ABM User Manual

ND-60.203.2 EN

ABM User Manual

ND-60.203.2 EN

PRINTING RECORD	
PRINTING	NOTES
12/84	Version 1 EN
06/86	Version 2 EN

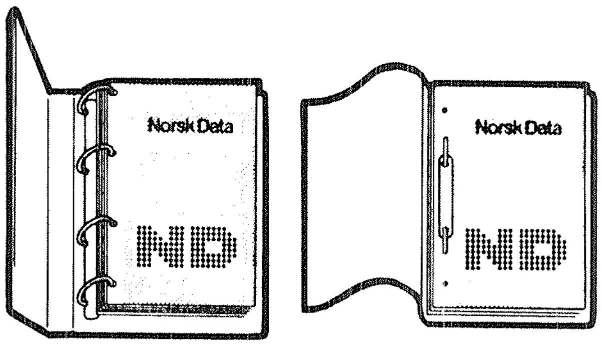
ABM User Manual
Publ.No. ND–60.203.2 EN

UPDATING

Manuals can be updated in two ways, new versions and revisions. New versions consist of a completely new manual which replaces the old one, and incorporate all revisions since the previous version. Revisions consist of one or more single pages to be merged into the manual by the user, each revised page being listed on the new printing record sent out with the revision. The old printing record should be replaced by the new one.

New versions and revisions are announced in the ND Customer Support Information and can be ordered from the address below.

The reader's comments form at the back of this manual can be used both to report errors in the manual and give an evaluation of the manual. Both detailed and general comments are welcome.



RING BINDER OR PLASTIC COVER

The manual can be placed in a ring binder for greater protection and convenience of use. Ring binders may be ordered at a price of NKr. 45.- per binder.

The manual may also be placed in a plastic cover. This cover is more suitable for manuals of less than 100 pages than for larger manuals.

Please send your order, as well as all types of inquiries and requests for documentation to the local ND office, or (in Norway) to:

Norsk Data A.S
Graphic Center
P.O.Box 25 BOGERUD
N-0621 OSLO 6 - Norway

I would like to order

.... Ring Binders, 40 mm, at NOK 45.- per binder

.... Plastic Covers, at NOK 10.- per cover

Name: ....

Company: ....

Address: ....

PREFACE

THE PRODUCT

This manual describes ABM - Application Building and Maintenance. ABM is used for online building and maintenance of applications for the SIBAS database.

ABM is registered in the ND Software Library as the following module:

ABM - Application Building and Maintenance
ND-210713B for ND-100
ND-210718B for ND-500

The product is delivered on diskettes containing all the necessary files and programs. The procedure for loading and implementing the files on the computer is described in the Program Description Sheet.

THE READER

The ABM User Manual should be of interest to database managers, system analysts and programmers.

PREREQUISITE KNOWLEDGE

The reader should be familiar with either FORTRAN or COBOL. The reader should also be familiar with SIBAS DRL (database Definition/ Redefinition Language) and FOCUS (screen handling system). General descriptions of these are found in:

FORTRAN Reference Manual	ND-60.145
COBOL Reference Manual	ND-60.144
SIBAS User Manual	ND-60.127
FOCUS Reference Manual	ND-60.137

THE MANUAL

CHAPTER 1 gives a general introduction to ABM.

CHAPTER 2 gives practical information for using ABM.

CHAPTER 3 gives general information for writing programs using ABM.

