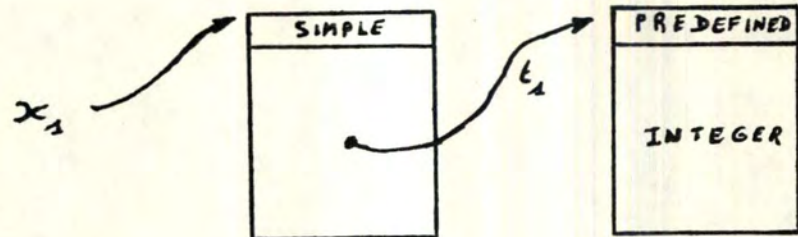
We will now offer full ability for user type definition but the idea here is to expend the description fully until, we arrive to a predefined type . This do not solve the problem of the records as defined in I . 1 . 1 . and also do not solve the problem of the recursive type .

example 1

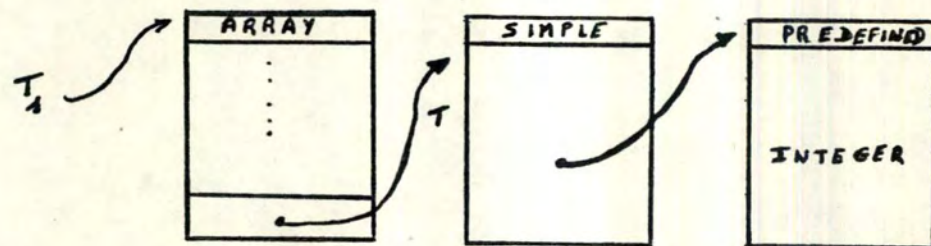
<pre> P1 : proc t₁ = integer t₂ = array [1...10] of t₁ x : t₂ call P2 (x) end;</pre>	<pre> P2 : proc(y) t₃ = integer t₄ = array [1...10] of t₃ y : t₄ end;</pre>
---	--

We test to see if we have a constructor we have a structure type, if not we have a simple type .

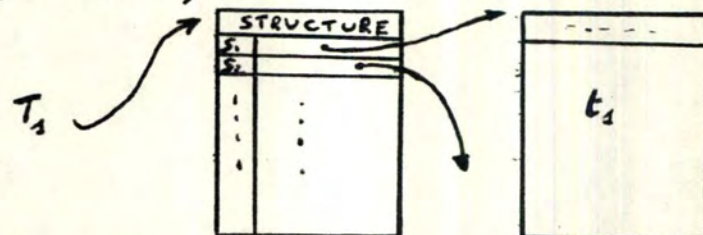
. t_1 : integer
 x_1 : t_1



. T : integer
 T_1 : array [e_1 , ...] of T



. T_1 : record (s_1 : t_1 , ... s_n : t_n)



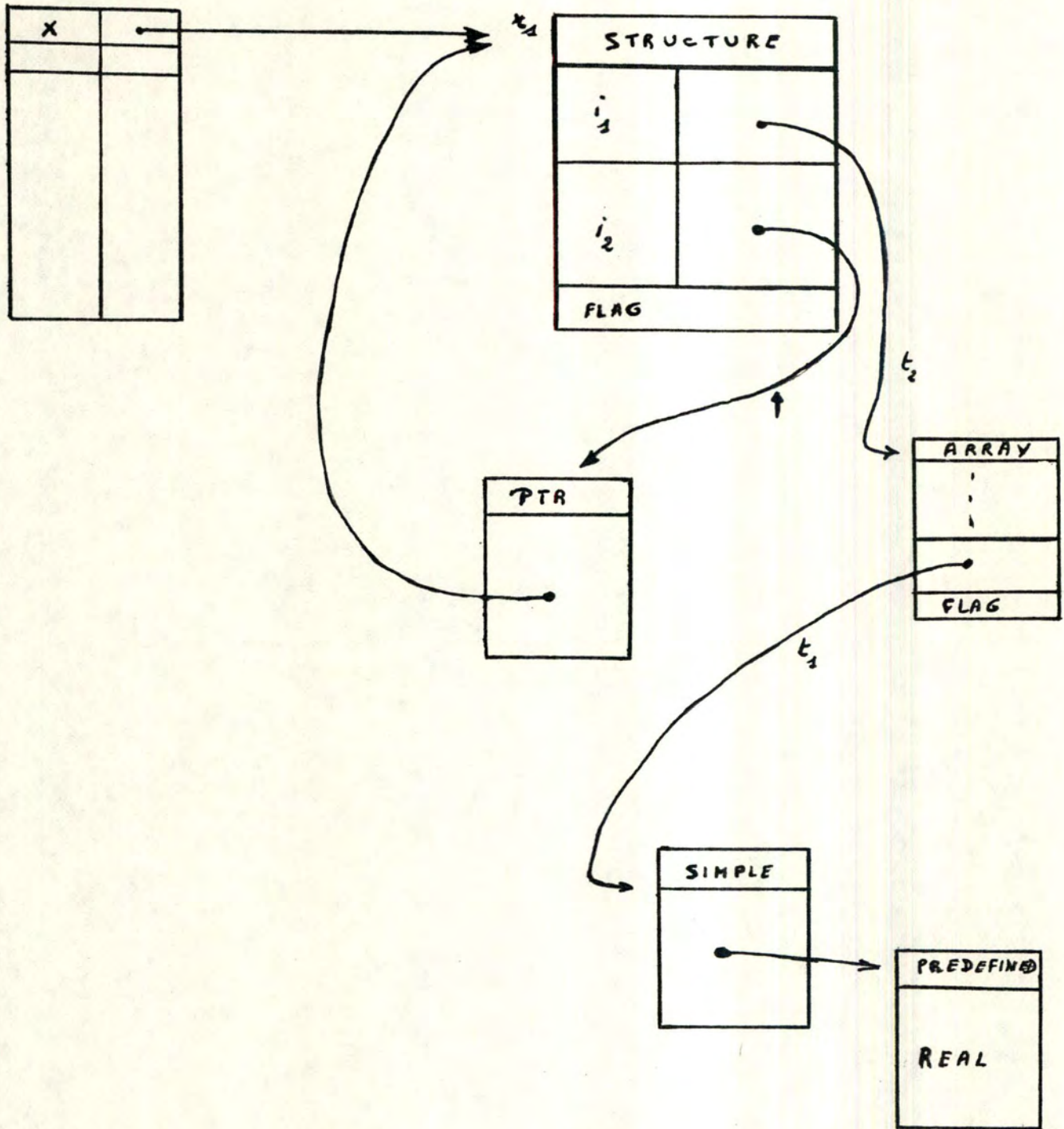
After defining this, I will show a complex example and how we can deal with it .

```
const m = 2
type t1 = array [1 ... m] of t2
      t2 = real
      r1 = record { i1 : t1
                  { i2 : ↑r1
```

```
x = r1
call P(x)
```

```
proc P(y)
const n = 2
type t3 = real
      t4 = array [1 ... n] of t3
      r2 = record { i1 : t3
                  { i2 : ↑r2
```

```
y = r2
```



As shown, I have defined a type description which can accept recursion and all other features of user defined type .

The flag will serve when checking the correspondance for avoiding to detect the entering into an infinite loop .

I.2 Handling the check at link time.

I.2.I What is expected the compilers do?

First, as I mentioned in II.4.I.2, compilers and linkage editors should be modified to handle type checking at link time.

The compiler has to save informations about each external procedure call.

In this implementation, two files will be used :

File 1 : contains informations for each entry point

File 2 : contains informations for associated
interface description

(for detailed layout, see fig I2. page 72)

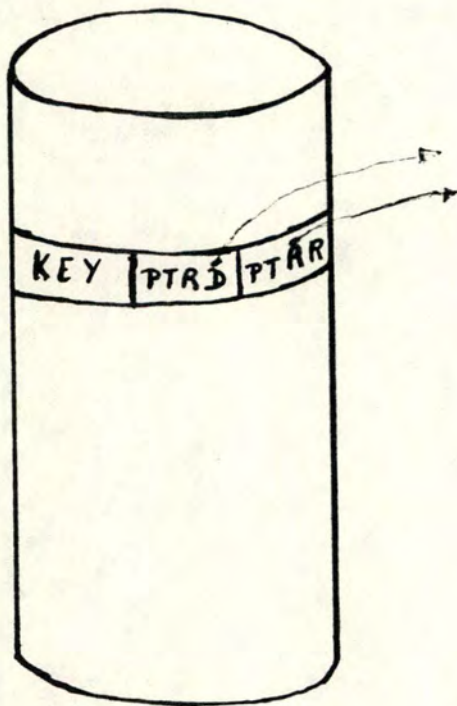
Second, I will show what are the changes involved by the type checking at ESD cards generated by the compilers. (There is no other changes for all other kind of cards, the old format is given in fig I3 a. page 73 and the new one in fig I3 b. page 73)

Before going further, I would like to give some remarks about the next figures.

In the figures, there will be two kinds of things : the existing situation what will be written in a quite thin way, the modifications will be written in a thick way.

In the figures, there will be several steps shown, each of them will have a number associated with each. These numbers will be used within the schemas.

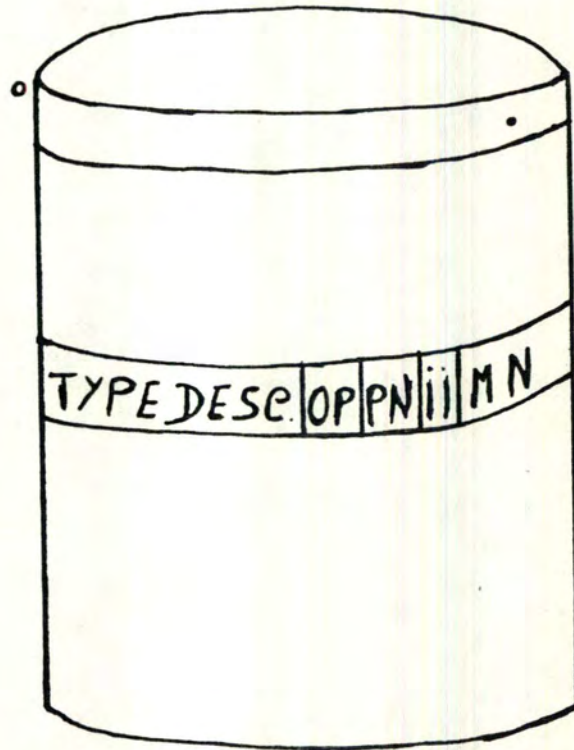
Keyed file



File DEF-REF

key : access-key = entryname
 PTRD : pointer to definition(s)
 of caller and callee
 PTRR : pointer to references
 to interfaces

Direct access file



File DESCRIPT

Typedesc : descriptor
 of types
 OP : overflow pointer
 NP : next pointer
 II : interface
 identifier
 MN : module name
 or
 pointer to module code
 Record 0 is the pointer
 to the next free space

Remark : The OP field will never be used here, because it does not add anything for explaining the implementation but it would keep away the clarity of the schemas.

The OP field will be used if the interface description is too big for the place useful in the typedesc field of the descript file records.

Fig 13 a. Actual ESD Cards.

<u>SYMBOL</u>	<u>TYPE</u>	<u>ID</u>	<u>ADDR</u>	<u>LENGTH</u>	<u>LDID</u>
P	ER	0002	

Fig 13 b. New ESD cards.

<u>SYMBOL</u>	<u>INTID</u>	<u>CSECTNAME</u>	<u>NBRSTMT</u>	<u>TYPE</u>	<u>ID</u>	<u>ADDR</u>
P	I		5	ER	0002		

The symbols used here will be used in the next figures.

Third, I will show what are the actions performed by the compilers when encountering a procedure definition and when encountering a call. The associated algorithms are described in I.2.2. (see figure I4, page 75).

Fourth, I will show what are the actions performed by the compilers when encountering a procedure definition which is used in an other module as a subroutine. This will show how procedure definition are handled. The associated algorithms are described in I.2.2. (see figure I5, page 76).

Fifth, I will show what are the actions performed by the compilers when encountering more than one procedure with the same name (this is known as overloading). This will show how different procedures with the same name can be described without confusion. The associated algorithms are described in I.2.2. (see figure I6, page 77).

Fig 14.

SOURCE

```

Proc A (B,C,D)
  ⋮
Line 5 Call P (E,F,G)
  ⋮
Line 12 Call P (E',F')
  ⋮
End A;
    
```

Where A,B,C,D,E,F,G,P
E',F' are supposed
correctly defined.

OUTPUT OF THE COMPILER

```

ESD : I A CM 0001 0000 002B
      2 P I ER 0002 5
      3 P 2 ER 0003 I2

TXT :
      .
      .
      .
      L I4,=V(PI)
      BALR I4,I5
      .
      .
      .
      L I4,=V(P2)
      BALR I4,I5
      .
      .
      .

RLD : 0001 ....
      0002 ....
      0003 ....
    
```

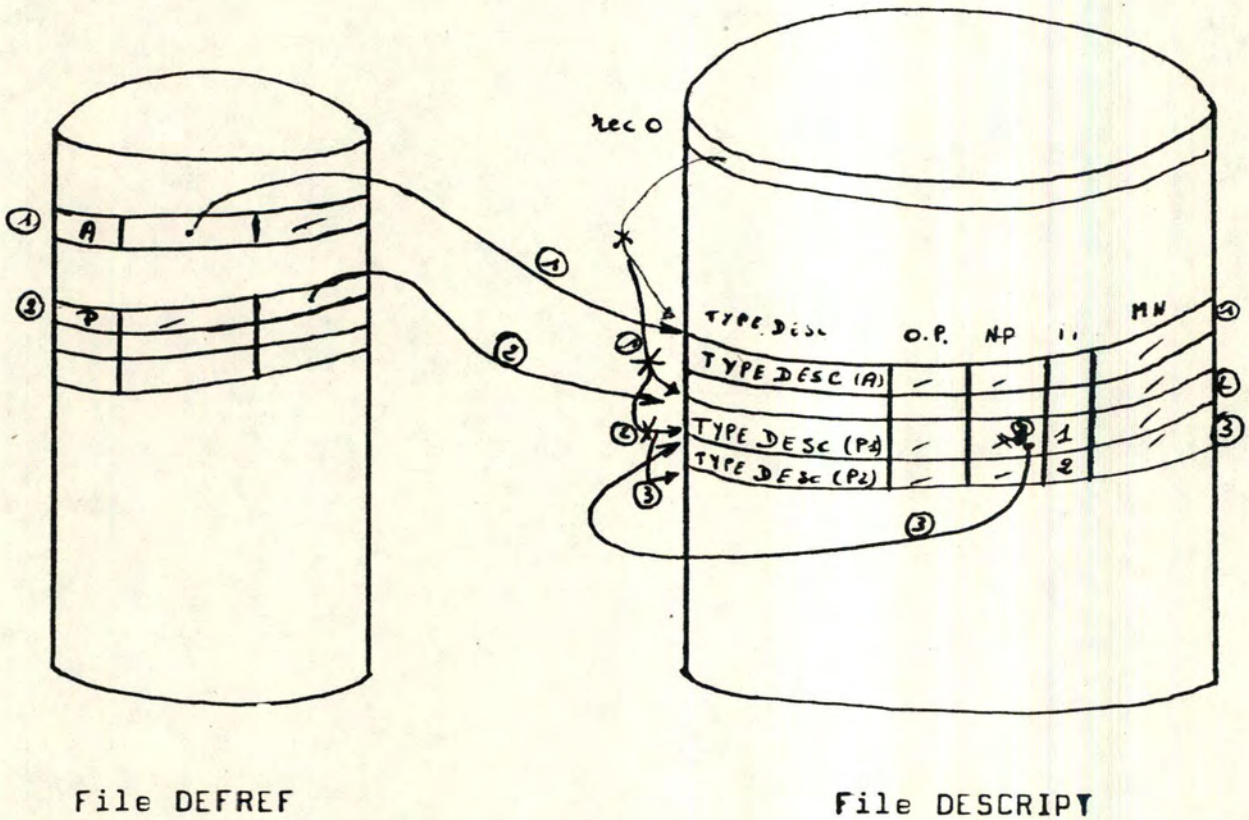


Fig 14.Explanation.

1.When encountering Proc A : ① The compiler looks in DEFREF file in order to see whether an entry exist with as key the entry name. Here, as it is supposed that we begin with a new situation, the compiler does not have such entry, then it creates one and store into the DEFREF file a record with the needed informations. After that, it stores the interface description into DESCRIPT file (after getting an entry in this file).

2.When encountering a first call as in line 5 :

② The compiler looks in DEFREF file, using the entry name as key, in order to see if the entry yet exist. Here of course, it does not exist. The compiler stores the needed informations into DEFREF file and gets an entry in DESCRIPT file and stores the informations in this file. The compiler updates the interface identifier and put it into the ESD card.

3.When encountering a call as in line 12 :

③ The compiler looks in DEFREF file, using the entry name as key, in order to see if the entry yet exist. Here, of course, the entry exists because it was yet created when compiling line 5. It takes the PTRR pointer to have the corresponding entry in DESCRIPT file. The compiler compares the interface description. If the descriptions are the same, the only thing to do is to put the interface number into the ESD card. If the descriptions are not equal the compiler gets a new entry in DESCRIPT file, updates the interface identifier number, stores the informations into the DESCRIPT file and generates a ESD card.

Fig 15.

SOURCE

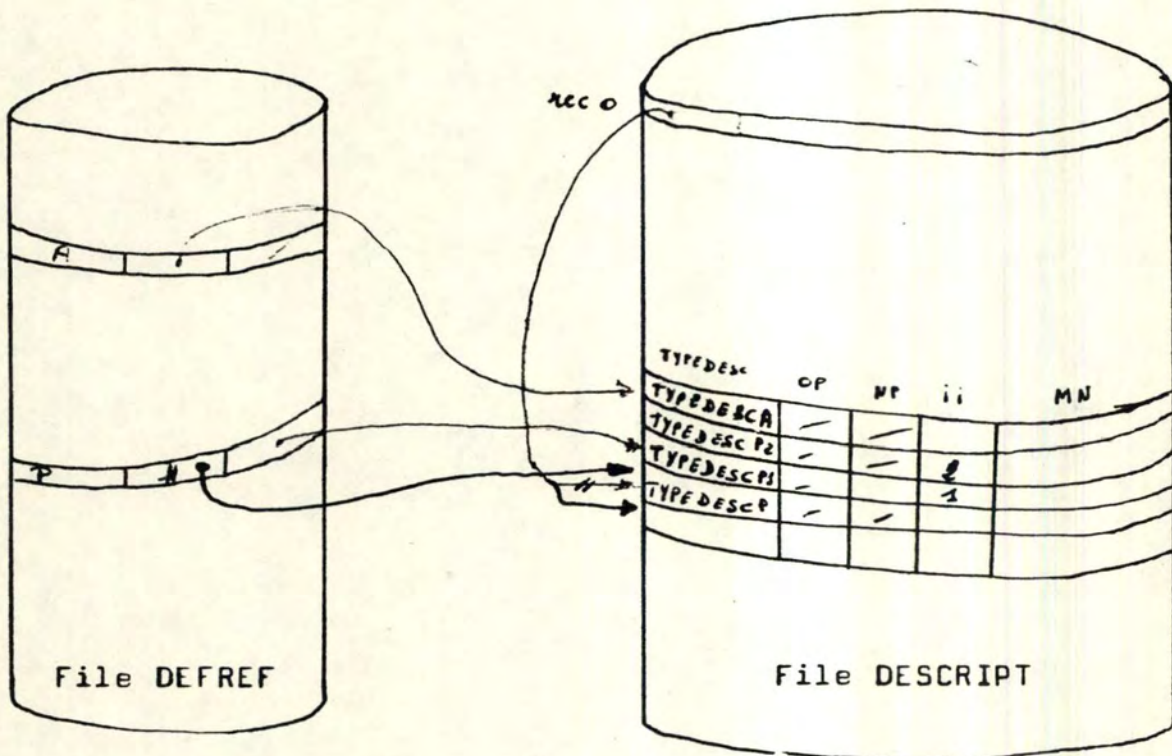
```
Proc P (A,B)
  .
  .
  .
END P;
```

OUTPUT OF THE COMPILER

```
ESD : P      CM      000I      000I      004C

TXT :      . . . . .
      . . . . .
      . . . . .

RLD : 000I
```



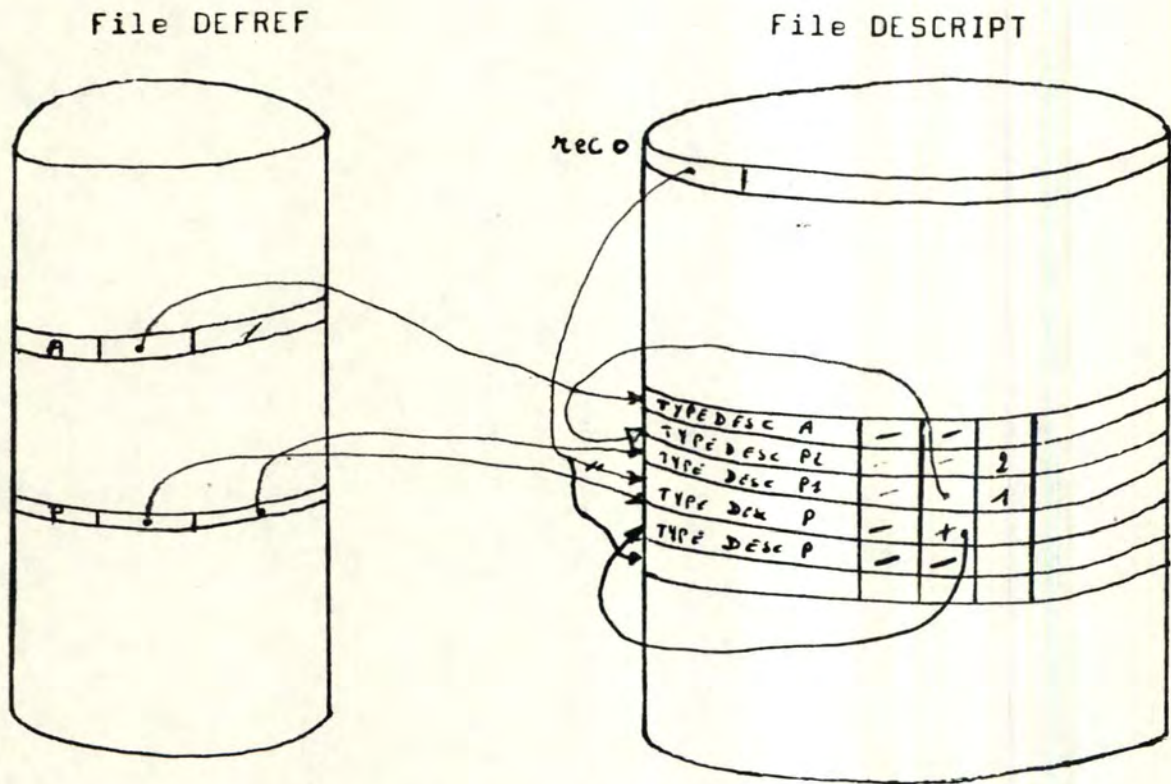
When the compiler compiles the procedure P, it looks in DEFREF file, using the entry name P as key. It finds then that the PTRD in this file is not used. The compiler gets an entry in DESCRIPT file and store in it the informations. The compiler update the PTRD pointer.

Fig 16.

SOURCE
 Proc P (A,B,C)
 ⋮
 END P;

OUTPUT OF THE COMPILER
 ESD : P CM 000I 0000 0I4B
 TXT :

 RLD : 000I



When the compiler compiles the procedure P, it looks in DEFREF file, using the entry name P as key, It finds that the PTRD is yet in use. It gets then the corresponding entry in DESCRIPT file and compare the descriptions of the file and the descriptions under examination. If the descriptions are equal, then there is no special actions must be undertaken by the compiler. If the descriptions are not equal, the description under compilation is added in DESCRIPT file and the NP pointer of the existing record is updated.

I.2.2 General algorithms of the new compiler functions

In the next figures (Fig I7 a,b,c), I will show the general algorithms for handling a call.

There are three algorithms because the compilers have to handle three kinds of things. The first one is the management of the two files, the second is the generation of the ESD card and the third is the generation of the RLD card. The figures are in page 79.

The following figures (Fig I8 a,b), I will show the general algorithms for handling procedure header.

There are two figures because I will describe what happen at files level, that will say the consultation of both files, and what happen at ESD card level. The figures are in page 80.

I.2.3 What is expected the linkage editor do ?

I will describe shortly how the linkage editor uses the informations given by the compilers. For doing it, I will use the files described in fig I6 and show with it the actions taken by the linkage editor in order to provide the type checking. The figure and its explanations can be found in page 81 & 82. (see fig I9)

I.2.4 General algorithms of the new functions of the linkage editor

In the next figure (Fig 20), I will give the general algorithm for handling the type checking by the linkage editor. I will show how the informations given by the compilers are used by the linkage editor for doing it. the figure will be given in page 83.

FIG. I7a

Is the entryname within the reference list?

Y -----> WORK.INTID := DESCRIPT.INTID

N -----> Is the list empty?

Y ---> WORK.INTID := 1 ;
DESCRIPT.INTID := 1;

N ---> WORK.INTID := LAST.INTID + 1;
DESCRIPT.INTID := WORK.INTID;

Add the reference into the reference list;

end;

FIG. I7b

GENERATION OF ESD CARDS.

ESD.SYMBOL := ENTRYNAME;
ESD.INTID := WORK.INTID;
ESD.CSECT := MODULENAME;
ESD.STMTNBR := SOURCELINENUMBER;

Nothing else have changed.

FIG. I7c

GENERATION OF RLD CARDS.

if ref does not exist into referencelist,
then create a new RLD card
endif;

FIG. I8a

Is the entryname within the DEFREF file?

Y -----> DESCRIPT.TYPE.P is within type-descriptor-list?

Y ---> RETURN

N ---> Add the reference to the reference
list;

N -----> Add the key entryname and set the PTRD;
Add the definition in the definition list
of DESCRIPT file;

end;

FIG. I8b

GENERATE ESD CARDS.

Do nothing else then actually done.

Fig I9.

- A link A instruction is given.
- The linkage editor takes the ESD cards of type ER of the compiled module A.
- For the other kinds of cards, nothing have changed.
- When it encounters the first ESD card P, it takes its II number and its entry name as key for accessing DEFREF file where it finds both pointers, one for the references and the other for the definitions.(1a)
- With the PTRR pointer, it goes through the reference list and when it finds the corresponding II, it stops the search through the list.(1b)
- With the PTRD pointer, it goes through the definition list and it test to see wether a II number is yet setted If it is the case, it compares the definition II and the reference II, if there are equal, no problem continue the link, if not go through the list.(1c)
- When the linkage editor encounter, the corresponding description and when it is not yet encountered (no II value), it puts the II value of the reference in the corresponding field of definition record.(1d)
- If no corresponding description can be found, an error is flagged.
- This process must be also done, if the module P has some external references, for P.

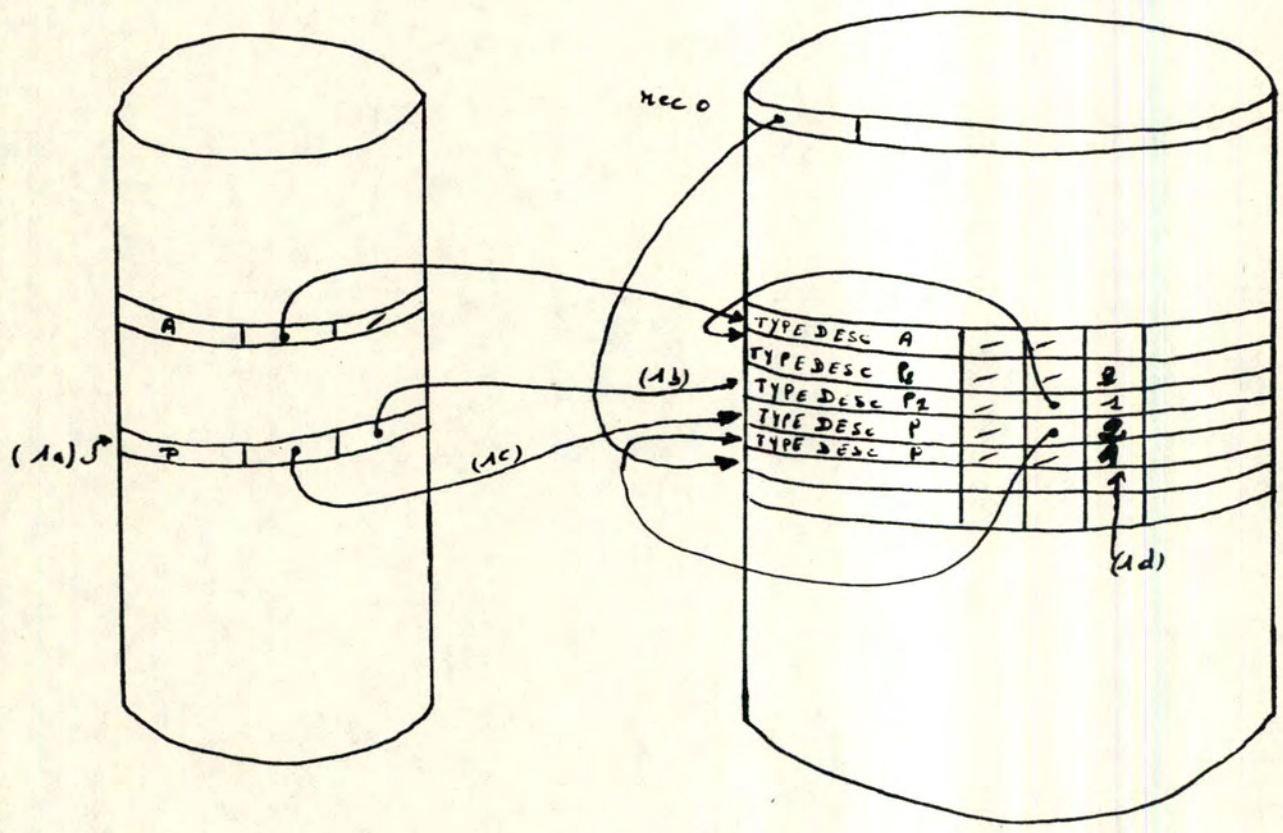
Fig 19.

LINK A is the instruction given.

```

ESD : A      CM      000I  0000  002B      (output
      P I ER  A      0002  5        of the
      P 2 ER  A      0003  I2       compilers)
      P I ER  A      0004  I8      (line added for expla nations
                                   purposes)
      .....
      .....
      .....

```



The symbols used here are the same that have been used in figure 14, 15 & 16, page 75, 76, 77. The corresponding general algorithm can be found in figure 20 page 83.

FIG 20

For each ESD cards of type "ER"

do

find corresponding entry in reference list;

if does not exist, then do

send error message to user;

enddo

else find same interface in definition list

if does not exist then do

send error;

enddo

else do

take module-
name;

load this;

enddo

endif

endif

enddo

I.3 Detailed algorithms

The figures showing these algorithms are in the following pages. The detailed algorithm corresponding to figure I7a, is shown in figure 21 page 85.

The detailed algorithm corresponding to figure I7b, (page 79) is shown in figure 22 page 86.

The detailed algorithm corresponding to figure I7c, (page 79) is shown in figure 23 page 87.

The detailed algorithm corresponding to figure I8a, (page 80) is shown in figure 24 page 88.

The detailed algorithm corresponding to figure 20, (page 83) is shown in figure 25 page 89.

I.4 Handling the dynamic interface.

As mentioned before, we are only concerned by the fact that the only dynamism, that we accept, handles a different number of parameters in a parameter list but we assume that there is no difference, at parameter type point of view, between the two lists.

The most easiest way for implementing it, is to modify the type checking algorithm for stopping the checks once a parameter list is completely examined. This modification will not change much of the algorithms described here, so I will not give a detailed way for implementing such thing.

I will mention that there is another way to do it. The alternative way of doing it, is to ask the linkage editor to continue the checks until all elements of the shortest parameter list were seen as correct. This has as consequences that the system will no more detect the cohesion of the actual parameter list and the formal one. It can only offer or guaranty that the common part of both parameter list have the same type.

It is left to the programmer the responsibility of the validity of the old call (comparing it with the new ones). This is may be not what is wished by the people who are concerned by the maintenance or by the reliability of a software.

Fig_21.

```

Defref_type = record [ Entryname : Char(8);
                        Defptr    : ↑ Descript_type;
                        Refptr    : ↑ Descript_type ]
Descript_type = record [ Typedesc : Interface;
                        Typeop    : ↑ Descript_type;
                        Typenp    : ↑ Descript_type;
                        Intid     : Integer;
                        Modname   : Char(8) ]

Refbuf : Defref_type;
Descrbuf : Descript_type;
Out : Descript_type;
Outptr : ↑ Descript_type;
Workintid : Integer;

getfile DEFREF (entryname) into Refbuf;
if record not found then create an empty record with entryname;
endif
if Refptr = NIL then do
    getfile DESCRIPT (0) into Descrbuf;
    Refbuf.Refptr := Descrbuf.Nextfree; / Nextfree is
                                        a field only
                                        in record 0 /
    Descrbuf.Nextfree := Descrbuf.Nextfree + 1;
    putfile DESCRIPT (0) from Descrbuf;
    putfile DEFREF (entryname) from Refbuf;

/ We have just updated the pointer to the next free record in DESCRIPT
file and also updated the pointer to the references in DEFREF file /

/ The next few lines create the new Descript record and written it
in the DESCRIPT file /

    Descrbuf.Typedesc := convert (interface);
    Descrbuf.NP := NIL;
    Workintid := 1;
    Descrbuf.Intid := 1;
    Descrbuf.Modname := #;
    putfile DESCRIPT (Refbuf.Refptr) from Descrbuf;
enddo

/ convert is a primitive of the system for transforming the source
description of the interface in the internal one. /

```

```
else do
```

```
    getfile DESCRIPT (0) into Descrbuf;  
    Refbuf.Refptr := Descrbuf.Nextfree;  
    Descrbuf.Nextfree := Descrbuf.Nextfree I;  
    putfile DESCRIPT (0) from Descrbuf;  
    Out.NP := Refbuf.Refptr;  
    putfile DESCRIPT (Outptr) from Out;
```

```
/ We have just updated the pointer to the next free record in  
  DESCRIPT file and also updated the pointer to the next reference  
  of the reference list in DESCRIPT file . /
```

```
/ The next few lines create the new Descript record and written  
  it in the DESCRIPT file. /
```

```
    Descrbuf.Typedesc := convert (interface);  
    Descrbuf.NP := NIL;  
    Workintid := Out.Intid + 1;  
    Descrbuf.Intid := Workintid;  
    Descrbuf.Modname := #;  
    putfile DESCRIPT (Refbuf.Refptr) from Descrbuf;  
  enddo
```

```
endif
```

Fig_22.

```

ESD_type = record [ Symbol : Char(8);
                    Intid  : Integer;
                    Csectname : Char(8);
                    Nbrstmt  : Integer;
                    Type     : Char(8);
                    Id       : Integer;
                    Addr     : Integer;
                    . . . . .
                    . . . . .
                    . . . . . ]

```

/ The rest of the ESD_type has not changed, this is the reason why it is not described here nor will be described the existing code for managing it. The only things that are described are those which are not yet handled. /

```
ESD : ESD_type;
```

```

ESD.Symbol := Entryname;
ESD.Intid  := Workintid;
ESD.Csectname := Source.Modulename;
ESD.Nbrstmt := Source.Linenumber;
ESD.Type := "ER";

```

```

. . . . .
. . . . .
. . . . .

```

/ There is no other changes, so I will not give more details. /

Fig_23.

```
if is_in_list (interface,Refbuf.Refptr,Out,Outptr) then do noop
                                                    else do
                                                        create a RLD card
                                                    enddo
endif
```

/ As there are no changes in the treatment of the RLD cards,I will not give any additional details. /

/ The is_in_list is a primitive of the system which provides two things.First,if the first parameter is in the list of the Descript record list pointed by the Refbuf.Refptr,the primitive returns in the Outptr parameter,the pointer to the record where the match was found,in the Out parameter,the contains of the record where the match was found.

Second,if the first parameter is not in the list of the Descript record list pointed by the Refbuf.Refptr,the primitive returns in the Out parameter,the value of the highest Intid used until now.There is no more information needed because we can use the Nextfree pointer of record 0. /

Fig 24.

```

Defref_type = record [ Entryname : Char(8);
                       Defptr   : Descript_type;
                       Refptr   : Descript_type ]
Descript_type = record [ Typedesc : Interface;
                        Typeop   : ↑ Descript_type;
                        Typenp   : ↑ Descript_type;
                        Intid    : Integer;
                        Modname  : Char(8) ]

Refbuf : Defref_type;
Descrbuf : Descript_type;
Out : Descript_type;
Outptr : ↑ Descript_type;
Workintid : Integer;

getfile DEFREF (entryname) into Refbuf;
if record not found then create an empty record withentryname;
endif
if Defptr = NIL then do
    getfile DESCRIPT (0) into Descrbuf;
    Refbuf.Defptr := Descrbuf.Nextfree;
/Nextfree is a field that exists only in record 0 and is a pointer
to the next free record in the DESCRIPT file. /
    Descrbuf.Nextfree := Descrbuf.Nextfree + 1;
    putfile DESCRIPT (0) from Descrbuf;
    putfile DEFREF (entryname) from Refbuf;
/ We have just updated the pointer to the next free record in DESCRIPT
file and also updated the pointer to the definitions in DEFREF file./

/ The next few lines create the new Descript record and written it
in the DESCRIPT file. /

    Descrbuf.Typedesc := convert (interface);
    Descrbuf.Typenp := NIL;
    Workintid := 1;
    Descrbuf.Intid := Workintid;
    Descrbuf.Modname := Catalogued_name;
    putfile DESCRIPT (Refbuf.Defptr) from Descrbuf;
enddo

/ convert is a primitive of the system for translating the source

```

Fig_24.

interface description in an other one more useful and easier
to handle. /

else do

```

getfile DESCRIPT (0) into Descrbuf;
Refbuf.Defptr := Descrbuf.Nextfree;;
Descrbuf.Nextfree := Descrbuf.Nextfree I;
putfile DESCRIPT (0) from Descrbuf;
Out.Typenp := Refbuf.Defptr;
putfile DESCRIPT (Outptr) from Out;

```

/ We have just updated the pointer to the next free record in
DESCRIPT file and also updated the pointer to the next definition
of the definition list in DESCRIPT file . /

/ The next few lines create the new Descript record and written
it in the DESCRIPT file. /

```

Descrbuf.Typedesc := convert (interface);
Descrbuf.Typenp := NIL;
Workintid := Out.Intid + 1;
Descrbuf.Intid := Workintid;
Descrbuf.Modname := Catalogued_name;
putfile DESCRIPT (Refbuf.Defptr) from Descrbuf;
enddo

```

endif

Fig_25.

```

ESD_type = record [ Symbol : Char(8);
                    Intid  : Integer;
                    Csectname : Char(8);
                    Nbrstmt  : Integer;
                    Type     : Char(8);
                    Id       : Integer;
                    Addr     : Integer;
                    . . . . .
                    . . . . .
                    . . . . . ]
Defref_type = record [ Entryname : Char(8);
                      Defptr     : ↑ Descript_type;
                      Refptr     : ↑ Descript_type ]
Descript_type = record [ Typedesc : Interface;
                       Typeop    : ↑ Descript_type;
                       Typenp    : ↑ Descript_type;
                       Intid     : Integer;
                       Modname   : Char(8) ]

Out : Descript_type;
Out1 : Descript_type;
Refbuf : Defref_type;
Descrbuf : Descript_type;
ESD : ESD_type;
Outptr : ↑ Descript_type;
Out1ptr : ↑ Descript_type;

for each ESD.Type = "ER" do
    getfile DEFREF (ESD.Symbol) into Refbuf;
    if record not found
        then do
            send errormsg1
        enddo
    else do

```

Fig_25.

```

if is_in_list (ESD.Intid,Refbuf.Refptr,Out,Outptr)
  then do
    if is_in_list (Out.Typedesc,Refbuf.Defptr,Out1,Out1ptr)
      then do
        Out1.Intid := Out.Intid;
        putfile DESCRIPT (Out1ptr) from Out1;
        load Out1.Modname;
        . . . . .
      / Here should be done the current treatment /
        enddo
      else do
        send errormsg2
        enddo
      endif
    else do
      send errormsg3
      enddo
    endif
  enddo
endif
endfor

```

Exemple of error message

```
errormsg1 := "CALL FROM LINE ESD.Nbrstmt ESD.Symbol DOES NOT EXIST !"
```

```
errormsg2 := "CORRESPONDING DEFINITION FOR THE REFERENCE ESD.Symbol
              CALLED IN LINE ESD.Nbrstmt IN ESD.Csectname DOES NOT
              NOT EXIST !"
```

```
errormsg3 := "REFERENCES OF ESD.Symbol CALLED IN LINE ESD.Nbrstmt
              HAVE BEEN DELETED SINCE LAST RUN !". /
```

2. Conclusions of part 4.

As described within the algorithms and the file description I have taken most of the practical details as ESD cards from IBM machines. This is not because they are better but only because I have many manuals from IBM and that it was easy to find informations about their systems. Here most of the details were taken from their VM370 system. In all cases, the functions of such parts of the operating system like linkers, compilers and loaders are the same even with other systems, even they differ in the way of doing it.

These are the reasons why I have taken such decision.

As it can be shown, the solution adopted here can also be done even with fetches. The only modification that must be done is the fact that the compilers must no more address the library of load modules but the library of executable module for having the coding.

There are also two possibilities for accessing the load modules. The first one, is to introduce no changes to the operating system for handling the code of the load module. This involves that the name of the module catalogued must be directly put into the record that describes the interfaces.

The second one, is to put directly an access to the module code to the same record. The first possibility will perhaps see as better because there are no changes and then, the old ways are compatible with it.

Part 5 : Conclusion .

At this point, I have to mention that as it is the evolution today, to write in high level language and to separate a software into pieces, the interface description is important .

When designing a software, the first part of the job is to define precisely what are the requirements of the software . The second part is to establish all the interfaces in order to be able to give some module to all people working on the project . The third one, is to implement the functions . While implementing them, the program only refer them selves to the interfaces as defined earlier . If some interfaces are changed, may be a lot of module should also change .

What this thesis solve in two things :

1) Type checking : We are now able to construct systems that can check the correspondance between the type definition as done in the caller and the callee program . This inforce the reliability of the system because no misconception nor ambiguities at interface level can be passed without be flagged .

This is done at a low price and at quite no execution price what is the most important . It has no difficult or special handling of files . It does not mean to change any logic of the operating system . I have only add for the compilers the duty of copying some part of the table they handle and for the linkage editor the type checking .

These modifications are easy to implement and will involve very strong assurance of having fully coherent interfaces of the software .

2) Variable interface : We can now use different interfaces . Not all cases are handled, but the most common flexibility wanted . This ability is adding a function to the module and replacing the old by the new or suppress a function to the module and replacing the old by the new .

As interfaces can be seen as specifications of modules or as the functionality of the modules, if we permit to change them in any way, this will increase the software crisis what is not our aim.

Indeed, if the interfaces have to change too often or too completely, is it not the symptom that the software was not understood (requirements point of view) . This also will not be better to reconstruct logically the whole software first . I think that if this happens, it is the symptom that it is better to stop, to reconstruct logically the whole, to redefine the interfaces in a convenient way and after that to redistribute the work .

As this work has been done with the idea of not changing the whole existent software such as operating systems, this involves that the solution found will may be not be theoretically elegant or that there is no other solution . In fact, it should be possible to construct an operating system that leads to totally different considerations and which can handle the checking directly as a requirement, but for doing this I would need a few years and not only a few months .

My hope is that this thesis will serve as a means for helping in solving the software crisis just a little . If this is the case or if this will serve anyone in its work, then I think that this thesis was fruitful .

INSTITUT D'INFORMATIQUE
FNOP NAMUR

ABOUT DYNAMIC
INTERFACES
TODAY

(ANNEXES)

ANNEE ACADEMIQUE
1984/85

MEMOIRE PRESENTE PAR
OLIVIER STAES
EN VUE DE L'OBTENTION
DU DIPLOME DE
LICENCIE ET MAITRE
EN INFORMATIQUE

GLOSSARY

GLOSSARY

- Actual parameter : parameter which definition is done in the caller program . It is not seen here as the value of the parameter at execution .
- Compiler : Software that translates a program from a high level language into machine language .
- Completeness : the degree to which a program can handle input variants .
- Correctness : the degree to which a program matches its specification .
- Database : a set of structured data used for a variety of purposes .
- Debugging : location and fixing an error in a program .
- External procedure : is a procedure (subroutine or function) defined "after" another procedure but which can be involved by the others . Here, when dealing with it, we mean procedure that are compiled separately from the caller . These procedures are then concerned by this thesis .
- Formal parameter : parameter which definition is done in the caller program . It is not seen here as the value of the parameter at execution time .
- High level language : programming language near the logical information or mathematical one . Must be translated into a machine code by a program called compiler .
- Internal procedure : can be subroutine or function . It is called internal because these programs will be compiled together .
- Module : the basic functional unit of a program . Modules are designed added and tested separately .
- Overloading : is using two times or more a same symbol . This symbol can represent a function call, an operator definition as "+" . These symbols can be distinguish by the objects type or the number of objects they can manipulate .
- Specification : of software is a way of hiding information and post-passing implementation decision . Here, I use specification as a mean of defining precisely the software such that both user and designer can discuss each other in an unambiguous way .

REFERENCES

Debugging

- Bats P. C. & all : An Approach to High Level Debugging of Distributed System
 in ACM Sigsoft/Sigplan : Symposium on High Level Debugging (1983)
 : High Level Debugging of Distributed System
 in The Journal of System and Software, vol 3 n^r 4 (Dec 1983)
- Beander : Van debug : An Interactive, Symbolic, Multilingual Debugger
 in ACM Sigsoft/Sigplan : Symposium on High Level Debugging (1983)
- Blain J. : Extendable non Interactive Debugging
- Brugger B. & all : Generalized Path Expression : A High Level Mechanism
 in The Journal of System and Software, vol 3 n^r 4 (Dec 1983)
- Briands & all : Development of a Debugger for Concurrent Language
 in ACM Sigsoft/Sigplan : Symposium on High Level Debugging (1983)
- Cargill T. A. : The Blit Debugger
 in The Journal of System and Software, vol 3 n^r ' (Dec 1983)
- Clarcke, Richardson : The Application of Error - Sensitive Testing
 Strategics to Debugging
 in ACM Sigsoft/Sigplan : Symposium on High Level Debugging (1983)
- Coak, Ler : Dymos : A Dynamic Modification System
 in ACM Sigsoft/Sigplan : Symposium on High Level Debugging (1983)
- Cordell : Multilingual Debugging with the SWAT High Level Debugger
 in ACM Sigsoft/Sigplan : Symposium on High Level Debugging (1983)
- Elliot B. : A High Level Debugger for PL/1, Fortran & Basic
 in Software Practice & Experience, vol 12 (1982)
- Ferrante : High Level Debugging with a Compiler
 in Software Practice & Experience, vol 12 (1982)
- Fritzon P. : A Systematic Approach to Advanced Debugging through incre-
 mental compilation
 in Software Practice & Experience, vol 12 (1982)

- Fritzon P. : Symbolic Debugging through Incremental Compilation in
an Integrated Environment
in The Journal of System and Software, vol 3 n^r 4 (Dec 1983)
- Gentleman W. M. & Haeksma H. : Hardware Assisted High Level Debugging
in The Journal of System and Software, vol 3 n^r 4 (Dec 1983)
- Gliksman R. : Criteria for a Debugging Language
- Hamelot : Debugging "Level" : Step Wise Debugging
in ACM Sigsoft/Sigplan : Symposium on High Level Debugging
(1983)
- Hill C.R. : A Real Time Microprocessor Debugging Technics
in ACM Sigsoft (Jul 1983)
- Johnson J.D. & Kenney G.W. : Implementation Issues for a Source Level
Symbolic Debugger
in ACM Sigsoft (Jul 1983)
- Kishimoto : An Experimental Debugger in a Limited Programming Environment
in ACM Sigsoft/Sigplan : Symposium on High Level Debugging
(1983)
- Kulzrad M. E. : Extending the Interactive Debugging System Helper
- Müllerburg : The Role of Debugging within Software Engineering Environment
in ACM Sigsoft/Sigplan : Symposium on High Level Debugging
(1983)
- Henstein O. : High Level Debugging Assistance via Optimizing Compiler
Technology
in ACM Sigsoft/Sigplan : Symposium on High Level Debugging
(1983)
- Powell M. L. & Linton M. A. : A Data Base Model of Debugging
in ACM Sigsoft/Sigplan : Symposium on High Level Debugging
(1983)
- Sedlmeyer R. L. & all : Knowledge Based Fault Localization in Debugging
in The Journal of System and Software, vol 3 n^r 4 (Dec 1983)
- Seidmer & Tindall : Interactive Debug Requirements
in ACM Sigsoft/Sigplan : Symposium on High Level Debugging
(1983)
- Supnik R. M. : Debugging under Simulation
- Tisher & all : Static Analysis of Programs as an Aid to Debugging
in ACM Sigsoft/Sigplan : Symposium on High Level Debugging
(1983)

- Walter : DELTA : A Universal Debugger for CP 6
in ACM Sigsoft/Sigplan : Symposium on High Level Debugging
(1983)
- Weber : Interactive Debugging of Concurrent Programs
in ACM Sigsoft/Sigplan : Symposium on High Level Debugging
(1983)
- Witscharsk C. A. : The Real Time Debugging Monitor for the Bell System 1A
in Software Practice and Experience, vol 13 (1983)
- Zellweger : An Interactive High Level Debugger for Control Flow Optimized
Programs
in ACM Sigsoft/Sigplan : Symposium on High Level Debugging
(1983)

Testing

- Elmendorf W. R. : Disciplined Software Testing
- Foster K. A. : Comment on "The Application of Error Sensitive Testing
Strategics to Debugging"
in ACM Sigsoft Engineering Notes, vol 8 n^F 5 (Oct 1983)
- Ginzberg G. : Notes on Testing Real Time System Programs
in IBM System Journal, vol 4 n^F 1 (1965)
- Heuermann C. A. & all : Automated Test and Verification
in IBM Technical Disclosure Bulletin, vol 17 n^F 7 (Dec 1974)
- Heuermann C. A. & all : Automated Test with Interface Verification
Simulation
in IBM Technical Disclosure Bulletin, vol 17 n^F 7 (Dec 1974)
- Heuermann C. A. & all : Verification of Test Case Output
in IBM Technical Disclosure Bulletin, vol 17 n^F 7 (Dec 1974)
- Jard C. & Boehmann G. V. : An Approach to Testing Specifications
in ACM Sigsoft/Sigplan : Symposium on High Level Debugging
(1983)
- King N. J. : Testing Conversional Systems
- Myers G. : The Art of Software Testing
A Wiley Interscience Publication (1978)
- Taylor R. N. : An Integrated Verification and Testing Environment
in Software Practice and Experience, vol 13 (1983)

Used for Solving the Problem

- Ancona M. & all : Integrating Library Modules into Pascal Programs
in Software Practice and Experience, vol 14 (May 1984)
- Andrews G. R. : An Alternation Approach to Arrays
in Software Practice and Experience, vol 12 (1982)
- Beech D. : Modularity of Computer Languages
in Software Practice and Experience, vol 12 (1982)
- Booch G. : Software Engineering with ADA
in Benjamin/Cummings Publication (1983)
- Briggs J. S. : Two Implementations of the ADA Program Library
in Software Practice and Experience, vol 14 (May 1984)
- Hennesay J. & Elmquist M. : The Design and Implementation of Parametric
Types in Pascal
in Software Practice and Experience, vol 12 (1982)
- Holdevorth D. : A System for Analysing ADA Programs at Run Time
in Software Practice and Experience, vol 13 (1983)
- King J. : A Verifying Compiler
- Knuth E. : Software Structuring : A Pragmatic Approach
in Lecture Notes in Computer Science : Specification and Design
of Software Systems Conference 82 . Springer Verlag (1983)
- Kowalkowski T. : Parameter Passing Mechanisms and Run Time Data Structures
in Software Practice and Experience, vol 11 (1981)
- Kozma L. & Laborczi L. : An Implementation Problems of Shared Abstract
Data Types
in Lecture Notes in Computer Science : Specification and Design
of Software Systems Conference 82 . Springer Verlag (1983)
- Kriskensen : Syntax-Directed Program Modularization in Integrated
Interactive Computing System
ECICS
- Mateti : "Pascal" Versus "C" : a Subjective Comparison
in Lecture Notes in Computer Science : Language Design and
Programming Methodology . Springer Verlag (1980)
- Meyer B. : Principles of Package Design
in CACM, vol 25 n^F 7 (Jul 1982)
- Parnas D. L. : A Technique for Software Module Specification with Examples
in Communication of ACM, vol 15 n^F 5 (may 1972)

- Richardson : A Critique of Module
in Lecture Notes in Computer Science Language Design and
Programming Methodology . Springer Verlag (1980)
- Robinson : The Design of a Successor to Pascal
in Lecture Notes in Computer Science Language Design
Programming Methodology . Springer Verlag (1980)
- Sale A. H. J. : Forward Declared Procedures; Parameter-lists and Scape
in Software Practice and Experience, vol 11 (1981)
- Shaw : The Logical Design of Operating Systems (1974)
- Stroustup B. : Adding Classes to the "C" Language
in Software Practice and Experience, vol 13 (1983)
- Stroustup B. : "Long Return" : A Technique for Improving the efficiency
of Inter Module Communication
in Software Practice and Experience, vol 11 (1981)
- Tsin Y. M. : Extending the Power of Pascal's External Procedure Mechanism
in Software Practice and Experience, vol 12 (1982)
- Wallis P. J. L. : Handling Type Information when Compiling Language with
User Defined Types
in Software Practice and Experience, vol 11 (1981)
- Welsh & all : A comparison of two Notations for Process Communication
in Lecture Notes in Computer Science : Language Design and
Programming Methodology . Springer Verlag (1980)
- Welsh & all : Pascal Plus - Another Language for Modular Multiprogramming
in Software Practice and Experience, vol 9 (1979)
- Wirth : The Module : A system Structuring Facility in High Level
Programming Language
in Lecture Notes in Computer Science : Language Design and
Programming Methodology . Springer Verlag (1980)

Maintainance

- Aron : The Program Development Process (vol I & II)
Addison Wesley (1983)
- Baber : Software Reflected
North Holland (1982)
- Bauer & all : Towards a Large Spectrum Language to Support Program Specification and Program Development
in Lecture Notes on Program Construction . Springer Verlag(1978)
- Bauer : Program Development by Stepwin Transformations
in Lecture Notes on Program Construction . Springer Verlag(1978)
- Bauer : Systematics of Transformation Rules
in Lecture Notes on Program Construction . Springer Verlag(1978)
- Bauer : Special Transformation Techniques
in Lecture Notes on Program Construction . Springer Verlag(1978)
- Bachm & all : Characteristics of Software Quality
North Holland (1978)
- Brooks : The Mythical Main Month
Addison Wesley (1975)
- Chambers J. M. : Evolution and Maintainance are both Semantically insoud
in ACM Sigsoft Engineering Notes, vol 3 n^r 3 (Jul 1983)
- Chapin N. :Software Maintainance with Four Generations Languages
in ACM Sigsoft Engineering Notes, vol 3 n^r 5 (Jan 1984)
- Couger : Evolution of Business Systems Analysis Techniques
in Writings of the Revolution. Yourdon (1982)
- Darlington : A System which Automatically Improves Programs
in Programming Methodology. Springer Verlag (1978)
- De Balbine : Beller Manpower Utilization Using Automatic Restructuring
in Writings of the Revolution. Yourdon (1982)
- Dershowitz : The Evolution of Programs
Brichtrouses Boston (1983)
- Deutsch M. S. :Software Verification and Validation . Realistic Project Approach
Prentice Hall (1982)
- Donelson: Project Planning and Control
in Writings of the Revolution. Yourdon (1982)

- Dunn, Ulmann : Quality Assurance for Computer Software
Mc Growhill (1982)
- Fagan : Design and Cods Inspections in Reducing Errors in Program
Development
in Writings of the Revolution. Yourdon (1982)
- Fox : Software and its Development
Printice Hall
- Hatstead : Toward a Theorical Basis for Estimating Programming Effort
in Writings of the Revolution. Yourdon (1982)
- Jones : Measuring Programming Quality and Productivity
in Writings of the Revolution. Yourdon (1982)
- Martin & Estrim : Models of Computations and Systems
in Writings of the Revolution. Yourdon (1982)
- Partsch : Examples for Change of Types and Object Structures
in Lecture Notes on Program Construction (1978)
- Putmann & Fitzimmens : Estimating Software Casts
in Writings of the Revolution. Yourdon (1982)
- Randell : System Structure for Software Fault Tolerance
in Programming Methodology. Springer Verlag (1978)
- Sidner, Uttal : Knowledge Representation Tools for the Design ,
Training and Use of Information System
in Integrated Interactive Computing System (ECICS)
- Van Tassel : Program Style, Design, Efficiency, Debugging and Testing
Printice Hall (1978)
- Walstone & Felix : A Method of Programming Measurement and Estimation
in Writings of the Revolution. Yourdon (1982)
- Walberg R. : Conversion of Computer Software
Printice Hall (1982)

General references

- Attardi : Extending the Power of Programming by Examples
in Integrated Interactive Computing Systems ECICS
- Balog K. & all : Software Development in LDM
in Lecture Notes in Computer Science Specification and Design
of Software System . Springer Verlag (1983)
- Barnard : Experiences with Building a SoftwareFramework for Interactive
System
in Integrated Interactive Computing System ECICS
- Barnes J. C. : An Overview of ADA :
in Software Practice and Experience, vol 10 (1980)
- Barra & Dahl O. J. : A Portable Toolbox for Observation of Simula Execution
in ACM Sigsoft/Sigplan Symposium on High Level Debugging (1983)
- Bates : A Case Study of a Methode for Determing the Necessary Characteris-
tics of a Natural Language Interface
in Integrated Interactive Computing Systems ECICS
- Baner : Detailization and Lazy Evaluation, Infinite Objects and Pointer
Representation
in Lecture Notes Program Construction. Springer Verlag (1978)
- Belady and Lehman : A Model of Large Program Development
in Writing of the Revolution. Yourdon Press (1982)
- Bevan : The Design of User-friendly Systems for generating Intelligent
Dialogues
in Integrated Interactive Computing Systems ECICS
- Bird R. : Programs and Machines
in Wiley - Interscience Publication(1976)
- Blank & all : Software Engineering : Methods and Techniques
John Wiley & Sons(1983)
- Bochmann : Concepts for Distributed Systems Design(1983)
- Bochm B. W. & all : A Primer in Structured Design
North Holland(1978)
- Bochm B. W. : Software and its Impact
in Writing of the Revolution. Yourdon Press (1982)
- Boyer R. S. & Strother Moore J. : Program Verification Prize
in Software Practice and Experience, vol 8 n^r 3 (Jul 1983)

- Boyer R. S. & Strother Moore J. : The Correctness Problem in Computer Science
Academic Press (1981)
- Braun G. L. : ADA : Programming in the 80's
in Computer Journal IEEE (1981)
- Brender R. F. & all : What is ADA?
in Computer Journal IEEE (1981)
- Brooks F. P. : The Mythical Man-Month
Addison Wesley (1975)
- Brooks F. P. : The Mythical Man-Month Revisited
in ACM Sigsoft Software Engineering Notes, vol 7 n^r 2 (April 82)
- Broy : Semantics of Nondeterministics and Noncontinuous Constructs
in Lecture Notes, Program Construction, Springer Verlag (1978)
- Buxton : Software Engineering
in Programming Methodology (IFLP WG 2-3), Springer Verlag (1978)
- Cardelli : Two-dimensional Syntax for Functionnal Languages
in Integrated Interactive Computing Systems ECICS
- Carlson W. E. : ADA : A Promising Beginning
in Computer Journal IEEE (June 1981)
- Chrustonsen K. & all : A Perspective on Software Science
in IBM System Journal, vol 20 n^r 4 (1981)
- Cointe : A VLISP Implementation of SMALLTALK -76
in Integrated Interactive Computing Systems ECICS
- Cooper : Böhm and Jacopini's Reduction of Flow Charts
in Writing of the Revolution, Yourdon Press (1982)
- Costello S. H. : Software Engineering under Deadline Pressure
in ACM Sigsoft Software Engineering Notes, vol 9 n^r 5 (Oct 1984)
- Crinc : PDL - A Tool for Software Design
in Writing of the Revolution, Yourdon Press (1982)
- Dahl O. J. : An Approach to Correctness Proofs for Semi Coroutines
in Programming Methodology (IFLP WG 2-3), Springer Verlag (1978)
- Dahl O. J. & Dijkstra E. W. : Structured Programming
Academic Press (1975)
- Davis W. S. : System Analysis
Academic Press (1983)

- Diaz Herrera J. L. : Pragmatic Problem with Stepwise Refinement Program Development
in Software Practice and Experience, vol 9 n^F 2 (April 1984)
- Dijkstra E. W. : A More Formal Treatment of a Less Simple Example
in Lecture Notes, Program Construction . Springer Verlag(1978)
- Dijkstra E. W. : Stationary Behaviour of Same Ternary Networks
in Lecture Notes, Program Construction : Springer Verlag(1978)
- Dijkstra E. W. : Finding the Correctness Proof of a Concurrent Program
in Lecture Notes, Program Construction . Springer Verlag(1978)
- Dijkstra E. W. : On the Interplay Between Mathematics and Programming
in Lecture Notes, Program Construction . Springer Verlag(1978)
- Dijkstra E. W. : A Theorem About Add Powers of Odd Integers
in Lecture Notes, Program Construction . Springer Verlag(1978)
- Dijkstra E. W. : In Honour of Fibonacci
in Lecture Notes; Program Construction . Springer Verlag(1978)
- Dijkstra E. W. : On the Foolishness of Natural Language Programming
in Lecture Notes, Program Construction . Springer Verlag(1978)
- Dijkstra E. W. : Program Inversion
in Lecture Notes, Program Construction . Springer Verlag(1978)
- Dijkstra E. W. : The Humble Programmer
in Programming Methodology (IFLP WG 2-3). Springer Verlag(1978)
- Dijkstra E. W. : Guarded Commands, Non Determining and Formal Derivation of Programs
- Dijkstra E. W. : The Structure of "the" Multiprogramming System
in Writing of the Revolution. Yourdon Press (1982)
- Dyer M. : Software Development Practices
in IBM System Journal, vol 19 n^F 4 (1980)
- Emery : Cost/Benefit Analysis of Information Systems
in Writing of the Revolution. Yourdon Press (1982)
- Fisher & Böcker : The Nature of Design Processes and How Computer System can support them
in Integrated Interactive Computing Systems ECICS
- Follett : Describing the Complete Effect of Programs
in Lecture Notes in Computer Science : Language Design and Programming Methodology . Springer Verlag (1980)

- Foo : Algebraic Specifications and Transition Graphs
 in Lecture Notes in Computer Science : Language Design and
 Programming Methodology . Springer Verlag (1980)
- Franckel E. C. : A Contracts Course for Software Engineers
 in ACM Sigsoft Software Engineers Notes, vol '7 n^F 4 (Oct 1982)
- Gerhart : A Derivation Oriented Proof of the Schare - Warriier Marking
 Algorithm
 in Lecture Notes : Program Construction . Springer Verlag(1978)
- Giacalone : Toward a Formally Based Programming Environment
 in Integrated Interactive Computing System ECICS
- Glass R. L. : Software Engineering Economics : Learning from Failures
 in Software Forum 1983
- Graham : A Software Design and Evaluation System
 in Writing of the Revolution. Yourdon Press (1982)
- Gries D. : On Structured Programming
 in Programming Methodology (IFLP WG 2-3). Springer Verlag(1978)
- Gries D. & Owicki : An Axiomatic Proof Technique for Parallel Programs
 in Programming Methodology (IFLP WG 2-3). Springer Verlag(1978)
- Gries D. : The Schore - Warriier Graph Marking Algorithm
 in Lecture Notes, Program Construction . Springer Verlag(1978)
- Gries D. : Eliminate the Chaff
 in Lecture Notes, Program Construction . Springer Verlag(1978)
- Gries D. : Current Ideas in Programming Methodology
 in Lecture Notes, Program Construction . Springer Verlag(1978)
- Gries D. : Basic Axiomatic Definitions
 in Lecture Notes, Program Construction . Springer Verlag(1978)
- Gries D. : The Mutiple Assignment Statement
 in Lecture Notes, Program Construction . Springer Verlag(1978)
- Gries D. : Is Sometimes ever Better then Always ?
 in Lecture Notes, Program C_0nstruction . Springer Verlag(1978)
- Griffiths : Development of the Schare - Warriier Algorithm
 in Lecture Notes, Program Construction . Springer Verlag(1978)
- Griffiths : Programming Methodology and Language Implication
 in Lecture Notes, Program Construction . Springer Verlag(1978)
- Guttag : Notes on Types Abstractions
 in Lecture Notes, Program Construction . Springer Verlag(1978)

- Halassy B. : SZIAM : A Data Model Design Aid
in Lecture Notes in Computer Science : Specification and Design
of Software System, Springer Verlag (1983)
- Hansen B. : Structured Multiprogramming
in Programming Methodology (IFLP WG 2-3), Springer Verlag (1978)
- Hansen B. : The Programming Language Concurrent Pascal
in Programming Methodology (IFLP WG 2-3), Springer Verlag (1978)
- Hansen B. : The Architecture of Concurrent Programs
Prentice Hall (1977)
- Herbinson E. : A Human Movement Language for Computer Animation
in Lecture Notes in Computer Science : Language Design and
Programming Methodology, Springer Verlag (1980)
- Hext : Pattern Matching Commands
in Lecture Notes in Computer Science : Language Design and
Programming Methodology, Springer Verlag (1980)
- Hoare C. A. R. : The Engineering of Software : a Startling Contradiction
in Programming Methodology (IFLP WG 2-3), Springer Verlag (1978)
- Hoare C. A. R. : An Axiomatic Basis for Computer Programming
in Programming Methodology (IFLP WG 2-3), Springer Verlag (1978)
- Hoare C. A. R. : Proof of a program : FIND
in Programming Methodology (IFLP WG 2-3), Springer Verlag (1978)
- Hoare C. A. R. : Towards a Theory of Parallel Programming
in Programming Methodology (IFLP WG 2-3), Springer Verlag (1978)
- Hoare C. A. R. : Monitors : an Operating System Structuring Concept
in Programming Methodology (IFLP WG 2-3), Springer Verlag (1978)
- Hoare C. A. R. : Proof of Correctness of Data Representation
in Programming Methodology (IFLP WG 2-3), Springer Verlag (1978)
- Hoare C. A. R. : The Emperor's Old Cloths
in Writing of the Revolution, Yourdon Press (1982)
- Horning : Verification of EUCLID Programs
in Lecture Notes, Program Construction, Springer Verlag (1978)
- Horning : A Case Study in Language Design : EUCLID
in Lecture Notes, Program Construction, Springer Verlag (1978)
- Horning : Programming Language for Reliable Computing Systems
in Lecture Notes, Program Construction, Springer Verlag (1978)

- Ichbiah J. : ADA : Past, Present, Futur
in CACM Oct 1984, vol 27 n^F 10
- Jackson M. A. : System Development
Prentice Hall (1983)
- Jemers R. A. : System Design for Usability
in CACM, vol 24 n^F 8 (Aug 1982)
- Jensen K & Wirth N. : Pascal User Manual and Report
Springer Verlag (1978)
- Johnston & Lister : An Experiment in Software Science
in Lecture Notes in Computer Science : Language Design and
Programming Methodology. Springer Verlag (1980)
- Joslin P. H. : System Productivity Facility
IBM System Journal, vol 20 n^F 4 (1981)
- Katz : A Two-way Natural Language Interface
in Integrated Interactive Computing Systems ECICS
- Knuth D. E. : Notes on Avoiding "GOTO" Statements
in Writing of the Revolution. Yourdon Press (1982)
- Komorovski : An Abstract Prolog Machin
in Integrated Interactive Computing Systems ECICS
- Kramer J. : Distributed Computer Systems
in Lecture Notes in Computer Science : Specification and
Design of Software Systems. Springer Verlag (1983)
- Lee : DOPLS : A New Type of Programming Language
in Lecture Notes in Computer Science : Language Design and
Programming Methodology. Springer Verlag (1980)
- Lehman M. M. & all : Another Look at Software Design Methodology
in ACM Sigsoft Software Engineering Notes, vol 9 n^F 2 (April 1984)
- Lehman M. M. : Programs, Cities, Students - Limits to Growth ?
in Programming Methodology (IFLP WG 2-3). Springer Verlag (1978)
- Leveson N. G. : Software Safety
in ACM Sigsoft Software Engineering Notes, vol 7 n^F 2 (April 1982)
- Liebermann : Designing Interactive Systems from the User's Viewpoint
in Integrated Interactive Computing Systems ECICS
- Linger R. C. : Software Design Practices
in IBM System Journal, vol 19 n^F 4 (1980)

- London & all : Proof Rules for the Programming Language Euclid
in Lecture Notes, Program Construction. Springer Verlag (1978)
- Lowell J. A. : Programmer Productivity
Wiley Interscience Publication (1983)
- Ludewig J. : ESPRESO : A System for Process Control Software Specification
in Lecture Notes in Computer Science : Specification and
Design of Software Systems. Springer Verlag (1983)
- Mac Cracken D. D. & Jackson M. A. : Life Cycle Concept Considered Harmful
in ACM Sigsoft Software Engineering Notes, vol 7 n^r 2 (April 82)
- Maddison & all : Information System Methodologies (1983)
- Maté L. L. & all : System Cars and its Description Language
in Lecture Notes in Computer Science : Specification and
Design of Software Systems. Springer Verlag (1983)
- Mathews & all : The Computers in Cartoons
CACM, vol 27 n^r 11
- Merlin : The Practical Guide to Structured Systems Design
- Meyer B. : Towards a Two Dimensional Programming Environment
in Integrated Interactive Computing Systems EGICS
- Mikelsons : Interactive Program Execution
in ACM Sigsoft Sigplan (1983)
- Miller G. M. : The Magical Number Seven, Plus or Minus Two
in Writing of the Revolution. Yourdon Press (1982)
- Mills H. D. : Mathematical Foundations for Structured Programming
in Writing of the Revolution. Yourdon Press (1982)
- Mills H. D. & all : The Management of Software Engineering
in IBM System Journal, vol 13 n^r 4 (1980)
- Mills H. D. : Top Down Programming in Large Systems
- Mitchell R. S. : Program Design - A Practical Approach
in Lecture Notes in Computer Science : Specification and
Design of Software Systems. Springer Verlag (1983)
- Musa D. & all : Stimulating Software Engineering Progress : A Report of
the Software Engineering Planning Group
in ACM Sigsoft Software Engineering Notes, vol 8 n^r 2 (April 83)
- Myers G. J. : Program Design Validation System
in IBM Technical Disclosure Bulletin, vol 19 n^r 10 (March 1977)

- Myers G. J. : Program Design Query System
in IBM Technical Disclosure Bulletin, vol 19 n^r 10 (March 1977)
- Neuhold E. S. : Development Methodologics for Event and Message Based Application System
in Lecture Notes in Computer Science : Specification and Design of Software System. Springer Verlag (1983)
- Newman : Office-Talk-Zero : An Experimental Integrated Office System
in Integrated Interactive Computing Systems ECICS
- O'Neill D. : Software Engineering Progress
in IBM System Journal, vol 19 n^r 4 (1980)
- Owicki : Specifications and Proofs for Abstract Data Types in Concurrent Programs
in Lecture Notes, Program Constructions. Springer Verlag (1978)
- Owicki : Specification and Verification of a Network Mail System
in Lecture Notes, Program Constructions. Springer Verlag (1978)
- Pair : Some Theoretical Aspects of Program Construction
in Lecture Notes, Program Constructions. Springer Verlag (1978)
- Parnas D. L. : On a "Buzzword" : Hierarchical Structure
in Programming Methodology (IFLP WG 2-3). Springer Verlag (1978)
- Parnas D. L. : On the Design and Development of Program Families
in Programming Methodology (IFLP WG 2-3). Springer Verlag (1978)
- Peeters : Software Design : Methods and Techniques
Yourdon Press (1982)
- Pritchard : On the Prime Example of Programming
in Lecture Notes in Computer Science : Language Design and programming Methodology. Springer Verlag (1980)
- Quinnan R. E. : Software Engineering Management Practices
in IBM System Journal, vol 19 n^r 4 (1980)
- Quint : An interactive System for Editing Mathematical Documents
in Integrated Interactive Computing Systems ECICS
- Rajamaran M. K. : A Characterization of Software Design Tools
in ACM Sigsoft Software Engineering Notes, vol 7 n^r 4 (Oct 1982)
- Rechenberg & all : Softwareengineering
Osterreichish Computer Gesellschaft (1983)
- Reynolds : Programming with Transition Diagrams
in Programming Methodology (IFLP WG 2-3). Springer Verlag (1978)

- Rice : Build Program Technique
Wiley Interscience Publication (1982)
- Richardson G. L. & all : A Primer on Structured Design (1983)
- Riddle W. E. : A Study of Software Technical Maturation
in ACM Sigsoft Software Engineering Notes, vol 9 n^r 2 (April 1984)
- Ritchie : The Evolution of the UNIX Time Sharing Systems
in Lecture Notes in Computer Science : Language Design and
Programming Methodology. Springer Verlag (1980)
- Rohl : Why Recursion ?
in Lecture Notes in Computer Science : Language Design and
Programming Methodology. Springer Verlag (1980)
- Ross : Structured Analysis : A Language for Communicating Ideas
in Programming Methodology (IFLP WG 2-3). Springer Verlag (1978)
- Ross. : A Garbage Collecting Association Memory for Interactive Database
Systems
in Integrated Interactive Computing Systems ECICS
- Sakman : Explorating Experimental Studies Comparing Online and Offline
Programming Performance
in Writing of the Revolution. Yourdon Press (1982)
- Sammat & Cohen : A Language for Describing Concepts as Program
in Lecture Notes in Computer Science : Language Design and
programming Methodology. Springer Verlag (1980)
- Sansonnet : A Hardware Support for Interactive Programming Environments
in Integrated Interactive Computing Systems ECICS
- Schneider L. : Systems Design for, with and by the User
North Holland (1983)
- Schwartz J. T. : An Overview of Bugs
- Sergot : A Query-the-User Facility for Logic Programming
in Integrated Interactive Computing Systems ECICS
- Shaw B. C. & all : Introduction of a Formal Technic into a Software
Development Environment
in ACM Sigsoft Software Engineering Notes, vol 9 n^r 2 (April 1984)
- Shooman M. L. : Software Reliability : A Historical Perspective
in IEEE, vol R-33 n^r-1 (April 1984)

- Smith : Designing the STAR User Interface
in Integrated Interactive Computing Systems ECICS
- Smoliar : An Interactive Approach to Software Specification
in Integrated Interactive Computing Systems ECICS
- Sommerville I. : Software Engineering
Addison Wesley (1982)
- Sommerville I. : System Development Process
Addison Wesley (1982)
- Spicer J. C. : A Spiral Approach to Software Engineering Project Management
Education
in ACM Sigsoft Software Engineering Notes, vol 8 n^r 3 (Jul 1983)
- Steels : Orbit : An Application View of Object Oriented Programming
in Integrated Interactive Computing Systems ECICS
- Stemming V. & all : The ADA Environment : A Perspective
Computer Journal IEEE (1981)
- Stevens W. P. & all : Structured Design
in IBM System Journal n^r 2 (1974)
- Thadans A. J. : Interactive User Productivity
in IBM System Journal, vol 20 n^r 4 (1981)
- Tichy : RCS : A Revision Control System
in Integrated Interactive Computer Systems ECICS
- Turski : Software Engineering : Some Principles and Problems
in Programming Methodology (IFLP WG 2-3). Springer Verlag (1978)
- Walstone & Felix : A Method of Programming Measurement and Estimation
in Writing of the Revolution. Yourdon Press (1982)
- Weber : Programmentwurf and Programdokumentation
UDI - Verlag (1982)
- Weinberg M. A. : The Psychology of Computer Programming 1979
- Wertz : An Integrated, Interactive and Incremental Programming
Environment for the Development of Complex Systems
in Integrated Interactive Computing Systems ECICS
- Wertz : An Integrated LISP Programming Environment
in ACM Sigsoft/Sigplan Symposium on High Level Debugging (1983)

- Winograd T. : Beyond Programming Languages
in CACM, vol 22 n^r 7 (Jul 1979)
- Wirth N. : Program Development by Stepwise Refinement
in Programming Methodology (IFLP WG 2-3). Springer Verlag (1978)
- Wirth N. : A Personal Computer Based on a High-Level Language
in Lecture Notes in Computer Science : Language Design and
Programming Methodology. Springer Verlag (1980)
- Wirth N. : Algorithmes + Data Structures = Programs
Prentice Hall (1977)
- Woodward : A Measure of Flow Compliance on Program Text
in Writing of the Revolution. Yourdon Press (1982)
- Yourdon E. & Constantine L. L. : Structured Design
Prentice Hall (1979)
- Yourdon E. : The Practical Guide to Structured Design
Yourdon Publication (1981)
- Yourdon E. : The Future of Software
in Software Forum (1983)
- Ziegler C. A. : Programming System Methodologies
Prentice Hall (1983)
- Ziegler S. : ADA for the Intel 432 Micro Computer
Computer Journal IEEE (June 1981)

AID / AIDSYS

This is often what is given to the users, in order to propose to them the new functions and their interfaces.

VOLUME 800 : PROGRAM MANAGEMENT
 CHAPTER 52 : external Interface
 SECTION 88.8 : AIDSYS V7.5

P R O P O S A L

DISTRIBUTION
 FREE

3.2.4 Funktion SYSTEM-EDIT

- Die folgenden Werte des Feldes ASASEKEY haben sich geaendert:

ASASETSK	X'24'	-->	X'18'
ASASEQU	X'18'	-->	X'1C'
ASASEPND	X'1C'	-->	X'20'
ASASEALL	X'20'	-->	X'24'
ASASEKMX	X'2C'	-->	X'28'
ASASEFTT	nil	-->	X'28'
ASASEFTT	X'28'	-->	gestrichen
ASASETL	X'2C'	-->	gestrichen

- Die Anzeigen ASASEFB.ASASEFIT bzw. .ASASEFTS werden nicht mehr abgefragt.

- Neue Anzeigen:

ASASEFB.ASASESTB EQU X'02' wird nur bei Keyword AUDIT ausgewertet in der Bedeutung: 'nur Original-Tabelle gewuenscht'

ASASEFB.ASASEIXV EQU X'01' angegebener Index ist gueltig. wird nur dann ausgewertet, falls der Index gleich Null ist.

- Neue Rueckkehr-Parameter:

ASASENAL	EQU	X'3C'	nur bei Keyword AUD1/AUD2 moeglich. wegen Speichermangel wird nur die Originaltabelle geliefert.
ASASELØP	EQU	X'40'	PCB-Schleife entdeckt. Puffer wurde mit Information gefuellte, es ist aber kein Fortsetzungsaufwurf mehr moeglich
ASASESER	EQU	X'44'	Tabelle ist wegen kaputter Pointer nicht ausgebar (Systemfehler)
ASASENP	EQU	X'48'	Nur fuer PCB-Zugriffe: Die gelieferte Information ist moeglicherweise kein PCB. (Plausibilitaetscheck liefert Fehler)

3.2.5 Funktion 'Open Memory Pool'

This should always be provided, in order to see what are the modules that can call this one and that are called by this one .

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT	
000000						1		DPACCESS START	00004000
						2		TITLE 'INTERFACE BETWEEN USER AND AIDSYS'	00009000
						3		*****	00019000
						4	*		00029000
						5	*	MODULE FUNCTIONAL	00039000
						6	*		00049000
						7	*	DESCRIPTION	00059000
						8	*		00069000
						9		*****	00079000
						10		*****	00099000
						11	*	*****	00109000
						12	*		00119000
						13	*	THE PURPOSE OF THIS MODULE IS TO PROVIDE FOR P1	00129000
						14	*	PROGRAMS, ACCESS TO DUMPFIL.	00139000
						15	*		00149000
						16	*	THE FOUR FUNCTIONS PROVIDED BY THIS PROGRAM ARE :	00159000
						17	*		00169000
						18	*	1.CMFCT : GET INFORMATION ABOUT CSECT MAP	00179000
						19	*		00189000
						20	*	2.GAFCT : GET INFORMATION ABOUT THE CSECT CONTAINING	00199000
						21	*	THE ADDRESS GIVEN BY THE CALLER	00209000
						22	*		00219000
						23	*		00229000
						24	*	3.PEFCT : PROVIDE THE USER THE INFORMATION ABOUT	00239000
						25	*	THE PROGRAM UNIT(PE)	00249000
						26	*		00259000
						27	*	4.CRIAFCT : PROVIDE THE USER THE INFORMATION	00269000
						28	*	ABOUT THE CONCISE REPS INFORMATION AREA	00279000
						29	*		00289000
						30	*	*****	00299000
						31	*		00309000
						32	*		00319000
						33	*	*****	00329000
						34	*		00339000
						35	*		00349000
						36	*	FOR MOST PRECISE INFORMATION ABOUT THE FUNCTION	00359000
						37	*	PLEASE, SEE WHERE THEY ARE CALLED.	00369000
						38	*		00379000
						39	*		00389000
						40	*	*****	00399000
						41	*		00409000
						42	*		00419000
						43	*	*****	00429000
						44	*		00439000
						45	*		00449000
						46	*	THE USER IS EXPECTED TO GIVE :	00459000
						47	*		00469000
						48	*	#1.THE REGISTER 1 HAS TO POINT TO THE CALLER	00479000
						49	*	PARAMETER LIST	00489000
						50	*		00499000
						51	*	#2.THE CALL HAS TO BE DONE VIA A BAL(R) 14,15(=..)	00509000
						52	*		00519000
						53	*	#3.THE CALLER HAS TO GIVE OR AN ADDRESS OF HIS READ	00529000

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT	
						54	*	DUMPFIL ROUTINE AND THE ITH OF HIS ACCESSED TASK	** 00539000
						55	*	OR NO ADDRESS OF A READ ROUTINE BUT A VALID LINK	** 00549000
						56	*	NUMBER OF HIS DUMPFIL.	** 00554000
						57	*		** 00559000
						58	*	#4. FOR THE OTHER FUNCTION THEN GAFCT, HE HAS TO	** 00569000
						59	*	GIVE A BUFFER ADDRESS AND THE LENGTH OF THE	** 00579000
						60	*	LENGTH OF THE BUFFER.	** 00589000
						61	*		** 00599000
						62	*	#5.!!!!!! IMPORTANT !!!!!!!	** 00609000
						63	*		** 00619000
						64	*	THE USER CAN NOT GIVE AN NULL VALUE	** 00629000
						65	*	=====	** 00639000
						66	*		** 00649000
						67	*	TO A MANDATORY PARAMETER, IT WILL BE UNDERSTANDED	** 00651000
						68	*	=====	** 00654000
						69	*		** 00656000
						70	*	AS IF NO PARAMETER WAS GIVEN!!!	** 00656700
						71	*	=====	** 00657500
						72	*		** 00658200
						73	*		** 00659900
						74	*	*****	** 00669000
						75	*		** 00679000
						76	*		** 00689000
						77	*	*****	** 00699000
						78	*		** 00709000
						79	*		** 00719000
						80	*	THE MODULE ALWAYS GIVE THE RETURN CODE INTO THE	** 00729000
						81	*	SPECIAL FIELD OF THE PARAMETER LIST.	** 00739000
						82	*		** 00749000
						83	*	THE RETURNING PARAMETER ARE SET INTO THE USER	** 00759000
						84	*	BUFFER. OF COURSE NOT THE GAFCT, WHERE THERE IS	** 00769000
						85	*	NO BUFFER, IN THIS CASE THE RETURNING INFORMATIONS	** 00779000
						86	*	ARE AFTER THE CALLER ARGUMENTS	** 00789000
						87	*		** 00799000
						88	*		** 00809000
						89	*	THE RETURN CODE FROM THIS MODULE ARE :	** 00819000
						90	*		** 00829000
						91	*	\$1.X'00' NO PROBLEMS.	** 00839000
						92	*		** 00849000
						93	*	\$2.X'04' FORMAL ERROR	** 00859000
						94	*		** 00869000
						95	*	-BUFFER NOT WORD ALIGNED	** 00879000
						96	*	-NO BUFFER GIVEN	** 00889000
						97	*	-NO BUFFER LENGTH GIVEN	** 00890000
						98	*	-NO TCB ADDRESS GIVEN	** 00902000
						99	*	-NO SYSBASE ADDRESS GIVEN	** 00903000
						100	*	-NEITHER A(READ) NOR LINK# GIVEN	** 009095000
						101	*	-INVALID ITH GIVEN	** 00919000
						102	*	-INVALID SUBFUNCTION CODE	** 00929000
						103	*		** 00939000
						104	*	\$3.X'08' MODULE ERROR, THIS IS WHEN AN UNEXPECTED	** 00941000
						105	*	ERROR HAS OCCURED IN A OTHER MODULE CALL.	** 00944000
						106	*		** 00946000
						107	*	\$4.X'0C' A(TCB) OR A(XVT) OR A(SYSBASE) ARE NOT	** 00946200
						108	*	WORD ALIGNED.	** 00946400

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT		
						109	**		**	00946600
						110	**	\$5.X'10' FILE IS NOT OPEN.	**	00946900
						111	**		**	00947100
						112	**	\$6.X'14' SPECIFIED TASK NOT FOUND.	**	00947300
						113	**		**	00947600
						114	**	\$7.X'18' TASK QUALIFICATION NECESSARY.	**	00947800
						115	**		**	00948000
						116	**	\$8.X'1C' INVALID LINK NUMBER.	**	00948300
						117	**		**	00948500
						118	**	\$9.X'20' INVALID INDICATOR OF DUMPFIL TYPE.	**	00948700
						119	**		**	00948800
						120	**	\$10.X'24' A(READ) GIVEN BUT NO ITH GIVEN	**	00948850
						121	**		**	00948900
						122	**	\$11.X'28' BUFFER FULLFILED WITH INFORMATION BUT	**	00949000
						123	**	FURTHER CALL ARE NEEDED IF ALL INFORMATION ARE NEEDED	**	00959000
						124	**	IF GETADDRESS FUNCTION, THEN THAT WILL SAY ADDRESS NOT	**	00962000
						125	**	ALLOCATED.	**	00965000
						126	**		**	00969000
						127	**	\$12.X'2C' BUFFER WRITE UNACCESSIBLE IN FULL LENGTH	**	00979000
						128	**	IF GETADDRESS FUNCTION, THEN THAT WILL SAY INPUT	**	00982000
						129	**	ADDRESS IN CONFLICT WITH FLAG.	**	00985000
						130	**		**	00989000
						131	**	\$13.X'30' PE NOT FOUND BUFFER IS EMPTY	**	01039000
						132	**	IF GETADDRESS FUNCTION, THEN THAT WILL SAY THAT	**	01042000
						133	**	NO PAGE IS DUMPED.	**	01045000
						134	**		**	01049000
						135	**	\$14.X'34' BUFFER LENGTH TOO SHORT TO CONTAIN	**	01059000
						136	**	ONLY ONE ANSWER.	**	01069000
						137	**	IF GETADDRESS FUNCTION, THEN THAT WILL SAY THAT	**	01069700
						138	**	ERROR IN SYSTEM.	**	01070500
						139	**		**	01071300
						140	**	\$15.X'38' NO PROGRAM IS LOADED.	**	01072000
						141	**		**	01072800
						142	**	\$16.X'3C' NO CSECT MAP FOUND, BUFFER IS EMPTY.	**	01073600
						143	**		**	01074300
						144	**	\$17.X'40' PAGE NOT DUMPED.	**	01075100
						145	**		**	01075900
						146	**	\$18.X'44' ERROR IN SYSTEM.	**	01076600
						147	**		**	01077400
						148	**	\$19.X'48' OLD FORMAT OF LOAD-INFORMATION	**	01078200
						149	**		**	01110000
						150	**	\$20.X'4C' INFORMATIONS ARE MISSING IN DUMPFIL.	**	01150000
						151	**		**	01189000
						152	**		**	01199000
						153	*	*****	*	01209000
						154	*		*	01219000
						155	*		*	01229000
						156	*	*****	*	01239000
						157	**		**	01249000
						158	**		**	01259000
						159	**	THE MODULE CALL : AIDSYS45.MOD AND A MODIFIED ROUTINE	**	01269000
						160	**	AS4ITNDP AND AS4RDPGE.	**	01274000
						161	**	ERROR IN SYSTEM.	**	01279000
						162	**		**	01289000
						163	**	\$15.X'38' NO PROGRAM IS LOADED.	**	01299000

FLAG	LOCN	OBJECT CODE	ADDR1	ADDR2	STMNT M	SOURCE STATEMENT	
					164	* *	* * 01309000
					165	* *	* * 01319000
					166	* *	* * 01329000
					167	* *	* * 01339000
					168	* *	* * 01349000
					169	* *	* * 01359000
					170	* *	* * 01369000
					171	* *	* * 01379000
					172	* *	* * 01389000
					173	* *	* * 01399000
					174	* *	* * 01409000
					175	* *	* * 01419000
					176	* *****	* * 01429000
					177	*	* * 01429600
					178	*	* * 01449000
					179	* *****	* * 01459000
					180	* *	* * 01469000
					181	* *	* * 01479000
					182	* *	* * 01489000
					183	* *	* * 01499000
					184	*	* * 01509000
					185	*****	* * 01519000
					186	R#00 EQU 0	01539000
					187	R#01 EQU 1	01549000
					188	R#02 EQU 2	01559000
					189	R#03 EQU 3	01569000
					190	R#04 EQU 4	01579000
					191	R#05 EQU 5	01589000
					192	R#06 EQU 6	01599000
					193	R#07 EQU 7	01609000
					194	R#08 EQU 8	01619000
					195	R#09 EQU 9	01629000
					196	R#10 EQU 10	01639000
					197	R#11 EQU 11	01649000
					198	R#12 EQU 12	01659000
					199	R#13 EQU 13	01669000
					200	R#14 EQU 14	01679000
					201	R#15 EQU 15	01689000

THE MODULE CALL : AIDSYS45.MOD AND A MODIFIED ROUTINE AS4ITNDP AND AS4RDPGE.

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT	
	202							*****	01709000
	203						*		01719000
	204						*		01729000
	205						*	MOST OF THE REGISTERS HAVE ALWAYS THE SAME MEANING	01739000
	206						*	WHICH ARE DESCRIBE HERE :	01749000
	207						*		01759000
	208						*	REGISTER 01 : COVER PARAMETER LIST TO CALL	01769000
	209						*		01779000
	210						*	REGISTER 02 : WILL SERVE TO PASS THE VALUE OF CERTAIN PARAM	01789000
	211						*		01799000
	212						*	REGISTER 03 : WORK REGISTER TO TEST THE USER INPUT	01809000
	213						*		01819000
	214						*	REGISTER 04 : BASE (DYNDATA)	01829000
	215						*		01839000
	216						*	REGISTER 05 : BASE (STATDATA)	01849000
	217						*		01859000
	218						*	REGISTER 06 : UNUSED	01869000
	219						*		01879000
	220						*	REGISTER 07 : WORK REGISTER TO TEST THE USER INPUT	01889000
	221						*		01899000
	222						*	REGISTER 08 : BASE (PARAMLIST OF USER)	01909000
	223						*		01919000
	224						*	REGISTER 09 : UNUSED	01929000
	225						*		01939000
	226						*	REGISTER 10 : BASE (USER BUFFER)	01949000
	227						*		01959000
	228						*	REGISTER 11 : UNUSED	01969000
	229						*		01979000
	230						*	REGISTER 12 : BASE REGISTER OF THE PROGRAM	01989000
	231						*		01999000
	232						*	REGISTER 13 : ADDRESS OF SAVE AREA	02009000
	233						*		02019000
	234						*	REGISTER 14 : RETURN ADDRESS	02029000
	235						*		02039000
	236						*	REGISTER 15 : FORWARD BRANCH REGISTER	02049000
	237						*		02059000
	238						*	*****	02069000
	239						*	PRINT GEN,XREF	02079000

The DSECT MALIB, defines the modifications done by adding the new functions to AIDSYS . This modifications refer to the DSECT SERVE . This DSECT SERVE is used by all functions . If someone is interested in seeing what is a flexible interface he (she) can look for more details in this DSECT . What should be interesting is how to do it in a high level language .

In all cases, I will say : "Good luck !"

This kind of interfaces is, perhaps, one reason of the problem of dealing with the understanding of the interfaces . For me, it took a lot of time to understand it and to be able to modify it .

This also will show what can be difficult in dealing with a so complex and so used interfaces . Indeed, how can be sure that the modification you are doing has no influence on the rest of the interfaces ?

FLAG LUCTN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

```

240 MALIB DSDPA
241 1 *****
242 1 *
243 1 * THE &ID DOES NEVER HAVE *
244 1 * *
245 1 * MORE THEN 3 CHARACTERS *
246 1 * *
247 1 *****
    
```

02099000

000000

```

248 1 MALIB DSECT
249 1 *****
250 1 *
251 1 * THE FIRST PART OF THE MACRO DESCRIBES THE COMMON *
252 1 * PARAMETERS USED BY ALL FOUR SUBFUNCTIONS: *
253 1 * -CMECT:FUNCTION USED TO GET THE CSECT MAP INFORMATION *
254 1 * *
255 1 * -GAFCT:FUNCTION USED TO GET INFORMATION ABOUT THE CSECT *
256 1 * CONTAINING THE GIVEN ADDRESS. *
257 1 * *
258 1 * -PEFCT:FUNCTION USED TO GET THE POINTER TO THE PROGRAM UNIT *
259 1 * OR ALSO CALLED PROGRAMM EINHEIT GIVEN *
260 1 * *
261 1 * FOLLOWING THE COMMON INPUT PARAMETERS, THERE ARE THE FUNCTION *
262 1 * SPECIFIC RETURN PARAMETERS. *
263 1 * *
264 1 *****
    
```

000000 00

000000
000004
000008
000008

```

265 1 DPAFCT DC X'00' NUMBER OF FUNCTION TO BE CALLED
266 1 DPACMSFT EQU X'00' NUMBER OF CSECT MAP FUNCTION
267 1 DPAGASFT EQU X'04' NUMBER OF GET ADDRESS FUNCTION
268 1 DPAFUSFT EQU X'08' NUMBER OF PU FUNCTION
269 1 DPAMAXNF EQU X'08' MAXIMAL NUMBER ADMITTED FOR A FUNCTION
270 1 DPARTH DC X'00' RETURN CODE,SEE EQU'S GIVEN AT THE
271 1 * SPECIFIC FUNCTION
272 1 DPALINK DC F'0' DUMPFIL-LinkNUMBER.HAVE TO BE GIVEN IF
273 1 * NO A(READROUTINE) 'READ' WAS GIVEN.
274 1 DPAREAD DC A(0) A(DUMPFIL'S READ ROUTINE OF CALLER)
275 1 * IF GIVEN,THEN 'DPACCES' DOES NOT NEED
276 1 * HAVE IT'S OWN DUMPFIL-ACCESS ROUTINE
277 1 DPAATCB DC A(0) A(TCB) TO BE PROVIDED BY THE CALLER
278 1 *
279 1 DPASYSBS DC A(0) A(SYSBASE) TO BE PROVIDED BY THE CALLER
280 1 *
281 1 DPAAXVT DC A(0) A(XVT) TO BE PROVIDED BY THE CALLER
282 1 *
283 1 DPAIND DC F'0' INDICATOR OF THE DUMPFIL TYPE.
284 1 *
285 1 DPAITH DC F'0' THE ITN NUMBER (JUSTIFIED RIGHT) OF THE
286 1 * TASK TO BE ACCESSED.
    
```

000001 00

000004 00000000

000008 00000000

00000C 00000000

000010 00000000

000014 00000000

000018 00000000

00001C 00000000

FLAG LCTN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

```

000020 00000000
000024 00000000
000028 00
000004
000008
00000C
000020
00002C
000029
327 1 *
328 1 *
329 1 *
290 1 DPABUFAD DC A(0)
291 1 DPABUFLE DC F'0'
292 1 DPAFLAG DC X'00'
293 1 *
294 1 DPAUSER EQU X'04'
295 1 DPASYST EQU X'08'
296 1 *
297 1 *
298 1 *
299 1 *
300 1 *
301 1 *
302 1 *
303 1 DPAALL EQU DPAUSER+DPASYST
304 1 DPASUCC EQU X'20'
305 1 DPAFLAGM EQU DPASUCC+DPAALL
306 1 DPADIV2 EQU *

```

```

THIS IS NEEDED ONLY FOR SLED-FILES, IF
A TASK SPECIFIC INFORMATION SHOULD BE
ACCESSED.
A(CALLER BUFFER);WORD-ALIGNED
LENGTH OF CALLER BUFFER
FLAG BYTE
PART TO BE ACCESSED
ONLY NOT PRIVILEGED TO BE ACCESSED
ONLY PRIVILEGED TO BE ACCESSED
IF NEITHER IS SET THEN ACCESS OF
PRIVILEGED PU IS ASSUMED, IF BOTH
ARE SPECIFIED, THE FIRST PU THAT
WILL BE FOUND WITH THE GIVEN NAME
WIL BE TAKEN, REGARDLESS OF PRIVILEGE
STATE. SEARCH WIL STARTED WITH NON-
PRIVILEGED PU'S.
IF BOTH ARE DESIRED
SUCCESSIVE CALL
MAXIMAL VALUE OF THE FLAG

```

```

307 1 *****
308 1 *
309 1 * ADDITIONAL INPUT PARAMETERS NEEDED TO CALL THE CSECT MAP *
310 1 * FUNCTION. *
311 1 * *
312 1 *****

```

000029 4040404040404040

```

313 1 DPACMHAM DC CL41' '
314 1 *
315 1 * NAME OF PROGRAM UNIT. IF NO NAME IS
316 1 * GIVEN, THE DEFAULT IS
317 1 * USER PROGRAM IF &ID.FLAG = &ID.USER
SYSTEM NUCLEUS IF &ID.FLAG = &ID.SYST

```

```

318 1 *****
319 1 *
320 1 * OUTPUT PARAMETERS FOR THE CMFCT SUBFUNCTION. *
321 1 * *
322 1 *****

```

```

000001
000000
000004
323 1 DPACMRTN EQU DPARTN
324 1 DPACMOK EQU X'00'
325 1 DPACMFE EQU X'04'
RETURN CODE NAME FOR CMFCT
O.K. NO PROBLEM.
FORMAL ERROR

```


INTERFACE BETWEEN USER AND AIDSYS

15:36:23 84-12-13 PAGE 0010

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE	STATEMENT	
	000020					378	1	DPACMRCL	EQU	*-DPACMCSN LENGTH OF ONE RECORD

000058

379 1 MALIB DSECT

FLAG LCTN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

```

380 1 *****
381 1 *
382 1 * INPUT PARAMETERS NEEDED BY THE GET ADDRESS FUNCTION. *
383 1 *
384 1 *****
    
```

00002C 00000000 000029

```

385 1 GRG DPADIV2 CONTINUE PARAMETER INPUT
386 1 DPAGAVA DC A(0) ADDRESS FOR WHICH THE INFORMATION
387 1 * SHOULD BE FETCHED (MANDATORY)
    
```

```

388 1 *****
389 1 *
390 1 * OUTPUT PARAMETERS FOR GAFCT *
391 1 *
392 1 *****
    
```

000001
000000
000004

```

393 1 DPAGARTN EQU DPARTN
394 1 DPAGARK EQU X'00'
395 1 DPAGAFE EQU X'04'
396 1 *
397 1 *
398 1 *
399 1 *
400 1 *
401 1 *
    
```

```

O.K. NO PROBLEM
FORMAL ERROR
-NO TCB ADDRESS GIVEN
-NO XVT ADDRESS GIVEN
-NO SYSBASE ADDRESS GIVEN
-NEITHER A(READ) NOR LINK# GIVEN
-INVALID SUBFUNCTION NUMBER
-INVALID ITN GIVEN
    
```

000008
00000C

```

402 1 DPAGAME EQU X'08'
403 1 DPAGANWA EQU X'0C'
404 1 *
    
```

```

MODULE ERROR
A(TCB) OR A(XVT) OR A(SYSBASE) NOT
WORD ALIGNED
    
```

000010
000014
000018
00001C
000020
000024
000028
00002C
000030
000034
000038
00003C
000040

```

405 1 DPAGAGF EQU X'10'
406 1 DPAGAS EQU X'14'
407 1 DPAGATQN EQU X'18'
408 1 DPAGAILN EQU X'1C'
409 1 DPAGAIND EQU X'20'
410 1 DPAGARNI EQU X'24'
411 1 DPAGANA EQU X'28'
412 1 DPAGIAF EQU X'2C'
413 1 DPAGAPRE EQU X'30'
414 1 DPAGAERS EQU X'34'
415 1 DPAGANPL EQU X'38'
416 1 DPAGANID EQU X'3C'
417 1 DPAGAPND EQU X'40'
    
```

```

FILE IS NOT OPEN
SPECIFIED TASK NOT FOUND
TASK QUALIFICATION NECESSARY
INVALID LINK NUMBER GIVEN
INVALID INDICATOR OF DUMPFIL TYPE
A(READ) GIVEN BUT NO ITN GIVEN
ADDRESS NOT ALLOCATED
INPUT ADDRESS IN CONFLICT WITH FLAG
ERROR IN PAGE READ ROUTINE
ERROR IN SYSTEM
NO PROGRAM LOADED
INFORMATION NOT IN DUMP
PAGE NOT DUMPED
    
```

000030 4040404040404040
000038 00000000
00003C 00000000
000040 00000000

```

418 1 DPAGASEC DC CL8'
419 1 DPAGASTA DC A(0)
420 1 DPAGAREL DC A(0)
421 1 *
422 1 DPAGACSL DC F'0'
423 1 DPAGALEN EQU *-DPAFCT
    
```

```

NAME OF THE CSECT
START ADDRESS OF THE CSECT
RELATIVE ADDRESS(GIVEN ADDRESS
MINUS START ADDRESS)
LENGTH OF CSECT
LENGTH OF THE PARAMETER LIST
    
```

000044

FLAG LCTN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

```

424 1 *****
425 1 *
426 1 *          ADDITIONAL INPUT PARAMETERS NEEDED FOR PU SUBFUNCTION.
427 1 *
428 1 *****
    
```

```

429 1 *          N O N E
    
```

```

430 1 *****
431 1 *
432 1 *          OUTPUT PARAMETERS OF THE PU SUBFUNCTION.
433 1 *
434 1 *****
    
```

000001
000000
000004

```

435 1 DPAPURTN EQU DPARTN
436 1 DPAPUGK EQU X'00'
437 1 DPAPUFE EQU X'04'
    
```

```

O.K. NO PROBLEM
FORMAL ERROR
-BUFFER ADDRESS NOT WORD-ALIGNED
-NO BUFFER ADDRESS GIVEN
-NO BUFFER LENGTH GIVEN
-ILLEGAL SUBFUNCTION NUMBER
-NO TCB ADDRESS GIVEN
-NO XVT ADDRESS GIVEN
-NO SYSBASE ADDRESS GIVEN
-NEITHER A(READ) NOR LINK# GIVEN
-INVALID ITN GIVEN
    
```

000008

```

447 1 DPAPUNWA EQU X'08'
448 1 *
    
```

```

A(TCB) OR A(XVT) OR A(SYSBASE) NOT
WORD ALIGNED
    
```

00000C
000010
000014
000018

```

449 1 DPAPUGF EQU X'0C'
450 1 DPAPUSTN EQU X'10'
451 1 DPAPUTQN EQU X'14'
452 1 DPAPUGC EQU X'18'
453 1 *
454 1 *
455 1 *
456 1 *
    
```

```

FILE IS NOT OPEN
SPECIFIED TASK NOT FOUND
TASK QUALIFICATION NECESSARY
O.K. BUT BUFFER FILLED WITH
INFORMATION AND INFORMATION IS NOT
COMPLETE, SUCCESSIVE CALL NECESSARY
WITH SAME PARAMETERS, BUT PUSUCC FLAG
SET.
    
```

00001C
000020
000024
000028

```

457 1 DPAPUILN EQU X'1C'
458 1 DPAPUIND EQU X'20'
459 1 DPAPURHI EQU X'24'
460 1 DPAPUIA EQU X'28'
461 1 *
    
```

```

INVALID LINK NUMBER GIVEN
INVALID INDICATOR OF DUMPFIL TYPE
A(READ) GIVEN BUT NO ITN GIVEN
BUFFER NOT WRITE ACCESSIBLE IN FULL
LENGTH.
    
```

00002C
000030

```

462 1 DPAPUNGFP EQU X'2C'
463 1 DPAPUTS EQU X'30'
464 1 *
    
```

```

NO PU FOUND, BUFFER IS EMPTY
BUFFER LENGTH TOO SHORT TO CONTAIN
ONLY ONE ANSWER.
    
```

000044 00000000

000048

```

465 1 DPAPUNRT DC F'0'
466 1 DPAPULGT EQU *-DPAFCT
    
```

```

NUMBER OF RETURNED RECORDS
    
```

FLAG LCTN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

```

467 1 *****
468 1 *
469 1 *      DESCRIPTION OF BUFFER LAYOUT
470 1 *
471 1 *****
    
```

```

000000
000000 00000000
000004 4040404040404040
00002D 404040
          000004
          000008
          00002C
    
```

```

472 1 DPAGUTPU DSECT
473 1 DPAPUCRI DC      A(0)          POINTER TO CRIA
474 1 DPAPUNAM DC     CL41' '      NAME OF PROGRAM UNIT
475 1 DPAPUST DC     CL3' '        PU STATE
476 1 DPAPUPRI EQU   DPAUSER      PU IS IN PRIVILEGED STATE
477 1 DPAPUNPR EQU   DPASYST      PU IS IN NON PRIVILIGED STATE
478 1 DPAPURCL EQU   *-DPAPUNAM
    
```

```

479 1 *****
480 1 *
481 1 *      THE FOLLOWING EQUATES ARE DONE IN ORDER TO BE ABLE
482 1 *      TO VALIDATE SOME INPUT PARAMETERS
483 1 *
484 1 *****
    
```

```

000040      485 1 DPABOTH EQU   X'40'   FLAG USED TO SEE IF BOTH PRIVILEGED AND
486 1 *                                     NON PRIVILEGED INFORMATION ARE NEEDED
0000FD      487 1 *
488 1 DPARESET EQU   X'FD'
489 1 *                                     B'11111101' IT IS USED FOR RESETTING THE
490 1 *                                     SYSTEM BIT SET IF BOTH WAS SPECIFIED
000003      491 1 *                                     NOW THE FIRST CALL DONE IS WITH NON PRIVILEGE
492 1 DPALNKL EQU   3               NUMBER OF BYTES WHICH MUST BE TESTED IN ORDER
493 1 *                                     TO VALIDATE THE CALLER GIVEN IND AND ALSO THE
494 1 *                                     NUMBER OF BYTES TO BE TRANSFERED FRGM A FW TO
495 1 *                                     THE ASERP CORRESPONDING ZONE>
000003      496 1 DPAITNL EQU   3               NUMBER OF BYTES WHICH MUST BE TESTED IN
497 1 *                                     ORDER TO VALIDATE THE CALLER GIVEN ITN
0000F3      498 1 DPAVALFN EQU   X'FF'-DPAGASFT-DPAUSFT
499 1 *                                     THIS FIELD IS USED AS A BINARY MASK OF
500 1 *                                     B'11110011' TO VALIDATE THE FUNCTION
501 1 *                                     NUMBER WHICH ONLY CAN BE
502 1 *                                     B'00000000' GR B'000000100' GR B'00001000'
503 1 *                                     GR B'00001100' ALL BITS SETS ARE NOT TESTED!
000003      504 1 DPAVALFW EQU   X'03'   THE CORRESPONDING BINARY MASK IS B'00000011'
505 1 *                                     WHICH IS THE MASK TO TEST IF THE ADDRESS
506 1 *                                     GIVEN BY THE CALLER IS FULLWORD ALIGNED
    
```

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMT	M	SOURCE STATEMENT
						507	1 *	THAT MEANS DIVIDABLE BY 4
0000DF						508	1	DPAFLAG EQU X'FF'-DPASUC
						509	1 *	B'11101111' IT IS USED IS TO TEST IF A
						510	1 *	IT IS THE FIRST CALL OR A SUCCESSIVE ONE
000003						511	1	DPAINDL EQU 3 THIS IS THE NUMBER OF BYTES TO BE BYPASSED
						512	1 *	BECAUSE GIVEN IS FW AND RECEIVED IS A BYTE
0000DF						513	1	DPAUSSET EQU DPAFLAG B'11011111' IT IS USED TO TEST IF THE FLAG
						514	1 *	IS SET IN THE MEANING OF USER OR SYSTEM
						515	1 *	DUMPFIL ACCESS IS WANTED.WHY NOT USE THE
						516	1 *	FOLLOWING MASK B'11111111'? BECAUSE THE
						517	1 *	FLAG IS ALSO USED FOR SETTING IF FIRST CALL
0000D3						518	1	DPAVALFL EQU X'FF'-DPASUC-DPASYS-DPAUSER
						519	1 *	B'11010011' IN ORDER TO VALIDATE THE VALUE
						520	1 *	OF THE FLAG BECAUSE THE ONLY PERMITTED
						521	1 *	BITS TO BE SET ARE B'00*0**00'.IF OTHERS
						522	1 *	ARE SET ...THERE IS AN ERROR
0000FF						523	1	DPARTNT EQU X'FF' B'11111111' THESE FIELD IS USED TO SEE
						524	1 *	WETHER THERE IS A NULL RETURN CODE FROM
						525	1 *	AIDSYS05 OR AIDSYS02 OR AIDSYS04 IF
						526	1 *	IT IS NOT NULL,THEN GOTG TRANSLATION OF
						527	1 *	THE RETURNED CODE TO THE CALLER EXPECTED
						528	1 *	ONE
000007						529	1	DPALINK# EQU X'07' THIS FIELD IS USED FOR HANDLING THE MAXIMUM OF
						530	1 *	GULTIG LINK NUMBER.IT IS USED TO SEE WETHER
						531	1 *	THE CALLER HAS GIVEN A VALID LINK NUMBER TO
						532	1 *	US FOR HIS DUMPFIL.THESE NUMBER CAN VARY
						533	1 *	THAT IS THE RAISON WHY IT IS DONE BY THIS WAY

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT	
						534		SERVE ASERP DSECTAID	02119000
000000						535	1	SERVE DSECT	
						536	1	IDLKG VER=022	
						537	2	*,VERSION 022	
000000						538	2	CNSP 0,4	00001300
000000						539	2	DS GF	00002800
						540	1	*****	00003500
						541	1	*	
						542	1	* ASERP IS THE ONLY INTERFACE BETWEEN AID AND AIDSYS. *	
						543	1	* THE PARAM-LIST CONSISTS OF TWO PARTS: A COMMON *	
						544	1	* HEADER FOR ALL FUNCTIONS AND THE PARAMETERS FOR *	
						545	1	* EACH FUNCTION. TO CALL AIDSYS FOR A SERVICE-REQUEST, *	
						546	1	* THE CALLER JUST HAS TO SET SOME HEADER-PARAMS AND *	
						547	1	* ALL NECESSARY PARAMS FOR THAT SPECIFIC FUNCTION. *	
						548	1	* ALL PARAMS OF ANY OTHER FUNCTION MUST NOT BE SET, *	
						549	1	* BECAUSE DOING SO WOULD OVERWRITE THE OTHER PARAMS. *	
						550	1	* A SPECIAL NOTE TO THE HEADER: *	
						551	1	* ASACAIDS IS ALWAYS SET BY AIDSYS AND MUST NOT BE *	
						552	1	* ALTERED, BECAUSE THIS FIELD HOLDS THE ADDRESS OF *	
						553	1	* THE ENTRY IN AIDSYS FOR ALL SERVICE-REQUESTS. *	
						554	1	* ASAPC IS SET BY AIDSYS, TOO, FOR EVERY RETURN TO *	
						555	1	* AID. ASAFCT AND ASASBFACT MUST ALWAYS BE SET BY AID. *	
						556	1	* *	
						557	1	* PARAMS: I = PREFIX; A MAXIMUM OF 3 CHARACTERS *	
						558	1	* IS ALLOWED; DEFAULT IS 'ASA' *	
						559	1	* *	
						560	1	* FCT = SPECIFIES THE FUNCTION(S) TO BE *	
						561	1	* SELECTED (WITH COMPLETE HEADER). *	
						562	1	* DEFAULT IS 'ALL' *	
						563	1	* = ALL COMPLETE ASERP WITH ALL *	
						564	1	* FUNCTIONS; THIS IS *	
						565	1	* COMPATIBLE WITH V6/V7.0 *	
						566	1	* = SETTP SET TESTPOINTS *	
						567	1	* = DELTP DELETE TESTPOINTS *	
						568	1	* = OUTPUT DATA OUTPUT *	
						569	1	* = GETADDR GET ADDRESS *	
						570	1	* = OPENDUMP OPEN DUMPED TASK *	
						571	1	* = REQM REQUEST MEMORY *	
						572	1	* = RELM RELEASE MEMORY *	
						573	1	* = CALLSYST CALL SYSTEM *	
						574	1	* = CLOSEDMP CLOSE DUMPFIL *	
						575	1	* = CLOSEGUT CLOSE OUTPUT-FILE & *	
						576	1	* AID-MEMORY-POOLS *	
						577	1	* = SETSW SET SWITCHES *	
						578	1	* = STS LOAD/UNLOAD STS-MODULE *	
						579	1	* = INFORM INFORM *	
						580	1	* = MOVE DATA MANIPULATION *	
						581	1	* = HELP ACCESS HELP-FILE *	
						582	1	* = TMODE GET TERMINAL-INFORMATION *	
						583	1	* = LADINF GET LOADER-INFORMATION *	
						584	1	* = ALPOINT CHANGE ACTIONLISTPOINTER *	
						585	1	* = SYSEDT SYSTEM - EDIT *	
						586	1	* = DATE GET DATE *	
						587	1	* = HNI HARDWARE-INFORMATION *	
						588	1	* = HIA HARDWARE-INFORMATION F.A. *	

FLAG LOC TN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

```

589 1 * = SAVE TABELLENSICHERUNG *
590 1 * * *
591 1 * INF = YES DESCRIPTION OF ASERP WILL *
592 1 * BE PRINTED OUT (DEFAULT). *
593 1 * = NO NO PRINT-OUT OF DESCRIPTION *
594 1 * * *
595 1 * *****

```

000000

```

596 1 ASASERVE DS OF USER INTERFACE FOR AID TO AIDSYS
597 1 *****
598 1 ** **
599 1 ** GENERAL EQUATES FOR AIDSYS-INTERNAL USE ONLY **
600 1 ** **
601 1 *****

```

```

000010
000004
000000
000000
000004
000000
000022

```

```

602 1 ASADUMP EQU X'10' ZUGRIFF AUF DUMPDATEI
603 1 ASAFE EQU X'04' ANZEIGE FUER FORMALER FEHLER
604 1 ASAITN EQU X'00' NICHT VERWENDET
605 1 ASATSN EQU X'00' NICHT VERWENDET
606 1 ASAUQUAL EQU X'04' EINGABE IST UEBERQUALIFIZIERT
607 1 ASANRG EQU X'80' WRITE INTO R/G-PAGE
608 1 ASAVERS# EQU X'022' ACTUAL MACRO-VERSION#

```

```

609 1 *****
610 1 ** **
611 1 ** STANDARD - H E A D E R FOR ALL FUNCTIONS **
612 1 ** **
613 1 *****

```

```

000000 00000000
000004 00000000
000008 00000000
00000C 00000000

```

000001

```

614 1 ASACAIDS DC A(0) A(LINK-CODING AID --> AIDSYS)
615 1 ASACALL1 DC A(0) FREE FOR CALLER
616 1 ASACALL2 DC A(0) FREE FOR CALLER
617 1 ASAPC DC A(0) PC OF INTERRUPTED PROGRAM
618 1 ASANGPCB EQU X'000001' FALLS 'ASAPC'=X'00000001', DANN IST
619 1 * KEIN PROGRAMM GELADEN

```

```

000010 22
000011 00

```

```

620 1 ASAVERS DC ALL(ASAVERS#) MACRO-VERSION#
621 1 ASAFCT DC X'00' FUNKTION; DIE EINZELNEN FUNKTIONS-
622 1 * CODES STEHEN JEWEILS AM ANFANG DER
623 1 * FUNKTIONSBESCHREIBUNG

```

000012 00

```

624 1 ASASBFCT DC X'00' SUBFUNKTION; SIEHE BEI DEN EINZELNEN
625 1 * HAUPTFUNKTIONEN BZGL. DES CODES

```

000013 00

```

000014
000014

```

```

626 1 ASARETRN DC X'00' RUECKKEHR-CODE
627 1 ASAHEAD EQU *-ASASERVE LENGTH OF HEADER
628 1 ASALABEL EQU *

```

```

629 1 *****
630 1 ** **
631 1 ** SET TESTPOINT **
632 1 ** ----- **
633 1 ** DIESER SERVICE DIENST ZUM SETZEN VON AID-TESTPUNKTEN (SVC 129), **
634 1 ** ZUM ANMELDEN EINES MASCHINEN-TRACE UND VON ASYNCHRONEN **
635 1 ** EREIGNISSEN. **
636 1 ** **
637 1 *****

```

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE	STATEMENT
				000014		638	1	ORG	ASALABEL
						639	1	*****	*****
						640	1	**	HEADER - PARAMETER **
						641	1	*****	*****
		000000				642	1	ASAST EQU X'00'	FUNCTION=SET TESTPGINT
		000000				643	1	ASASTL EQU X'00'	SUBFUNCTION=EXPLICIT LOCATION
		000004				644	1	ASASTE EQU X'04'	SUBFUNCTION=EVENTS (ANY,SVC,...)
						645	1	*	& MACHINE-TRACE
		000008				646	1	ASASTFT EQU X'08'	SUBFUNCTION=STATEMENT-TRACE
		000008				647	1	ASASTMAX EQU X'08'	MAX. # OF SUBFUNCTION
						648	1	*****	*****
						649	1	**	AUFRUF - PARAMETER **
						650	1	*****	*****
000014						651	1	ASASTLGC DS OF	L: LGC GF EXPL TESTPGINT
000014						652	1	ASASTEVT DS GC	E: EVENT CLASS
	000040					653	1	ASASTCLO EQU X'40'	MACHINE-TRACE
	000020					654	1	ASASTCL1 EQU X'20'	AIDSYS-CLASS1: SVC-INTERRUPTS
						655	1	*	ERROR-CODE X'50'
	000010					656	1	ASASTCL2 EQU X'10'	AIDSYS-CLASS2: TERMINATION
						657	1	*	- EVENT-CODES X'00' - X'02'
	000008					658	1	ASASTCL3 EQU X'08'	AIDSYS-CLASS3: ERROR-FLAG
						659	1	*	- EVENT-CODES: X'48'
						660	1	*	X'54'-X'78'
	000004					661	1	ASASTCL4 EQU X'04'	AIDSYS-CLASS4: ILLSTXIT
						662	1	*	- EVENT-CODES: X'03' - X'05'
						663	1	*	X'0C'
	000002					664	1	ASASTCL5 EQU X'02'	AIDSYS-CLASS5: STXIT
						665	1	*	- EVENT-CODES: X'06'
						666	1	*	X'44'
						667	1	*	X'4C'
	000001					668	1	ASASTCL6 EQU X'01'	AIDSYS-CLASS6: LPOV/LINK
						669	1	*	- EVENT-CODES: X'07'
	000080					670	1	ASASTCL7 EQU X'80'	AIDSYS-KLASSE 7: AUDIT
000014	00000000					671	1	ASASTALM DC A(0)	FT: A(LMIR); KANN AUS AIDSYS-DESCRIPTOR
						672	1	*	ENTNOMMEN WERDEN
000018	00000000					673	1	ASASTTRC DC A(0)	FT: A(TRACE-PUNKT-PUFFER)
00001C	00000000					674	1	ASASTTLN DC A(0)	FT: ANZAHL DER SAETZE IM TRACEPUNKT-
						675	1	*	PUFFER INCLUSIVE ALLER FOLGEAUFRUFE.
						676	1	*	FALLS EIN FORTSETZUNGSAUFRUF VORGE-
						677	1	*	SEHEN IST, MUSS DIE ANZEIGE 'STLNK'
						678	1	*	GESETZT WERDEN. BEI FOLGEAUFRUFEN
						679	1	*	WIRD 'STLNK' NICHT MEHR BETRACHTET.
000020	0000					680	1	ASASTITH DC H'0'	ALL: (OPTIONAL) ITH. X'0000' BEDEUTET
						681	1	*	ZUGRIFF AUF DEN DEFAULT-TASK.
000024	00000000					682	1	ASASTTSN DC A(0)	ALL: (OPTIONAL) TSN. A(0) BEDEUTET
						683	1	*	ZUGRIFF AUF DEN DEFAULT-TASK.
000028	00					684	1	ASASTFB DC X'00'	ALL: FLAGBYTE
	000000					685	1	ASASTFSN EQU ASATSN	NICHT VERWENDET
	000000					686	1	ASASTFTN EQU ASAITN	NICHT VERWENDET
	000040					687	1	ASASTALL EQU X'40'	ALL TASKS MUST REACT
	000020					688	1	ASASTEXC EQU X'20'	EXCLUDE TASK SPECIFIED BY STTSK

FLAG	LCTN	OBJECT CODE	ADDR1	ADDR2	STMT	M	SOURCE STATEMENT
			000010		689	1	ASASTLNK EQU X'10'
			000008		690	1	ASASTGWN EQU X'08'
			000002		691	1	ASASTPH EQU X'02'
			000001		692	1	ASASTP2 EQU X'01'
					693	1	*
					694	1	*
					695	1	*
000029	00				696	1	ASASTTSK DC XL1'00'
00002A					697	1	ASASTLL DS 0CL3
00002A	00				698	1	ASASTTYP DC X'00'
					699	1	*
					700	1	*
00002B	00				701	1	ASASTFB2 DC X'00'
			000080		702	1	ASASTWRG EQU ASAWRG
00002C	00000000				703	1	ASASTUAD DC A(0)
					704	1	*****
					705	1	** RUECKKEHR - PARAMETER **
					706	1	*****
000013					707	1	ASASTRTN EQU ASARETRN ALL: RETURN CODE
000000					708	1	ASASTGK EQU X'0' ALL: ALL GK
000004					709	1	ASASTFE EQU X'4' ALL: FORMALER FEHLER.
					710	1	*
					711	1	*
					712	1	*
					713	1	*
					714	1	*
					715	1	*
					716	1	*
000008					717	1	ASASTTTS EQU X'8' ALL: TEST PRIVILEGIERUNG IST ZU KLEIN.
					718	1	*
					719	1	*
00000C					720	1	ASASTTAS EQU X'C' ALL: TEST PRIVILEGIERUNG IST ZU KLEIN.
					721	1	*
					722	1	*
000010					723	1	ASASTADE EQU X'10' L: TESTPUNKT-ADRESSE NICHT ALLGKIERT
000014					724	1	ASASTENA EQU X'14' E: EREIGNIS NICHT ERLAUBT, DA 'IDA'
					725	1	*
000018					726	1	ASASTNFI EQU X'18' ALL: ITN NICHT AKTIV
00001C					727	1	ASASTGDD EQU X'1C' L: TESTPUNKT-ADRESSE NICHT AUF HALB-
					728	1	*
000020					729	1	ASASTTPE EQU X'20' L: TESTPUNKT EXISTS ALREADY, A(1ST AL)
					730	1	*
000024					731	1	ASASTMNF EQU X'24' L/FT: MODUL NICHT GEFUNDEN
000028					732	1	ASASTNMA EQU X'28' ALL: KEIN SPEICHER VERFUEGBAR.
00002C					733	1	ASASTNFT EQU X'2C' ALL: TASK MIT ANGEGEBENER TSN IST
					734	1	*
000030					735	1	ASASTNT EQU X'30' L: EIN ANDERER TASK BLOCKIERT ZUGRIFF
					736	1	*
000034					737	1	ASASTNBT EQU X'34' ALL: ANFORDERUNG NICHT ERFUELLT.
					738	1	*
000038					739	1	ASASTREJ EQU X'38' ALL: FREMDTASK-ZUGRIFF NOCH NICHT
					740	1	*
00003C					741	1	ASASTIDT EQU X'3C' L: AID-TESTPUNKT KANN NICHT GESETZT

FLAG LOC TN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

				742	1	*			WERDEN, DA SCHON 'IDA'-TESTPUNKT
				743	1	*			AN ANGEGEBENER ADRESSE
	000040			744	1	ASAST1CL	EQU	X'40'	FT: ERSTER AUFRUF DER SUBFKT. 'TFT'
				745	1	*			UND 'STTLN' IST NICHT GESETZT.
	000044			746	1	ASASTTMA	EQU	X'44'	L/FT: TPLIST NOT ALLGCATED
	000048			747	1	ASASTVRR	EQU	X'48'	ALL: VALIDATION-ERROR
	00004C			748	1	ASASTPHY	EQU	X'4C'	ALL: ZUGRIFF MIT REALER ADRESSE NICHT
				749	1	*			MOEGLICH
	000050			750	1	ASASTIRG	EQU	X'50'	L/FT: ILLEGAL WRITE-ACCESS OF READ-ONLY PAGE
000030				751	1	ASASTXXX	DS	0X	
000030	00			752	1	ASASTFBR	DC	XL1'00'	ALL: FLAG BYTE
	000080			753	1	ASASTRCT	EQU	X'80'	ALL: REACTING TASK(S)-INDICATOR IS ...
				754	1	*			IN CONFLICT WITH INDICATOR OF TP
	000040			755	1	ASASTGTK	EQU	X'40'	L: TP EXISTS ALREADY AND WAS SET BY
				756	1	*			ANOTHER TASK.
	000020			757	1	ASASTMPL	EQU	X'20'	E: ALL EVENTS REJECTED, BECAUSE
				758	1	*			THERE IS NO PROGRAM LOADED
	000010			759	1	ASASTIDA	EQU	X'10'	E: MACHINE-TRACE REJECTED, BECAUSE
				760	1	*			THERE IS A PARALLEL IDA-TRACE
	000004			761	1	ASASTQUL	EQU	ASAUQUAL	ALL: UEBERQUALIFIZIERTE EINGABE
000034	00000000			762	1	ASASTMRD	DC	7A(0)	ALL: A(ACTIONLISTS). IF EVENTS ARE TO BE
				763	1	*			SET, THEN 1ST WORD REFERS TO CLASS1,
				764	1	*			THE N-TH WORD TO CLASS N. OTHERWISE
				765	1	*			ONLY 1ST WORD WILL BE SET.
000050	00000000			766	1	ASASTCHT	DC	A(0)	FT: FOR TRACE ONLY; #(TRAPCODES), THAT
				767	1	*			COULD NOT GET SET
	000024			768	1	ASASTLEN	EQU	*-ASASTXXX	LENGTH(RET.PARAMS)

```

769 1 *****
770 1 **
771 1 **          DELETE TESTPOINT          **
772 1 **          -----          **
773 1 ** MIT DIESEM SERVICE KANN EIN AID-TESTPUNKT (SVC 129), EIN ODER **
774 1 ** MEHRERE STATEMENTTRACE-KLASSEN, DER MASCHINENTRACE ODER EIN **
775 1 ** ODER MEHRERE EREIGNISKLASSEN GELGESCHT WERDEN. **
776 1 ** **
777 1 *****
000014 778 1          GRG  ASALABEL
779 1 *****
780 1 **          HEADER - PARAMETER          **
781 1 *****
000004 782 1 ASADT   EQU  X'04'   ALL: FUNCTION = DELETE TESTPOINT
000000 783 1 ASADTL  EQU  X'00'   L:  SUBFUNCTION = EXPLICIT LOCATION
000004 784 1 ASADTE  EQU  X'04'   E:  SUBFUNCTION = EVENTS (ANY,SVC,...)
000008 785 1 ASADTFT EQU  X'08'   FT: SUBFUNCTION = FULLTRACE
000008 786 1 ASADTMAX EQU X'08'   MAX. # OF SUBFUNCTIONS

787 1 *****
788 1 **          AUFRUF - PARAMETER          **

```

FLAG	LOC TN	OBJECT CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT
					789	1	*****
000014	00000000				790	1	ASADTLGC DC A(0) L: LOC OF EXPL TESTPOINT
000018	00				791	1	ASADTEVT DC X'00' E: EVENT-CLASS
000019	00				792	1	ASADTCTM DC X'00' FT: TRACE-TYPE
00001A	0000				793	1	ASADTITH DC H'0' ALL: (OPTIONAL) ITM. X'0000' BEDEUTET
					794	1	* ZUGRIFF AUF DEN DEFAULT-TASK.
00001C	00000000				795	1	ASADTTSN DC A(0) ALL: (OPTIONAL) TSN. A(0) BEDEUTET ZUGRIFF
					796	1	* AUF DEN DEFAULT-TASK
000020	00				797	1	ASADTFB DC X'00' ALL: FLAG BYTE
	000000				798	1	ASADTFTH EQU ASAITN NICHT VERMENDET
	000000				799	1	ASADTFSH EQU ASATSN NICHT VERMENDET
	000040				800	1	ASADTALL EQU X'40' NOT USED
	000020				801	1	ASADTPH EQU X'20' ALL: 'DTLGC' IST REALE ADRESSE
	000001				802	1	ASADTP2 EQU X'01' E: SET: EVENT IS FOR STATE=P2
					803	1	* RESET: EVENT IS FOR STATE=P1
000024	00000000				804	1	ASADTALM DC A(0) FT: A(LMIR) FOR TRACE-FCI
					805	1	*****
					806	1	** RUECKKEHR - PARAMETER **
					807	1	*****
	000013				808	1	ASADTRTH EQU ASARETRN ALL: RETURN CODE
	000000				809	1	ASADTCK EQU X'00' ALL: ALL O.K., USER WORD IN DTWRD
	000004				810	1	ASADTFE EQU X'04' ALL: FORMALER FEHLER.
					811	1	* - AUFRUFER IST NICHT AID.
					812	1	* - UNBEKANNTER SUBFKT-CODE
					813	1	* - ITM GROESSER ALS 255
					814	1	* - FREMDTASKZUGRIFF
					815	1	* - SUBFKT = 'E' UND KEINE EREIGNIS
					816	1	* KLASSE ANGEGEBEN
	000008				817	1	ASADTTPF EQU X'08' L/FT: LGC/TRACE DOES NOT EXIST
	00000C				818	1	ASADTNA EQU X'0C' L: TASK IS NOT ALLOWED TO DELETE TP
	000010				819	1	ASADTSHR EQU X'10' L: TP DELETED, BUT SOURCE COULD NOT
					820	1	* BE RESTORED (NO TRAP SET)
	000014				821	1	ASADTNFI EQU X'14' ALL: ITM/TSN NOT FOUND
	000018				822	1	ASADTTTS EQU X'18' ALL: TESTPRIVILEGE TOO SMALL
	00001C				823	1	ASADTTAS EQU X'1C' ALL: TESTPRIVILEGE TOO SMALL IN ANY CASE
	000020				824	1	ASADTFHY EQU X'20' L: REALE ADRESSE IST NICHT ERLAUBT
000028					825	1	ASADTXXX DS 0X
000028	00				826	1	ASADTNEX DC X'00' E: FLAGBYTE WITH SAME LAYOUT AS DTEVT.
					827	1	* THE FLAGGED EVENTS WERE NOT SET.
000029	00				828	1	ASADTRFB DC X'00' E: FLAGBYTE WITH SAME LAYOUT AS DTEVT.
					829	1	* THE FLAGGED EVENTS COULD NOT BE
					830	1	* REMOVED.
					831	1	* THE FLAGGED EVENTS WERE NOT SET.
00002C	00000000				832	1	ASADTAAL DC 7A(0) ALL: ADDRESS OF 1ST ACTION-LIST
					833	1	* IF EVENTS GOT REMOVED, THERE WILL BE RETURNED
					834	1	* THE ADDRESS(ACTIONLIST) FOR EACH EVENT.
000048	00				835	1	ASADTFB2 DC X'00' ALL: ANZEIGENBYTE
	000004				836	1	ASADTQUL EQU ASAUQUAL ALL: UEBERQUALIFIZIERTE EINGABE
	000021				837	1	ASADTLEN EQU *-ASADTXXX LENGTH(RET.PARAMS)

FLAG	LOC TN	OBJECT	CODE	ADDR1	ADDR2	STMT	M	SOURCE	STATEMENT
						838	1	*****	*****
						839	1	**	**
						840	1	**	DATA OUTPUT
						841	1	**	-----
						842	1	**	MIT DIESEM SERVICE KANN EIN LOGISCHER SATZ AUF EINES ODER
						843	1	**	MEHRERE AUSGABEMEDIEN GLEICHZEITIG AUSGEBEN WERDEN. AUSSERDEM
						844	1	**	KOENNEN MESSAGES AUS DEM MSG-FILE AUF 'SYSOUT' ODER DIE HAUPT-
						845	1	**	KONSOLE AUSGEBEN WERDEN.
						846	1	**	FALLS CONSOL-I/O EINGESTELLT IST (SERVICE 'SET SWITCH'), ERFOLGT
						847	1	**	JEDE FUER SYSOUT BESTIMMTE AUSGABE AUF DIE HAUPT-KONSOLE.
						848	1	**	**
						849	1	*****	*****
				000014		850	1	GRG	ASALABEL
						851	1	*****	*****
						852	1	**	HEADER - PARAMETER
						853	1	*****	*****
				000008		854	1	ASADG EQU X'08'	ALL: FUNCTION = DATA OUTPUT
				000000		855	1	ASADGMX EQU X'00'	MAX. # OF SUBFUNCTION
						856	1	*****	*****
						857	1	**	AUFRUF - PARAMETER
						858	1	*****	*****
000014	00					859	1	ASADGMED DC X'00'	MEDIUM, INTO WHICH THE OUTPUT IS DONE
				000001		860	1	ASADGHC EQU X'01'	HC: HARDCOPY
				000002		861	1	ASADGFIL EQU X'02'	FILE: FILE, SEE DGFNR
				000004		862	1	ASADGMEM EQU X'04'	MEM: MEMGRY-PGOL, SEE DGMN
				000008		863	1	ASADGTER EQU X'08'	TER: SYSOUT
				000010		864	1	ASADGPRT EQU X'10'	PRT: SYSLST
				000020		865	1	ASADGMSG EQU X'20'	MSG: MSG, NUMBER SEE DGMGN, THIS BIT IS EXCLUSIVE
						866	1	*	
000015	00					867	1	ASADGFNR DC X'00'	FILE: NUMBER OF FILES: LEFTMOST BIT IS FILE #0. A MAXIMUM OF 8 FILES IS POSSIBLE. AT LEAST 1 FILE MUST BE GIVEN
						868	1	*	
						869	1	*	
						870	1	*	
000016	00					871	1	ASADGMN DC X'00'	MEM: # OF MEM'S AT LEAST 1 MUST BE GIVEN.
						872	1	*	
						873	1	*	LEFTMOST BIT IS MEM #0
000018	00000000					874	1	ASADGADR DC A(0)	ALL BUT MSG: ADDR. OF V-FORMATED OUTPUT ADRESSE MUSS HALBWGRTAUSGERICHTET SEIN
						875	1	*	
00001C	00					876	1	ASADGIND DC X'00'	ALL BUT MSG: INDICATOR BYTE
				000018		877	1	GRG	ASADGADR
000018	00000000					878	1	ASADGMGN DC A(0)	MSG: MESSAGE ID
						879	1	ASADGBC DC X'00'	MSG: FLAGBYTE
				000000		880	1	ASADGNIL EQU X'00'	MSG: NO INDICATORS
				000080		881	1	ASADGRPE EQU X'80'	MSG: REPLY EXPECTED
				000040		882	1	ASADGCG EQU X'40'	MSG: OUTPUT TO CONSOLE
00001D	00					883	1	ASADGIN DC X'00'	MSG: NUMBER OF INSERTS
00001E	0000					884	1	DC XL2'00'	MSG: NOT USED
000020	00					885	1	ASADSRP DC X'00'	MSG: SIZE OF REPLY (OPTIONAL - ABER FALLS REPLY EXPECTED, MUSS LAENGE >0 SEIN)
						886	1	*	
000021	000000					887	1	ASADGARP DC AL3(0)	MSG: ADDRESS(REPLY) (OPTIONAL- ABER FALLS REPLY EXPECTED, MUSS ADRESSE HALBWGRTAUSGERICHTET SEIN)
						888	1	*	

FLAG LCTN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

000024 00000000

889 1 ASADGINS DC 15A(0) MSG: MAX. 15 INSERTS. (OPTIONAL)
890 1 * 1ST BYTE HOLDS SIZE(INSERT)
891 1 * NEXT 3 BYTES A(INSERT)

000013
000000
000004

892 1 *****
893 1 ** RUECKKEHR - PARAMETER **
894 1 *****
895 1 ASADGRTN EQU ASARETRN ALL: RETURN CODE
896 1 ASADGGK EQU X'0' ALL: ALL G.K.
897 1 ASADGFE EQU X'4' ALL: FORMALER FEHLER
898 1 * - STRINGLAENGE < 5
899 1 * - AUSGABE IN MEMORY-POLLS
900 1 * - A(AUSGABESTRING) NICHT HALBWORT-
901 1 * AUSGERICHTET
902 1 * - A(AUSGABESTRING) NICHT ALLOKIIERT
903 1 * - A(MSG-INSERT) NICHT ALLOKIIERT
904 1 * - A(MSG-ANTWORT-PUFFER) NICHT
905 1 * ALLOKIIERT
906 1 * - FALLS REPLY EXPECTED:
907 1 * A(REPLY) NICHT HALBWORT-
908 1 * AUSGERICHTET

000008
00000C

909 1 ASADGIMS EQU X'8' MSG: ERROR IN MESSAGE-PARAMS
910 1 ASADGBT EQU X'C' MSG: NON-CONVERSATIONAL USER ASKED FOR
911 1 * REPLY
912 1 ASADGAIQ EQU X'10' MSG: ABNORMAL I/O TERMINATION
913 1 ASADGEGM EQU X'14' ALL BUT MSG: ERROR ON MEDIUM, SEE DGMER
914 1 ASADGHHM EQU X'18' ALL: NO MEDIUM DEFINED
915 1 ASADGHHX EQU X'1C' MSG: MSG BIT NOT SET EXCLUSIVELY
916 1 ASADGHP EQU X'20' FIL: NO FILE# GIVEN
917 1 ASADGHHM EQU X'24' MEM: NO MEM# GIVEN
918 1 ASADGCHQ EQU X'28' FIL: FILE COULD NOT BE OPENED.
919 1 * SEE "DGFMM" FOR FILE# AND 'DERR'
920 1 * FOR DMS-RETURNCODE

00002C
000030

921 1 ASADGNT EQU X'2C' FIL: FCB-TYPE NOT SUPPORTED ("DGFMM")
922 1 ASADGFIX EQU X'30' FIL: RECORDFORM IS NOT 'V'
923 1 ASADGXXX DS GC
924 1 ASADGMR DC X'00' ALL: MEDIUM WITH ERROR, LAYOUT AS
925 1 * DGMED, FILE/MEM # IN DGFMM

000060
000060 00

000061 00
000062 00

926 1 ASADGFMN DC X'00' FIL/MEM: FILE/MEM #, IF THERE ANY ERROR
927 1 ASADGFB DC X'00' TER: INDICATOR FOR TERMINAL OVERFLOW
928 1 ASADGQVF EQU X'80' TER: TERMINAL OVERFLOW
929 1 ASADGCTG EQU X'40' TER: CONTINUE OUTPUT
930 1 ASADGHHG EQU X'20' TER: TERMINATE OPERAND; NEXT OPERAND
931 1 ASADGHHX EQU X'10' TER: TERMINATE COMMAND
932 1 ASADGTRM EQU X'08' TER: TERMINATE ACTIONLIST
933 1 ASADGUPD EQU X'04' TER: UPDATE
934 1 ASADGUSF DC A(0) TER: A(BUFFER WITH UPDATE STRING)
935 1 ASADGERR DC H'0' FIL: DMS-ERROR-CODE, FALLS RTN='DGCN0'
936 1 ASADGLEN EQU *-ASADGXXX LENGTH(RET.PARAM)

000064 00000000
000068 0000

00000A

937 1 *****

FLAG LGCTH OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

FLAG	LGCTH	OBJECT CODE	ADDR1	ADDR2	STMT	M	SOURCE STATEMENT
					938	1	**
					939	1	**
					940	1	**
					941	1	**
					942	1	**
					943	1	*****
		000014			944	1	ORG ASALABEL
					945	1	*****
					946	1	*****
					947	1	*****
		00000C			948	1	ASASE EQU X'0C' FUNKTION = SYSTEM-EDIT
		000000			949	1	ASASEINF EQU X'00' SUBFUNKTION = NUR INFORMIEREN
		000004			950	1	ASASEEDT EQU X'04' SUBFUNKTION = TABELLEN AUFBEREITEN UND
					951	1	* IN PUFFER UEBERTRAGEN
		000004			952	1	ASASEMAX EQU X'04' MAX. NUMMER EINER SUBFUNKTION
					953	1	*****
					954	1	**
					955	1	*****
000014	00				956	1	ASASEKEY DC X'00' ALL: AIDSYS-INTERNE VERSCHLUESSELUNG DES
					957	1	* KEYWORDS. WIRD VON AIDSYS BEI AUFRUF
					958	1	* DER INFORMIERE-FUNKTION HIER ABGELEGT.
					959	1	* KANN BEI AUFRUF MITTELS 'ASASEEDT'
					960	1	* ANGEGEBEN WERDEN. IST BEI 'ASASEEDT'
					961	1	* SONDHL 'ASASEKEY' ALS AUCH 'ASASESTR'
					962	1	* ANGEGEBEN, WIRD 'ASASESTR' IGNORIERT.
		000000			963	1	ASASENGK EQU X'00' ALL: KEINE KEY# GEGEBEN
		000004			964	1	ASASESTK EQU X'04' ALL: STACK
		000004			965	1	ASASEPCB EQU X'04' ALL: PCB
		000008			966	1	ASASEPL EQU X'08' ALL: PCBLS
		00000C			967	1	ASASEAU1 EQU X'0C' ALL: P1 AUDIT
		000010			968	1	ASASEAU2 EQU X'10' ALL: P2 AUDIT
		000014			969	1	ASASETT EQU X'14' ALL: TT (TRACE TABLE)
		000018			970	1	ASASETSK EQU X'18' ALL: TASK
		00001C			971	1	ASASEQU EQU X'1C' ALL: QUEUE
		000020			972	1	ASASEPRD EQU X'20' ALL: PEND
		000024			973	1	ASASEALL EQU X'24' ALL: ALL
		000028			974	1	ASASEPTT EQU X'28' ALL: FULL TRACE TABLE
		000028			975	1	ASASEKMX EQU X'28' MAXIMALE KEY#
					976	1	ASASESTR DC CL8'STRNG' ALL: %-LOSSES SCHLUESSELWORT. MUSS BEI
					977	1	* INFORMIERE-FUNKTION GESETZT SEIN.
					978	1	* WIRD BEI EDIT-FUNKTION IGNORIERT,
					979	1	* FALLS 'ASASEKEY' GESETZT IST.
00001D	00				980	1	ASASEINX DC X'00' EDT: WAHLWEISE: INDEX FUER DEN KEY
00001E	00				981	1	ASASEPB DC X'00' EDT: INDIKATOR
		000000			982	1	ASASEFIT EQU ASAITN NICHT AUSGEWERTET
		000000			983	1	ASASEFTS EQU ASATSN NICHT AUSGEWERTET
		00004C			984	1	ASASEBYS EQU X'40' EDT: SCHLUESSELWORT SOLL AUF SYSTEMADRESS-
					985	1	* RAUM ANGENENDET WERDEN.
		000020			986	1	ASASECNT EQU X'20' EDT: FOLGEAUFRUF DER LETZTEN EDIT-FUNKTION
		000010			987	1	ASASEDP EQU ASADUMP EDT: ZUGRIFF AUF DUMPFIL
		000002			988	1	ASASESTB EQU X'02' EDT: NUR ORIGINALTABELLE GEWUENSCHT
					989	1	* (NUR FUER AUDIT UNTERSTUETZT)

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE	STATEMENT	EDT:
				000001		990	1	ASASEIXV EQU	X'01'	EDT: INDEX IST GUELTIG
00001F	00					991	1	ASASEDF DC	X'00'	EDT: LINKNAME EINES DUMPFILERS
000020	00					992	1	ASASELL DC	X'00'	EDT: ZEILENLAENGE. ES WERDEN NUR 80/132
						993	1	*		ZEICHEN PRG ZEILE UNTERSTUETZT.
000022	0000					994	1	ASASEITN DC	H'0'	EDT: ITN DES ANZUSPRECHENDEN TASKS.
						995	1	*		X'0000' BEDEUTET DEN DEFAULT-TASK
000024	00000000					996	1	ASASETSN DC	A(0)	EDT: TSN DES ANZUSPRECHENDEN TASKS
						997	1	*		A(0) BEDEUTET DEN DEFAULT-TASK
000028	00000000					998	1	ASASEBUF DC	A(0)	EDT: A(PUFFER). MUSS IN KLASSE5/6 LIEGEN
00002C	00000000					999	1	ASASEBLN DC	A(0)	EDT: LAENGE DES PUFFERS.
000030	00000000					1000	1	DC	4A(0)	NICHT VERWENDET
						1001	1	*****		
						1002	1	**	RUECKKEHR - PARAMETER	**
						1003	1	*****		
000013						1004	1	ASASERTN EQU	ASARETRN	
000000						1005	1	ASASEGK EQU	X'00'	ALL: ALLES OK, ABER SIEHE 'ASASERFB'
000004						1006	1	ASASEFE EQU	X'04'	ALL: FORMALER FEHLER
						1007	1	*		- KEINE ZEILENLAENGE ANGEGBEN
						1008	1	*		- UNBEKANNTE SUBFUNKTIONSNUMMER
						1009	1	*		- KEINE PUFFERLAENGE ANGEGBEN
						1010	1	*		- KEIN PUFFER ANGEGBEN
						1011	1	*		- KEIN KEY ANGEGBEN
						1012	1	*		- FEHLERHAFTER LINK#
						1013	1	*		- PUFFER IST SO KLEIN, DASS KEIN
						1014	1	*		EINZIGER SATZ AUSGEGBEN WIRD
000008						1015	1	ASASEKNF EQU	X'08'	ALL: UNBEKANNTES SCHLUESSELWORT/INTERNER
						1016	1	*		SCHLUESSEL
00000C						1017	1	ASASENIA EQU	X'0C'	EDT: UNERLAUBTER INDEX ANGEGBEN
000010						1018	1	ASASEBNA EQU	X'10'	EDT: PUFFER NICHT IN VOLLER LAENGE
						1019	1	*		SCHREIBZUGREIFBAR, ODER NICHT IN
						1020	1	*		KLASSE-5/6-SPEICHER
000014						1021	1	ASASEFOL EQU	X'14'	EDT: FOLGEAUFRUF ANGEGBEN OHNE VORAN-
						1022	1	*		GEGANGENEN ORIGINALAUFRUF
000018						1023	1	ASASEKNA EQU	X'18'	EDT: TABELLE EXISTIERT NICHT IM ANGEGBENEN
						1024	1	*		ADRESSRAUM ODER WURDE NICHT GESAVED
						1025	1	*		REALER SLED: AUF TABELLE KONNTE NICHT
						1026	1	*		ZUGEGRIFFEN WERDEN.
						1027	1	*		(SEITE NICHT IM DF)
00001C						1028	1	ASASEDFN EQU	X'1C'	EDT: DUMPFILERS NICHT OFFEN
000020						1029	1	ASASETNF EQU	X'20'	EDT: ITN/TSN EXISTIERT NICHT
000024						1030	1	ASASENGT EQU	X'24'	EDT: KEIN TASK SPEZIFIZIERT (NUR FUER
						1031	1	*		DUMPFILERSZUGRIFFE)
000028						1032	1	ASASEMEM EQU	X'28'	EDT: FUNKTION WEGEN SPEICHERMANGEL NICHT
						1033	1	*		AUSFUHRBAR.
00002C						1034	1	ASASEPV1 EQU	X'2C'	EDT: PRIVILEGIERUNG ZU KLEIN, KANN ABER
						1035	1	*		NOCH AUSREICHEND ERHOEHET WERDEN.
000030						1036	1	ASASEPV2 EQU	X'30'	EDT: PRIVILEGIERUNG ZU KLEIN, KANN NICHT
						1037	1	*		MEHR AUSREICHEND ERHOEHET WERDEN.
000034						1038	1	ASASERPT EQU	X'34'	NICHT MEHR VERWENDET
000038						1039	1	ASASENYI EQU	X'38'	EDT: FUNKTION NOCH NICHT IMPLEMENTIERT
00003C						1040	1	ASASENAL EQU	X'3C'	EDT: WEGEN SPEICHERMANGEL NUR TEIL DER

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE	STATEMENT
						1041	1	*	AUDIT-TABELLE GELIEFERT
			000040			1042	1	ASASELGP EQU X'40'	EDT: PCB-SCHLEIFE ENTDECKT. PUFFER
						1043	1	*	WURDE MIT INFORMATION GEFUELLT,
						1044	1	*	ABER ES IST KEIN FORTSETZUNGSAUFRUF
						1045	1	*	MEHR MOEGLICH
			000044			1046	1	ASASESER EQU X'44'	EDT: TABELLE MEGEN FALSCHER POINTER NICHT
						1047	1	*	AUSGEBBAR (SYSTEMFEHLER)
			000048			1048	1	ASASENP EQU X'48'	EDT: NUR FUER PCB-ZUGRIFFE. DIE GELIEFERTE
						1049	1	*	INFORMATION IST VERMUTLICH KEIN PCB
			00004C			1050	1	ASASETEY EQU X'4C'	EDT: TABELLE LEER. EVENTUELLEN INHALT
						1051	1	*	IM AUSGABEBUFFER IGNORIEREN
000040						1052	1	ASASEXXX DS DC	
000040	00					1053	1	ASASERFB DC X'00'	ALL: INDIKATOR
			000080			1054	1	ASASEIND EQU X'80'	INF: SCHLUESSELWORT KANN INDEX HABEN
			000040			1055	1	ASASERSU EQU X'40'	INF: TRENNUNG NACH USER/SYSTEM MOEGLICH
			000020			1056	1	ASASEBF EQU X'20'	EDT: PUFFER IST VOLL; WEITERER AUFRUF MIT
						1057	1	*	CONTINUE-ANZEIGE IST NOETIG
			000004			1058	1	ASASEQL EQU ASAUQUAL	ALL: UEBERQUALIFIZIERTE EINGABE
000041	00					1059	1	ASASERX DC X'00'	INF: MAXIMALWERT DES INDEX
			000002			1060	1	ASASELEN EQU *-ASASEXXX	LAENGE DER RUECKKEHR-PARAMETER
			000042			1061	1	ASASEGLN EQU *-ASASERVE	GESAMTLAENGE (EIN/AUSGABE-PARAMS)
						1062	1	*****	
						1063	1	**	
						1064	1	** H A R D W A R E - I N F O R M A T I O N F U E R A N W E N D E R **	
						1065	1	** ----- **	
						1066	1	** DER AUFRUFER KANN SICH MIT DIESEM SERVICE AUSGEWAELTE SPEICHER- **	
						1067	1	** BEREICHE AUS SLEDFILES IN EINEN VON IHM BEREITGESTELLTEN PUFFER **	
						1068	1	** AUSGEBEN LASSEN: **	
						1069	1	** - SCRATCHPAD (BZW. PSM) **	
						1070	1	** - SPECIAL REGISTER SECTION (NUR BEI X-ANLAGEN) **	
						1071	1	** - CPU-/IOP-LOGGOUT-AREAS **	
						1072	1	** - HARDWARE-TRACE **	
						1073	1	** - AUDIT-ADRESS-REGISTER **	
						1074	1	** **	
						1075	1	** BEMERKUNG: DIE AUDIT ADRESS REGISTER WERDEN NICHT - WIE ALLE **	
						1076	1	** ANDEREN SPEICHERBEREICHE - IN EINEN VOM AUFRUFER BEREITGESTELL- **	
						1077	1	** TEN PUFFER UEBERTRAGEN, SONDERN DIREKT IN DIE ASERP-RETURNPA- **	
						1078	1	** RAMETER GESCHRIEBEN. **	
						1079	1	** **	
						1080	1	** DIESER SERVICE IST UEBER TAM AUFRUFBAR. **	
						1081	1	** ER SOLL IN ZUKUNFT DEN SERVICE HARDWARE-INFORMATION (FUNKTION **	
						1082	1	** X'30') WEITGEHEND ERSETZEN, DA ER DESSEN FUNKTIONEN VOLL MIT **	
						1083	1	** ABDECKT. **	
						1084	1	** **	
						1085	1	** HINWEIS: ES WERDEN STETS NUR ZUSAMMENHAENGENDE STUECKE AUS **	
						1086	1	** DEN SPEICHERBEREICHEN IN DEN VOM ANWENDER BEREIT- **	
						1087	1	** GESTELLTEN (AUF WORTGRENZE AUSGERICHTETEN) PUFFER **	

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE	STATEMENT
						1088	1	**	UEBERTRAGEN. VERSCHIEDENE SPEICHERBEREICHE KOENNEN **
						1089	1	**	NUR MIT MEHREREN SERVICE-AUFRUFEN UEBERTRAGEN WER- **
						1090	1	**	DEN. **
						1091	1	*****	*****
				000014		1092	1		GRG ASALABEL
						1093	1	*****	*****
						1094	1	**	HEADER-PARAMETER **
						1095	1	*****	*****
				000010		1096	1	ASAH1	EQU X'10' FUNKTION= HARDWARE-INFORMATION
						1097	1	*	FUER DEN ANWENDER
				000000		1098	1	ASAHISP	EQU X'00' SUBFUNKTION: AUSGABE SCRATCHPAD
				000004		1099	1	ASAHILA	EQU X'04' SUBFUNKTION: AUSGABE LOGGOUT-AREA
				000008		1100	1	ASAHISPR	EQU X'08' SUBFUNKTION: AUSGABE SPECIAL REGISTER
						1101	1	*	SECTION
				00000C		1102	1	ASAHIAAR	EQU X'0C' SUBFUNKTION: AUSGABE ALLER AUDIT ADRESS
						1103	1	*	REGISTER IN AUSGABEPARAMETER(!)
				000010		1104	1	ASAHIHWT	EQU X'10' SUBFUNKTION: AUSGABE HARDWARE-TRACE
						1105	1	*	(NOCH NICHT UNTERSTUETZT)
				000010		1106	1	ASAHIMAX	EQU X'10' HOECHSTE SUBFUNKTIONSNUMMER
						1107	1	*****	*****
						1108	1	**	AUFRUF-PARAMETER **
						1109	1	*****	*****
000014	00					1110	1	ASAHILNK	DC X'00' ALL: SLEDFILE-LINKNUMMER
000015	00					1111	1	ASAHIPRC	DC X'00' LA: TYP DES PROZESSORS (CPU ODER IOP),
						1112	1	*	DESSEN LOGGOUT UEBERTRAGEN WERDEN SOLL
				000080		1113	1	ASAHICPU	EQU X'80' PROZESSOR IST CPU
				000040		1114	1	ASAHIIOP	EQU X'40' PROZESSOR IST IOP
000016	00					1115	1	ASAHIPR#	DC X'00' BUT AAR: NUMMER DES PROZESSORS, DESSEN SPEICHER-
						1116	1	*	BEREICHE UEBERTRAGEN WERDEN SOLL
						1117	1	*	! IST DIE PR0Z# NICHT ANGEGBEN, WIRD
						1118	1	*	STANDARDMAESSIG PR0Z#=0 ANGENOMMEN
000018	00000000					1119	1	ASAHIBFA	DC A(0) BUT AAR: A(UEBERGEBENEM PUFFER); PUFFER MUSS
						1120	1	*	AUF WORTGRENZE AUSGERICHTET SEIN.
00001C	0000					1121	1	ASAHIBFL	DC Y(0) BUT AAR: LAENGE DES UEBERGEBENEM PUFFERS
00001E	0000					1122	1	ASAHIRIA	DC Y(0) BUT AAR: REL. ANFANGSADR.(BZGL. TABEL-
						1123	1	*	LENANFANG) DER GEW. INFORMATION
						1124	1	*	! DEFAULTWERT: 0
000020	0000					1125	1	ASAHIIL	DC Y(0) BUT AAR: GEWUENSCHTE UEBERTRAGUNGSLAENGE
						1126	1	*	! STANDARDWERT: PUFFERLAENGE, MAXIMAL
						1127	1	*	DIE TABELLENLAENGE
						1128	1	*****	*****
						1129	1	**	RUECKKEHR-PARAMETER **
						1130	1	*****	*****
				000013		1131	1	ASAHIRTN	EQU ASARETRN ALL: RETURNCODE
				000000		1132	1	ASAHIAOK	EQU X'00' ALL: ALLES G.K.
				000004		1133	1	ASAHIAFE	EQU X'04' ALL: FORMALER FEHLER:
						1134	1	*	- UNQUELTIGE SUBFUNKTION
						1135	1	*	- UNQUELTIGE LINK#
						1136	1	*	- PUFFERADRESSE 0 ODER NICHT GESETZT
						1137	1	*	- PUFFER NICHT ALLOKIIERT
						1138	1	*	ODER PUFFER IM SYSTEMRAUM
						1139	1	*	- PUFFER FUER INFO ZU KURZ

FLAG LCTN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

```

1140 1 * - UNGUELTIGE PROZESSGR#
1141 1 * - REL. ANFANGSADR. VON INFO
1142 1 * GRÖßER ALS TABELLENLAENGE
1143 1 * - UNGUELTIGER PROZESSORTYP
000008 1144 1 ASAHINYT EQU X'08' HWT: SUBFUNKTION NOCH NICHT UNTERSTUETZT
00000C 1145 1 ASAHINGI EQU X'0C' ALL: INFORMATION KONNTE NICHT UEBERTRAGEN
1146 1 * WERDEN ODER IST IN VORLIEGENDEM SLED NICHT VGRHANDEN
000010 1147 1 ASAHILIL EQU X'10' ALL: INFORMATION WIRD HOECHSTENS
1148 1 * IN TABELLENLAENGE UEBERTRAGEN
000014 1149 1 ASAHIFNG EQU X'14' ALL: DUMPFIL NICHT OFFEN
000018 1150 1 ASAHING3 EQU X'18' ALL: KEIN SLEDFILE
000024 1151 1 ASAHIXXX DS OF
000024 00000000 1152 1 ASAHIAA0 DC A(0) AAR: AUDIT ADRESS REGISTER (CPU-0)
000028 00000000 1153 1 ASAHIAA1 DC A(0) AAR: " " " (CPU-1)
00002C 00000000 1154 1 ASAHIAA2 DC A(0) AAR: " " " (CPU-2)
000030 00000000 1155 1 ASAHIAA3 DC A(0) AAR: " " " (CPU-3)
000034 0000 1156 1 ASAHIMVL DC Y(0) BUT AAR: ANZAHL DER UEBERTRAGENEN BYTES
000012 1157 1 ASAHILEN EQU *-ASAHIXXX LAENGE DER RETURNPARAMETER

1158 1 *****
1159 1 ** **
1160 1 ** OPEN DUMPFILE **
1161 1 ** ----- **
1162 1 ** DIESER SERVICE MUSS VOR DEM ERSTEM DUMPDATEI-ZUGRIFF ERFOLGEN. **
1163 1 ** DAMIT KANN SICH DER AUFRUFER EINE DUMPDATEI FUER ZUKUENFTIGE **
1164 1 ** ZUGRIFFE ANMELDEN UND ERGEBEN. EIN UND DIESELBE DATEI KANN **
1165 1 ** UNTER VERSCHIEDENEN KURZNAMEN MEHRMALS ERGEBNET WERDEN. DER **
1166 1 ** GLEICHE KURZNAME IST IMMER NUR FUER GENAU EINE OFFENE DATEI **
1167 1 ** VERWENDBAR. **
1168 1 ** **
000014 1169 1 *****
1170 1 GRG ASALABEL

1171 1 *****
1172 1 ** HEADER - PARAMETER **
1173 1 *****
000018 1174 1 ASAGT EQU X'18' OPEN DUMP-FILE
1175 1 * ES GIBT KEINE SUBFUNKTION.
000000 1176 1 ASAGTMAX EQU X'00' MAX. # OF SUBFUNCTION

1177 1 *****
1178 1 ** AUFRUF - PARAMETER **
1179 1 *****
000014 4040404040404040 1180 1 ASAGTFIL DC CL54' ' DUMP FILE NAME
00004A 00 1181 1 ASAGTLN# DC X'00' LINK # OF SPECIFIED DUMPFIL

1182 1 *****
1183 1 ** RUECKKEHR - PARAMETER **
1184 1 *****
000013 1185 1 ASAGTRTN EQU ASARETRN RETURN CODE
000000 1186 1 ASAGTOK EQU X'00' DUMPFIL OPENED
000004 1187 1 ASAGTTMD EQU X'04' FORMAL ERROR
    
```

FLAG	LOC	OBJECT	CODE	ADDR1	ADDR2	STMT	M	SOURCE STATEMENT
						1188	1	* - INVALID LINK#
						1189	1	* - INVALID SUBFUNCTION
						1190	1	ASAGTNE EQU X'08'
						1191	1	ASAGTNDT EQU X'0C'
						1192	1	ASAGTHGM EQU X'10'
						1193	1	ASAGTIVB EQU X'14'
						1194	1	* FILE COULD NOT BE OPENED
						1195	1	ASAGTFAG EQU X'18'
						1196	1	* FILE IS NO DUMPED TASK
						1197	1	ASAGTBSV EQU X'1C'
						1198	1	* NO MEMORY FOR DUMPFILAREA
						1199	1	ASAGTTPE EQU X'20'
						1200	1	ASAGTXXX DS 0X
						1201	1	ASAGTVER DC CL3'Y70'
						1202	1	ASAGTHT DC X'00'
						1203	1	ASAGTTYP DC X'00'
						1204	1	ASAGTSLD EQU X'08'
						1205	1	ASAGTCUS EQU X'04'
						1206	1	ASAGTCSY EQU X'02'
						1207	1	ASAGTHTK DC H'00'
						1208	1	ASAGTTIT DC A(0)
						1209	1	ASAGTBAS DC A(0)
						1210	1	ASAGTXVT DC A(0)
						1211	1	ASAGTMEM DC A(0)
						1212	1	ASAGTERR DC H'0'
						1213	1	*****
						1214	1	*****
						1215	1	*****
						1216	1	ASAGTSLN EQU *-ASAGTXXX
						1217	1	ASAGTXSL DS Y(0)
						1218	1	ASAGTSRL DS Y(0)
						1219	1	ASAGTOPL DS Y(0)
						1220	1	ASAGTIPL DS Y(0)
						1221	1	ASAGTCBM DS X'00'
						1222	1	ASAGTIBM DS X'00'
						1223	1	ASAGTLEN EQU *-ASAGTXXX
						1224	1	*****
						1225	1	**
						1226	1	**
						1227	1	**
						1228	1	**
						1229	1	**
						1230	1	**
						1231	1	**
						1232	1	**
						1233	1	**
						1234	1	**
						1235	1	**
						1236	1	**

REQUEST STORAGE

DIESER SERVICE DIENST ZUM ANFORDERN VON KLASSE- 5 - SPEICHER.
 EIN P1-AUFRUFER ERHAELT AUTOMATISCH NICHT-PRIVILEGIERTEN, EIN
 P2-AUFRUFER PRIVILEGIERTEN SPEICHER. IN JEDEM FALL ERFOLGT NUR
 EINE GANZWORT - AUSRICHTUNG DES DELIEFERTEN BEREICHS.

GRG ASALABEL

HEADER - PARAMETER

FLAG	LGCTN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT
						1237	1	*****
			00001C			1238	1	ASARQ EQU X'1C' ALL: FUNCTION = REQUEST MEMORY
			000000			1239	1	ASARQD EQU X'00' D: SUBFUNCTION = DYNAMIC STORAGE
			000004			1240	1	ASARQS EQU X'04' S: SUBFUNCTION = STATIC STORAGE
			000004			1241	1	ASARQMAX EQU X'04' MAX. # OF SUBFUNCTION
						1242	1	*****
						1243	1	** AUFRUF - PARAMETER **
000014	0000					1244	1	*****
						1245	1	ASARQNB DC H'0' ALL: # OF REQUESTED BYTES
						1246	1	*****
						1247	1	** RUECKKEHR - PARAMETER **
						1248	1	*****
000013						1249	1	ASARQRTN EQU ASARETRN ALL: RETURN CODE
000000						1250	1	ASARQCK EQU X'00' ALL: REQUEST HONORED, ADDR IN RQADR
000004						1251	1	ASARQFE EQU X'04' ALL: FORMALER FEHLER
						1252	1	* - UNGUELTIGE SUBFUNCTION
						1253	1	* - AUFRUFER IST NICHT AID
000008						1254	1	ASARQNB EQU X'08' ALL: NO STORAGE AVAILABLE
000018						1255	1	ASARQXXX DS OF
000018	00000000					1256	1	ASARQADR DC A(0) ALL: ADDRESS OF RETURNED AREA
			00001C			1257	1	ASARQLGH EQU *-ASASERVE LEN(RQ PARAMS)
			000004			1258	1	ASARQLEN EQU *-ASARQXXX LEN(RET.PARAMS)
						1259	1	*****
						1260	1	**
						1261	1	** R E L E A S E M E M O R Y **
						1262	1	** ----- **
						1263	1	** DIESER SERVICE DIENST ZUM FREIGEBEN VON SPEICHER, DER VORHER **
						1264	1	** MIT DEM 'REQUEST MEMORY' - SERVICE ANGEFORDERT WURDE. **
						1265	1	**
						1266	1	*****
000014						1267	1	GRG ASALABEL
						1268	1	*****
						1269	1	** HEADER - PARAMETER **
						1270	1	*****
000020						1271	1	ASARL EQU X'20' FUNCTION = RELEASE MEMORY
						1272	1	* ES GIBT KEINE SUBFUNCTIONEN
000000						1273	1	ASARLMAX EQU X'00' MAX. # OF SUBFUNCTION
						1274	1	*****
						1275	1	** AUFRUF - PARAMETER **
						1276	1	*****
000014	00000000					1277	1	ASARLADR DC A(0) ADDR OF AREA TO RELEASE
						1278	1	* RLADR MUST BE A RQADR, OF AN
						1279	1	* EARLIER REQUEST
						1280	1	ASARLFB DC X'00' INDICATOR
000018	00					1281	1	ASARLSGL EQU X'80' RELEASE ONLY SPECIFIED (BY ADDRESS)

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT
						1282	1	*
						1283	1	*
						1284	1	*
						1285	1	*
						1286	1	*
						1287	1	*****
						1288	1	** RUECKKEHR - PARAMETER **
						1289	1	*****
000013						1290	1	ASARLRN EQU ASARETRN RETURN CODE
000000						1291	1	ASARLQK EQU X'00' AREA COULD BE RELEASED
000004						1292	1	ASARLFE EQU X'04' FORMALER FEHLER
						1293	1	* - AUFRUFER IST NICHT 'AID'
						1294	1	* - UNGUELTIGE SUBFUNKTION
000008						1295	1	ASARLLE EQU X'08' THERE IS NO REQUEST WITH RLADR
						1296	1	*
						1297	1	* NOTE 1: THE IMPLICIT RELEASE OF ALL KINDS OF STORAGE IS DESCRIBED
						1298	1	* IN 'BENUTZERSCHNITTSTELLE FUER AIDSYS'.
						1299	1	* NOTE 2: DYNAMIC STORAGE CAN BE RELEASED ONLY FOR THE ACTUAL
						1300	1	* TESTLEVEL.
000006						1301	1	ASARLLEN EQU *-ASARLRN LENGTH(RET.PARAMS)
						1302	1	*****
						1303	1	**
						1304	1	** CALL SYSTEM **
						1305	1	**
						1306	1	** DIESER SERVICE DIENST ZUR RUECKKEHR ZUM SYSTEM, UM:
						1307	1	** - EIN KOMMANDO ZU LESEN,
						1308	1	** - EIN SYSTEMKOMMANDO AUSFUEHREN ZU LASSEN,
						1309	1	** - DAS PROGRAMM ZU STARTEN.
						1310	1	** DIE RUECKKEHR ZU 'AID' NACH AUSFUEHRUNG EINER FUNKTION KANN
						1311	1	** SOFORT (KDD LESEN), VERZOEGERT (PROGRAMM-START) ODER GAR NICHT
						1312	1	** ERFOLGEN (MANCHE SYSTEM-KDDs).
						1313	1	** DER ERSTE AUFRUF VON 'AID' BZGL. EINER STACK-EBENE ERFOLGT
						1314	1	** UEBER EINE PSEUDO-RUECKKEHR DIESER SERVICE.
						1315	1	**
						1316	1	*****
000014						1317	1	ORG ASALABEL
						1318	1	*****
						1319	1	** HEADER - PARAMETER **
						1320	1	*****
000024						1321	1	ASACS EQU X'24' ALL: FUNCTION = CALL SYSTEM
000000						1322	1	ASACSRQ EQU X'00' RC: SUBFUNCTION = READ NEXT COMMAND
000004						1323	1	ASACSESC EQU X'04' ESC: SUBFUNCTION = EXEC SYSTEM CMD
000008						1324	1	ASACSRSM EQU X'08' RSM: SUBFUNCTION = RESUME
000008						1325	1	ASACSMAX EQU X'08' MAXIMUM SUBFUNCTION #
						1326	1	*****
						1327	1	** AUFRUF - PARAMETER **
						1328	1	*****

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMT	M	SOURCE	STATEMENT	
000014	00000000					1329	1	ASACSDR	DC A(0)	ESC: ADDR OF V-FORMATED CMD
						1330	1	*		MUST BEGIN WITH 1ST 4 BYTES UNUSED
						1331	1	*		(FOLLOVED BY 4 BYTES V-FORMAT AND
						1332	1	*		CMD-STRING.) LENGTH GIVEN IN V-FORMAT
						1333	1	*		IS WITHGUT 1ST 4 BYTES
				000014		1334	1	GRG	ASACSDR	
000014	00					1335	1	ASACSF8	DC X'00'	RC: ANZEIGENBYTE
				000080		1336	1	ASACSSP8	EQU X'80'	RC: JOB STEP TERMINATION DURCHFUEHREN
				000018		1337	1	GRG	ASACSDR+4	
						1338	1	*****		
						1339	1	**	RUECKKEHR - PARAMETER	**
						1340	1	*****		
		000013				1341	1	ASACSRIN	EQU ASARETRN	ALL: RETURN CODE
		000000				1342	1	ASACSCMD	EQU X'00'	ALL: FIRST CHECK, WHETHER ASACSTRM IS
						1343	1	*		SET. IF YES, THEN PSEUD8-RETURN
						1344	1	*		FOR TERMINATION-PROCESSING AND
						1345	1	*		NOTHING ELSE. OTHERWISE REGULAR
						1346	1	*		RETURN FROM CALL WITH SUBFUNCTION.
				000004		1347	1	ASACSF8	EQU X'04'	ALL: FORMALER FEHLER:
						1348	1	*		- AUFRUFER IST NICHT 'AID'
						1349	1	*		- UNGUELTIGE SUBFUNKTION
						1350	1	*		- ESC: KOMMAND8-PUFFER IST NICHT
						1351	1	*		ALLKTIERT
						1352	1	*		- ESC: KOMMAND8-STRING IST NICHT
						1353	1	*		MO8T-AUSGERICHTET
		000008				1354	1	ASACSINT	EQU X'08'	RC: INTERRUPT FOR AID, INT-KEY IN 'CSIKY'
		00000C				1355	1	ASACSEVT	EQU X'0C'	RC: EVENT FOR AID, EVENT-KEY IN 'CSEKY'
		000010				1356	1	ASACSPV	EQU X'10'	RC: PROGRAM GOT LOADED WITH LSD
		000014				1357	1	ASACSLNK	EQU X'14'	RC: MODULE GOT LINKED
000018						1358	1	ASAC8XXX	DS 0X	
000018	00					1359	1	ASAC8INI	DC X'00'	RC: INDICATOR
		000080				1360	1	ASAC88IN	EQU X'80'	RC: INIT. OF STATIC DATA
		000040				1361	1	ASAC88IN	EQU X'40'	RC: INIT. OF DYN. DATA
		000020				1362	1	ASAC88RN	EQU X'20'	RC: MONITG8D PROGRAM DID RUN SINCE
						1363	1	*		LAST TIME'S CALL OF AID
		000010				1364	1	ASAC88RM	EQU X'10'	RC: TERMINATION-PROCESSING
		000008				1365	1	ASAC88EX	EQU X'08'	ESC: SYSTEM-KOMMAND8 WURDE NICHT AUSGEFUEHRT
		000004				1366	1	ASAC88J8T	EQU X'04'	ESC: JOB STEP TERMINATION GEF8RDERT
		000002				1367	1	ASAC88NAL	EQU X'02'	ESC: DAS ANGE8EBENE SYSTEM-KOMMAND8
						1368	1	*		IST NICHT FUER DIE BEARBEITUNG
						1369	1	*		DURCH 'EMCLP' ERLAUBT
000019	00					1370	1	ASAC88IKY	DC X'00'	RC: INTERRUPT-KEY
		000080				1371	1	ASAC88TPI	EQU X'80'	RC: TESTPGINT INTERRUPT
		00007C				1372	1	ASAC88TNI	EQU X'7C'	RC: TEST MODE INTERRUPT
		000084				1373	1	ASAC888XT	EQU X'84'	RC: STIXIT IN USER PROGRAM
		000088				1374	1	ASAC88897	EQU X'88'	RC: SVC 97 (TG AID ?)
		000050				1375	1	ASAC888VC	EQU X'50'	RC: SVC HAS TAKEN PLACE
		000060				1376	1	ASAC88880	EQU X'60'	RC: DATA ERR8R
		000064				1377	1	ASAC88884	EQU X'64'	RC: EXPONENT OVERFLOW
		000068				1378	1	ASAC88888	EQU X'68'	RC: DIVIDE ERR8R
		00006C				1379	1	ASAC8888C	EQU X'6C'	RC: SIGNIFICANCE ERR8R

FLAG LCTN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

FLAG	LCTN	OBJECT CODE	ADDR1	ADDR2	STMT	M	SOURCE STATEMENT	RC:
			000070		1380	1	ASACSW70 EQU X'70'	RC: EXPONENT UNDERFLOW
			000074		1381	1	ASACSW74 EQU X'74'	RC: DECIMAL OVERFLOW
			000078		1382	1	ASACSW78 EQU X'78'	RC: FIXED-POINT OVERFLOW
			000006		1383	1	ASACSW6 EQU X'06'	RC: 2. STXIT (STXIT IN STXIT)
			000044		1384	1	ASACSW44 EQU X'44'	RC: EXIT, AND NO STXIT SVC BEFORE
			00004C		1385	1	ASACSW4C EQU X'4C'	RC: EXIT, AND NO INTERRUPT BEFORE
			000003		1386	1	ASACSW3 EQU X'03'	RC: ERROR IN SVC CALL
			000004		1387	1	ASACSW4 EQU X'04'	RC: ILLEGAL SVC CALL
			000005		1388	1	ASACSW5 EQU X'05'	RC: THERE IS NO STXIT ROUTINE
			00000C		1389	1	ASACSW0C EQU X'0C'	RC: BYTE BOUNDARY
			000048		1390	1	ASACSW48 EQU X'48'	RC: PAGING ERROR
			000054		1391	1	ASACSW54 EQU X'54'	RC: PRIVILEGED OPERATION
			000058		1392	1	ASACSW58 EQU X'58'	RC: ILLEGAL OPERATION CODE
			00005C		1393	1	ASACSW5C EQU X'5C'	RC: ADDRESS ERROR
			000000		1394	1	ASACSW0 EQU X'00'	RC: TERM SVC
			000001		1395	1	ASACSW1 EQU X'01'	RC: TERMD SVC
			000002		1396	1	ASACSW2 EQU X'02'	RC: TERMJ SVC
			000007		1397	1	ASACSW7 EQU X'07'	RC: LPOV / LINK SVC
			000019		1398	1	ASACSEKY EQU ASACSIKY	
00001A	00				1399	1	ASACSLK DC X'00'	RC: SET ONLY, IF ASACSW7 IS SET
					1400	1	*	=C'0': LPOV-SVC
					1401	1	*	=C'L': LINK-SVC
00001B	00				1402	1	ASACSS# DC X'00'	RC: SVCH, SET ONLY IF ASACSSVC IS SET
00001C	00000000				1403	1	ASACSSAD DC A(0)	RC: A(STAT. AID-DATA); VALID ONLY IF
					1404	1	*	CSSIN IS SET
000020	00000000				1405	1	ASACSDAD DC A(0)	RC: A(DYN. AID-DATA); VALID ONLY IF
					1406	1	*	CSDIN IS SET
000024	00000000				1407	1	ASACSCVF DC A(0)	RC: CSCVF<0 : CSCVF POINTS TO V-FORMATED
					1408	1	*	CMD IN SYSTEM
					1409	1	*	CSCVF>=0: CSCVF POINTS TO V-FORMATED
					1410	1	*	IN DYNAMIC STORAGE
		000080			1411	1	ASACSCNC EQU X'80'	ANZEIGE FUER 'KOMMANDO WURDE NICHT
					1412	1	*	KOPIERT'
000028	000000000000				1413	1	ASACSORG DC XL6'00'	RC: ORIGINAL CODE, IF CSIKY =CSTPI
000030	00000000				1414	1	ASACSAM DC A(0)	RC: PROGRAM RELATIVE ADDRESS OF MODULE
000034	00000000				1415	1	ASACSSAC DC A(0)	RC: MODULE RELATIVE ADDRESS OF CSECT
000038	00000000				1416	1	ASACSLGC DC A(0)	RC: A(ENTRY IN LOCALISIERUNGSLISTE)
00003C	00000000				1417	1	ASACSSGLB DC A(0)	RC: A(1ST A-BLOCK)
000040	4040404040404040				1418	1	ASACSSCS DC CL8' '	RC: CSECT-NAME OF ACTUAL LOCATION
000048	4040404040404040				1419	1	ASACSSMD DC CL8' '	RC: MODUL-NAME OF ACTUAL LOCATION
000050	4040404040404040				1420	1	ASACSLPM DC CL8' '	RC: NAME OF MODULE THAT GOT LOADED BY
		000050			1421	1	ASACSLPM BRG ASACSLPM	
000050	E2E5C34040404040				1422	1	ASACSVCH DC CL8'SVC'	RC: NAME OF SVC (SET ONLY, IF 'ASACSSVC'
					1423	1	*	IS SET.
000058	00				1424	1	ASACSTTP DC X'00'	RC: TRACE-TYPE; VALID ONLY IF TRACE IS SET
000059	00				1425	1	ASACSSC DC X'00'	RC: CONDITION-CODE; SET ONLY WHEN TRACE
00005C	00000000				1426	1	ASACSAAL DC 4A(0)	RC: A MAXIMUM OF 4 ADDRESSES OF ACTION-
					1427	1	*	LISTS MAY BE SPECIFIED, IF ASACSRTH = ASACSSINT
					1428	1	*	GR = ASACSEVT.
		000080			1429	1	ASACSTAL EQU X'80'	RC: IF SET (LEFTMOST BYTE OF ABOVE
					1430	1	*	ADDRESS) THEN ACTION-LIST BELONGS TO TRACE-
					1431	1	*	INTERRUPT
00006C	0000				1432	1	ASACSLN DC H'0'	RC: LENGTH OF CMD-STRING

FLAG	LCTN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE	STATEMENT
				000056		1433	1	ASACSLN EQU	*-ASACSXXX LENGTH(RET.PARAMS)
						1434	1	*****	*****
						1435	1	**	**
						1436	1	**	C L O S E D U M P F I L E
						1437	1	**	-----
						1438	1	**	MIT DIESEM SERVICE KANN EIN VORHER MIT 'OPEN DUMPFIL' ANGE-
						1439	1	**	HELDETER DUMPFIL EXPLIZIT GESCHLOSSEN WERDEN.
						1440	1	**	**
						1441	1	*****	*****
			000014			1442	1	ORG	ASALABEL
						1443	1	*****	*****
						1444	1	**	HEADER - PARAMETER
						1445	1	*****	*****
			000028			1446	1	ASACD EQU	X'28' FUNCTION = CLOSE DUMPFIL
			000000			1447	1	ASACDSGL EQU	X'00' SUBFUNCTION = CLOSE 1 SINGLE FILE
			000004			1448	1	ASACDALL EQU	X'04' SUBFUNCTION = CLOSE ALL FILES
			000004			1449	1	ASACDMAX EQU	X'04' MAXIMUM SUBFUNCTION #
						1450	1	*****	*****
						1451	1	**	AUFRUF - PARAMETER
						1452	1	*****	*****
			000014	00		1453	1	ASACDLN# DC	X'00' SGL: LINK# OF DUMPFIL TO BE CLOSED
						1454	1	*****	*****
						1455	1	**	RUECKKEHR - PARAMETER
						1456	1	*****	*****
			000013			1457	1	ASACDRIN EQU	ASARETRN RUECKKEHRCODE
			000000			1458	1	ASACDRK EQU	X'00' ALL: DATEI WURDE GESCHLOSSEN
			000004			1459	1	ASACDFE EQU	X'04' ALL: FORMALER FEHLER:
						1460	1	*	- UNGUELTIGE SUBFUNCTION
			000008			1461	1	ASACDNC EQU	X'08' ALL: DATEI KONNTE NICHT GESCHLOSSEN WERDEN
			000000			1462	1	ASACDIL EQU	X'00' ALL: UNGUELTIGE LINKNUMMER
			000010			1463	1	ASACDFH EQU	X'10' SGL: DATEI WAR NICHT OFFEN
						1464	1	ASACDXXX DS	0X
			000015			1465	1	ASACDFIL DC	CL54' SGL: NAME OF FILE
			000015	4040404040404040		1466	1	ASACDLN EQU	*-ASACDXXX LENGTH(RETURN PARAMS)
						1467	1	*****	*****
						1468	1	**	**
						1469	1	**	C L O S E O U T P U T F I L E
						1470	1	**	-----
						1471	1	**	DIESER SERVICE DIENT ZUM EXPLIZITEN SCHLIESSEN EINER AUSGABE-
						1472	1	**	DATEI.
						1473	1	**	**
						1474	1	*****	*****
			000014			1475	1	ORG	ASALABEL

FLAG LCTN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

```

1476 1 *****
1477 1 **          HEADER - PARAMETER          **
1478 1 *****
1479 1 ASACL    EQU   X'2C'      ALL: FCT =CLOSE OUTPUTFILE
1480 1 ASACLOF  EQU   X'00'      GF: SUBFUNCTION = CLOSE OUTPUT-FILE
1481 1 ASACLMAX EQU   X'00'      MAXIMUM SUBFUNCTION #

1482 1 *****
1483 1 **          AUFRUF - PARAMETER          **
1484 1 *****
1485 1 ASACLLN# DC   X'00'      ALL: LINKNUMMER DER AUSGABEDATEI

1486 1 *****
1487 1 **          RUECKKEHR - PARAMETER        **
1488 1 *****
1489 1 ASACLRTN EQU  ASARETRN    ALL: RETURN CODE
1490 1 ASACLOK  EQU  X'00'      ALL: DATEI WURDE GESCHLOSSEN
1491 1 ASACLFE  EQU  X'04'      ALL: FORMALER FEHLER:
1492 1 *          - AUFRUFER IST NICHT 'AID'
1493 1 *          - UNGUELTIGE SUBFUNCTION
1494 1 ASACLILK EQU  X'08'      ALL: UNGUELTIGE LINKNUMMER
1495 1 ASACLINC EQU  X'0C'      ALL: DATEI KONNTE NICHT GESCHLOSSEN WERDEN
1496 1 ASACLOPN EQU  X'10'      ALL: DATEI WAR NICHT OFFEN

1497 1 ASACLXXX DS    0X
1498 1 ASACLFIL DC   CL54'FILE' ALL: DATEINAME
1499 1 ASACLLEN EQU  *-ASACLXXX  LENGTH(RETURN PARAMS)
    
```

00002C
000000
000000

000014 00

000013
000000
000004

000008
00000C
000010

000015
000015 C6C9D3C540404040
000036

```

1500 1 *****
1501 1 **
1502 1 **          S E T   S W I T C H          **
1503 1 **          -----          **
1504 1 **  DURCH AUFRUF DIESER FUNKTION KOENNEN GLOBALE VOREINSTELLUNGEN **
1505 1 **  IN 'AIDSYS' GEAENDERT WERDEN.          **
1506 1 **  - EIN/AUSGABE AUF HAUPTKONSOLE          **
1507 1 **  - EIN/AUSGABE UEBER 'SYSOUT'          **
1508 1 **  - 'AID' = GLD (NOCH NICHT UNTERSTUETZT) **
1509 1 *****
1510 1          GRG  ASALABEL

1511 1 *****
1512 1 **          HEADER - PARAMETER          **
1513 1 *****
1514 1 ASACB    EQU   X'30'      ALL: FUNCTION = SET SWITCH
1515 1 ASACBIN  EQU   X'00'      IN:  SUBFUNCTION =SWITCH TO CONSOLE-I/O
1516 1 ASACBOUT EQU   X'04'      OUT: SUBFUNCTION =SWITCH TO SYSOUT
1517 1 ASACBGD  EQU   X'08'      GLD: SUBFUNCTION =SWITCH TO AID=GLD
1518 1 *          (NOT YET SUPPORTED)

1519 1 *****
1520 1 **          AUFRUF - PARAMETER          **
    
```

000014

000030
000000
000004
000008

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT
						1521	1	*****
						1522	1	*
						1523	1	* ES GIBT KEINE AUFRUF-PARAMETER
						1524	1	*****
						1525	1	** RUECKKEHR - PARAMETER **
						1526	1	*****
000013						1527	1	ASACBRTN EQU ASARETRN ALL: RETURN CODE
000000						1528	1	ASACBCK EQU X'00' ALL: REQUEST HONORED
000004						1529	1	ASACDFE EQU X'04' ALL: FORMALER FEHLER:
						1530	1	* - AUFRUFER IST NICHT 'AID'
						1531	1	* - UNGUELTIGE SUBFUNKTION
000008						1532	1	ASACPRV EQU X'08' IN: CALLER IS NOT TSGS
						1533	1	*****
						1534	1	** **
						1535	1	** SAVE SYSTEM TABLES **
						1536	1	** ----- **
						1537	1	** **
						1538	1	** TCB, PCBS AND P2 AUDIT TABLES ARE SAVED FOR AN **
						1539	1	** SPECIFIED TASK. **
						1540	1	** **
000014						1541	1	*****
						1542	1	ORG ASALABEL
						1543	1	*****
						1544	1	** HEADER - INFORMATION **
						1545	1	*****
000034						1546	1	ASASV EQU X'34' ALL: FUNCTION = SAVE SYSTEMTABLES
000000						1547	1	ASASVSAV EQU X'00' SAV: SUBFUNCT: SAVE SYSTABLES FOR SPEC.TASK
000004						1548	1	ASASVUSV EQU X'04' USV: SUBFUNCT: UNSAVE SYSTABLES F.SPEC.TASK
000008						1549	1	ASASVUSA EQU X'08' USA: SUBFUNCT: UNSAVE SYSTABLES F.ALL TASKS
000008						1550	1	ASASVMAX EQU X'08' MAXIMUM SUBFUNCTION
						1551	1	*****
						1552	1	** AUFRUF - PARAMETER **
						1553	1	*****
000014	00000000					1554	1	ASASVTSN DC F'0' ALL: TSN
000018	0000					1555	1	ASASVITH DC X'0000' ALL: ITN
						1556	1	*****
						1557	1	** RUECKKEHR - PARAMETER **
						1558	1	*****
000013						1559	1	ASASVRTN EQU ASARETRN
000000						1560	1	ASASVCK EQU X'00' ALL: REQUEST HONORED
000004						1561	1	ASASVFE EQU X'04' ALL: FORMALER FEHLER:
						1562	1	* ALL: - ILLEGAL SUBFUNCTION
						1563	1	* SAV/USV: - 2 BYTE ITN
						1564	1	* SAV/USV: - NEITHER TSN NOR ITN GIVEN
000008						1565	1	ASASVONT EQU X'08' SAV: FOR OWN TASK FUNCTION NOT EXECUTED

FLAG	LOC TN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE	STATEMENT
				00000C		1566	1	'ASASVMEM EQU X'0C'	SAV: NO MEMORY AVAILABLE FOR SAVE
				000010		1567	1	ASASVNSV EQU X'10'	USV: NOTHING SAVED FOR THIS TASK
				000014		1568	1	ASASVNAP EQU X'14'	SAV: NOT ALL PCB'S ARE SAVED
				000018		1569	1	ASASVCAN EQU X'18'	ALL: FUNCTION NOT AVAILABLE FOR SOME DAYS
						1570	1	*	THIS RC CAN ONLY BE ACTIVATED BY REPS
				00001C		1571	1	ASASVNMR EQU X'1C'	SAV: FUNCTION CALLED BY TOO MANY TASKS
				000020		1572	1	ASASVTNF EQU ASASVTNF	SAV/USV: TSN/ITN DOES NOT EXIST
				000024		1573	1	ASASVHTT EQU X'24'	SAV/USV: ANOTHER TASKS IS EXECUTING
						1574	1	*	THIS FUNCTION. TRY IT LATER.
00001A	0000					1575	1	ASASVPB# DC H'0'	ANZAHL DER GESAVTEN PCB'S. NUR DANN
						1576	1	*	GESETZT, IF RTH = 'SVGK' ODER 'SVNAP'
				00001C		1577	1	ASASVLEN EQU *-ASASERVE	
						1578	1	*****	*****
						1579	1	**	**
						1580	1	**	**
						1581	1	**	**
						1582	1	**	**
						1583	1	**	**
						1584	1	*****	*****
000014						1585	1	GRG ASALABEL	
						1586	1	*****	*****
						1587	1	**	**
						1588	1	*****	*****
000038						1589	1	ASAGD EQU X'38'	FUNCTION = GET DATE
						1590	1	*	ES GIBT KEINE SUBFUNCTIONS
000000						1591	1	ASAGD MAX EQU X'00'	MAXIMALE SUBFUNCTION-#
						1592	1	*****	*****
						1593	1	**	**
						1594	1	*****	*****
						1595	1	*	*
						1596	1	*	*
						1597	1	*	*
						1598	1	*****	*****
						1599	1	**	**
						1600	1	*****	*****
000013						1601	1	ASAGDRTH EQU ASARETRN	ALL: RETURN-CODE
000000						1602	1	ASAGDOK EQU X'00'	ALL: ALLES OK
000004						1603	1	ASAGEFE EQU X'04'	ALL: FORMALER FEHLER
						1604	1	*	- SUBFUNCTION UNGLEICH X'00'
000014						1605	1	ASAGDXXX DS 0X	
						1606	1	*	*
						1607	1	*	DATUM IST IN ZEICHENDARSTELLUNG IN DER FORM:

FLAG LOCIN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

```

1608 1 * YY/MM/TTIIIIB
1609 1 * WOBEI: YY = JAHR
1610 1 * MM = MONAT
1611 1 * TT = TAG
1612 1 * III = JULIANISCHES DATUM
1613 1 * B = BLANK
1614 1 *
000014 F8F161F1F161F2F0 1615 1 ASAGDDAT DC CL12'81/11/20321 ' TAGESDATUM

1616 1 *
1617 1 * DIE TAGESZEIT IST IN ZEICHENDARSTELLUNG IN FOLGENDER
1618 1 * FORM GEGEBEN:
1619 1 * HHMMSS
1620 1 * WOBEI: HH = STUNDE
1621 1 * MM = MINUTE
1622 1 * SS = SEKUNDE
1623 1 *
000020 F1F3F5F4F0F1 1624 1 ASAGDTIM DC CL6'135401' TAGESZEIT

1625 1 *
1626 1 * DIE VERBRAUCHTE CPU-ZEIT WIRD IN ZEICHENDARSTELLUNG
1627 1 * IN FOLGENDER FORM GEGEBEN:
1628 1 * HHMMSS
1629 1 *
000026 F0F0F0F1F4F2 1630 1 ASAGDCPU DC CL6'000142' CPU-ZEIT
000018 1631 1 ASAGDLEN EQU *-ASAGDXXX LAENGE(RUECKKEHRPARAMETER)

1632 1 *****
1633 1 **
1634 1 ** H A R D W A R E - I N F O R M A T I O N **
1635 1 ** ----- **
1636 1 ** DIESER SERVICE IST NUR FUER SLEDFILE-ZUGRIFFE REALISIERT. ER **
1637 1 ** LIEFERT SELEKTIERTE HARDWARE-INFORMATION ZURUECK: **
1638 1 ** - AUDIT ADRESS REGISTER **
1639 1 ** - ALLE CPU - LOGGOUT - REGISTER **
1640 1 ** - ALLE IGC - LOGGOUT - REGISTER **
1641 1 ** DER SERVICE IST AUCH UEBER 'TAM' AUFRUFBAR. **
1642 1 *****
000014 1643 1 BRG ASALABEL

1644 1 *****
1645 1 ** HEADER - PARAMETER **
1646 1 *****
00003C 1647 1 ASAHW EQU X'3C' FUNKTION = HARDWARE-INFORMATION
1648 1 * ES GIBT KEINE SUBFUNKTIONEN. DAS FELD
1649 1 * 'SBFCT' MUSS GLEICH X'00' SEIN.
000000 1650 1 ASAHWMAX EQU X'00' HOECHSTE SUBFUNKTIONSNUMMER

1651 1 *****
1652 1 ** AUFRUF - PARAMETER **
1653 1 *****
000014 00 1654 1 ASAHWLNK DC X'00' SLEDFILE#
    
```

FLAG	LOC TN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT
						1655	1	*****
						1656	1	** RUECKKEHR - PARAMETER **
						1657	1	*****
000013						1658	1	ASAHWRTN EQU ASARETRN
000000						1659	1	ASAHWTK EQU X'00'
000004						1660	1	ASAHWFE EQU X'04'
						1661	1	*
						1662	1	*
000008						1663	1	ASAHWFNG EQU X'08'
						1664	1	*
00000C						1665	1	ASAHWNOS EQU X'0C'
						1666	1	*
000010						1667	1	ASAHWINA EQU X'10'
								ALLES OK; SERVICE AUSGEFUEHRT FORMALER FEHLER: - UNGUELFIGE SUBFUNKTION - LINK# > X'07' ANGEGEBENE DUMPDATEI WURDE NOECH NICHT UEBER DEN OPEN-SERVICE ANGEMELDET DUMPDATEI IST KEINE SLED-DATEI ODER DUMP WURDE AUF FUJITSU-ANLAGE GEZOGEN INFORMATION KONNTE NICHT ERZEUGT WERDEN
000018						1668	1	ASAHWXXX DS 0F
000018	00000000					1669	1	ASAHWAA0 DC A(0) CPU-0-AUDIT-ADRESS-REGISTER
00001C	00000000					1670	1	ASAHWAA1 DC A(0) CPU-1-AUDIT-ADRESS-REGISTER
000020	00000000					1671	1	ASAHWAA2 DC A(0) CPU-2-AUDIT-ADRESS-REGISTER
000024	00000000					1672	1	ASAHWAA3 DC A(0) CPU-3-AUDIT-ADRESS-REGISTER
000028	00000000					1673	1	ASAHWCRO DC A(0) CPU-0-LOGGOUT-REGISTER
00002C	00000000					1674	1	ASAHWCRI DC A(0) CPU-1-LOGGOUT-REGISTER
000030	00000000					1675	1	ASAHWCRI DC A(0) CPU-2-LOGGOUT-REGISTER
000034	00000000					1676	1	ASAHWCRI DC A(0) CPU-3-LOGGOUT-REGISTER
000038	00000000					1677	1	ASAHWIR0 DC A(0) IGC-0-LOGGOUT-REGISTER
00003C	00000000					1678	1	ASAHWIR1 DC A(0) IGC-1-LOGGOUT-REGISTER
000040	00000000					1679	1	ASAHWIR2 DC A(0) IGC-2-LOGGOUT-REGISTER
000044	00000000					1680	1	ASAHWIR3 DC A(0) IGC-3-LOGGOUT-REGISTER
000048	00000000					1681	1	ASAHWIR4 DC A(0) IGC-4-LOGGOUT-REGISTER
00004C	00000000					1682	1	ASAHWIR5 DC A(0) IGC-5-LOGGOUT-REGISTER
000050	00000000					1683	1	ASAHWIR6 DC A(0) IGC-6-LOGGOUT-REGISTER
000054	00000000					1684	1	ASAHWIR7 DC A(0) IGC-7-LOGGOUT-REGISTER
		000040				1685	1	ASAHWLEN EQU *-ASAHWXXX LAENGE DER RETURN-PARAMETER
						1686	1	*****
						1687	1	**
						1688	1	** DATA TRANSPORT **
						1689	1	**
						1690	1	** DIESER SERVICE TRANSPORTIERT ZUSAMMENHAENGENDE SPEICHERBEREICHE **
						1691	1	** VON EINER AUSGANGS-ADRESSE AUF EINE ZIEL-ADRESSE. **
						1692	1	*****
000014						1693	1	GRG ASALABEL
						1694	1	*****
						1695	1	** HEADER - PARAMETER **
						1696	1	*****
000040						1697	1	ASAMC EQU X'40' FUNCTION = DATA MANIPULATION
000000						1698	1	ASAMCM EQU X'00' SUBFUNCTION = MOVE DATA
000000						1699	1	ASAMCMAX EQU X'00' MAX. # OF SUBFUNCTION
						1700	1	*****
						1701	1	** AUFRUF - PARAMETER **
						1702	1	*****

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE	STATEMENT
	000014	0000				1703	1	ASAMCLGH DC	H'0' # OF BYTES TO PROCESS
						1704	1	*	
						1705	1	*	DEF OF SOURCE (S)
						1706	1	*	
	000018					1707	1	ASAMCSRC DS	OF START OF SOURCEPARAMS
	000018	00000000				1708	1	ASAMCSAD DC	A(0) CH ADDR OF S; IF 'ASAMCSKY'='ASAMCSPL'
						1709	1	*	THEN ADDRESS OF BUFFER (WORD-ALIGNED)
	00001C	00000000				1710	1	ASAMCSTS DC	A(0) CH TSN OF S; A(0) = DEFAULTTASK
	000020	00000000				1711	1	ASAMCSP# DC	A(0) CH # OF PAM PAGE, IF ACCESS TO DUMPFIL
	000024	00000000				1712	1	ASAMCSPA DC	A(0) CH REL. ADDRESS WITHIN 1 MB RANGE
	000028	0000				1713	1	ASAMCSIT DC	H'0' CH ITN OF S; X'0000' = DEFAULT-TASK
	00002A	0000				1714	1	ASAMCSER DC	H'0' DSECT-RELATIVE ADDRESS IF MCSST IS SET
	00002C	00				1715	1	ASAMCSLN DC	X'00' LINKNAME OF DUMPFIL, IF MCSST SET
	00002D	00				1716	1	ASAMCSFB DC	X'00' FLAGBYTE FOR S
		000000				1717	1	ASAMCSTN EQU	ASAITN CH NICHT VERWENDET
		000000				1718	1	ASAMCSTV EQU	ASATSN CH NICHT VERWENDET
		000000				1719	1	ASAMCSTI EQU	ASAMCSTV+ASAMCSTN NICHT VERWENDET
		000040				1720	1	ASAMCSPV EQU	X'40' CH MCSF# & MCSFA ARE VALID
		000020				1721	1	ASAMCSPH EQU	X'20' CH 'MCSAD' IST REALE ADRESSE
		000010				1722	1	ASAMCSDT EQU	ASADUMP READ IN DUMPED TASK, ID # IN
		000008				1723	1	ASAMCSTP EQU	X'08' TRANSFORM TESTPOINTS IN S
		000001				1724	1	ASAMCSOQ EQU	X'01' CHECK BIT, NO VALIDATION FOR S
	00002E	00				1725	1	ASAMCSF2 DC	X'00' FLAGBYTE 2; NOT YET USED
	000030					1726	1	DS	0H ALIGNMENT
	000030	00				1727	1	ASAMCSKY DC	X'00' INDICATOR FOR KEYWORDS/REGISTERS
		000000				1728	1	ASAMCSNR EQU	X'00' NO KEYWORD
		000004				1729	1	ASAMCSR EQU	X'04' REGISTER, SEE MCSNR FOR NUMBER
		000008				1730	1	ASAMCSOC EQU	X'08' CONDITION CODE
		00000C				1731	1	ASAMCSPC EQU	X'0C' PROGRAM COUNTER
		000010				1732	1	ASAMCSPI EQU	X'10' PROGRAM MASK
		000014				1733	1	ASAMCSFB EQU	X'14' PROCESS CONTROL BLOCK;SEE MCPNR
		000018				1734	1	ASAMCSPL EQU	X'18' NOT USED
		00001C				1735	1	ASAMCSFR EQU	X'1C' INT. FLAG REGISTER
		000020				1736	1	ASAMCSMR EQU	X'20' INT. MASK REGISTER
		000024				1737	1	ASAMCSBR EQU	X'24' INT. STATUS REGISTER
		000028				1738	1	ASAMCSJC EQU	X'28' JOB CONTROL BLOCK
		00002C				1739	1	ASAMCSJT EQU	X'2C' JOB TO BE PROCESSED BLOCK
		000030				1740	1	ASAMCSIB EQU	X'30' TASK CONTROL BLOCK
		000034				1741	1	ASAMCSFC EQU	X'34' FCB; LINK IS IN ASAMCFLK
		00003E				1742	1	ASAMCSKS EQU	X'3E' LIMIT # FOR TASK-DEPENDAND KEYS
		000040				1743	1	ASAMCSKV EQU	X'40' EXECUTIVE VECTOR TABLE
		000044				1744	1	ASAMCSSC EQU	X'44' SPEICHER-KLASSE; GIB IN 'MCSPI'
						1745	1	*	DIE SPEICHERKLASSE AN
		000048				1746	1	ASAMCSGR EQU	X'48' FLOATING-POINT-REGISTERS
		000048				1747	1	ASAMCSMX EQU	X'48' MAX. # OF KEY
						1748	1	ASAMCSST DC	X'00' INDICATOR FOR SYST.TABLE SYMBOLS
		000000				1749	1	ASAMCSHS EQU	X'00' NO SYMBOL
						1750	1	*	&I.MCSST HOLDS AN IDENTIFICATION# FOR THE PARENT-
						1751	1	*	DSECT (X'00'-X'FF') OF THE SYSTEM-TABLE-SYMBOL
						1752	1	*	IN QUESTION.
		000000				1753	1	ASAMCSSS EQU	X'00' MAXIMUM ID-NUMBER OF PARENT-
						1754	1	*	DSECT OF TASK-INDEPENDAND STS-
						1755	1	*	SYMBOL. DSECTS MUST BE SORTED

FLAG LCTN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

```

1756 1 *
1757 1 *
1758 1 *
1759 1 *
1760 1 *
00001B 1761 1 ASAMCSSX EQU X'1B'
000032 00 1762 1 ASAMCSRH DC X'00'
000010 1763 1 ASAMCSAR EQU X'10'
000033 00 1764 1 ASAMCSFN DC X'00'
1765 1 *
1766 1 *
1767 1 *
000000 1768 1 ASAMCSC1 EQU X'00'
000004 1769 1 ASAMCSC2 EQU X'04'
000003 1770 1 ASAMCSC3 EQU X'08'
00000C 1771 1 ASAMCSC5 EQU X'0C'
000010 1772 1 ASAMCSC6 EQU X'10'
000034 D3C9D5D2D5C1D4C5 1773 1 ASAMCFLK DC CL8'LINKNAME'
```

```

IN A WAY THAT ALL TASK-INDE-
PENDAND TABLES HAVE LOWER NUM-
BERS, WHILE ALL TASK-DEPENDAND
TABLES ARE PUT TOGETHER WITH
HIGHER NUMBERS.
MAX. ALLOWED IDENTIFICATION#
0-15: NR. OF REGISTER
ALL REGISTERS
0 : USER GAVE NO INDEX
N : - N-TER STACK
FALLS 'MOSKY'='MCSSC', DANN MUSS
HIER DIE SPEICHERKLASSE STEHEN.
SPEICHER-KLASSE 1
SPEICHER-KLASSE 2
SPEICHER-KLASSEN 3/4
SPEICHER-KLASSE 5
SPEICHER-KLASSE 6
LINKNAME FOR FCB(ASAMCSFC)
```

```

1774 1 *****
1775 1 ** DEFINITION OF DESTINATION **
1776 1 *****
```

```

00003C 00000000 00003C 1777 1 ASAMCDRC EQU *
1778 1 ASAMCDAD DC A(0)
1779 1 *
000040 00000000 1780 1 ASAMCDTS DC A(0)
000044 00000000 1781 1 ASAMCDP# DC A(0)
000048 00000000 1782 1 ASAMCDPA DC A(0)
00004C 0000 1783 1 ASAMCDIT DC H'0'
00004E 0000 1784 1 ASAMCEDR DC H'0'
000050 00 1785 1 ASAMCDLN DC X'00'
000051 00 1786 1 ASAMCDFB DC X'00'
000001 1787 1 ASAMCDCQ EQU ASAMCSCQ
000000 1788 1 ASAMCDTN EQU ASAMCSTN
000008 1789 1 ASAMCDTP EQU ASAMCSTP
000010 1790 1 ASAMCDDT EQU ASAMCSDT
000020 1791 1 ASAMCDPH EQU ASAMCSPH
000040 1792 1 ASAMCDPV EQU ASAMCSPV
000000 1793 1 ASAMCDTV EQU ASAMCSTV
000000 1794 1 ASAMCDTI EQU ASAMCSTI
000052 00 1795 1 ASAMCDF2 DC X'00'
000054 000054 1796 1 ASAMCDRG EQU ASAMRG
000054 00 1797 1 ASAMCDR# EQU ASAMR#
1798 1 ASAMCEKY DC X'00'
000000 1799 1 ASAMCDNG EQU ASAMCSNG
000004 1800 1 ASAMCDR EQU ASAMCSR
000008 1801 1 ASAMCDCC EQU ASAMCSCC
00000C 1802 1 ASAMCDPC EQU ASAMCSPC
000010 1803 1 ASAMCDPM EQU ASAMCSPM
000014 1804 1 ASAMCDPB EQU ASAMCSPB
00001C 1805 1 ASAMCDPR EQU ASAMCSPR
000020 1806 1 ASAMCDMR EQU ASAMCSMR
```

```

BEGINNING OF DESTINATION PARAMS
ADDR OF D; WILL NOT BE REGARDED, IF
MCDKY OR MCDST IS SET
TSN OF D; A(0) = DEFAULT-TASK
# OF PAM PAGE IF ACCESS TO DUMP FILE
REL. ADDRESS WITHIN 1 MB RANGE
ITN OF D; X'0000' = DEFAULT - TASK
DSECT-RELATIVE ADDRESS, IF MCDST SET
LINKNUMBER OF DUMPFIL, IF MCDDT SET
FLAGBYTE FOR D
CHECK BIT, NO VALIDATION FOR D
NICHT VERWENDET
TRANSFORM TESTPOINTS IN D
READ IN DUMPED TASK, ID # IN 'MCDIT'
'MCDAD' IST REALE ADRESSE
MCDP# & MCDPA ARE VALID
NICHT VERWENDET
NICHT VERWENDET
FLAGBYTE 2
IGNORE READ-ONLY-PROTECTION
ALIGNMENT
INDICATOR FOR KEYWORDS/REGISTERS
NO KEYWORD
REGISTER, SEE MCDNR FOR NUMBER
CONDITION CODE
PROGRAM COUNTER
PROGRAM MASK
PROCESS CONTROL BLOCK;SEE MCPNR
INT. FLAG REGISTER
INT. MASK REGISTER
```

FLAG	LCTN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE	STATEMENT
				000024		1807	1	ASAMCDSR EQU	ASAMCSSR INT. STATUS REGISTER
				000028		1808	1	ASAMCDJC EQU	ASAMCSJC JOB CONTROL BLOCK
				00002C		1809	1	ASAMCDJT EQU	ASAMCSJT JOB TO BE PROCESSED BLOCK
				000030		1810	1	ASAMCDTB EQU	ASAMCSTB TASK CONTROL BLOCK
				000040		1811	1	ASAMCDXV EQU	ASAMCSXV EXECUTIVE VECTOR TABLE
				000048		1812	1	ASAMCDGR EQU	ASAMCSGR FLOATING-POINT-REGISTERS
				000048		1813	1	ASAMCDMX EQU	ASAMCSMX MAX. # OF KEY
000055	00					1814	1	ASAMCDST DC	X'00' INDICATOR FOR SYST.TABLE SYMBOLS
				000000		1815	1	ASAMCDNS EQU	0 NO SYMBOL
000056	00					1816	1	ASAMCDRN DC	X'00' 0-15: NR OF REGISTER
				000010		1817	1	ASAMCDAR EQU	16 ALL REGISTERS
000057	00					1818	1	ASAMCDPN DC	X'00' 0 : CURRENT STACK
						1819	1	*	N : - N-TER PCB
						1820	1	*****	*****
						1821	1	**	RUECKKEHR - PARAMETER **
						1822	1	*****	*****
000013						1823	1	ASAMCRTN EQU	ASARETRN RETURN PARAM
000000						1824	1	ASAMCGK EQU	X'00' ALL G.K.; BUT SEE ASAMCFB
000004						1825	1	ASAMCFE EQU	X'04' FORMALER FEHLER:
						1826	1	*	- SUBFUNKTION UNGLEICH X'00'
						1827	1	*	- LAENGE = X'0000', ABER KEIN
						1828	1	*	KEYWORD GEGEBEN
						1829	1	*	- DUMPPFILE-LINK# > 7
						1830	1	*	- UNGUELTIGE KEY#
						1831	1	*	- SCHREIBEN IN TRACETABLE
						1832	1	*	- UNGUELTIGES REGISTER
000008						1833	1	ASAMCV EQU	X'08' ERROR IN S AND/OR D, SEE MCSER & MCDER
00000C						1834	1	ASAMCDT1 EQU	X'0C' DUMPDATEI EXISTIERT NICHT
000010						1835	1	ASAMCDT2 EQU	X'10' NICHT VERWENDET
000014						1836	1	ASAMCIT1 EQU	X'14' ITN IN S NOT FOUND
000018						1837	1	ASAMCIT2 EQU	X'18' ITN IN D NOT FOUND
00001C						1838	1	ASAMCPH1 EQU	X'1C' INDICATED STACK NOT FOUND IN S
						1839	1	*	OR NOT SAVED
000020						1840	1	ASAMCPN2 EQU	X'20' INDICATED STACK NOT FOUND IN D
000024						1841	1	ASAMCBD1 EQU	X'24' SOURCE BOUNDARIES VIOLATED
000028						1842	1	ASAMCBD2 EQU	X'28' D - BOUNDARIES VIOLATED
00002C						1843	1	ASAMCTS1 EQU	X'2C' TSN IN S NOT FOUND
000030						1844	1	ASAMCTS2 EQU	X'30' TSN IN D NOT FOUND
000034						1845	1	ASAMCTK1 EQU	X'34' NO TASK SPECIFIED FOR S
000038						1846	1	ASAMCTK2 EQU	X'38' NO TASK SPECIFIED FOR D
00003C						1847	1	ASAMCPH1 EQU	X'3C' ZUGRIFF UEBER EINE REALE ...
000040						1848	1	ASAMCPH2 EQU	X'40' ... ADRESSE IST NICHT MOEGLICH
000044						1849	1	ASAMCN4 EQU	X'44' NO CLASS4 MEMORY AVAILABLE; ACCESS
						1850	1	*	TO SYSTEM-ADDRESS IMPOSSIBLE
000048						1851	1	ASAMCCM EQU	X'48' UEBERTRAGUNG ZWISCHEN 2 FREMDEN TASKS
						1852	1	*	IST NICHT ERLAUBT
00004C						1853	1	ASAMCSEM EQU	X'4C' ANOTHER TASK ACCESSES SAME PAGE
						1854	1	*	YOU MAY RETRY LATER
000050						1855	1	ASAMCCM0 EQU	X'50' ACCESS OF DIFFERENT SEGMENTS OF
						1856	1	*	FOREIGN TASKS NOT ALLOWED
000054						1857	1	ASAMCSCH EQU	X'54' FGN ADDR SPACE CANNOT BE ACCESSED
000058						1858	1	ASAMCKNE EQU	X'58' SPECIFIED TABLE DOES NOT EXIST OR
						1859	1	*	NOT ACCESSABLE OR NOT SAVED

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMT	M	SOURCE	STATEMENT	
				00005C		1860	1	ASAMCLNA EQU	X'5C'	ILLEGAL WRITE-ACCESS TO READ-ONLY PAGE
						1861	1	*		
				000060		1862	1	ASAMCHYI EQU	X'60'	FUNCTION NOT YET IMPLEMENTED
				000064		1863	1	ASAMCSDA EQU	X'64'	ALLOCATION ERROR IN S/D
				000068		1864	1	ASAMCTNM EQU	X'68'	NO MEMORY FOR TRACETABLE
				00006C		1865	1	ASAMCNLS EQU	X'6C'	FCB REQUESTED WITHOUT SPECIFYING LINK-NAME
						1866	1	*		
				000070		1867	1	ASAMCFNG EQU	X'70'	FILE WITH SPECIFIED LINKNAME NOT OPEN (FCB REQUESTED)
						1868	1	*		
				000074		1869	1	ASAMCMCH EQU	X'74'	ANGEGEBENE SPEICHERKLASSE EXISTIERT NICHT
						1870	1	*		
				000074		1871	1	ASAMCPL EQU	X'74'	MAXIMUM VALUE OF RETURN-CODE
000058						1872	1	ASAMCXXX DS	0X	
000058	00					1873	1	ASAMCFB DC	X'00'	FLAGBYTE
				000080		1874	1	ASAMCLA EQU	X'80'	GIVEN LENGTH VIOLATED S- OR D- BOUNDARIES; LENGTH HAS BEEN ADJUSTED ACCORDINGLY
						1875	1	*		
						1876	1	*		
				000040		1877	1	ASAMCINF EQU	X'40'	NO LENGTH GIVEN; INFORM ONLY
						1878	1	*		SEE MCSAD, MCDAD, MCLGH FOR RETURN
				000020		1879	1	ASAMCHC1 EQU	X'20'	TOO MANY ENTRIES IN SAVEMOVE; D COULD NOT BE SAVED
						1880	1	*		
				000010		1881	1	ASAMCHC2 EQU	X'10'	NO MEMORY FOR COPY OF D; D COULD NOT BE SAVED
						1882	1	*		
				000008		1883	1	ASAMCSEG EQU	X'08'	MOVE INTO/OUT OF FOREIGN TASK VIOLATES SEGMENT-BOUNDARY; ONLY 'MCTMP' BYTES MOVED; REPEAT NEW ADDRESS
						1884	1	*		
						1885	1	*		
						1886	1	*		
				000004		1887	1	ASAMCQUL EQU	ASAUQUAL	UEBERQUALIFIZIERTE EINGABE
000059	00					1888	1	ASAMCCC DC	X'00'	CONDITION CODE, (NOT YET IMPLEMENTED)
				000080		1889	1	ASAMCEQ EQU	X'80'	S = D) USEFUL AS MASK
				000040		1890	1	ASAMCLT EQU	X'40'	S < D > FOR A EX GN A BC
				000020		1891	1	ASAMCGT EQU	X'20'	S > D) OR A BCR
00005A						1892	1	ASAMCERR DS	0H	
00005A	00					1893	1	ASAMCSER DC	X'00'	ERROR FLAGBYTE FOR SOURCE
				000080		1894	1	ASAMCSNA EQU	X'80'	PAGE NOT DUMPED, COULD NOT BE ACCESSED
				000040		1895	1	ASAMCSOB EQU	X'40'	PAGE NOT DUMPED, NOT WITHIN LIMITS OF DUMP
						1896	1	*		
				000020		1897	1	ASAMCSNE EQU	X'20'	PAGE NOT ALLOCATED
				000010		1898	1	ASAMCSDP EQU	X'10'	DUMP PRIVILEGE TOO SMALL, PAGE NOT DUMPED
						1899	1	*		
				000001		1900	1	ASAMCSPR EQU	X'01'	TESTPRIV TOO SMALL
				000002		1901	1	ASAMCSPG EQU	X'02'	TESTPRIV IN ANY CASE TOO SMALL
00005B	00					1902	1	ASAMCDBR DC	X'00'	ERROR FLAGBYTE FOR D
				000080		1903	1	ASAMCDNA EQU	ASAMCSNA	PAGE NOT DUMPED, COULD NOT BE ACCESSED
				000040		1904	1	ASAMCDOB EQU	ASAMCSOB	PAGE NOT DUMPED, NOT WITHIN LIMITS OF DUMP
						1905	1	*		
				000020		1906	1	ASAMCDNE EQU	ASAMCSNE	PAGE NOT ALLOCATED
				000010		1907	1	ASAMCDBP EQU	ASAMCSDP	DUMP PRIVILEGE TOO SMALL, PAGE NOT DUMPED
						1908	1	*		
				000001		1909	1	ASAMCDPR EQU	ASAMCSPR	TESTPRIV TOO SMALL
				000002		1910	1	ASAMCDPG EQU	ASAMCSPG	TESTPRIV IN ANY CASE TOO SMALL
				000004		1911	1	ASAMCDBR EQU	X'04'	WRITE INTO DUMP-FILE

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMT	M	SOURCE	STATEMENT
00005C	00000000					1912	1	ASAMCBYT DC	A(0) ANZAHL DER BEARBEITETEN BYTES
				00005E		1913	1	ORG	ASAMCBYT+2
00005E	0000					1914	1	ASAMCTNP DC	H'0'
						1915	1	*	# OF TILL NOW PROCESSED BYTES
						1916	1	*	DIESES FELD IST NUR NOCH AUS
000060	00					1917	1	ASAMCSSB DC	X'00'
000061	00					1918	1	ASAMCDSB DC	X'00'
000062	00					1919	1	ASAMCTYP DC	X'00'
000063	00					1920	1	ASAMCPND DC	X'00'
000064	0000					1921	1	ASAMCO DC	H'0'
000065	00000000					1922	1	ASAMCTCB DC	A(0)
				000014		1923	1	ASAMCLEN EQU	*-ASAMCXXX LENGTH(RET.PARAMS)
						1924	1	*****	*****
						1925	1	**	**
						1926	1	**	ZUGRIFF AUF AID-TEXTDATEI **
						1927	1	**	----- **
						1928	1	**	ZUGRIFF AUF VOM AUFRUFER ANGEGEBENE DATEI UEBER EINEN ISAM- **
						1929	1	**	SATZSCHLUESSEL. **
						1930	1	**	**
						1931	1	*****	*****
				000014		1932	1	ORG	ASALABEL
						1933	1	*****	*****
						1934	1	**	HEADER - INFORMATION **
						1935	1	*****	*****
				000044		1936	1	ASAFP EQU	X'44' FUNCTION = ACCESS AID-TEXT-DATEI
				000000		1937	1	ASAFP MAX EQU	X'00' MAXIMUM SUBFUNCTION #
						1938	1	*****	*****
						1939	1	**	AUFRUF - PARAMETER **
						1940	1	*****	*****
000014	0000					1941	1	ASAHPLSC DC	Y(0) LENGTH OF SEARCH-CRITERIA (VALUES
						1942	1	*	1 - 155 ALLOWED ONLY)
000016	C3D9C9E3C5D9C9C1					1943	1	ASAH PSC DC	'CRITERIA' SEARCH-CRITERIA
000020	00000000					1944	1	ASAH PABA DC	A(0) ADDRESS OF USER BUFFER
000024	0000					1945	1	ASAH PLBA DC	Y(0) LENGTH OF USER BUFFER
000026	C6C9D3C5D5C1D4C5					1946	1	ASAH PNAM DC	CL17 'FILENAME' NAME OF HELPPFILE
						1947	1	*****	*****
						1948	1	**	RUECKKEHR - PARAMETER **
						1949	1	*****	*****
				000013		1950	1	ASAH PRN EQU	ASARETRN RETURN-CODE
				000000		1951	1	ASAH POK EQU	X'00' RETURN G.K.
				000000		1952	1	ASAH PEGI EQU	X'00' RETURN G.K. END OF INFO
				000004		1953	1	ASAH PFE EQU	X'04' FORMALER FEHLER:
						1954	1	*	- CALLER IS NOT AID
						1955	1	*	- ANY INPUT EQUALS TO ZERO
						1956	1	*	- A(BUFFER) IS NOT ALLGATED
						1957	1	*	- BUFFER NOT WORD-ALIGNED
						1958	1	*	- LENGTH(SEARCH-CRITERIA) > 255
						1959	1	*	- UNGUELTIGE SUBFUNKTION
				000008		1960	1	ASAH PNIA EQU	X'08' RETURN NO INFO AVAIL
				00000C		1961	1	ASAH PFNO EQU	X'0C' FILE COULD NOT BE OPENED
				000010		1962	1	ASAH PTAP EQU	X'10' BANDDATEIZUGRIFF NICHT ERLAUBT

FLAG	LOC TN	OBJECT CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT
000037					1963	1	ASAHXXXX DS 0X
000037	0000				1964	1	ASAHXERR DC XL2'00'
			000002		1965	1	ASAHXLEN EQU *-ASAHXXXX DMS-ERRORCODE, FALLS RTN = 'HPFN0'
					1966	1	*****LENGTH OF RETURN-PARAMS*****
					1967	1	**
					1968	1	** GET TERMINAL INFORMATION **
					1969	1	** ----- **
					1970	1	**
			000014		1971	1	*****
					1972	1	ORG ASALABEL
					1973	1	*****
					1974	1	** HEADER - PARAMETER **
			000048		1975	1	*****
			000000		1976	1	ASATM EQU X'48' FUNCTION = GET TERMINAL INFORMATION
					1977	1	ASATMMAX EQU X'00' MAX. # OF SUBFUNCTION
					1978	1	*****
					1979	1	** AUFRUF - PARAMETER **
					1980	1	*****
					1981	1	*
					1982	1	* ----- K E I N E ----- *
					1983	1	*
					1984	1	*****
					1985	1	** RUECKKEHR - PARAMETER **
			000013		1986	1	*****
			000000		1987	1	ASATMRTN EQU ASARETRN RETURN-CODE
			000004		1988	1	ASATMSGK EQU X'00' REQUEST HONORED
					1989	1	ASATMFE EQU X'04' FORMALER FEHLER
					1990	1	* - AUFRUFER IST NICHT 'AID'
					1991	1	* - UNGUELTIGE SUBFUNKTION
000014					1992	1	ASATMXXX DS 0X
					1993	1	*****
					1994	1	* IF ANY ONE OF THE FOLLOWING ELEMENTS IS X'00', THEN *
					1995	1	* THE CORRESPONDING DEVICE IS NOT AVAILABLE. *
					1996	1	*****
000014	00				1997	1	ASATMTLL DC X'00' TERMINAL LINE LENGTH
000015	00				1998	1	ASATMHCL DC X'00' HARDCOPY LINE LENGTH
000016	00				1999	1	ASATMPLL DC X'00' PRINTER LINE LENGTH
000018	00000000				2000	1	ASATMTSN DC A(0) TSN OF TASK IN CONTROL
00001C	0000				2001	1	ASATMITN DC Y(0) ITN OF TASK IN CONTROL
		00000A			2002	1	ASATMLEN EQU *-ASATMXXX LENGTH OF RETURN PARAMS
					2003	1	*****
					2004	1	**
					2005	1	** A I D S Y S D E S C R I P T O R S E R V I C E **
					2006	1	** ----- **
					2007	1	**
					2008	1	** THIS FUNCTION SATISFIES ALL OF THE FOLLOWING AIDSYS- **
					2009	1	** DESCRIPTOR REQUESTS: **
					2010	1	**
					2011	1	** - CREATION OF AN AIDSYS-DESCRIPTOR TO A SPECIFIED **

FLAG	LOC TN	OBJECT CODE	ADDR1	ADDR2	STMT	M	SOURCE STATEMENT
					2012	1	** LEVEL FOR A GIVEN NAMED ITEM. THE POSSIBLE LEVELS **
					2013	1	** IN DESCENDING ORDER ARE: **
					2014	1	**
					2015	1	** LOAD MODULE (= ADDRESS OF LMIR) **
					2016	1	** SOURCE MODULE (= ADDRESS OF SMREC) **
					2017	1	** OBJECT MODULE (= ADDRESS OF MIREC) **
					2018	1	** CSECT (= ADDRESS OF CSREC); **
					2019	1	**
					2020	1	** - CREATION OF A COMPLETE AIDSYS-DESCRIPTOR FOR A **
					2021	1	** GIVEN VIRTUAL ADDRESS; AND **
					2022	1	**
					2023	1	** - DESCRIBE THE ENVIRONMENT OF A GIVEN AIDSYS- **
					2024	1	** DESCRIPTOR IN THE FOLLOWING FORM: **
					2025	1	**
					2026	1	** NAME OF THE PROGRAMM (FROM PCLST) **
					2027	1	** PROGRAMM LOAD ADDRESS (FROM PCLST) **
					2028	1	** NAME OF THE LOAD MODULE (FROM LMIR) **
					2029	1	** RELATIVE LOAD ADDRESS OF THE LM (FROM LMIR) **
					2030	1	** NAME OF THE SOURCE MODULE (FROM SMREC) **
					2031	1	** ADDRESS OF THE FIRST GM IN THE SM (FROM MIREC) **
					2032	1	** NAME OF THE OBJECT MODULE (FROM MIREC) **
					2033	1	** RELATIVE LOAD ADDRESS OF THE GM (FROM MIREC) **
					2034	1	** NAME OF THE CSECT (FROM CSREC) **
					2035	1	** RELATIVE ADDRESS OF THE CSECT (FROM CSREC) **
					2036	1	** INDICATOR AS TO WHETHER LSD OR ISD INFORMATION **
					2037	1	** IS AVAILABLE **
					2038	1	**
					2039	1	*****
000014					2040	1	ORG ASALABEL
					2041	1	* UNIVERSAL INPUT PARAMETERS
					2042	1	*
00004C					2043	1	ASADS EQU X'4C' FUNCTION: DESCRIPTOR SERVICES
000004					2044	1	ASADSCFN EQU X'04' SUBFCT: CREATE AD FROM NAME
000008					2045	1	ASADSCFA EQU X'08' SUBFCT: CREATE AD FROM VIRT. ADDR.
00000C					2046	1	ASADSENV EQU X'0C' SUBFCT: DESCRIBE THE AD ENVIRONMENT
000010					2047	1	ASADSDPG EQU X'10' SUBFCT: LIEFERE MAP-INFORMATION
000010					2048	1	ASADSMAX EQU X'10' MAX. # OF SUBFUNCTION
000014	40404040				2049	1	ASADSTSN DC CL4' ' ALL: TSN OF ACCESSED TASK; A(0)=DEFAULT
000018	0000				2050	1	ASADSITH DC XL2'0000' ALL: ITN OF ACCESSED TASK; X'00'=DEFAULT
00001A	00				2051	1	ASADSDFN DC XL1'00' ALL: DUMP FILE LINK NR OF ACCESSED TASK
00001B	00				2052	1	ASADSBUSE DC X'00' ALL: USAGE INDICATOR; VALUES CAN BE
000001					2053	1	ASADSBUDU EQU X'01' ALL: DOMAINE = USER
000002					2054	1	ASADSBUDS EQU X'02' ALL: DOMAINE = SYSTEM
000000					2055	1	ASADSBUTH EQU ASATSN NICHT VERWENDET
000000					2056	1	ASADSBUTH EQU ASAITH NICHT VERWENDET
000010					2057	1	ASADSBUDF EQU ASADUMP ALL: DUMPFILERE ACCESS IS REQUIRED
000020					2058	1	ASADSSCL EQU X'20' DPG: SUPPLEMENTARY CALL
					2059	1	*****
					2060	1	** UNIVERSAL INPUT PARAMETERS WHICH MAY BE ALTERED (SUBFCTS CREATE)*
					2061	1	*****
00001C					2062	1	DS OF
00001C					2063	1	ASADSD DS OCL16 ENV+DPG: AIDSYS-DESCRIPTOR (MUST BE ALL

FLAG LCTN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

```

00001C 00000000      2064 1 * ZEROS FOR SUBJECTS 'CFN' & 'CFA'
000020 00000000      2065 1 ASADSADL DC A(0) ENV+DPG: ADDRESS OF LMIR
000024 00000000      2066 1 ASADSADS DC A(0) ENV+DPG: ADDRESS OF SMREC
000028 00000000      2067 1 ASADSADQ DC A(0) ENV+DPG: ADDRESS OF MIREC
000028 00000000      2068 1 ASADSADC DC A(0) ENV+DPG: ADDRESS OF CSREC

2059 1 *****
2070 1 ** UNIVERSAL OUTPUT PARAMETERS **
2071 1 *****
000013 2072 1 ASADSRTH EQU ASARETRN RETURN CODES
2073 1 *
000000 2074 1 ASADSGK EQU X'00' ALL: REQUEST SATISFIED
000004 2075 1 ASADSPF EQU X'04' ALL: FORMAL ERROR
2076 1 *
2077 1 * - CALLER IS NOT AID
2078 1 * - INVALID SUBFUNCTION-CODE
2079 1 * - ONE OF INPUT-ADDRESSES IS NOT
2080 1 * ALLOCATED
2081 1 * - ONE OF 'RECORDPOINTERS' DOES NOT
2082 1 * POINT TO INDICATED RECORD
2083 1 * - BOTH DOMAINE-USER & DOMAINE-
2084 1 * =SYSTEM ARE SPECIFIED
2085 1 * - DOMAINE-SYSTEM & LEVEL<>CSECT
2086 1 * - DESCRIPTOR<>0 & SUBJECT='CFN'
2087 1 * OR SUBJECT='CFA'
000008 2087 1 ASADSERN EQU X'08' CFN: THE NAMED ITEM WAS NOT FOUND
00000C 2088 1 ASADSERA EQU X'0C' CFA: NO MATCH FOUND FOR VIRTUAL ADDRESS
000010 2089 1 ASADSEAD EQU X'10' ENV+DPG: THE AIDSYS-DESCRIPTOR IS INCORRECT
000014 2090 1 ASADSNPG EQU X'14' ALL: NO PROGRAM LOADED
000018 2091 1 ASADSSNA EQU X'18' ALL: - BUFFER NOT CLASS5 OR 6
2092 1 * - NO BUFFER SPECIFIED THOUGH
2093 1 * SUBJECT=ASADSDPG
00001C 2094 1 ASADSEGF EQU X'1C' ALL: OLD FORMAT OF LOAD-INFORMATION
000020 2095 1 ASADSPFG EQU X'20' ALL: SPECIFIED FILE NOT OPEN
000024 2096 1 ASADSCNF EQU X'24' CFA: INPUT-ADDRESS IS IN CONFLICT WITH
2097 1 * SPECIFIED (OR DEFAULT) DOMAINE
2098 1 * ALL: SPECIFIED TASK NOT FOUND
000028 2098 1 ASADSTNF EQU X'28' ALL: ERROR IN SYSTEM
00002C 2099 1 ASADSSYS EQU X'2C' DPG: NOT YET IMPLEMENTED
000030 2100 1 ASADSNYI EQU X'30' DIE GEWUNSCHETE INFORMATION IST
000034 2101 1 ASADSNIF EQU X'34' NICHT IM DUMPPFILE-UMFANG ENTHALTEN
2102 1 * TESTPRIV IST ZU KLEIN, KANN ABER
000038 2103 1 ASADSPV1 EQU X'38' ERHOEHT WERDEN.
2104 1 * TESTPRIV IST ZU KLEIN UND KANN NICHT
00003C 2105 1 ASADSPV2 EQU X'3C' MEHR GENUEGEND ERHOEHT WERDEN
2106 1 * DPG: ANGEBEBENER LEVEL IST NICHT ERLAUBT
000040 2107 1 ASADSSLK EQU X'40' ALL: NO CSECT INFO AVAILABLE
000044 2108 1 ASADSNCI EQU X'44' ALL: PAGE NOT DUMPED (REAL SLEDFILE)
000048 2109 1 ASADSPND EQU X'48' END OF UNIVERSAL PARAMETERS
00002C 2110 1 ASADSSXX ES 0CL1

```

```

2111 1 *****
2112 1 *
2113 1 * SUBFUNCTION: CREATION OF AN AIDSYS-DESCRIPTOR TO A *

```

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT
						2114	1	* SPECIFIED LEVEL FOR A GIVEN NAMED ITEM *
						2115	1	* *
						2116	1	*****
			00002C			2117	1	GRG ASADSXXX
						2118	1	* *
						2119	1	* INPUT PARAMETERS: PARTIAL AIDSYS-DESCRIPTOR (&I.DSAD)
						2120	1	* *
00002C	C9E3C5D4D5C1D4C5					2121	1	ASADSNAM DC CL8'ITEMNAME' NAME OF THE ITEM
000034	00					2122	1	ASADSLEV DC XL1'00' LEVEL OF THE ITEM, THAT IS:
						2123	1	* *
			000004			2124	1	ASADSLLM EQU X'04' - LOAD MODULE
			000008			2125	1	ASADSLSM EQU X'08' - SOURCE MODULE
			00000C			2126	1	ASADSLGM EQU X'0C' - OBJECT MODULE
			000010			2127	1	ASADSLCS EQU X'10' - CSECT
						2128	1	* *
						2129	1	* OUTPUT PARAMETERS: AIDSYS-DESCRIPTOR (SEE &I.DSAD)
						2130	1	* RETURN CODES (SEE &I.DSRTN)
						2131	1	* *
						2132	1	*****
						2133	1	* *
						2134	1	* SUBFUNCTION: CREATION OF A COMPLETE AIDSYS-DESCRIPTOR FOR A *
						2135	1	* GIVEN VIRTUAL ADDRESS *
						2136	1	* *
			00002C			2137	1	*****
						2138	1	GRG ASADSXXX
						2139	1	* *
						2140	1	* INPUT PARAMETERS
						2141	1	* *
00002C	00000000					2142	1	ASADSADR DC A(0) VIRTUAL ADDRESS FOR WHICH AN AIDSYS
						2143	1	* DESCRIPTOR IS REQUESTED
						2144	1	* *
						2145	1	* OUTPUT PARAMETERS: AIDSYS-DESCRIPTOR (SEE &I.DSAD)
						2146	1	* RETURN CODES (SEE &I.DSRTN)
						2147	1	* *
000030	00					2148	1	ASADSDGM DC X'00' DOMAIN-INDICATOR
			000001			2149	1	ASADSDUS EQU ASADSUDU DOMAIN = USER
			000002			2150	1	ASADSDSY EQU ASADSUDS DOMAIN = SYSTEM
						2151	1	*****
						2152	1	* *
						2153	1	* SUBFUNCTION: DESCRIBE THE ENVIRONMENT OF A GIVEN AIDSYS- *
						2154	1	* DESCRIPTOR *
						2155	1	* *
			00002C			2156	1	*****
						2157	1	GRG ASADSXXX
						2158	1	* *
						2159	1	* INPUT PARAMETERS: AIDSYS-DESCRIPTOR (SEE &I.DSAD)
						2160	1	* *

FLAG LCTN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

			2161	1	*				
			2162	1	*		OUTPUT PARAMETERS:		RETURN CODES (SEE &I.DSRTN)
			2163	1	*				
00002C			2164	1	ASADSEB	DS	0L65		ENVIRONMENT BLOCK
00002C	D7D9D6C7D5C1D4C5		2165	1	ASADSPRN	DC	CL8'PRGNAME'		NAME OF THE PROGRAMM
000034	00000000		2166	1	ASADSPRA	DC	A(0)		INITIAL LOAD ADDRESS OF THE PROGRAMM
000038	D3D4D6C4D5C1D4C5		2167	1	ASADSLMN	DC	CL8'LMODNAME'		NAME OF THE LOAD MODULE
000040	00000000		2168	1	ASADSLMA	DC	A(0)		RELATIVE LOAD ADDRESS THEREOF
000044	E2D4D6C4D5C1D4C5		2169	1	ASADSSMN	DC	CL8'SMODNAME'		NAME OF THE SOURCE MODULE
00004C	00000000		2170	1	ASADSSMA	DC	A(0)		RELATIVE LOAD ADDRESS OF FIRST SM
000050	D6D4D6C4D5C1D4C5		2171	1	ASADSSMN	DC	CL8'GMODNAME'		NAME OF THE OBJECT MODULE
000058	00000000		2172	1	ASADSSMA	DC	A(0)		RELATIVE LOAD ADDRESS THEREOF
00005C	C3E2C5C3D5C1D4C5		2173	1	ASADSCSN	DC	CL8'CSECTNAME'		NAME OF THE CSECT
000064	00000000		2174	1	ASADSCSA	DC	A(0)		RELATIVE LOAD ADDRESS THEREOF
000068	00000000		2175	1	ASADSABA	DC	A(0)		ADDRESS OF THE FIRST A-BLOCK
00006C	00		2176	1	ASADSIND	DC	XL1'00'		INDICATOR ISD/LSD INFORMATION AVAIL
			2177	1	*				
		000004	2178	1	ASADSIISD	EQU	X'04'		- MEANING ISD INFORMATION IS AVAIL
		000008	2179	1	ASADSLSD	EQU	X'08'		- MEANING LSD INFORMATION IS AVAIL
000070	00000000		2180	1	ASADSPPL	DC	A(0)		PROGRAMM-LAENGE
000074	00000000		2181	1	ASADSLLL	DC	A(0)		LADEMÖDUL-LAENGE
000078	00000000		2182	1	ASADSSLL	DC	A(0)		SOURCE-MÖDUL-LAENGE
00007C	00000000		2183	1	ASADSSLL	DC	A(0)		OBJEKT-MÖDUL-LAENGE
000080	00000000		2184	1	ASADSECL	DC	A(0)		CSECT-LAENGE
		00002C	2185	1	GRG		ASADSXXX		

			2186	1	*****				
			2187	1	*				*
			2188	1	*		SUBFUNKTION: LIEFERE MAP - INFORMATION		*
			2189	1	*				*
			2190	1	*****				
00002C	00000000		2191	1	ASADSAUB	DC	A(0)		START ADDRESS OF BUFFER AREA
000030	0000		2192	1	ASADSLUB	DC	Y(0)		LENGTH OF BUFFER AREA
		000034	2193	1	ASADSKLV	EQU	ASADSLEV		LEVEL OF THE ITEM: I. E.
		000004	2194	1	ASADSKLM	EQU	ASADSKLM		LOAD MODULE LEVEL
		000008	2195	1	ASADSKSM	EQU	ASADSKSM		SOURCE MODULE LEVEL
		00000C	2196	1	ASADSKSM	EQU	ASADSKSM		OBJECT MODULE LEVEL
		000010	2197	1	ASADSKCS	EQU	ASADSKCS		CSECT LEVEL

			2198	1	*				THE FOLLOWING EQUATES DESCRIBE THE STRUCTURE OF
			2199	1	*				ONE RECORD RETURNED IN THE BUFFER
			2200	1	*				!!!! THIS EQU MUST CORRESPOND TO LENGTH !!!!
			2201	1	*				!!!! OF FIELDS DESCRIBED WITHIN 'ARBLK' !!!!
			2202	1	*				!!!! DSECT DEFINED IN MODULE AIDSYS05. !!!!
			2203	1	*				
		000000	2204	1	ASADSBNM	EQU	X'00'		OFFSET: NAME OF CSECT/LOADMODULE/
			2205	1	*				OBJECTMODULE
000008			2206	1	ASADSBAD	EQU	X'08'		OFFSET: A(ADDRESS OF OBJECT)
00000C			2207	1	ASADSBLN	EQU	X'0C'		OFFSET: A(LENGTH OF OBJECT)
000010			2208	1	ASADSETP	EQU	X'10'		OFFSET FUER 13-BYTE-ETPND
000020			2209	1	ASADSEBL	EQU	X'20'		OFFSET: LENGTH(RECORD)
000084			2210	1	GRG				

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMT	M	SOURCE STATEMENT
						2211	1	*****
						2212	1	**
						2213	1	ALLGEMEINE WEITERE RUECKKEHR - PARAMETER **
000084	00					2214	1	*****
				000004		2215	1	ASADSFBR DC X'00' ALL: ANZEIGENFELD
				000008		2216	1	ASADSQUL EQU ASAUQUAL UEBERQUALIFIZIERTE EINGABE
				000059		2217	1	ASADSMGR EQU X'08' BUFFER OVERFLOW; FURTHER CALL NECESS
						2217	1	ASADSLRN EQU *-ASADSXXX LENGTH OF SUBFUNCTION PARAMETERS
						2218	1	*****
						2219	1	**
						2220	1	**
						2221	1	GET ADDRESS **
						2222	1	**
						2222	1	** ZUGRIFF AUF DIE 'EGLDTAB'. LIEFERT ZU VORGEGEBENER VIRUELLER **
						2223	1	** ADRESSE EINEN MODULNAMEN UND EINE MODULRELATIVE ADRESSE. **
						2224	1	** LIEFERT BEI VORGEGEBENEM MODULNAMEN DIE ZUGEHORIGE MODUL- **
						2225	1	** ANFANGSADRESSE. **
						2226	1	**
						2227	1	*****
				000014		2228	1	ORG ASALABEL
						2229	1	*****
						2230	1	**
						2231	1	HEADER - PARAMETER **
				000050		2232	1	*****
				000000		2233	1	ASAGA EQU X'50' ALL: FUNCTION = GET ADDRESS
				000004		2234	1	ASAGAEG EQU X'00' EG: SUBFCT = ACCESS EGLDTAB BY NAME
				000004		2235	1	ASAGAEGA EQU X'04' EGA: SUBFCT = ACCESS EGLDTAB BY ADDRESS
						2235	1	ASAGANAX EQU X'04' MAX# OF SUBFUNCTION
						2236	1	*****
						2237	1	**
						2238	1	AUFRUF - PARAMETER **
						2239	1	*****
000014	00000000					2239	1	ASAGAADD DC A(0) EGA: ADDRESS FOR SEARCH IN EGLDTAB
				000014		2240	1	ORG ASAGAADD
000014	E2E8D4C2D6D34040					2241	1	ASAGANME DC CL8'SYMBOL' EG: SYMBOLIC NAME; MUST BE FILLED UP
						2242	1	** WITH BLANKS IF LESS THAN 8 CHARS. **
00001C	00					2243	1	ASAGAFIL DC X'00' ALL: LINKNAME OF DUMPFIL
00001D	00					2244	1	ASAGAFB DC X'00' ALL: INDICATOR
				000010		2245	1	ASAGAFBF EQU ASADUMP ALL: ACCESS DUMPFIL
						2246	1	*****
						2247	1	**
						2248	1	RUECKKEHR - PARAMETER **
						2249	1	*****
				000013		2249	1	ASAGARTN EQU ASARETRN ALL: RETURN CODE
				000000		2250	1	ASAGASK EQU X'00' ALL: REQUEST HONGURED
				000004		2251	1	ASAGAFE EQU X'04' ALL: FORMALER FEHLER
						2252	1	**
						2253	1	- UNGUELTIGE SUBFUNKTION
						2254	1	**
						2254	1	- UNGUELTIGE DUMPFIL LINK#
						2255	1	**
				000008		2255	1	ASAGANOE EQU X'08' ALL: EGLDTAB NOT FOUND
				00000C		2256	1	ASAGAFNG EQU X'0C' ALL: SPECIFIED FILE NOT OPEN
				000010		2257	1	ASAGANNF EQU X'10' EG: SPECIFIED NAME NOT FOUND

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE	STATEMENT	
				000014		2258	1	ASAGAANF EQU	X'14'	EGA: NO MODULE FOUND FOR SPECIFIED LOCATION
000020						2259	1	ASAGAXXX DS	OF	
000020	00000000					2260	1	ASAGAADR DC	A(0)	EG: ADDRESS OF SPECIFIED SYMBOL
						2261	1	*		EGA: RELATIVE ADDRESS OF SPECIFIED
						2262	1	*		LOCATION WITHIN CSECT
000024	C3E2C5C3E3404040					2263	1	ASAGAMGD DC	CL8'CSECT'	EGA: NAME OF CSECT THAT CONTAINS
						2264	1	*		SPECIFIED LOCATION
00002C	00000000					2265	1	ASAGAENT DC	A(0)	ALL: POINTER TO ENTRY IN EQLDTAB
				000010		2266	1	ASAGALEH EQU	*-ASAGAXXX	LENGTH(RETURN-PARAMS)
						2267	1	*****		
						2268	1	**		**
						2269	1	**	I N F O R M	**
						2270	1	**	-----	**
						2271	1	**	DIESER SERVICE LIEFERT VERSCHIEDENE TASK- ODER SYSTEM-BEZUGENE	**
						2272	1	**	INFORMATIONEN:	**
						2273	1	**	- ALLE FILE CONTROL BLOCKS EINES TASKS	**
						2274	1	**	- ALLE PROCESS CONTROL BLOCKS EINES TASKS	**
						2275	1	**	- ALLE TESTPUNKTE (TASK OR SYSTEM) DES ANGEGEBENEN TASKS	**
						2276	1	**	- ALLE AID/IDA-REPS	**
						2277	1	**	- ALLE TASKS GN DUMPDATEI	**
						2278	1	**		**
						2279	1	*****		
				000014		2280	1	GRG	ASALABEL	
						2281	1	*****		
						2282	1	**	HEADER - PARAMETER	**
						2283	1	*****		
000054						2284	1	ASAI2 EQU	X'54'	ALL: FUNCTION = INFORM
000000						2285	1	ASAI2FCB EQU	X'00'	FCB: SUBFUNCTION = GET ALL FCB'S
000004						2286	1	ASAI2PCB EQU	X'04'	PCB: SUBFUNCTION = GET ALL PCB'S
000008						2287	1	ASAI2CSC EQU	X'08'	UNUSED
00000C						2288	1	ASAI2TPT EQU	X'0C'	TPT: SUBFUNCTION = GET ALL TESTPOINTS
000010						2289	1	ASAI2MVE EQU	X'10'	MVE: SUBFUNCTION = GET ALL SYSTEM-MOVES
000014						2290	1	ASAI2DMP EQU	X'14'	DMP: SUBFUNCTION = ALLE TASKS VON DUMPDATEI
000014						2291	1	ASAI2MAX EQU	X'14'	MAXIMUM # OF SUBFUNCTION
						2292	1	*****		
						2293	1	**	AUFRUF - PARAMETER	**
						2294	1	*****		
000014	0000					2295	1	ASAI2ITN DC	H'0'	ALL: ITN; X'0000' BEDEUTET DEFAULT-TASK
000018	00000000					2296	1	ASAI2TSN DC	A(0)	ALL: TSN; A(0) BEDEUTET DEN DEFAULT-TASK
00001C	00000000					2297	1	ASAI2ADR DC	A(0)	ALL: A(BUFFER), INTO WHICH AIDSYS HAS TO
						2298	1	*		RETURN THE REQUESTED INFORMATION. THE
						2299	1	*		END OF THE LIST WILL BE MARKED BY
						2300	1	*		AIDSYS WITH C'*****'.
000020	00000000					2301	1	ASAI2BLN DC	A(0)	ALL: LENGTH OF BUFFER SPECIFIED BY ASAI2ADR
000024	00					2302	1	ASAI2FB DC	X'00'	ALL: FLAGBYTE
				000010		2303	1	ASAI2DT EQU	ASADUMP	ALL: ACCESS DUMPED TASK
				000040		2304	1	ASAI2SYS EQU	X'40'	TPT: FOR SUBFCT ASAI2TPT ONLY: GET TEST-

FLAG	L0CTN	OBJECT	C0DE	ADDR1	ADDR2	STMNT	M	SOURCE	STATEMENT	
						2305	1	*	POINTS SET IN SYSTEM'S SPACE BY	
						2306	1	*	SPECIFIED TASK	
	000020					2307	1	ASAI2ALS EQU	X'20'	TPT: FOR SUBFCT ASAI2TPT ONLY; GET ALL
						2308	1	*	TESTPOINTS IN SYSTEM WITHOUT REGARD	
						2309	1	*	TO ANY SPECIFIC TASK	
	000000					2310	1	ASAI2VIT EQU	ASAITN	NICHT VERWENDET
	000000					2311	1	ASAI2VTS EQU	ASATSN	NICHT VERWENDET
000025	00					2312	1	ASAI2LNK DC	X'00'	ALL: LINK# OF DUMPFIL (OPTIONAL)
000026	4040404040404040					2313	1	ASAI2MGD DC	CL8'	CSC/TPT: MODULE-NAME
						2314	1	*****		
						2315	1	**	RUECKKEHR - PARAMETER	**
						2316	1	*****		
000013						2317	1	ASAI2RTN EQU	ASARETRN	RETURN- CODE
000000						2318	1	ASAI2GK EQU	X'00'	ALL: ALL GK
000004						2319	1	ASAI2FE EQU	X'04'	ALL: FORMALER FEHLER:
						2320	1	*		- DER AUFRUFER IST NICHT 'AID'
						2321	1	*		- UNGUELTIGE SUBFUNKTION
						2322	1	*		- ITN > 255
000008						2323	1	ASAI2NMA EQU	X'08'	ALL: NO BUFFER SPECIFIED
00000C						2324	1	ASAI2ILK EQU	X'0C'	ALL: INVALID DUMPFIL-LINK
000010						2325	1	ASAI2FNG EQU	X'10'	ALL: DUMPFIL NOT OPEN
000014						2326	1	ASAI2INS EQU	X'14'	ALL: NO TASK SPECIFIED
000018						2327	1	ASAI2INF EQU	X'18'	ALL: SPECIFIED ITN/TSN NOT FOUND
00001C						2328	1	ASAI2NAC EQU	X'1C'	ALL: SPECIFIED TASK CANNOT BE ACCESSED
000020						2329	1	ASAI2FHI EQU	X'20'	TPT: FUNCTION NOT YET IMPLEMENTED
000024						2330	1	ASAI2NFS EQU	X'24'	ALL: NO FILE SPECIFIED
000028						2331	1	ASAI2BNA EQU	X'28'	ALL: BUFFER NOT WRITE-ACCESSABLE
00002C						2332	1	ASAI2NAV EQU	X'2C'	MVE: CONCISE REP AREA IST NICHT VORHANDEN
						2333	1	*		ODER MOMENTAN GESPERRT
00002E						2334	1	ASAI2XXX DS	0X	
00002E 00						2335	1	ASAI2RFB DC	X'00'	ALL: FLAG-BYTE
	000080					2336	1	ASAI2NCP EQU	X'80'	ALL: INFORMATION IS NOT COMPLETE BECAUSE
						2337	1	*		OF LACK OF MEMORY
000004						2338	1	ASAI2QUL EQU	ASAUQUAL	UEBERQUALIFIZIERTE EINGABE
000001						2339	1	ASAI2LEN EQU	*-ASAI2XXX	LENGTH OF RETURN-PARAMS
						2340	1	*****		
						2341	1	**	ES FOLGEN DIE PUFFER-FORMATE DER RECORDS	**
						2342	1	*****		
						2343	1	*****		
						2344	1	**	SUBFCT = ALL PCBs	**
						2345	1	*****		
000000						2346	1	ASAI2PAD EQU	X'00'	OFFSET FOR A(PCB)
000004						2347	1	ASAI2PC EQU	X'04'	OFFSET FOR P-COUNTER; ERSTES FELD DES
						2348	1	*		BEFEHLSZAEHLERS ENTHAELT DAS BYTE
						2349	1	*		'ESTKIND', MIT DER SYSTEM-EXIT-ANZEIGE
000008						2350	1	ASAI2ISR EQU	X'08'	OFFSET FOR ISR
00000C						2351	1	ASAI2PLN EQU	X'0C'	LENGTH OF RECORD

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMT	M	SOURCE STATEMENT
						2352	1	*****
						2353	1	* SUBFCT = ALL FCBS **
						2354	1	** 8 BYTES LINKNAME **
						2355	1	** 1 BYTE OPEN-INDICATOR **
						2356	1	*****
	000080					2357	1	ASAI2FOP EQU X'80' FILE IS OPEN
	000009					2358	1	ASAI2FL EQU X'09' LENGTH OF RECORD
						2359	1	*****
						2360	1	** SUBFCT = TPGINT **
	000000					2361	1	*****
						2362	1	ASAI2TLC EQU X'00' OFFSET OF LOCATION OF TESTPOINT
						2363	1	* CONTENTS = C'***T' THEN THIS IS TRACE
						2364	1	* = C'***1' : EVENT-1-INF0
						2365	1	* UNTIL = C'***7' : EVENT-7-INF0
	000004					2366	1	ASAI2TAL EQU X'04' OFFSET OF A(ACTIONLIST)
	000008					2367	1	ASAI2TLN EQU X'08' LENGTH OF RECORD
						2368	1	*****
						2369	1	** SUBFCT = SYSTEM-MOVES **
	000000					2370	1	*****
						2371	1	ASAI2SAD EQU 0 OFFSET FOR ADDRESS OF 1ST OVERRITTEN
						2372	1	* BYTE
	000004					2373	1	ASAI2SL EQU 4 OFFSET FOR LENGTH OF MOVE
	000005					2374	1	ASAI2SLN EQU 5 LENGTH OF RECORD
						2375	1	*****
						2376	1	** SUBFUNKTION: ALLE TASKS VON DUMPDAT1 **
	000000					2377	1	*****
	000002					2378	1	ASAI2DI EQU X'00' OFFSET FUER ITN
	000006					2379	1	ASAI2DTS EQU X'02' OFFSET FUER TSN
	00000A					2380	1	ASAI2DAT EQU X'06' OFFSET FUER TCB-ADRESSE
						2381	1	ASAI2DLN EQU X'0A' LAENGE EINES PUFFER-EINTRAGS
						2382	1	*****
						2383	1	**
						2384	1	** AENDERN AKTIONSLISTEN-ZEIGER **
						2385	1	** ----- **
						2386	1	**
	000014					2387	1	*****
						2388	1	ORG ASALABEL
						2389	1	*****
						2390	1	** HEADER - PARAMETER **
	000058					2391	1	*****
						2392	1	ASAPT EQU X'58' ALL: FUNKTION = AENDERN AKTIONSLISTEN-
						2393	1	* ZEIGER
	000000					2394	1	ASAPTTPT EQU X'00' TPT: SUBFCT = TESTPUNKTZEIGER
	000004					2395	1	ASAPTEVT EQU X'04' EVT: SUBFCT = EREIGNIS- ODER MASCHINEN-

FLAG	LOC TN	OBJECT CODE	ADDR1	ADDR2	STMT	M	SOURCE STATEMENT
					2396	1	* TRACE-ZEIGER
			000008		2397	1	ASAPTTRC EQU X'08' TRC: SUBFCT = STATEMENTTRACEZEIGER
			000008		2398	1	ASAPTMAX EQU X'08' MAX. # EINER SUBFUNKTION
					2399	1	*****
					2400	1	** AUFRUF - PARAMETER **
					2401	1	*****
000014	00000000				2402	1	ASAPTADR DC A(0) ALL: ADRESSE DER NEUEN AKTIONSLISTE
000018	00000000				2403	1	ASAPTTAD DC A(0) TPT: ADRESSE DES BETROFFENEN TESTPUNKTS
		000018			2404	1	ORG ASAPTTAD
000018	00				2405	1	ASAPTECL DC X'00' EVT: BETROFFENE EREIGNISKLASSE. SIND MEHR
					2406	1	* ALS EINE EREIGNISKLASSE ANGEBEN,
					2407	1	* SO WIRD FUER ALLE KLASSEN DIE SELBE
					2408	1	* AKTIONSLISTENADRESSE GESETZT.
					2409	1	*****
					2410	1	** RUECKKEHR - PARAMETER **
					2411	1	*****
000019					2412	1	ASAPTXXX EQU * ANFANG RUECKKEHRPARAMETER
000013					2413	1	ASAPTRTN EQU ASARETRN RUECKKEHRCODE
000000					2414	1	ASAPTOK EQU X'00' FUNKTION AUSGEFUEHRT. ASAPTRAD
					2415	1	* ENTHAELT ZUR KONTROLLE DIE ALTE
					2416	1	* ERSETZTE AKTIONSLISTENADRESSE
000004					2417	1	ASAPTFE EQU X'04' FORMALER FEHLER, D.H. ENTWEDER:
					2418	1	* - AUFRUFER IST NICHT AID
					2419	1	* - UNGUELTIGE UNTERFUNKTIONSNUMMER
					2420	1	* - ASAPTADR NICHT DEFINIERT
					2421	1	* - KEINE EREIGNISKLASSE DEFINIERT
					2422	1	* - UNGUELTIGE EREIGNISKLASSE DEF.
000008					2423	1	ASAPTERR EQU X'08' FUER DIE ANGEGEBENE UNTERFUNKTION
					2424	1	* EXISTIERT KEIN AKTIVER EINTRAG.
					2425	1	* AKTION WURDE NICHT AUSGEFUEHRT
00001C	00000000				2426	1	ASAPTRAD DC A(0) ERSETZTE AKTIONSLISTENADRESSE ZUR
					2427	1	* KONTROLLE. FALLS X'00000000' ZURUECK-
					2428	1	* GEgeben WIRD, DER RETURNCODE ABER
					2429	1	* X'00' IST, BEDEUTET DIES, DASS MEHR
					2430	1	* ALS EIN ALTER ZEIGER ERSETZT WURDE,
					2431	1	* ABER NICHT ALLE IDENTISCHE WAREN.
					2432	1	* MOEGLICHERWEISE LIEGT EIN AID-
					2433	1	* FEHLER VOR.
000007					2434	1	ASAPTLEN EQU *-ASAPTXXX LAENGE DER RUECKKEHRPARAMETER
000085					2435	1	ORG
000085					2436	1	ASALENGT EQU *-ASASERVE LENGTH(ASERP)
0000A0					2437	1	ASATTLEN EQU ((ASALENGT+X'1F')/32)*32 LEN(AIDSYS=TRACETABLE-ENTRY)
000058					2438	1	ASASEMAX EQU X'58' MAX. # OF SUBFUNCTION
000038					2439	1	ASAAID2 EQU X'38' MAX. # OF SUBFCT IN AIDSYS02
000040					2440	1	ASAAID3 EQU X'40' MAX. # OF SUBFCT IN AIDSYS03
000060					2441	1	ASAAID4 EQU X'60' MAX. # OF FCT IN AIDSYS04
000050					2442	1	ASAAID5 EQU X'50' MAX. # OF FCT IN AIDSYS05

I have given this program in order to show different things :

- 1) In a more or less complex modules, you have many entries what is not easy to handle because if you have an entry, you must have an external in the other module . There is of course a problem if you want to modify the description of SYSBASE, for example, which is defined here in this module, the changes must be seen by all other modules that use it . It is not easy to see what are the calling modules (here, in fact, it is not a problem because the calling modules are written in the beginning of the module, but sometimes it is not possible to do it so).
- 2) See page 102, the comments. Is it not so wonderful when dealing with modules ? It is of course not a criticism but only example of what we sometimes must handle .
- 3) See page 103, the comments . Is it not absurd that some modules presupposes that he receive the correct value in the register . In the called module, there are no saving attitude nor testing to see if no error .
- 4) There is also a problem because sometimes the decision if an error has occurred in the called module, it is in the called module that it is decided if it returns normally or if it returns to the next instruction which is a branch to the error routine. This is of course not normal because the programmer has then to know what are done in both module and not only in his own module .
- 5) I have putted after the program its flowchart, it is not the best solution, but it seems to me that if the flowchart is simple and therefore easy to understand, than the program is easy also . Indeed, in order to do this flowchart so "in one lime", I have had to rewritte various times the logic of the program and therefore, I was able to keep it simple and is it not right that simple is beautiful ?

FLAG	LCTN	OBJECT	CODE	ADDR1	ADDR2	STMT	M	SOURCE	STATEMENT		
						3910		*****		02270000	
						3911		*		* 02271000	
						3912		*	THE ENTRIES ARE:	* 02272000	
						3913		*		* 02274000	
						3914		*****		02275000	
000458						3915		ENTRY	SYSBASE	BOTH THIS ENTRIES HAVE A VALUE GIVEN	02276000
000460						3916		ENTRY	EXVTBASE	BY THE CALLER.	02277000
00045C						3917		ENTRY	AREADRT	ADDRESS OF CALLER READ ROUTINE	02278000
0004AC						3918		ENTRY	SAVE3	SAVEAREA ZONE FOR AS4RDPGE	02278500
0004F4						3919		ENTRY	SAVEREG8	SAVEAREA FOR REGISTER 08	02278700

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT	
000000						3920		DPACCESS CSECT	02289000
000000	90	EC	D00C			3921		STM R#14,R#12,12(R#13)	02309000
000004	05	C0				3922		BALR R#12,0	02319000
000006						3923		USING *,R#12	02329000
000006	50	D0	C462	000468		3924		ST R#13,SAVE+4 SAVE THE ADDRESS OF THE CALLER SAVE AREA	02339000
00000A	41	D0	C45E	000464		3925		LA R#13,SAVE LOAD THE ADDRESS OF THE NEW SAVE AREA	02349000
00000E	18	81				3926		LR R#08,R#01 USING OF DSECT BECAUSE OF MACRO	02359000
000010	50	80	C4EE	0004F4		3927		ST R#08,SAVEREG8 SAVE REGISTER 8 COULD BE DESTROYED !	02364000
000000						3928		USING MALIB,R#08 USE R#08 AS BASE REGISTER FOR THE DSECT	02369000
000014	D2	03	C4568008	00045C	000008	3929		MVC AREADRT,DPAREAD MOVE A(READ-ROUTINE) TO EXTERNAL FIELD	02375000
						3930		*****	02389000
						3931	*		* 02399000
						3932	*	THE FIRST TEST DONE HERE ARE VALIDATION OF SUBFUNCTION	* 02409000
						3933	*	NUMBER.	* 02419000
						3934	*		* 02439000
						3935	*	*****	* 02449000
00001A	91	F3	8000	000000		3936		TM DPAFCT,DPAVALFN X'F3' IS EQUATE OF B'11110011'	02469000
						3937	*	WICH IS THE MASK TO VALIDATE	02479000
						3938	*	THE NUMBER'S SUBFUNCTION.	02489000
00001E	47	70	C41C	000422		3939		BHZ RTNISFN	02499000
000022	95	08	8000	000000		3940		CLI DPAFCT,DPAMAXNF THIS TEST IS DONE IN ORDER TO AVOID THE	02510000
						3941	*	X'0C' VALUE WHICH IS NO MORE VALID.	02520000
000026	47	20	C41C	000422		3942		BH RTNISFN DPAMAXNF CONTAINS THE MAXIMUM VALID	02530000
						3943	*	NUMBER FOR A KNOWN FUNCTION.	02540000
						3944		*****	02729000
						3945	*		* 02739000
						3946	*	THE NEXT TESTS ARE DONE IN ORDER TO VALIDATE THE DIFFERENT	* 02744000
						3947	*	ADDRESSES TO BE PROVIDED BY THE CALLER.	* 02749000
						3948	*		* 02759000
						3949	*	*****	* 02769000
						3950	*	CLC DPAATCB,=A(0) THIS COMPARE IS DONE IN ORDER TO SEE	02799000
						3951	*	IF AN TCB ADDRESS WAS GIVEN	02889000
						3952	*	BE RTNFF IF NOT,BRANCH TO FORMAL FEHLER	02889400
						3953	*		02889800
						3954	*		02890200
						3955	*		02890600
						3956	*	TM DPAATCB+3,DPAVALFW COMPARE THE TWO RIGHTEST BITS	02891000
						3957	*	IF THE RESULT CONTAINS SOME	02891400
						3958	*	ONES,THEN THE ADDRESS IS NOT	02891900

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE	STATEMENT		
						3959	*		FULL WORD ALIGNED.	02892300	
						3960	*	BNZ	RTNWA	BRANCH TO NOT WORD ALIGNED.	
00002A	D5	03	8010C42A	000010	000430	3961		CLC	DPASYSBS,=A(0)	SYSBASE ADDRESS GIVEN ?	02893500
000030	47	80	C15A	000160		3962		BE	RTNFF	NO, BRANCH TO FORMAL FEHLER	02893900
000034	91	03	8013	000013		3963		TM	DPASYSBS+3, DPAVALFW	FW ALIGNED ADDRESS ?	02894800
000038	47	70	C162	000168		3964		BNZ	RTNWA	NO, BRANCH TO FORMAL FEHLER.	02895200
00003C	D5	03	8014C42A	000014	000430	3965		CLC	DPAAXVT,=A(0)	TEST IF XVT ADDRESS IS GIVEN ?	02896000
000042	47	80	C15A	000160		3966		BE	RTNFF	NO, BRANCSH TO FORMAL FEHLER.	02896400
000046	91	03	8017	000017		3967		TM	DPAAXVT+3, DPAVALFW	FW ADDRESS ALIGNED ?	02897300
00004A	47	70	C162	000168		3968		BNZ	RTNWA	NO, BRANCH TO NOT WORD ALIGNED	02897700

FLAG	LOC	CTN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT	
							3969		*****	02898500
							3970		*	02899000
							3971		*	02909000
							3972		*	02919000
							3973		*	02921000
							3974		*	02924000
							3975		*	02926000
							3976		*	02929000
							3977		*****	02939000
00004E	D5	03	8008C42A		000008	000430	3978		CLC DPAREAD,=A(0) IS THE READ ADDRESS GIVEN ?	02959000
000054	47	70	C068		00006E		3979		BNE READGIVN YES,BRANCH TO READ GIVEN	02969000
000058	D5	03	8004C42E		000004	000434	3980		CLC DPALINK,=F'0' IS A LINK NUMBER GIVEN ?	02979000
00005E	47	80	C15A		000160		3981		BE RTNFF YES,BRANCH TO FORMAL FEHLER	02989000
000062	95	07	8004		000004		3982		CLI DPALINK,DPALINK# IS THAT A VALID LINK NUMBER ?	02999000
000066	47	20	C182		000188		3983		BH RTHILN NO,BRANCH TO INVALID LINK NUMBER	03009000
00006A	47	F0	C096		00009C		3984		B VALIDPRM YES,BRANCH TO VALIDATION O.K.	03019000
00006E							3985		READGIVN DS OY WE ARE HERE ONLY IF A(READ) IS <> 0	03020200
00006E	D5	03	801CC42E		00001C	000434	3986		CLC DPAITH,=F'0' ITN GIVEN ?	03020800
000074	47	80	C192		000198		3987		BE RTNRNI NO,BRANCH TO A(READ) GIVEN BUT NO ITN.	03021500
000078	D5	03	800CC42A		00000C	000430	3988		CLC DPAATCB,=A(0) IS AN TCB GIVEN ?	03021560
00007E	47	80	C15A		000160		3989		BE RTNFF NO,BRANCH FORMAL FEHLER	03021590
000082	91	03	800F		00000F		3990		TM DPAATCB+3,DPAVALFW COMPARE THE TWO RIGHTEST BITS IF	03021660
							3991	*	NOT ZERGES, THEN BRANCH TO NOT WORD	03021690
							3992	*	ALIGNED	03021730
000086	47	70	C162		000168		3993		BNZ RTNHW NO,BRANCH TO NOT WORD ALIGNED	03021760
00008A	91	03	800B		00000B		3994		TM DPAREAD+3,DPAVALFW IS THE READ ADDRESS FW ALIGNED ?	03022100
00008E	47	70	C162		000168		3995		BNZ RTNHW NO,BRANCH TO NOT WORD ALIGNED	03022700

FLAG	LCTN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT	
000092	D5	02	801CC42E	00001C	000434	3996		CLC DPAITN(DPAITNL),=F'0' VALID ITN GIVEN ?	03023300
000098	47	70	C15A	000160		3997		BNE RTNFF NO,BRANCH TO FORMAL FEHLER	03024000
00009C						3998		VALIDPRM DS 0Y VALID PARAMETER	03025200
00009C	D2	03	C4528010	000458	000010	3999		MVC SYSBASE,DPASYSBS MOVE THE VALUE OF SYSBASE FOR THE	03025800
						4000	*	ENTRY TO BE CORRECT	03026500
0000A2	D2	03	C45A8014	000460	000014	4001		MVC EXVTBASE,DPAAXVT MOVE THE VALUE OF XVT FOR THE ENTRY	03027100
						4002	*	TO BE CORRECT	03027700

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT	
						4003		*****	03039000
						4004	*		* 03049000
						4005	*	THE DONE AFTER ARE VALIDATION OF IND,	* 03059000
						4006	*	VALIDATION OF BUFFER ADDRESS AND BUFFER LENGTH	* 03069000
						4007	*	VALIDATION OF FLAGUSER AND LAST VALIDATION OF FUNCTION	* 03079000
						4008	*		* 03089000
						4009		*****	03099000
0000A8	95	04	801B	00001B		4010		CLI DPAIN+DPAINDL,ASAGTCUS TEST IF THE DUMPFIL TYPE	03119000
						4011	*	IS A USER DUMPFIL	03124000
0000AC	47	80	C0BA	0000C0		4012		BE VALIDDFT YES,NO MORE TEST NEEDED IT IS O.K.	03129000
0000B0	95	02	801B	00001B		4013		CLI DPAIN+DPAINDL,ASAGTCSY TEST IF THE DUMPFIL TYPE	03139000
						4014	*	IS A SYSTEM DUMPFIL	03144000
0000B4	47	80	C0BA	0000C0		4015		BE VALIDDFT YES,NO MORE TEST NEEDED IT IS O.K.	03149000
0000B8	95	08	801B	00001B		4016		CLI DPAIN+DPAINDL,ASAGTSLD TEST IF THE DUMPFIL TYPE IS	03150000
						4017	*	IS A SLED FILE	03151000
0000BC	47	70	C18A	000190		4018		BNE RTNIIND NO,THEN THERE IS AN ERROR BECAUSE	03152000
						4019	*	IT IS THE NEITHER A USER DUMP NOR A	03153000
						4020	*	SYSTEM DUMP NOR A SLED AND NOTHING	03155000
						4021	*	ELSE IS KNOWN.	03157000

FLAG	LCTN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT	
						4022		*****	03157200
						4023	*		03157300
						4024	*	THE NEXT TESTS ARE DONE IN ORDER TO SEE IF THAT IS THE FIRST	03157500
						4025	*	CALL OR A SUCESSIVE ONE,IF BOTH SYSTEM AND USER SPACE IS	03157600
						4026	*	WANTED.	03157700
						4027	*		03157800
						4028		*****	03157860
0000C0						4029		VALIDDFT DS 0Y	03158000
0000C0	91	DF	8028	000028		4030		TM DPAFLAG,DPAUSSET SEE IF USER FLAG IS SET OR OMITTED	03159000
0000C4	47	70	COCA	0000D0		4031		BHZ GIVEN IF OMITTED DEFAULT IS TO BE SET	03169000
0000C8	96	08	8028	000028		4032		GI DPAFLAG,DPASYST DEFAULT SET	03179000
0000CC	47	F0	COE6	0000EC		4033		B GNESET NO CHECK IF FLAG CORRECTLY SET,THE	03181000
						4034	*	PROGRAM ONLY PROVIDE CORRECT VALUE...	03184000

FLAG	LOC TN	OBJECT	CODE	ADDR1	ADDR2	STMT	M	SOURCE STATEMENT	
						4035		*****	03186100
						4036	*		03186300
						4037	*	IF WE ARE HERE THEN WE ARE SURE THAT THE CALLER HAS GIVEN	03186500
						4038	*	A VALUE TO DPAFLAG, BUT WE ARE NOT SURE THAT IT IS A CORRECT	03186700
						4039	*	VALUE IN IT. THEN WE TEST IT HERE.	03186900
						4040	*		03187100
						4041		*****	03187300
0000D0						4042		GIVEN DS OY	03189000
0000D0	91	D3	8028	000028		4043		TM DPAFLAG,DPAVALFL TEST IF VALIDE VALUE OF THIS FLAG	03209000
						4044	*	THE MASK VALUE IS X'D3' OR B'11010011'	03219000
0000D4	47	70	C15A	000160		4045		BNZ RTNFF IF NOT 0, THEN BRANCH TO FORMAL FEHLER	03229000
0000D8	95	2C	8028	000028		4046		CLI DPAFLAG,DPAFLAGM TEST IF FLAG HAS A VALUE <= 2C	03249000
0000DC	47	20	C15A	000160		4047		BH RTNFF IF NOT <= 44, THEN BRANCH TO FORMAL FEHL.	03259000
0000E0	91	0C	8028	000028		4048		TM DPAFLAG,DPAALL TEST IF BOTH USER AND SYSTEM ARE SET	03279000
0000E4	47	E0	C0E6	0000EC		4049		BNZ GNESET IF NOT DOES NOT MATTER	03289000
0000E8	92	40	CE5E	000E64		4050		MVI ASUB,DPABOTH SET FLAG BOTH WANTED, WHAT WILL BE USED	03299000
						4051	*	LATER WHEN CALLING THE FUNCTIONS	03309000

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMT	M	SOURCE STATEMENT	
						4052		*****	03319600
						4053	*		03320200
						4054	*	WE ARE HERE ONLY IF THE DPAFLAG IS SET WITH A CORRECT VALUE	03320800
						4055	*	BUT WE HAVE TO TRANSFORM THIS VALUE BECAUSE IT IS GIVEN 04	03321500
						4056	*	08 GR 12 AS VALUE AND IT IS ONLY EXPECTED 01 GR 02 GR 03.	03322100
						4057	*		03322700
						4058		*****	03323300
0000EC						4059		GNESET DS 0Y	03329000
0000EC	17	33				4060		XR R#03,R#03 CLEAR REGISTER 3, OTHERWISE STRANGE	03339000
						4061	*	EFFECTS CAN BE OBSERVED...	03349000
0000EE	D2	00	CE5F8028	000E65	000028	4062		MVC ASUB+1(1),DPAFLAG TRANSFORM THE VALUE OF SYSTEM GR USER	03359000
0000F4	94	DF	CE5F	000E65		4063		NI ASUB+1,DPAUSSET FROM CALLER VALUE X'04' GR X'08' TO	03369000
0000F8	43	30	CE5F	000E65		4064		IC R#03,ASUB+1 EXPECTED VALUE FOR THE SAME THING IN	03379000
0000FC	88	30	0002	000002		4065		SRL R#03,2 AIDSYS05 WHERE IT IS EXPECTED THE	03389000
000100	94	F0	8028	000028		4066		NI DPAFLAG,X'F0' VALUE X'01' GR X'02', WITHOUT CHANGING	03399000
000104	42	30	CE5F	000E65		4067		STC R#03,ASUB+1 THE FIRST HALF OF THE FLAG WHICH	03409000
000108	D6	00	8028CE5F	000028	000E65	4068		GC DPAFLAG(1),ASUB+1 IS USED FOR OTHER PURPOSE.	03419000
						4069	*	NOW, WE CAN CONTINUE TESTING FOR OTHERS	03430000
						4070	*	FIELD. IF THE FUNCTION TO BE CALLED IS	03432000
						4071	*	GET ADDRESS THEN NO TEST TO BE DONE	03433000
						4072	*	ABOUT THE BUFFER BECAUSE NO GNE IS	03435000
						4073	*	TO BE PROVIDED.	03437000
00010E	91	DF	8028	000028		4074		TM DPAFLAG,DPAFLAGT FIRST CALL GR SUCCESSIVE CALL?	03519000
						4075	*	THE MASK TO TEST IT IS X'20'	03529000
000112	47	10	C14A	000150		4076		B0 NOTEST SUCCESSIVE CALL, THEN NO CONTROL NEEDED	03539000
						4077	*	FIRST CALL? GR WRONG VALUE ASSIGNED?	03549000
						4078	*	IF NOT 4 GR 8 GR C THEN FORMAL FEHLER	03559000
000116	95	04	8000	000000		4079		CLI DPAFCT,DPAGASFT TEST IF IT IS THE GET ADDRESS FUNCTION	03570000
00011A	47	80	C2BE	0002C4		4080		BE GAFCT IF YES, THEN NO MORE TEST NEEDED GO!	03572000
						4081	*	REMEMBER ADDRESS = 0 => NOT GIVEN	03579000
						4082	*		03589000

FLAG	LOC	CTN	OBJECT	CODE	ADDR1	ADDR2	STMT	M	SOURCE STATEMENT	
							4083		*****	03599900
							4084	*		03600800
							4085	*	ALL THE NEXT TEST ARE DONE IN ORDER TO VALIDATE THE BUFFER	03601700
							4086	*	1. BUFFER ADDRESS GIVEN ?	03602600
							4087	*	IF NOT, GO FORMAL FEHLER	03603500
							4088	*	2. BUFFER ADDRESS WORD ALIGNED ?	03604400
							4089	*	IF NOT, GO FORMAL FEHLER	03605300
							4090	*	3. BUFFER LENGTH GIVEN ?	03606200
							4091	*	IF NOT, GO FORMAL FEHLER	03607100
							4092	*	4. BUFFER LENGTH LESS THEN ONE REGRD LENGTH ?	03608000
							4093	*	IF YES, GO BUFFER TOO SHORT	03608090
							4094	*		03608180
							4095	*	*****	03608270
00011E	17	33					4096		XR R#03,R#03	03609000
000120	59	30	8020		000020		4097		C R#03,DPABUFAD	03619000
000124	47	80	C15A		000160		4098		BE RTNFF	03629000
									IF NOT GIVEN, THEN BRANCH TO FGRM. FEHLER	
000128	91	03	8023		000023		4099		TM DPABUFAD+3, DPAVALFW	03649000
00012C	47	70	C15A		000160		4100		BHZ RTNFF	03659000
							4101	*	TEST IF THE ADDRESS IS FULLWORD ALIGNED, IF NOT BRANCH TO FORMAL FEHLER (TEST OF TWO LEFTTEST BITS)	03669000
000130	59	30	8024		000024		4102		C R#03,DPABUFLE	03779000
000134	47	80	C15A		000160		4103		BE RTNFF	03789000
									TEST TO SEE IF A BUFFER LENGTH IS GIVEN, IF NOT BRANCH TO FORMAL FEHLER.	
000138	41	30	CE52		000E58		4104		LA R#03,TAB	03809000
							4105	*	LOAD ADDRESS OF THE TABLE CONTAINING THE MINIMAL BUFFER LENGTH FOR EACH FCT.	03814000
00013C	D2	00	CE5F8000		000E65	000000	4106		MVC ASUB+1(1),DPAFCT	03819000
000142	4A	30	CE5E		000E64		4107		AH R#03,ASUB	03829000
							4108	*	MOVE DONE BECAUSE OF HW ALIGNEMENT	03839000
							4109	*	ADD TO THE START ADDRESS THE SUBFUNCTION NUMBER IN ORDER TO OBTAIN THE MINIMAL BUFFER LENGTH NEEDED BY THE SPECIFIED FCT	03849000
000146	D5	03	80243000		000024		4110		CLC DPABUFLE,0(R#03)	03859000
							4111	*	COMPARE TO SEE IF LENGTH IS ENOUGH TO CONTAIN ONE ANSWER AT LEAST!	03869000
00014C	47	40	C17A		000180		4112		BL RTNBTS	03879000
									IF NOT, BRANCH TO BUFFER TOO SHORT FEHLER	

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMT	M	SOURCE STATEMENT	
000150						4113		NOTEST DS OY SECOND CALL DIRECT TO FUNCTION CALL.	03919000
						4114		*****	03929000
						4115	*	*	03939000
						4116	*	NO MORE TEST ALL ARE O.K. THEN CALL FUNCTION	03949000
						4117	*	*	03959000
						4118		*****	03969000
000150	95	04	8000	000000		4119		CLI DPAFCT,DPAGASFT IF SECOND CALL,THIS TEST IS IMPORTANT	03989000
000154	47	80	C2BE	0002C4		4120		BE GAFCT IF FIRST CALL,IF GAFCT IT SHOULD NOT	03999000
						4121	*	NO MORE COMMON TEST FOR ANY SUBFUNCTION	04019000
						4122	*	AS THERE IS NO MORE VALID VALUE OF FLAG	04029000
						4123	*	DPACSFCT THEN 0,4,8.THERE IS NO NEED TO	04039000
						4124	*	DO SOME OTHER TEST .	04049000
						4125	*		04059000
000158	47	20	C41C	000422		4126		BH PEFCT IF DPACSFCT > 4,THEN PEFCT	04069000
00015C	47	40	C19A	0001A0		4127		BL CMFCT IF < 4,THEN CSEC MAP FUNCTION	04079000

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMT	M	SOURCE STATEMENT	
						4128		*****	04099000
						4129	*		* 04109000
						4130	*	RETURN WITH FORMAL FEHLER.RETURN CODE IS X'04'	* 04119000
						4131	*		* 04129000
						4132		*****	04139000
000160						4133		RTNFF DS OY	04154000
000160	92 04 8001			000001		4134		MVI DPARTN,DPACMFE MOVE VALUE IN CORRECT RETURN CODE ZONE	04159000
000164	47 F0 C41C			000422		4135		B RTN	04169000
						4136		*****	04189000
						4137	*		* 04199000
						4138	*	NOT WORD ALIGNED ADDRESS.RETURN CODE IS X'0C'	* 04209000
						4139	*		* 04219000
						4140		*****	04229000
000168						4141		RTNNWA DS OY	04244000
000168	92 0C 8001			000001		4142		MVI DPARTN,DPACMWA MOVE VALUE IN RETURN CODE ZONE	04249000
00016C	47 F0 C41C			000422		4143		B RTN	04259000
						4144		*****	04259900
						4145	*		* 04260800
						4146	*	INVALID ADDRESS.RETURN CODE IS X'2C'	* 04261700
						4147	*		* 04262600
						4148		*****	04263500
000170						4149		RTNIA DS OY	04264800
000170	92 2C 8001			000001		4150		MVI DPARTN,DPACMIA MOVE VALUE IN RETURN CODE ZONE	04265300
000174	47 F0 C41C			000422		4151		E RTN RETURN TO THE CALLER	04266200
						4152		*****	04279000
						4153	*		* 04289000
						4154	*	FILE IS NOT OPEN.RETURN CODE IS X'10'	* 04299000
						4155	*		* 04309000
						4156		*****	04319000
000178						4157		RTNDFE DS OY	04334000
000178	92 10 8001			000001		4158		MVI DPARTN,DPACMDF MOVE VALUE IN RETURN CODE ZONE	04339000
00017C	47 F0 C41C			000422		4159		B RTN	04349000
						4160		*****	04369000
						4161	*		* 04379000
						4162	*	LENGTH OF BUFFER NOT GREAT ENOUGH TO CONTAIN ONE ANSWER	* 04389000
						4163	*	R C = X'34'	* 04399000
						4164		*****	04409000

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT	
000180						4165		RTNBTS DS OY	04424000
000180	92	34	8001	000001		4166		MVI DPARTN,DPACMTS MOVE VALUE IN RETURN CODE ZONE	04429000
000184	47	F0	C41C	000422		4167		B RTN	04434000
						4168		*****	04459000
						4169		*	* 04469000
						4170		* INVALID LINK NUMBER GIVEN.RETURN CODE IS X'1C'	* 04479000
						4171		*	* 04489000
						4172		*****	04499000
000188						4173		RTNILN DS OY	04514000
000188	92	1C	8001	000001		4174		MVI DPARTN,DPACMILN MOVE VALUE IN RETURN CODE ZONE	04519000
00018C	47	F0	C41C	000422		4175		B RTN	04529000
						4176		*****	04530200
						4177		*	* 04530800
						4178		* INVALID INDICATOR OF DUMPFIL TYPE GIVEN.RC=X'20'	* 04531500
						4179		*	* 04532100
						4180		*****	04532700
000190						4181		RTNIIND DS OY	04534000
000190	92	20	8001	000001		4182		MVI DPARTN,DPACMIND MOVE VALUE IN RETURN CODE ZONE	04534600
000194	47	F0	C41C	000422		4183		B RTN RETURN TO CALLER	04535200
						4184		*****	04536500
						4185		*	* 04537100
						4186		* A(READ) GIVEN BUT NOT THE ITH. RETURN CODE = X'24'	* 04537700
						4187		*	* 04538300
						4188		*****	04538370
000198						4189		RTNRNI DS OY	04538530
000198	92	24	8001	000001		4190		MVI DPARTN,DPACMRNI MOVE VALUE IN RETURN CODE	04538610
00019C	47	F0	C41C	000422		4191		B RTN RETURN TO CALLER	04538680

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT	
	0001A0					4192		CMFCT DS OY	04549000
						4193		*****	04559000
						4194		* *****	04569000
						4195		* *	* * 04579000
						4196		* *	* * 04589000
						4197		* * THIS IS THE CALL FOR EXECUTING THE CMFCT,THE LAYOUT	* * 04599000
						4198		* * OF THE CALLING PARAMETERS IS :	* * 04609000
						4199		* *	* * 04619000
						4200		* * DISPLACEMENT 00 NUMBER OF THE CMFCT = X'0'	* * 04629000
						4201		* *	* * 04639000
						4202		* * DISPLACEMENT 01 RETURN CODE FIELD.	* * 04649000
						4203		* *	* * 04659000
						4204		* * DISPLACEMENT 04 LINK NUMBER OF DUMPFIL	* * 04669000
						4205		* *	* * 04679000
						4206		* * DISPLACEMENT 08 ADDRESS OF THE DUMPFIL READ ROUTINE	* * 04689000
						4207		* *	* * 04699000
						4208		* * DISPLACEMENT 12 ADDRESS OF TCB	* * 04709000
						4209		* *	* * 04710000
						4210		* * DISPLACEMENT 16 ADDRESS OF SYSBASE	* * 04711000
						4211		* *	* * 04712000
						4212		* * DISPLACEMENT 20 ADDRESS OF XVT	* * 04713000
						4213		* *	* * 04714000
						4214		* * DISPLACEMENT 24 INDICATOR OF DUMPFIL TYPE	* * 04715000
						4215		* *	* * 04716000
						4216		* * DISPLACEMENT 28 THE ITN NUMBER	* * 04717000
						4217		* *	* * 04719000
						4218		* * DISPLACEMENT 32 ADDRESS OF THE BUFFER	* * 04720000
						4219		* *	* * 04721000
						4220		* * DISPLACEMENT 36 BUFFER LENGTH	* * 04723000
						4221		* *	* * 04724000
						4222		* * DISPLACEMENT 40 FLAG FOR USER OR SYSTEM REACHED	* * 04725000
						4223		* *	* * 04739000
						4224		* * DISPLACEMENT 41 NAME OF PROGRAM UNIT	* * 04749000
						4225		* *	* * 04759000
						4226		* * DISPLACEMENT 84 NUMBER OF RETURNED RECORD	* * 04762000
						4227		* *	* * 04765000
						4228		* *	* * 04769000
						4229		* *****	* * 04779000
						4230		* *	* * 04789000
						4231		* *	* * 04799000
						4232		* *****	* * 04809000
						4233		* *	* * 04819000
						4234		* *	* * 04829000
						4235		* * THE OUTPUT WILL HAVE AS LAYOUT :	* * 04839000
						4236		* *	* * 04849000
						4237		* *	* * 05083692
						4238		* * DISPLACEMENT 0 CSECT NAME	* * 05083702
						4239		* *	* * 05083712
						4240		* * DISPLACEMENT 8 CSECT START ADDRESS	* * 05083722
						4241		* *	* * 05083732
						4242		* * DISPLACEMENT 12 CSECT LENGTH	* * 05083742
						4243		* *	* * 05083752
						4244		* * DISPLACEMENT 16 ETPND INFORMATION	* * 05083761
						4245		* *	* * 05083771
						4246		* * THE RETURN CODE IS IN SPECIAL FIELD	* * 05083801

FLAG	LOC	CTN	OBJECT	CODE	ADDR1	ADDR2	STMT	M	SOURCE STATEMENT	
							4247	*	*	05083811
							4248	*	*	05083821
							4249	*	*	05083831
							4250	*	*	05083841
							4251	*	*****	05083851
							4252	*		05083860
							4253	*	*****	05083870
0001A0	41	10	C4F2		0004F8		4254		LA R#01,WORKAREA	05083890
000000							4255		USING SERVE,R#01	05083900
0001A4	92	10	1012		000012		4256		MVI ASASBFCT,ASADSDPG	05088355
							4257	*	CSECT LEVEL.	05199000
0001A8	D2	03	102C8020		00002C 000020		4258	*	MVC ASADSAUB(4),DPABUFAD	05109000
							4259	*	MOVE ADDRESS OF THE CALLER	05119000
0001AE	D2	01	10308026		000030 000026		4260	*	MVC ASADSLUB(2),DPABUFLE+2	05129000
							4261	*	LENGTH OF BUFFER	05134000
0001B4	D2	05	1004C44A		000004 000450		4262	*	MVC ASACALL1(6),=C'AIDSYS'	05139000
							4263	*	AS AIDSYS REQUEST!!!!!!!!!!!!	05144000
0001BA	92	10	101B		00001B		4264	*	MVI ASADSUSE,ASADUMP	05149000
							4265	*	TO SIMULATE DUMP FILE ACCESS	05154000
0001BE	92	10	1034		000034		4266	*	MVI ASADSLEV,ASADSLCS	05159000
							4267	*	CSECT LEVEL	05160000
0001C2	D2	00	101A8007		00001A 000007		4268	*	MVC ASADSDFN(4-DPALNKL),DPALINK+DPALNKL	05162000
							4269	*		05163000
0001C8	D2	00	1019801F		000019 00001F		4270	*	MVC ASADSIITN+4-DPAITNL(4-DPAITNL),DPAITN+DPAITNL	05165000
0001CE	91	40	CE5E		000E64		4271	*	TM ASUB,DPABOTH	05179000
							4272	*	TEST IF BOTH ARE SPECIFIED	05189000
0001D2	47	E0	C1D4		0001DA		4273	*	BHØ ENDPARAM	05199000
								*	BEFORE DOING THE MOVE	
								*	IF NOT MOVE WITHOUT CARE	
0001D6	94	FD	8028		000028		4274	*	NI DPAFLAG,DPARESET	05219000
							4275	*	RESET SYSTEM VALUE AND LET DO	05229000
							4276	*	THE CALL FOR NON PRIVILEGED	05239000
							4277	*	CSECT MAP,WHAT IS TGLD TO BE	05249000
								*	DONE IN THE EXTERNAL INTERFACE	
0001DA							4278		ENDPARAM DS 0Y	05269000
0001DA	D6	00	101B8028		00001B 000028		4279		ØC ASADSUSE,DPAFLAG	05279000
									SET EXPECTED FLAG FROM AIDSYS05	

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT	
						4280		*****	05299000
						4281		*	* 05309000
						4282		*	* 05319000
						4283		*	* 05329000
						4284		*	* 05339000
						4285		*	* 05349000
						4286		*****	05359000
0001E0	58	40	C432	000438		4287		L R#04,=A(MYDYNDTA) COVER DYNDATA NEEDED BY AIDSYS	05379000
000000						4288		USING DYNDATA,R#04	05382000
0001E4	D2	03	49408014	000940	000014	4289		MVC DXVTBASE,DPAAXVT MOVE THESE VALUES IN ORDER TO SIMULATE	05385000
0001EA	50	10	40F0	0000F0		4290		ST R#01,APARAM THIS FIELD IS USED IN ASDPRGG(ASDSERV)	05385200
						4291	*		05385300
0001EE	D2	03	4A7C800C	000A7C	00000C	4292		MVC DUMPATCB,DPAATCB MOVE IMPORTANT IF A ITN IS GIVEN	05385500
						4293	*		05386200
0001F4	D2	03	4008C436	000008	00043C	4294		MVC ARET,=A(WORKAREA+2000) AIDSYS02,WHICH IS THE MODULE	05387000
						4295	*	RESPONSABLE FOR SETTING THESE FIELD.	05388000
						4296	*		05388200
0001FA	D2	03	4B20C43A	000B20	000440	4297		MVC REQRQADR,=A(WRKETPND) BUFFER FOR ETPND FUNCTION.	05388500
						4298		*****	05399000
						4299	*		* 05409000
						4300	*	THE MEMORY REQUESTED BY AIDSYS IN ORDER TO COVER DYNDATA	* 05419000
						4301	*	IS RESERVED HERE AT THE END OF THE PROGRAM.IT'S LENGTH	* 05429000
						4302	*	IS GIVEN IN THE ASERP MACRO.	* 05439000
						4303	*		* 05449000
						4304		*****	05459000
000200	58	50	C43E	000444		4305		L R#05,=A(MYSTATDT) SAVE ADDRESS OF MEMORY COVERING STATD	05469000
						4306	*	LA R#01,WORKAREA COVER PARAM LIST	05479000
000204	50	D0	C45E	000464		4307		ST R#13,SAVE SAVE REGISTER 13	05489000
						4308	*	LA R#13,SAVE1 SAVE AREA FOR CALLING PARAMETERS	05519000
						4309		*****	05539000
						4310	*		* 05549000
						4311	*	THE PARAMETERS ARE READY FOR CALL AND ALSO THE REGISTERS	* 05559000
						4312	*		* 05569000
						4313		*****	05579000

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT	
						4314		*****	05610000
						4315	*		* 05630000
						4316	*	WHAT IS DONE HERE IS TO SIMULATE THE DISPOSITION OF DATAS	* 05660000
						4317	*	WHICH ARE NORMALLY PROVIDED BY AIDSYS02 AND WHICH HERE ARE	* 05680000
						4318	*	PROVIDED DIRECTLY BY THE CALLER	* 05700000
						4319	*		* 05730000
						4320		*****	05750000
000000						4321		USING STATDATA,R#05	05753000
						4322	*	COVER STATDATA FOR FINDING	05754000
						4323		A(DFTAB)	05755000
						4324	1	AIDSCOD GETDFTAB,P1=R#06,P2=R#07,P3=ASADSDFN,P4=DFTAB	
						4325	2	IDLKG VER=009	
						4326	2	*,VERSION 009	00001300
000208						4327	2	CNSP 0,4	00002800
000208						4328	1	DS OF	00003500
000208 17 66						4329	1	XR R#06,R#06 CLEAR REG	
00020A 43 60 101A	00001A					4330	1	IC R#06,ASADSDFN GET LINK #	
00020E 89 60 0003	000003					4331	1	SLL R#06,3 REL. A(ENTRY IN DF-TABLE)	
000212 41 70 5260	000260					4332	1	LA R#07,DFTAB A(DF-TABLE)	
000216 1A 67						4333	1	AR R#06,R#07 A(ENTRY IN DF-TABLE)	
000218 50 60 4278	000278					4333	1	ST R#06,DYDFPBIT SAVE ADDRESS	
00021C D2 03 6000C442	000448					4334		MVC 0(4,R#06),=A(WORKAREA+1024) SIMULATE THAT THIS FIELD	05756000
						4335	*	WAS FULFILLED WHEN OPEN(NORMALY	05757000
						4336	*	DONE BY AIDSYS02)	05757500
000222 58 70 6000						4337	*	L R#07,0(0,R#06) LOAD A(DUMPFIL-WORKAREA)	05758000
						4338	*		05759000
000226 12 77						4339		LTR R#07,R#07 DUMPFIL OPEN ?	05761000
000228 47 80 C172	000178					4340		BZ RTNGFE NG.	05764000
00022C 50 70 40EC	0000EC					4341		ST R#07,DWORK SAVE ADDRESS	05767000
000000						4342		USING WORK,R#07	05770000
						4343	*		05771000
000230 D2 03 72D88010	0002D8	000010				4344	*	MVC WAVIRT(4),DPASYSBS	05773000
						4345	*	STORE A(SYSBASE) IN THE RIGHT	05776000
						4346	*	PLACE SO THAT AIDSYS05 WILL GET	05776300
						4347	*	IT WITHOUT TROUBLE AS IF IT WAS	05776600
000236 D2 00 730B801B	00030B	00001B				4348	*	GONE THROUGH THE USUAL WAY.	05776900
						4349	*	WAIND IS ONLY A BYTE BUT DPAIND	05777200
00023C D2 03 72F48014	0002F4	000014				4350	*	MVC WAXVT,DPAAXVT	05777500
						4351	*	MOVE A(XVT) IN THE FIELD USED	05777600
						4352	*	LATER BY ADSERV.	05777800
								DROP R#07,R#05,R#04	

FLAG	LCTN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT		
						4353	*	*****	05789000	
						4354	*		05799000	
						4355	*	NEVER CHANGE THE VALUE	05809000	
						4356	*	OF THE REGISTER	05819000	
						4357	*		05829000	
						4358	*	THESE VALUES ARE NEEDED BY AIDSYS05!!!	05839000	
						4359	*	=====	05849000	
						4360	*		05859000	
						4361	*		05869000	
						4362	*	*****	05879000	
						4363	*		05899000	
						4364	*		05909000	
						4365	*		05919000	
000242	41	D0	C45E	000464		4366		LA R#13,SAVE	RESTORE ADDRESS OF SAVE AREA LOST	05929000
000246	50	C0	C45E	000464		4367		ST R#12,SAVE	BY AIDSYS02,AND SAVE R#12,DESTROYED	05939000
00024A	58	C0	C446	00044C		4368		L R#12,=V(ADSERV)	BY THE CALL TO AIDSYS05(WHICH	05949000
00024E	05	EC				4369		BALR R#14,R#12	EXSPECTS TO HAVE THE RIGHT VALUE IN	05959000
000250	58	CD	0000	000000		4370		L R#12,0(R#13)	REGISTER 12 RESTORE IT.	05969000
000254	58	80	C4EE	0004F4		4371		L R#08,SAVEREG8	RESTORE CALLER PARAMETER LIST	05975000
						4372	*		ADDRESS WHICH HAD BEEN LOOSED	05975400
000258	91	30	8001	000001		4373		TM DPARTH,DPACMPRE	TEST IF AN ERROR HAS OCCURED	05975800
00025C	47	10	C41C	000422		4374		BD RTH	DURING THE READ PAGE ROUTINE	05976600
						4375	*	*****	05989000	
						4376	*		05999000	
						4377	*	WHAT IS DONE HERE IS TO CHECK IF AN ERROR HAS OCCURED	06009000	
						4378	*	DURING THE EXECUTION OF AIDSYS05.THE PROBLEM IS THAT IF	06019000	
						4379	*	THE CALLER HAS SPECIFIED BOTH DOMAIN WANTED,IT IS POSSIBLE	06029000	
						4380	*	THAT WHAT WE ASSUMED FIRST WAS NOT THE CORRECT DECISION,THEN	06039000	
						4381	*	A FURTHER CALL IS NEEDED.IF IT IS NOT THE CASE,THEN IT IS	06049000	
						4382	*	A REAL ERROR AND THEN HAS TO BE PROCESSED LIKE A COMMON	06050000	
						4383	*	ERROR.IT IS ALSO THE PROBLEM OF TRANSLATING THE RETURN	06052000	
						4384	*	CODE OF AIDSYS05 IN THE RETURN CODE OF THIS TQGL.	06053000	
						4385	*		06059000	
						4386	*	*****	06069000	
000260	91	40	CE5E	000E64		4387		TM ASUB,DPABOTH	TEST IF THERE WAS BOTH USER	06089000
						4388	*		AND SYSTEM WANTED.	06099000
000264	47	E0	C2A2	0002A8		4389		BNG ERR	DON'T CARE ABOUT THIS	06109000
000268	91	08	1013	000013		4390		TM ASARETRN,ASADSERN	IF BOTH SET THE ONLY THREE	06129000
						4391	*		CASES TO TAKE IN CONSIDERATION	06139000
						4392	*		HAVE AS COMMON BIT SET IN THE	06149000
						4393	*		RETURN CODE IS X'08' WHICH	06159000
						4394	*		IS TESTED HERE,OTHERWISE	06169000

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE	STATEMENT			
00026C	47	E0	C2A2	0002A8		4395		BNG	ERR		IT IS A REAL ERROR	06179000
000270	97	40	CE5E	000E64		4396		XI	ASUB,DPABOTH		RESET THE FLAG BECAUSE	06189000
						4397	*				FIRST TRY WAS UNSUCCESSFULL	06199000
000274	17	DD				4398		XR	R#13,R#13		CLEAR R#13 FOR SECURITY	06209000
000276	43	D0	8028	000028		4399		IC	R#13,DPAFLAG		TAKE THE FLAG SET FOR USER	06219000
00027A	88	D0	0001	000001		4400		SRL	R#13,1		FOR SEARCH AND SET IT FOR	06229000
00027E	89	D0	0001	000001		4401		SLL	R#13,1		SEARCHING FOR SYSTEM	06239000
000282	96	08	000D	00000D		4402		GI	R#13,DPASYST			06249000
000286	42	D0	101B	00001B		4403		STC	R#13,ASADSUSE			06259000

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMT	M	SOURCE	STATEMENT	
						4404		*****		06260200
						4405		*		06260800
						4406	*	THE SUPPOSITION DONE DURING THE FIRST CALL WAS WRONG		06261500
						4407	*	IT WAS NOT A NON PRIVILEGE ACCESS TO BE DONE(OR IT IS		06262100
						4408	*	POSSIBLY A MISTAKE) THEN THE NEXT CALL IS DONE WITH		06262700
						4409	*	A PRIVILEGED ACCESS IF THIS ALSO FAILS, THEN RETURN WITH		06263300
						4410	*	THE APPROPRIATE RETURN CODE.		06264000
						4411	*			06264600
						4412		*****		06265200
00028A	41	D0	C45E	000464		4413		LA R#13,SAVE	RESTORE ADDRESS OF THE SAVE	06266500
00028E	50	C0	C45E	000464		4414		ST R#12,SAVE	AREA LGST BY AIDSYS05 AND	06267100
000292	58	C0	C446	00044C		4415		L R#12,=V(ADSERV)	SAVE R#12 DESTROYED BY THE	06267700
000296	05	EC				4416		BALR R#14,R#12	CALL TO AIDSYS05(THINKING TO	06268300
000298	58	CD	0000	000000		4417		L R#12,0(R#13)	HAVE THE RIGHT VALUE IN R#12)	06269000
00029C	58	80	C4EE	0004F4		4418		L R#08,SAVEREG8	RESTORE A(CALLER PARAMETER	06270000
						4419	*		LIST) HAD BEEN LOADED	06271000
0002A0	91	30	8001	000001		4420		TM DPARTN,DPAGAPRE	TEST IF AN ERROR IN THE	06273000
0002A4	47	10	C41C	000422		4421		BT RTN	CALLER PAGE READ ROUTINE HAS	06275000
						4422	*		OCCURED. IF YES, RETURN TO HIM	06277000
						4423	*		DIRECTLY FLAG WAS YET SET.	06278000
						4424		*****		06280000
						4425	*			06281000
						4426	*	THE NEXT TEST ARE THE COMMON TEST DONE TO SEE IF SOME		06282000
						4427	*	ERRORS HAD OCCURED DURING THE PERFORMING OF AIDSYS05		06284000
						4428	*	THESE TESTS ARE DONE IF ONLY ONE CALL WAS DONE.		06285000
						4429	*	BUT ALSO IF, AFTER THE SECOND CALL OF AIDSYS05(IF ANY),		06285090
						4430	*	AN ERROR WAS DISCOVERED WHEN EXECUTING IT TWICE(WHICH WILL		06285180
						4431	*	TELL US THAT NOTHING WAS FOUND).		06285270
						4432	*			06285360
						4433		*****		06285000
0002A8						4434		ERR DS OY		06289000
						4435	*			06294000
						4436	*			06296000
0002A8	91	FF	1084	000084		4437		TM ASADSFBR,DPARTNT	X'FF' NO MORE INFO	06299000
0002AC	47	80	C2AE	0002B4		4438		BZ REALERR	TEST IF ASARETRN IS SET	06309000
						4439	*			06314000
0002B0	92	08	1013	000013		4440		MVI ASARETRN,ASADSMOR	SET BUFFER OVFLW	06319000

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE	STATEMENT	
0002B4						4441		REALERR	DS 0Y	06329000
0002B4	91	FF	1013	000013		4442			TM ASARETRN,DPARTNT	06339000
0002B8	47	80	C41C	000422		4443			BZ RTN	06349000
						4444	*			06354000
0002BC	41	30	CE62	000E68		4445		LA	R#03,TABCMFCT	06359000
0002C0	47	F0	C3EE	0003F4		4446		B	ERRHANDL	06369000
									CONVERSION TABLE FOR ERROR HANDLING	
									GOTO COMMON ROUTINE FOR ERROR	

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMT	M	SOURCE STATEMENT	
0002C4				4447		GAFCT		DS OY	06609000
				4448		*****		*****	06619000
				4449		*		*	06629000
				4450		*		*****	06639000
				4451		**		**	06649000
				4452		**		**	06659000
				4453		**		THIS PART IS RESPONSABLE OF THE EXECUTION OF GA FUNCTION	06669000
				4454		**		**	06679000
				4455		**		IT IS EXPECTED THAT THE CALLER GIVES :	06689000
				4456		**		**	06699000
				4457		**		AT DISPLACEMENT 0 NUMBER OF THE FUNCTION,HERE X'04'	06709000
				4458		**		**	06719000
				4459		**		AT DISPLACEMENT 1 RETURN CODE FIELD.	06729000
				4460		**		**	06739000
				4461		**		AT DISPLACEMENT 4 DUMPFIL LINK NUMBER.	06749000
				4462		**		**	06759000
				4463		**		AT DISPLACEMENT 8 ADDRESS OF THE DUMPFIL READ ROUTINE	06769000
				4464		**		**	06770000
				4465		**		AT DISPLACEMENT 12 ADDRESS OF THE TCB.	06771000
				4466		**		**	06772000
				4467		**		AT DISPLACEMENT 16 ADDRESS OF SYSBASE .	06773000
				4468		**		**	06774000
				4469		**		AT DISPLACEMENT 20 ADDRESS OF XVT.	06775000
				4470		**		**	06776000
				4471		**		AT DISPLACEMENT 24 INDICATOR OF DUMPFIL TYPE	06777000
				4472		**		**	06778000
				4473		**		AT DISPLACEMENT 28 ITN NUMBER,IF ANY	06778100
				4474		**		**	06778200
				4475		**		AT DISPLACEMENT 40 THE FLAG BYTE	06778300
				4476		**		**	06778500
				4477		**		AT DISPLACEMENT 44 THE GIVEN ADDRESS	06778600
				4478		**		**	06779000
				4479		**		**	06789000
				4480		*		*****	06799000
				4481		*		*	06809000
				4482		*		*	06819000
				4483		*****		IT IS NOT CONTROLLED	06829000
				4484		*****		*****	06839000
				4485		*****		THAT THE USER HAS	06849000
				4486		*****		*****	06859000
				4487		*****		GIVEN THE PLACE	06869000
				4488		*****		*****	06879000
				4489		*****		ENOUGH FOR THE RETURNED	06889000
				4490		*****		*****	06899000
				4491		*****		INFORMATION ! ! ! ! !	06909000
				4492		*****		*****	06919000
				4493		*****		*****	06929000
				4494		*****		*****	06939000
				4495		*****		*****	06949000
				4496		*****		*****	06959000

FLAG	LOC	CTN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE	STATEMENT		
0002C4	41	10	C4F2		0004F8		4497		LA	R#01,WORKAREA	COVER PARAM LIST	06979000
000000							4498		USING	SERVE,R#01		06989000
0002C8	92	4C	1011		000011		4499		MVI	ASAFCT,ASADS	MØVE WHICH FUNCTION HAS TØ BE CALLED	06999000
							4500	*				07004000
0002CC	92	08	1012		000012		4501		MVI	ASASBFCT,ASADSCFA	SUBFUNCTION ENVIRONMENT	07009000
							4502	*				07014000
0002D0	D2	05	1004C44A		000004	000450	4503		MVC	ASACALL1(6),=C'AIDSYS'	SIMULATE THE NORMAL CALL	07019000
							4504	*				07024000
0002D6	92	10	101B		00001B		4505		MVI	ASADSUSE,ASADUMP	DUMPFIL ACCESS	07029000
							4506	*				07034000
0002DA	92	00	1034		000034		4507		MVI	ASADSLEV,X'00'	CLEAR THIS FIELD	07039000
							4508	*				07044000
0002DE	D2	03	102C802C		00002C	00002C	4509		MVC	ASADSADR(4),DPAGAVA	USER VIRTUAL ADDRESS WANTED	07049000
							4510	*				07049050
0002E4	D2	00	101A8007		00001A	000007	4511		MVC	ASADSDFN(4-DPALNKL),DPALINK+DPALNKL		07049100
							4512	*				07049200
0002EA	D2	00	1019801F		000019	00001F	4513		MVC	ASADSITN+4-DPAITNL(4-DPAITNL),DPAITN+DPAITNL		07049300
0002F0	91	40	CE5E		000E64		4514		TM	ASUB,DPABØTH	TEST IF BØTH PRIVILEGED AND	07050000
							4515	*			NON PRIVILEGED FLAG IS SET	07051000
0002F4	47	10	C414		00041A		4516		BØ	RTNAIC	IF YES,BRANCH TØ ADDRESS IN	07052000
							4517	*			CØNFLECT	07053000
							4518	*			WITH FLAG IS SET	07054000
0002F8	D6	00	101B8028		00001B	000028	4519		BØ	ASADSUSE,DPAFLAG	SET EXPECTED FLAG FROM AIDSYS05	07059000

FLAG	LOC TN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT	
						4520		*****	07079000
						4521	*		* 07089000
						4522	*	MYDYNDTA WILL SERVE AS BASE FOR DYNDATA	* 07099000
						4523	*		* 07109000
						4524		*****	07119000
0002FE	58	40	C432	000438		4525		L R#04,=(MYDYNDTA) COVER THE EXPECTED DYNDATA	07139000
000000						4526		USING DYNDATA,R#04 COVER DYNDATA IN ORDER TO BE ABLE	07140000
000302	D2	03	49408014	000940	000014	4527		MVC DXVTBASE,DPAAXVT TO SET FIELDS WHICH ARE NORMALLY	07142000
						4528	*		07142100
000308	50	10	40F0	0000F0		4529		ST R#01,APARAM FIELD USED IF SYSTEM DOMAIN FLAG SET	07142200
						4530	*		07142300
00030C	D2	03	493C8010	00093C	000010	4531		MVC DSYSBASE,DPASYSBS	07142600
						4532	*		07143000
000312	D2	03	4A7C800C	000A7C	00000C	4533		MVC DUMPATCB,DPAATCB MOVE IMPORTANT IF AN ITN IS GIVEN	07144000
						4534	*		07144500
000318	D2	03	4008C436	000008	00043C	4535		MVC ARET,=(WORKAREA+2000) SUPPOSED BEEN SET WHEN USED BY	07145000
						4536	*	AS4ITNDP AND BY AS4RDPGE	07146000
						4537		*****	07149000
						4538	*		* 07159000
						4539	*	MYSTATDTA COVER THE STATDATA	* 07169000
						4540	*		* 07179000
						4541		*****	07189000
00031E	58	50	C43E	000444		4542		L R#05,=(MYSTATDT) REGISTER 4 HAS ALWAYS TO COVER	07209000
						4543	*	DYNDATA,AND REGISTER 5 HAS ALWAYS	07219000
						4544	*	TO COVER STATDATA!!!!!!!!!!	07229000
						4545	*	LA R#13,SAVE1 LOAD A NEW SAVE AREA	07239000
000322	41	10	C4F2	0004F8		4546		LA R#01,WORKAREA REGISTER 1 WAS DESTROYED	07549000
000326	41	D0	C45E	000464		4547		LA R#13,SAVE	07559000
00032A	50	C0	C45E	000464		4548		ST R#12,SAVE REGISTER 12 IS BASE REGISTER FOR A5	07569000

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT	
						4549		*****	07589000
						4550		*	07599000
						4551		* ALL THINGS ARE READY NOW FOR THE FIRST CALL TO AIDSYS05	07609000
						4552		* THERE WILL BE ANOTHER ONE BECAUSE THE FIRST GET THE	07619000
						4553		* POINTERS AND THE SECOND ONE GETS THE INFORMATION.	07629000
						4554		*	07639000
						4555		*****	07649000

FLAG	LCTN	OBJECT	CODE	ADDR1	ADDR2	STMT	M	SOURCE	STATEMENT	
						4556		*****		07659400
						4557	*			* 07659900
						4558	*	WHAT IS DONE HERE IS TO SIMULATE THE DISPOSITION OF DATAS		* 07660400
						4559	*	WHICH ARE NORMALLY PROVIDED BY AIDSYS02 AND WHICH HERE ARE		* 07660900
						4560	*	PROVIDED DIRECTLY BY THE CALLER.		* 07661300
						4561	*			* 07661800
						4562	*	*****		* 07662300
000000						4563		USING STATDATA,R#05	COVER STATDATA FORT FINDING	07663700
						4564	*		A(DFTAB)	07664200
						4565		AIDSCOD GETDFTAB,P1=R#06,P2=R#07,P3=ASADSDFN,P4=DFTAB		07664700
						4566	1	IDLKG VER=009		
						4567	2	*,VERSION 009		00001300
00032E	0700					4568	2	CNDP 0,4		00002800
000330						4569	2	DS OF		00003500
000330	17 66					4570	1	X2 R#06,R#06	CLEAR REG	
000332	43 60	101A		00001A		4571	1	IC R#06,ASADSDFN	GET LINK #	
000336	89 60	0003		000003		4572	1	SLL R#06,3	REL. A(ENTRY IN DF-TABLE)	
00033A	41 70	5260		000260		4573	1	LA R#07,DFTAB	A(DF-TABLE)	
00033E	1A 67					4574	1	AR R#06,R#07	A(ENTRY IN DF-TABLE)	
000340	50 60	4278		000278		4575	1	ST R#06,DYDFPBIT	SAVE ADDRESS	
000344	D2 03	6000C442		000448		4576		MVC 0(4,R#06),=A(WORKAREA+1024)		07664900
						4577	*			07665000
00034A	58 70	6000				4578		L R#07,0(0,R#06)	LOAD A(DUMPFIL-WORKAREA)	07665100
00034E	12 77					4579		LTR R#07,R#07	DUMPFIL OPEN?	07665600
000350	47 80	C172		000178		4580		BZ RTNDFE	NO,GO AHEAD	07666100
000354	50 70	40EC		0000EC		4581		ST R#07,DWORK		07666600
000000						4582		USING WORK,R#07		07667000
000358	D2 03	72D88010		0002D8	000010	4583		MVC WAVIRT,DPASYSBS	STORE A(SYSBASE) IN THE RIGHT	07667500
						4584	*		PLACE SO THAT AIDSYS05 WILL GET	07668000
						4585	*		IT WITHOUT TROUBLE AS IF IT WAS	07668500
						4586	*		GONE THROUGH THE USUAL WAY.	07668580
00035E	D2 00	730B801B		00030B	00001B	4587		MVC WAIND,DPAIND+DPAINDL	WAIND IS ONLY A BYTE BUT DPAIND	07668660
						4588	*		IS A FULLWORD	07668740
000364	D2 03	72F48014		0002F4	000014	4589		MVC WAXVT,DPAAXVT		07668830
						4590		DROP R#04,R#05,R#07		07668910

FLAG LCTN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

```

4591 ***** 07669000
4592 ***** 07679000
4593 *** 07689000
4594 *** BETWEEN THE CALL OF AIDSYS05 *** 07699000
4595 *** 07709000
4596 *** DO NOT CHANGE THE CONTAIN *** 07719000
4597 *** OF THE REGISTERS. *** 07729000
4598 *** 07739000
4599 *** 07749000
4600 *** ALL ARE USED WITH THE VALUE *** 07759000
4601 *** 07769000
4602 *** LEFT IN IT BY THE LATEST *** 07779000
4603 *** 07789000
4604 *** AIDSYS CALL THAT WOULD BE *** 07799000
4605 *** 07809000
4606 *** A D E S A S T R O U S T H I N G ! ! ! *** 07819000
4607 *** 07829000
4608 *** 07839000
4609 ***** 07849000
4610 ***** 07859000
    
```

```

00036A 58 C0 C446 00044C 4611 L R#12,=V(ADSERV) CALL NOW AIDSYS05 07879000
00036E 05 EC 4612 BALR R#14,R#12 AIDSYS05,EXSPECT THE RIGHT VALUE 07889000
4613 * IN REGISTER 12 07899000
000370 58 CD 0000 000000 4614 L R#12,0(R#13) DGN'T TOUCH AT THIS,THIS IS DANGER. 07909000
    
```

```

000374 58 80 C4EE 0004F4 4615 L R#08,SAVEREG8 RESTORE A(CALLER PARAMETER LIST) 07912000
000378 91 30 8001 000001 4616 TM DPARTN,DPAGAPRE TEST IF AN ERROR HAS OCCURED WHEN 07915000
00037C 47 10 C41C 000422 4617 BR RTN EXECUTING THE CALLER PAGE READ 07917000
4618 * RETURN TO HIM DIRECTLY FLAG WAS YET 07918000
4619 * CORRECTLY SET 07918500
    
```

```

4620 ***** 07929000
4621 * 07939000
4622 * TEST IF NO ERRORS HAD OCCURED DURING THE EXECUTION OF 07949000
4623 * AIDSYS05,OTHERWISE IT IS NOT NECESSARY TO CALL IT AGAIN 07959000
4624 * KNOWING THAT SOMETHING HAS OCCURED IN IT. 07969000
4625 * 07979000
4626 ***** 07989000
    
```

INTERFACE BETWEEN USER AND AIDSYS

15:36:23 84-12-13 PAGE 0113

FLAG LCTN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

000388	41	30	CE76	000E7C		4629	*	LA	R#03, TABGAFCT	LA CONVERSION TABLE FOR ERROR	08026000
00038C	47	F0	C3EE	0003F4		4630		B	ERRHANDL	GG TO DG THE CONVERSION	08029000
						4631					08039000

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE STATEMENT	
						4632		*****	08059000
						4633	*		* 08069000
						4634	*	NOW THE DESCRIPTOR IS CREATED ,WE CAN THEN CALL AGAIN	* 08079000
						4635	*	AIDSYS05 IN ORDER TO GET OTHERS INFORMATION.	* 08089000
						4636	*		* 08099000
						4637		*****	08109000
000390						4638		PREPARTN DS OY	08129000
000390	41 10	C4F2		0004F8		4639		LA R#01,WORKAREA	08139000
						4640	*	RESTORE REGISTER 1 IN ORDER TO COVER PARAMLISTR FOR SECOND CALL	08149000
000000						4641		USING SERVE,R#01	08159000
000394	92 4C	1011		000011		4642		MVI ASAFCT,ASADS	08169000
000398	96 04	1012		000012		4643		GI ASASBFCT,ASADSCFN	08179000
						4644	*	ENVIRONMENT FUNCTION CALL SUBFUNCTION,WITH THE PARAMS FROM LAST CALL TO AIDSYS05	08189000
00039C	41 D0	C45E		000464		4645		LA R#13,SAVE	08199000
0003A0	50 C0	C45E		000464		4646		ST R#12,SAVE	08209000
						4647		*****	08229000
						4648	*		* 08239000
						4649	*	THE PARAMETERS ARE READY FOR THE NEXT CALL.	* 08249000
						4650	*		* 08259000
						4651		*****	08269000
0003A4	58 C0	C446		00044C		4652		L R#12,=V(ADSERV)	08289000
0003A8	05 EC			000000		4653		BALR R#14,R#12	08299000
0003AA	58 CD	0000		000000		4654		L R#12,0(R#13)	08309000
								AIDSYS05 HAS R12 AS BASE REGISTER BUT DOES NOT LOAD IT DON'T TOUCH AT THIS,THIS IS DANGER.	
0003AE	58 80	C4EE		0004F4		4655		L R#08,SAVEREG8	08312000
0003B2	91 30	8001		000001		4656		TM DPARTN,DPAGAPRE	08315000
0003B6	47 10	C41C		000422		4657		BG RTN	08317000
						4658	*	TEST IF AN ERRGR HAS OCCURED WHEN EXECUTING THE CALLER PAGE READ ROUTINE.THE FLAG IS YET SET.RETURN	08318000

AG	LOC	CTN	OBJ	CODE	ADDR1	ADDR2	STMT	M	SOURCE STATEMENT	
							4659		*****	08329000
							4660		*	08339000
							4661		* TEST TO SEE IF NO ERROR HAPPENED, IF THERE ARE SOME, THEN	08349000
							4662		* GO TO THE ROUTINE FOR ERROR HANDLING.	08359000
							4663		*	08369000
							4664		*****	08379000
0003BA	91	FF	1013		000013		4665		TM ASARETRN,DPARTNT TEST TO SEE IF NO PROBLEMS	08399000
0003BE	47	80	C3C4		0003CA		4666		BZ MOVERESU NO PROBLEM, RETURN	08409000
							4667	*		08416000
0003C2	41	30	CE76		000E7C		4668		LA R#03,TABGAFCT CONVERSION TABLE FOR ERROR HANDLING	08419000
0003C6	47	F0	C3EE		0003F4		4669		B ERRHANDL GO TO COMMON ROUTINE FOR ERROR	08429000
0003CA							4670		MOVERESU DS OY	08449000
							4671		*****	08459000
							4672	*		08469000
							4673	*	MOVE THE VALUES INTO THE EXPECTED ZONE	08479000
							4674	*		08489000
							4675		*****	08499000
0003CA	58	D0	C462		000468		4676		L R#13,SAVE+4 RESTORE ADDRESS OF CALLER SAVEAREA	08519000
0003CE	58	8D	0018		000018		4677		L R#08,24(R#13) TO FIND THE ADDRESS OF CALLER PARAMS	08529000
0003D2	D2	07	8030105C	000030	00005C		4678		MVC DPAGASEC,ASADSCSN MOVE NAME OF CSECT MODULE	08539000
							4679	*		08544000
0003D8	D2	03	80381064	000038	000064		4680		MVC DPAGASTA,ASADSCSA MOVE START ADDRESS OF CSECT	08549000
							4681	*		08554000
0003DE	D2	03	80401080	000040	000080		4682		MVC DPAGACSL,ASADSCSL MOVE CSECT LENGTH	08559000
0003E4	58	B0	802C		00002C		4683		L R#11,DPAGAVA ADDRESS GIVEN BY THE CALLER	08569000
0003E8	5B	B0	1064		000064		4684		S R#11,ASADSCSA SUBTRACT FROM ADDRESS GIVEN THE	08579000
							4685	*	START ADDRESS OF THE CSECT FOUND	08589000
0003EC	50	B0	803C		00003C		4686		ST R#11,DPAGAREL PUT IT IN RETURN PARAMS	08599000

INTERFACE BETWEEN USER AND AIDSYS

15:36:23 84-12-13 PAGE 0116

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE	STATEMENT		
						4691		*****		08659000	
0003F0	47	F0	C41C	000422		4692		B	RTN	ALL IS O.K	08679000

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMT	M	SOURCE STATEMENT	
						4693		*****	08679600
						4694		*	08679900
						4695		*	08680200
						4696		* THE FOLLOWING IS DONE IN ORDER TO TRANSLATE THE RETURN CODE *	08680600
						4697		* RETURNED BY THE AIDSYS05 INTO THE RETURN CODE EXPECTED BY *	08680900
						4698		* THE CALLER. THIS IS CALLED BY ALL FUNCTION WHICH CALLED *	08681200
						4699		* AIDSYS05. *	08681500
						4700		*****	08681900
						4701		ERRHANDL DS OY	08682500
0003F4						4702		XR R#02,R#02	08682800
0003F4	17	22				4703		IC R#02,ASARETRN	08683100
0003F6	43	20	1013	000013		4704		SRL R#02,2	08683500
0003FA	88	20	0002	000002		4705		AR R#03,R#02	08683800
0003FE	1A	32				4706		*	08683900
						4707		TM 0(R#03),DPARTNT	08684100
000400	91	FF	3000			4708		BQ MODULERR	08684400
000404	47	10	C40C	000412		4709		*	08684600
						4710		MVC DPARTN(1),0(R#03)	08684800
000408	D2	00	80013000	000001		4711		*	08684900
						4712		B RTN	08685100
00040E	47	F0	C41C	000422					

INTERFACE BETWEEN USER AND AIDSYS

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMT	M	SOURCE	STATEMENT	
						4713		*****		08685170
						4714		*		08685210
						4715		*	YOU ARE HERE ONLY IF THERE IS AN UNEXPECTED RETURN CODE	08685250
						4716		*	FROM AIDSYS05.	08685280
						4717		*		08685320
						4718		*****		08685360
000412						4719		MODULERR	DS 0Y	08685400
000412	92	08	8001	000001		4720		MVI	DPARTN,DPAMODE	08685700
000416	47	F0	C41C	000422		4721		B	RTN	08686000
									MOVE VALUE IN RETURN CODE ZONE	
									BRANCH TO COMMON EXIT	

FLAG	LOCN	OBJECT	CODE	ADDR1	ADDR2	STMNT	M	SOURCE	STATEMENT			
						4722		*****		08686600		
						4723	*			08586800		
						4724	*	INPUT ADDRESS IN CONFLICT WITH FLAG, THAT WILL SAY THAT		08687100		
						4725	*	THE GIVEN ADDRESS IS IN SYSTEM ZONE AND THE FLAG IS SET		08687300		
						4726	*	TO USER ZONE OR IS IN USER ZONE AND THE FLAG IS SET TO		08687500		
						4727	*	SYSTEM SPACE TO BE ACCESSED. THE SECOND CASE IS WHEN THE		08687800		
						4728	*	CALLER SPECIFIES BOTH SYSTEM AND USER IS WANTED WHEN CALLING		08688000		
						4729	*	THE GET ADDRESS FUNCTION. THE RETURN CODE IS X' '		08688200		
						4730	*			08688500		
						4731		*****		08688700		
00041A						4732		RTNAIC	DS	0Y	08688770	
00041A	92	2C	8001	000001		4733			MVI	DPAGARTN, DPAGAIAP	MOVE VALUE IN RETURN CODE ZONE	08688800
00041E	47	F0	C41C	000422		4734			B	RTN	RETURN TO THE CALLER	08688850

FLAG	LOCN	OBJECT CODE	ADDR1	ADDR2	STMT	M	SOURCE STATEMENT	
					4735		*****	08699000
					4736	*		* 08709000
					4737	*	HERE IS THE ONLY ONE POINT WHERE YOU CAN GET OUT OF	* 08719000
					4738	*	ALL THE MODULE.	* 08729000
					4739	*		* 08739000
					4740		*****	08749000

000422					4741	RTN	DS	OY		08769000
000422	58	DO	C462	000468	4742		L	R#13,SAVE+4	RESTORE REGISTERS OF CALLING PGM	08779000
000426	98	EC	D00C		4743		LM	R#14,R#12,12(R#13)		08789000
00042A	07	FE			4744		BR	R#14		08799000
					4745		LTORG			08809000
000430	00000000				4746			=A(0)		
000434	00000900				4747			=F'0'		
000438	00000E90				4748			=A(MYDYNDA)		
00043C	00000CC8				4749			=A(MGRKAREA+2000)		
000440	000022F0				4750			=A(MRKETPND)		
000444	00001BF0				4751			=A(MYSTATDT)		
000448	000008F8				4752			=A(MGRKAREA+1024)		
00044C	00000000				4753			=V(ADSERV)		
000450	C1C9C4E2E8E2				4754			=C'AIDSYS'		

FLAG	LOC TN	OBJECT CODE	ADDR1	ADDR2	STMT M	SOURCE	STATEMENT		
000458	00000000				4755	SYSBASE DC	A(0)	SIMULATION OF SYSTEM START ADDRESS	08820000
00045C	00000000				4756	AREADRT DC	A(0)	A(CALLER DUMPFIL READ ROUTINE)	08821000
000460	00000000				4757	EXVTBASE DC	A(0)	A(XVT)	08822000
000464					4758	*AVE1 DS	18F	SAVEAREA WHEN CALLING AIDSYS02	08829000
00046C					4759	SAVE DS	18F	SAVEAREA WHEN CALLING AIDSYS05	08839000
					4760	SAVE3 DS	18F	SAVEAREA USED TO SAVE REGISTERS	08844000
					4761	*		WHEN CALLING AS4RDPGE	08846000
0004F4					4762	SAVEREG8 DS	F	SAVE IF TWO CALL TO AS4ITNDP !!	08847000
0004F8					4763	WORKAREA DS	600F	COVERING PARAMETER LIST, A(RET) STACK	08849000
000E58			000E58		4764	TAB DS	4F	THIS TABLE CONTAINS THE MINIMAL	08859000
					4765	GRG	TAB	VALUE OF BUFFER LENGTH	08869000
000E58	00000021				4766	CMLN DC	F'33'	MINIMAL VALUE FOR CMFCT	08879000
000E5C	00000014				4767	GALEN DC	F'20'	MINIMAL VALUE FOR GAFCT	08889000
					4768	*	F'0'	NO MORE USED WAS USED FOR CRFCT	08899000
000E60	00000013				4769	PULEN DC	F'19'	MINIMAL VALUE FOR PUFCT(+CRFCT)	08909000
					4770	*LAGUS DS	X	NO MORE USED	08919000
					4771	*ABALPHA DS	XL41	NO MORE USED WAS USED ONLY TO FIND	08929000
					4772	*	TABALPHA	IF THE FIRST CHARACTER OF THE FILE	08939000
					4773	*	DC	X'AABBCCDDEEFF0F1F200000000000000' NAME GIVEN WHEN	08949000
					4774	*	DC	X'F3F4F5F6F7F8F9FAFB00000000000000' THE CALLER HAD TO	08959000
					4775	*	DC	X'00FCFDFFFAABBCCDD' TO GIVE US A A(FCB) WAS CORRECT	08969000
000E64	0000				4776	ASUB DC	H'0'	FIELD COVER TWO DISTINCT THINGS:	08979000
					4777	*		1. IF THE CALLER SPECIFIES BOTH DOMAIN.	08984000
					4778	*		2. MANAGERING THE TRANSFORMATION OF FLAG	08986000
000E68					4779	TABCMFCT DS	0F	RETURN CODE TRANSLATION FOR CMFCT	08989000
000E68	000428FFFF382C48				4780	DC	X'000428FFFF382C4810FF1444184CFFFFFFF3C50'	08999000	
000E7C					4781	TABGAFCT DS	0F	RETURN CODE TRANSLATION FOR GAFCT	09009000
000E7C	0004FF28FF38FFFF				4782	DC	X'0004FF28FF38FFFF102C1434183CFFFFFFF40'	09019000	
000E90			001BF0		4783	MYDYNDTA DS	0F	SPACE RESERVED FOR COVERING DYNDATA	09049000
					4784	GRG	*+DYNLEN		09059000

FLAG	LOCN	OBJECT CODE	ADDR1	ADDR2	STMNT M	SOURCE STATEMENT	
0022F0					4787	WRKETPND DS OF	09085000
			0227F0		4788	GRG **X'20500'	09086000
			000422		4789	RTNISFN EQU RTN	09089000
			000422		4790	PEFCT EQU RTN	09109000
					4791	ETPND DPACCESS,VER=800,DATE=850131,PATCH=200,LTORG=NO	09119000
					4792	PRINT ON,GEN	
					4793	1 * THIS MACRO IS USED TO GIVE THE STANDARD TERMINATE FOR A MODULE	
					4794	1 * N MANDATORY POSITIONAL OPERAND 1 - MODULE NAME	
					4795	1 * B OPTIONAL POSITIONAL OPERAND 2 - LENGTH OF MODULE	
					4796	1 * PATCH= OPTIONAL OPERAND - LENGTH OF PATCHAREA	
					4797	1 * (ZERO MAY BE SPECIFIED)	
					4798	1 * DEFAULT - 200 BYTES	
					4799	1 * IF B AND &PATCH PARAMS NOT USED	
					4800	1 * DATE= OPTIONAL OPERAND - DATE (FORMAT YYMMDD)	
					4801	1 * VER= MANDATORY OPERAND - MODULE VERSION NUMBER (3 DIGITS)	
					4802	1 * COMPNR= OPTIONAL OPERAND - TO IDENTIFY COMPONENT NUMBER	
					4803	1 * CSECT= OPTIONAL OPERAND	
					4804	1 * YES (DEFAULT) ORIGINAL CSECT FOR ETPND	
					4805	1 * NO NO CSECT IS CREATED	
					4806	1 * NAME SPECIAL CSECT NAME	
					4807	1 * LTORG= OPTIONAL OPERAND - DEFAULT=YES. NO=NOT IN EXPANSION	
			0227F0		4808	1 GRG	
0227F0	00				4809	1 DC 200X'00'	
0228B8					4810	1 DS CD	
0228B8	C4D7C1C3C3C5E2E2				4811	1 DC CL8'DPACCESS'	MODULE NAME
0228C0	F8F0F0				4812	1 DC CL3'800'	MODULE VERSION NUMBER ASSIGNED
0228C3	C7				4813	1 DC CL1'G'	MACRO LIBRARY VERSION
0228C4	F8F5F0F1F3F1				4814	1 DC CL6'850131'	ASSEMBLY SUBMISSION DATE
0228CA	F3F4F8				4815	1 DC CL3'348'	CURRENT JULIAN ASSEMBLY DATE
0228CD	000000				4816	1 DC AL3(DPACCESS)	LOAD ADDRESS OF MODULE

FLAG LOCN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

```

4817 ***** 09132000
4818 * * 09136000
4819 * WHAT IS DESCRIBED NOW IS THE TRANSLATION TABLES FOR * 09140000
4820 * CHANGING THE RETURN CODE RETURNED BY ADSERV IN RETURN * 09144000
4821 * CODE EXPECTED BY THE USER. * 09148000
4822 * * 09152000
4823 ***** 09155000

```

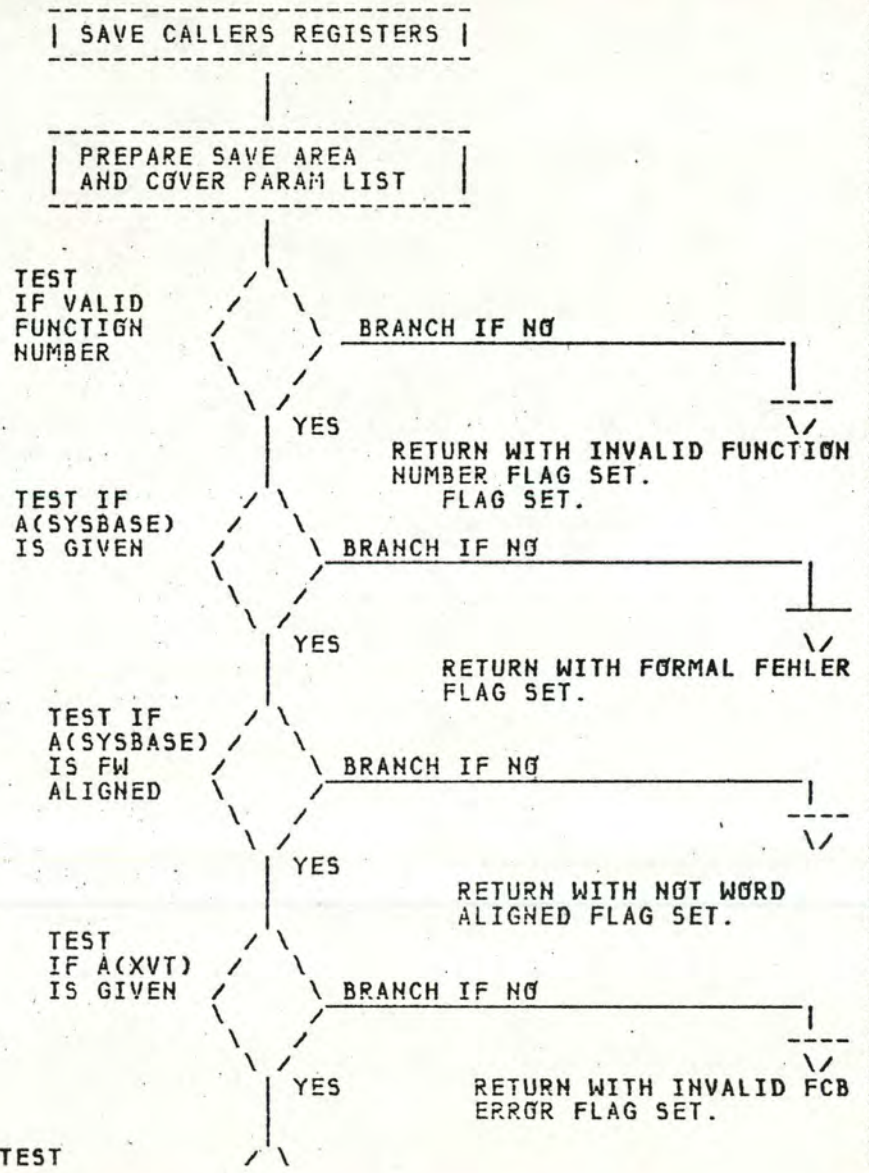
```

4824 ***** 09163000
4825 * * 09167000
4826 * VALUE OF R.C. * VALUE OF R.C. * VALUE OF R.C. * SIGNIFICATION * 09171000
4827 * FROM ADSERV * FROM ADSERV * FROM ADSERV * OF R.C. * 09175000
4828 * TO CMFCT * TO GAFCT * * 09179000
4829 ***** 09182000
4830 * 00 * 00 * 00 * NO ERROR * 09186000
4831 *-----*-----*-----*-----*-----*
4832 * 04 * 04 * 04 * FORMAL FEHLER * 09194000
4833 *-----*-----*-----*-----*-----*
4834 * 08 * 28 * ERROR * BUFFER OVFLW * 09202000
4835 *-----*-----*-----*-----*-----*
4836 * 0C * ERROR * 28 * ADDRESS UNALOC * 09206070
4837 *-----*-----*-----*-----*-----*
4838 * 10 * ERROR * ERROR * UNEXPECTED * 09206150
4839 *-----*-----*-----*-----*-----*
4840 * 14 * 38 * 38 * NO PGM LOADED * 09206230
4841 *-----*-----*-----*-----*-----*
4842 * 18 * 2C * ERROR * BUFFER UNACC * 09206310
4843 *-----*-----*-----*-----*-----*
4844 * 1C * 48 * ERROR * OLDFMT LOADING * 09206390
4845 *-----*-----*-----*-----*-----*
4846 * 20 * 10 * 10 * FILE NOT OPEN * 09206470
4847 *-----*-----*-----*-----*-----*
4848 * 24 * ERROR * 2C * ADDRESS<>FLAG * 09206540
4849 *-----*-----*-----*-----*-----*
4850 * 28 * 14 * 14 * SPECIF TASK NF * 09206620
4851 *-----*-----*-----*-----*-----*
4852 * 2C * 44 * 34 * SYSTEM ERROR * 09206700
4853 *-----*-----*-----*-----*-----*
4854 * 30 * 18 * 18 * TASK SPEC NEC * 09206780
4855 *-----*-----*-----*-----*-----*
4856 * 34 * 4C * ERROR * INFO NOT IN DF * 09207170
4857 *-----*-----*-----*-----*-----*
4858 * 38 * ERROR * ERROR * UNEXPECTED * 09207250
4859 *-----*-----*-----*-----*-----*
4860 * 3C * ERROR * ERROR * UNEXPECTED * 09207330
4861 *-----*-----*-----*-----*-----*
4862 * 40 * ERROR * ERROR * UNEXPECTED * 09207410
4863 *-----*-----*-----*-----*-----*
4864 * 44 * 3C * ERROR * NO CSECT FOUND * 09207480
4865 *-----*-----*-----*-----*-----*
4866 * 48 * 50 * 40 * PAGE NOT DUMPD * 09207560
4867 *-----*-----*-----*-----*-----*
4868 * * * * * 09207640
4869 ***** 09208760

```

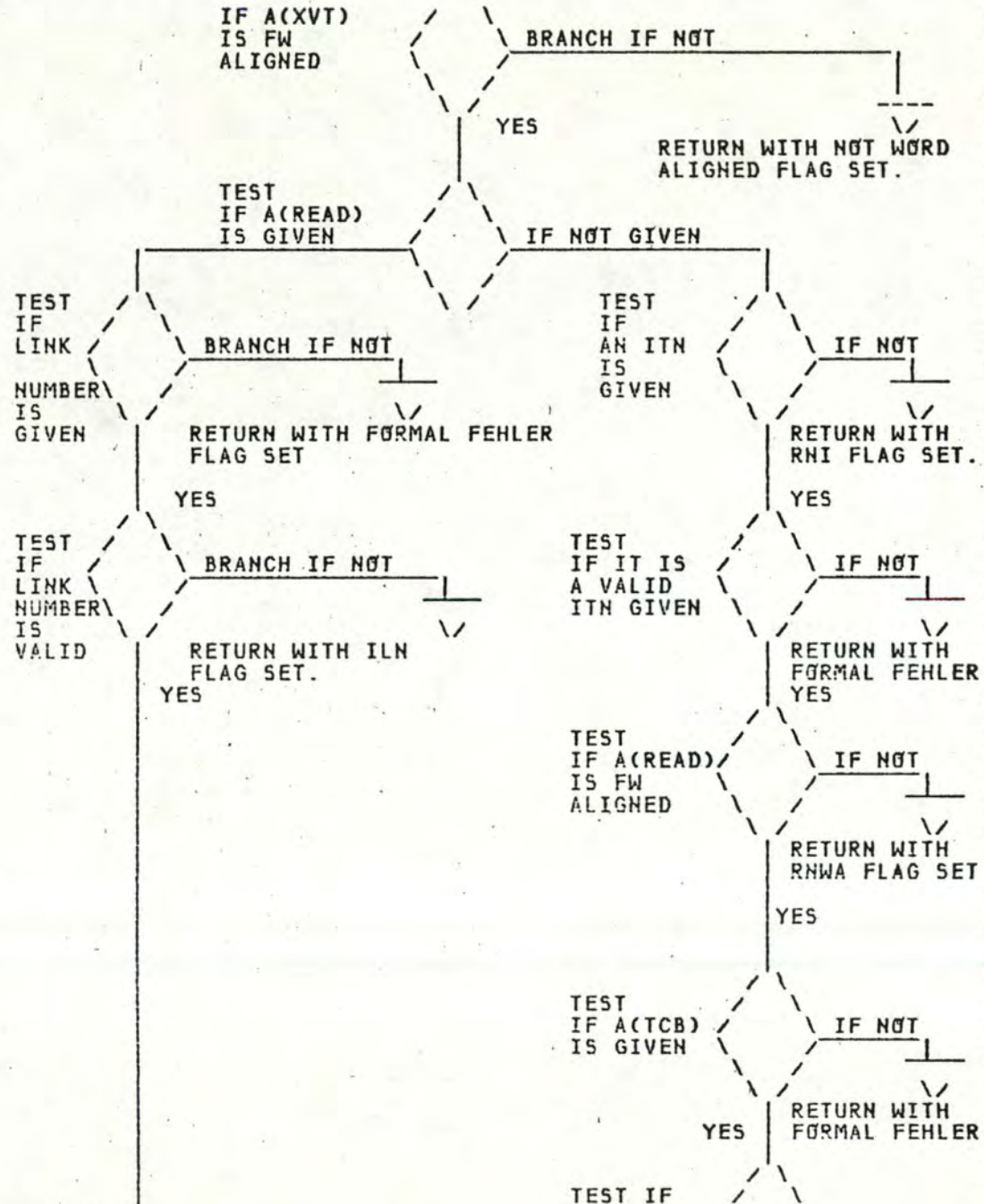
FLAG LOC TN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

ADDR1	ADDR2	STMT	M	SOURCE STATEMENT	ADDRESS
4870			*	*****	09600000
4871			*		09800000
4872			*	THE NEXT PART IS SET IN COMMENTS AND DESCRIBE THE FLOW CHART	10000000
4873			*	OF ALL THE PARTS OF THE MODULE. ALL LOGICAL PART WILL HAVE	10100000
4874			*	IT'S OWN DESCRIPTION IN ONE SHEET OF PAPER.	10300000
4875			*		10500000
4876			*	*****	10700000
4877			*		10900000
4878			*	SAVE CALLERS REGISTERS	11100000
4879			*		11200000
4880			*		11400000
4881			*		11600000
4882			*		11800000
4883			*	PREPARE SAVE AREA	12000000
4884			*	AND COVER PARAM LIST	12100000
4885			*		12300000
4886			*		12500000
4887			*		12700000
4888			*		12900000
4889			*	TEST	13000000
4890			*	IF VALID	13200000
4891			*	FUNCTION	13400000
4892			*	NUMBER	13600000
4893			*		13800000
4894			*	BRANCH IF NO	13900000
4895			*	YES	14100000
4896			*	RETURN WITH INVALID FUNCTION	14300000
4897			*	NUMBER FLAG SET.	14500000
4898			*	FLAG SET.	14700000
4899			*		14900000
4900			*	TEST IF	15000000
4901			*	A(SYSBASE)	15200000
4902			*	IS GIVEN	15400000
4903			*	BRANCH IF NO	15600000
4904			*	YES	15800000
4905			*	RETURN WITH FORMAL FEHLER	15900000
4906			*	FLAG SET.	16100000
4907			*		16300000
4908			*	TEST IF	16500000
4909			*	A(SYSBASE)	16700000
4910			*	IS FW	16800000
4911			*	ALIGNED	17000000
4912			*	BRANCH IF NO	17200000
4913			*	YES	17400000
4914			*	RETURN WITH NOT WORD	17600000
4915			*	ALIGNED FLAG SET.	17700000
4916			*		17900000
4917			*	TEST	18100000
4918			*	IF A(XVT)	18300000
4919			*	IS GIVEN	18500000
4920			*	BRANCH IF NO	18700000
4921			*	YES	18800000
4922			*	RETURN WITH INVALID FCB	19000000
4923			*	ERROR FLAG SET.	19200000
4924			*	TEST	19400000



FLAG LOC TN OBJECT CODE ADDR1 ADDR2 STMNT M SOURCE STATEMENT

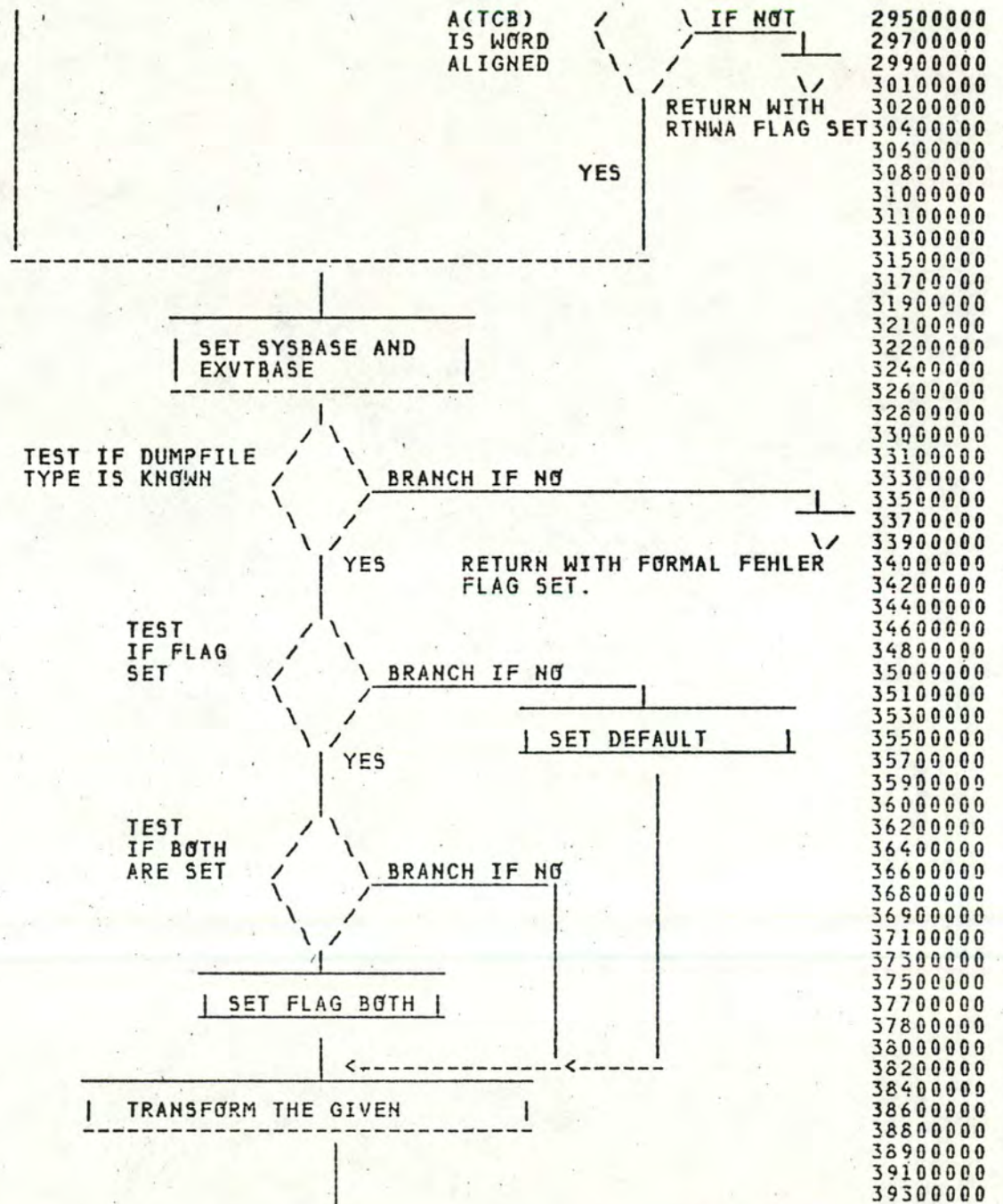
4925 *
4926 *
4927 *
4928 *
4929 *
4930 *
4931 *
4932 *
4933 *
4934 *
4935 *
4936 *
4937 *
4938 *
4939 *
4940 *
4941 *
4942 *
4943 *
4944 *
4945 *
4946 *
4947 *
4948 *
4949 *
4950 *
4951 *
4952 *
4953 *
4954 *
4955 *
4956 *
4957 *
4958 *
4959 *
4960 *
4961 *
4962 *
4963 *
4964 *
4965 *
4966 *
4967 *
4968 *
4969 *
4970 *
4971 *
4972 *
4973 *
4974 *
4975 *
4976 *
4977 *
4978 *
4979 *



19600000
19700000
19900000
20100000
20300000
20500000
20600000
20800000
21000000
21200000
21400000
21600000
21700000
21900000
22100000
22300000
22500000
22600000
22800000
23000000
23200000
23400000
23500000
23700000
23900000
24100000
24300000
24400000
24600000
24800000
25000000
25200000
25400000
25500000
25700000
25900000
26100000
26300000
26400000
26600000
26800000
27000000
27200000
27300000
27500000
27700000
27900000
28100000
28300000
28400000
28600000
28800000
29000000
29200000
29300000

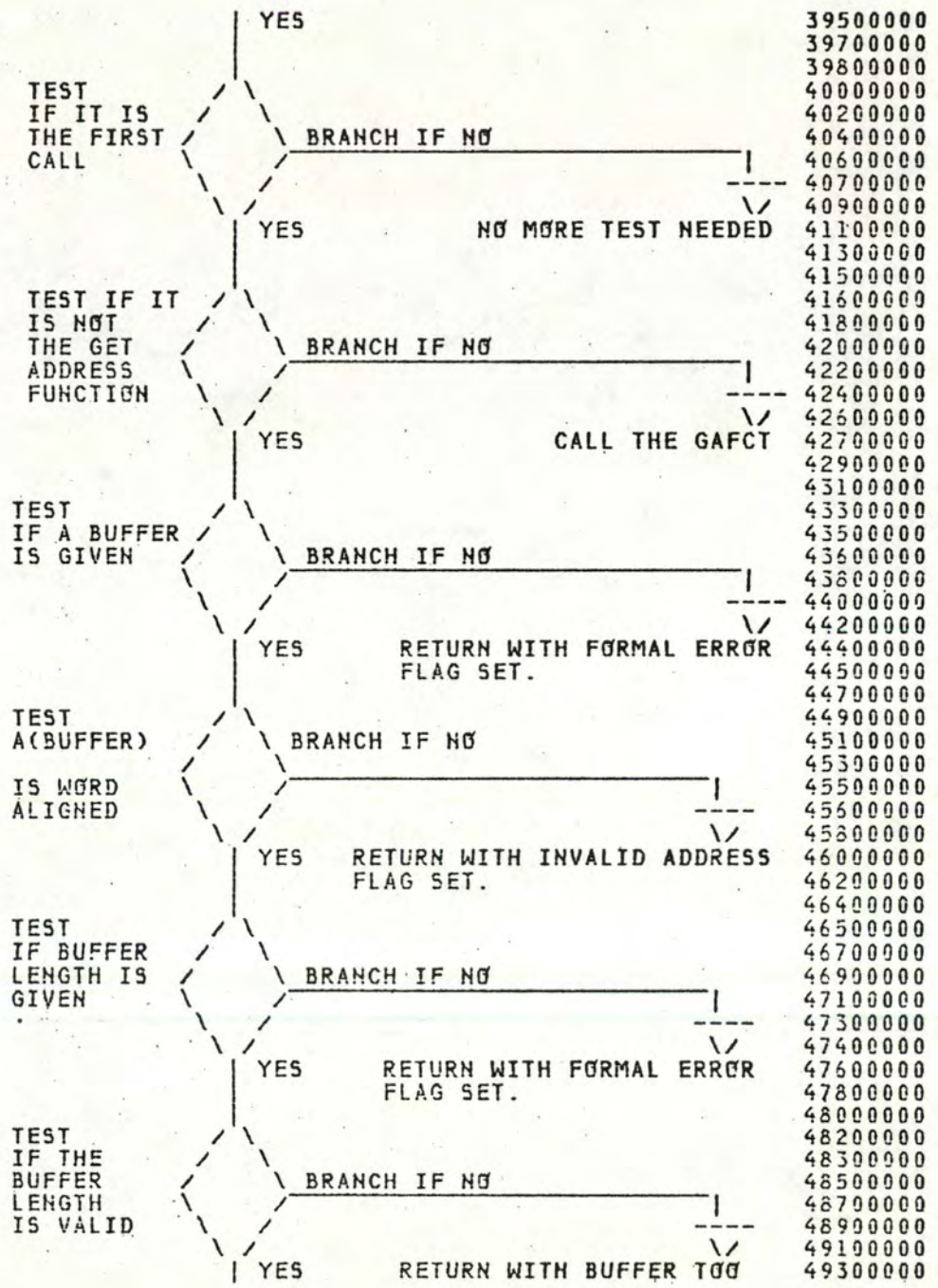
FLAG LOC TN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

4980 *
4981 *
4982 *
4983 *
4984 *
4985 *
4986 *
4987 *
4988 *
4989 *
4990 *
4991 *
4992 *
4993 *
4994 *
4995 *
4996 *
4997 *
4998 *
4999 *
5000 *
5001 *
5002 *
5003 *
5004 *
5005 *
5006 *
5007 *
5008 *
5009 *
5010 *
5011 *
5012 *
5013 *
5014 *
5015 *
5016 *
5017 *
5018 *
5019 *
5020 *
5021 *
5022 *
5023 *
5024 *
5025 *
5026 *
5027 *
5028 *
5029 *
5030 *
5031 *
5032 *
5033 *
5034 *



FLAG LOCN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

5035 *
5036 *
5037 *
5038 *
5039 *
5040 *
5041 *
5042 *
5043 *
5044 *
5045 *
5046 *
5047 *
5048 *
5049 *
5050 *
5051 *
5052 *
5053 *
5054 *
5055 *
5056 *
5057 *
5058 *
5059 *
5060 *
5061 *
5062 *
5063 *
5064 *
5065 *
5066 *
5067 *
5068 *
5069 *
5070 *
5071 *
5072 *
5073 *
5074 *
5075 *
5076 *
5077 *
5078 *
5079 *
5080 *
5081 *
5082 *
5083 *
5084 *
5085 *
5086 *
5087 *
5088 *
5089 *



FLAG LOC TN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

5090 *
 5091 *
 5092 *
 5093 *
 5094 *
 5095 *
 5096 *
 5097 *
 5098 *
 5099 *
 5100 *
 5101 *
 5102 *
 5103 *
 5104 *
 5105 *
 5106 *
 5107 *
 5108 *
 5109 *
 5110 *

NO MORE TEST ---->

TEST WHICH FUNCTION HAS TO BE CALLED

SHORT FLAG SET.

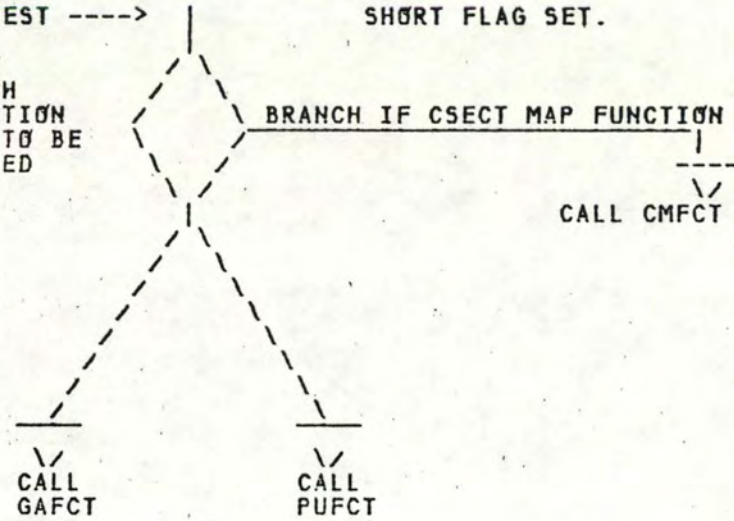
BRANCH IF CSECT MAP FUNCTION

CALL CMFCT

CALL GAFCT

CALL PUFCT

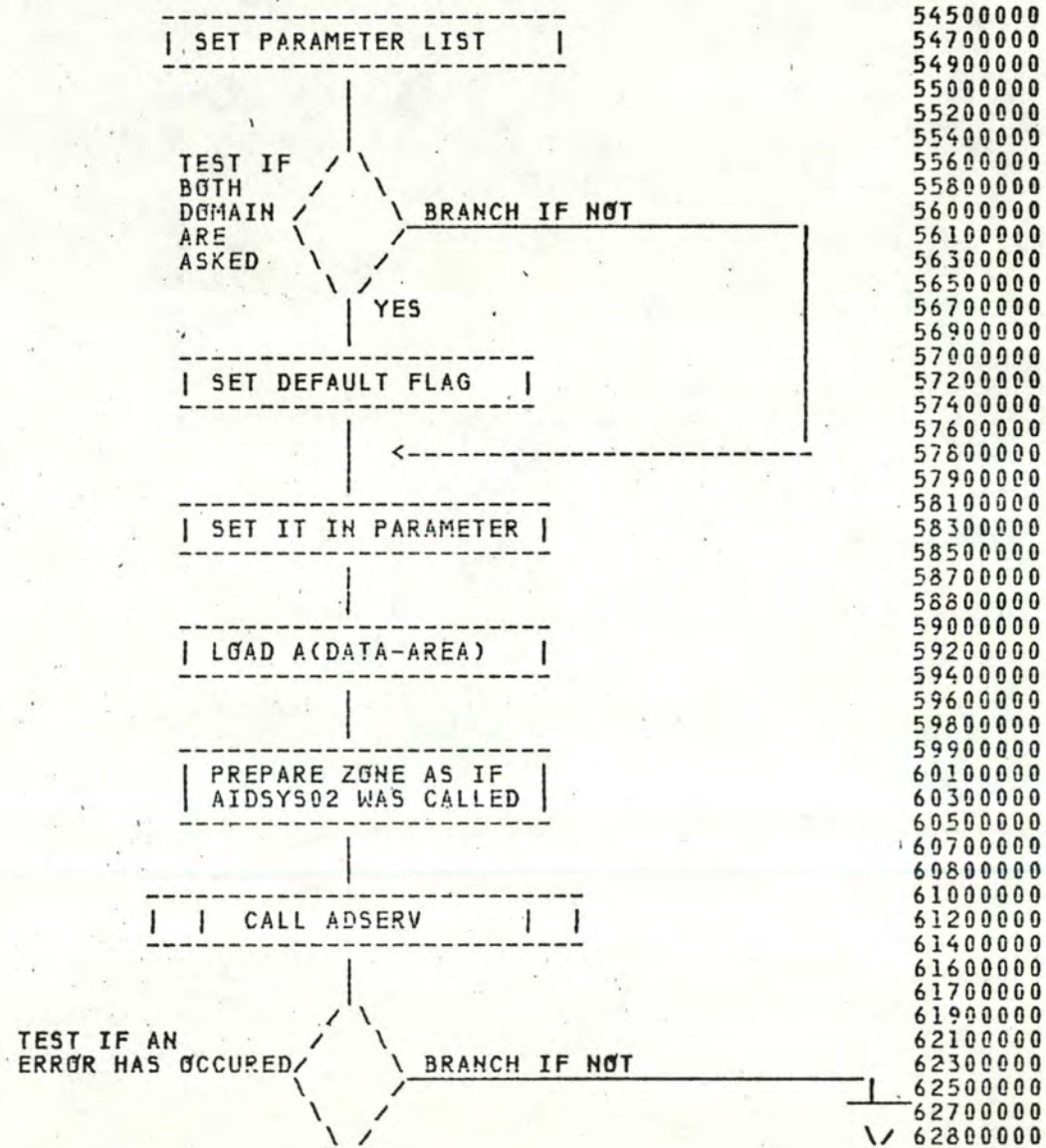
49400000
 49600000
 49800000
 50000000
 50200000
 50300000
 50500000
 50700000
 50900000
 51100000
 51200000
 51400000
 51600000
 51800000
 52000000
 52200000
 52300000
 52500000
 52700000
 52900000
 53100000



FLAG LOCIN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

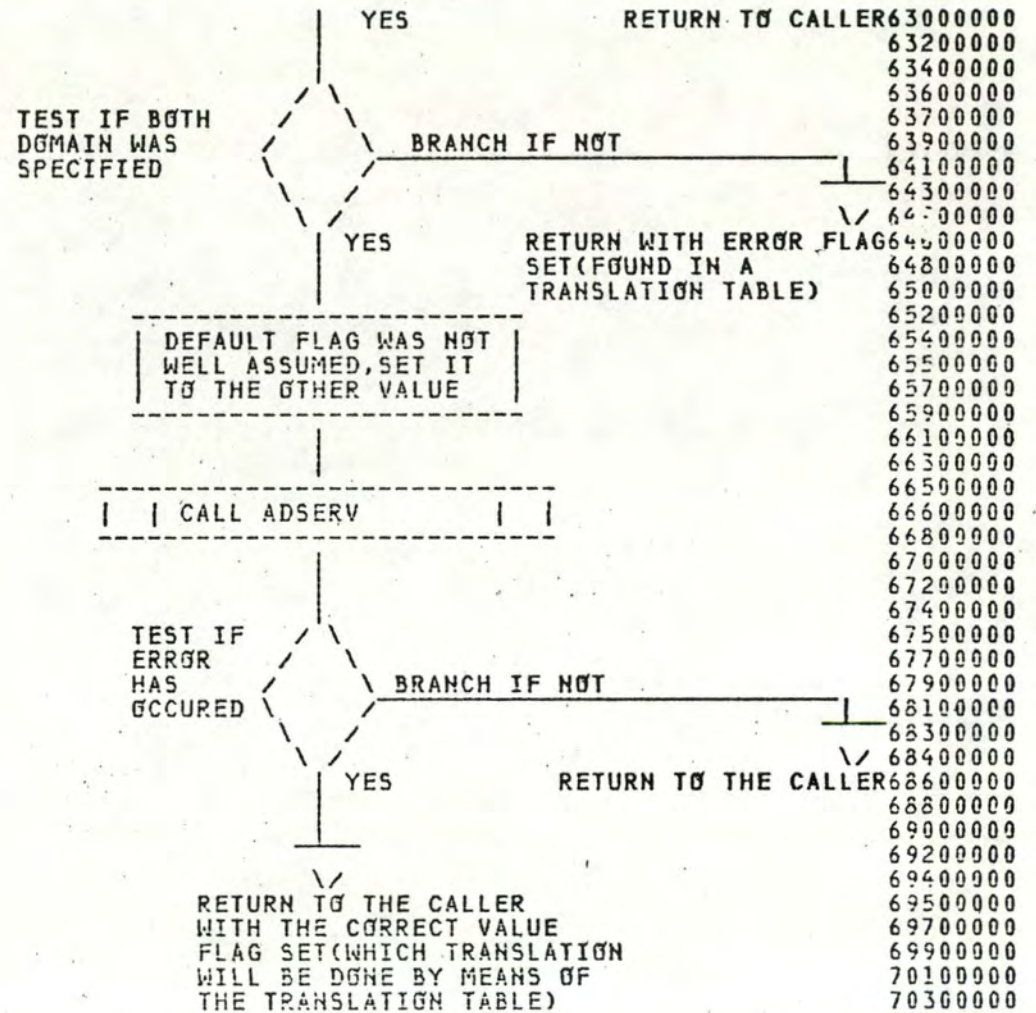
5111 ***** 53400000
 5112 * * 53600000
 5113 * THE NEXT FLOWCHART IS THE FLOW OF CMFCT * 53800000
 5114 * * 54000000
 5115 ***** 54100000

5116 * 54500000
 5117 * 54700000
 5118 * 54900000
 5119 * 55000000
 5120 * 55200000
 5121 * 55400000
 5122 * 55600000
 5123 * 55800000
 5124 * 56000000
 5125 * 56100000
 5126 * 56300000
 5127 * 56500000
 5128 * 56700000
 5129 * 56900000
 5130 * 57000000
 5131 * 57200000
 5132 * 57400000
 5133 * 57600000
 5134 * 57800000
 5135 * 57900000
 5136 * 58100000
 5137 * 58300000
 5138 * 58500000
 5139 * 58700000
 5140 * 58800000
 5141 * 59000000
 5142 * 59200000
 5143 * 59400000
 5144 * 59600000
 5145 * 59800000
 5146 * 59900000
 5147 * 60100000
 5148 * 60300000
 5149 * 60500000
 5150 * 60700000
 5151 * 60800000
 5152 * 61000000
 5153 * 61200000
 5154 * 61400000
 5155 * 61600000
 5156 * 61700000
 5157 * 61900000
 5158 * 62100000
 5159 * 62300000
 5160 * 62500000
 5161 * 62700000
 5162 * 62800000



FLAG LOCN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

5163 *
 5164 *
 5165 *
 5166 *
 5167 *
 5168 *
 5169 *
 5170 *
 5171 *
 5172 *
 5173 *
 5174 *
 5175 *
 5176 *
 5177 *
 5178 *
 5179 *
 5180 *
 5181 *
 5182 *
 5183 *
 5184 *
 5185 *
 5186 *
 5187 *
 5188 *
 5189 *
 5190 *
 5191 *
 5192 *
 5193 *
 5194 *
 5195 *
 5196 *
 5197 *
 5198 *
 5199 *
 5200 *
 5201 *
 5202 *
 5203 *



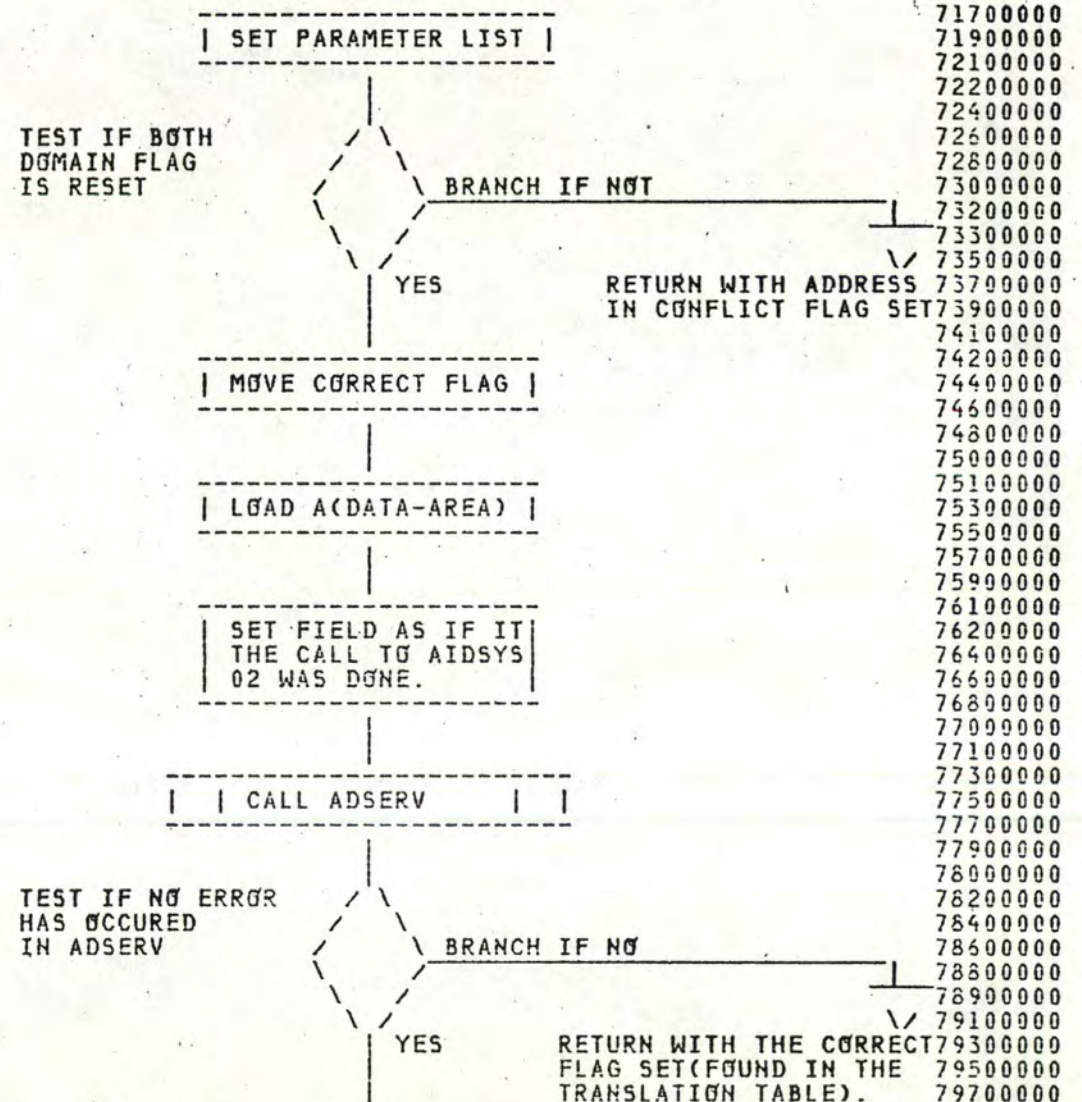
FLAG LOCN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

```

5204 ***** 70600000
5205 * * 70800000
5206 * THE NEXT IS THE FLOWCHART FOR THE GAFCT. * 71000000
5207 * * 71200000
5208 ***** 71300000
    
```

```

5209 *
5210 *
5211 *
5212 *
5213 *
5214 *
5215 *
5216 *
5217 *
5218 *
5219 *
5220 *
5221 *
5222 *
5223 *
5224 *
5225 *
5226 *
5227 *
5228 *
5229 *
5230 *
5231 *
5232 *
5233 *
5234 *
5235 *
5236 *
5237 *
5238 *
5239 *
5240 *
5241 *
5242 *
5243 *
5244 *
5245 *
5246 *
5247 *
5248 *
5249 *
5250 *
5251 *
5252 *
5253 *
    
```



FLAG LCTN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT

5254 *
 5255 *
 5256 *
 5257 *
 5258 *
 5259 *
 5260 *
 5261 *
 5262 *
 5263 *
 5264 *
 5265 *
 5266 *
 5267 *
 5268 *
 5269 *
 5270 *
 5271 *
 5272 *
 5273 *
 5274 *
 5275 *
 5276 *
 5277 *
 5278 *
 5279 *
 5280 *
 5281 *
 5282 *
 5283 *
 5284 *