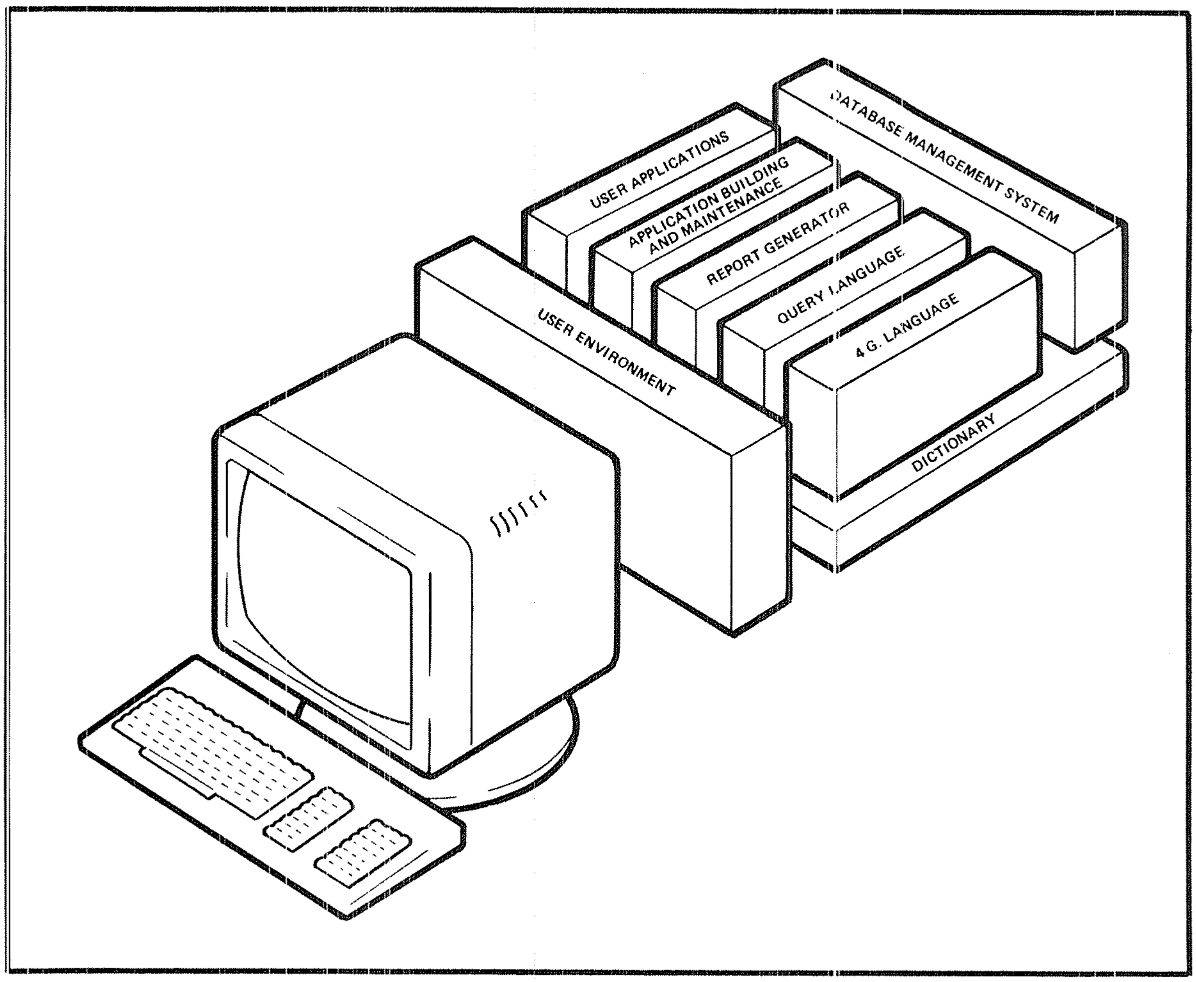
CHAPTERS 4 to 8 give details of the ABM-SIBAS, ABM-FOCUS and ABM-UTILITY library routines.

CHAPTER 9 gives an example of using ABM. The example gives a good overview of using the various modules of ABM.

STANDARD NOTATION

In the text you see:	What it means or what it is used for:
▨	● Areas shaded grey represent screen pictures.
<u>@ABM</u>	● Text typed in by the user is <u>underlined</u>. All operating system commands must be terminated by ↵.
@	● This symbolizes the SINTRAN III prompt sign. It indicates that you are in touch with the computer's operating system and can give it commands.
↵	● This represents the carriage return key. On the terminal it may be marked ↵, CR, RETURN or ENTER.
↳	● This represents the EXECUTE key.
↖	● This represents the HOME key.
CTRL + W	● This is an example of a CTRL combination. It means you press the CTRL key and hold it down while you press W.
><<>	● This key is used for example when moving from one level of the Subschema pictures down to the next.
SHIFT + ><<>	● These keys are used for example when moving from one level of the Subschema pictures up to the next.
FUNK @	● This will clear the screen picture. Note: The keys should be pressed one at a time.

TABLE OF CONTENTS



DATABASE MANAGEMENT SYSTEM
USER APPLICATIONS
APPLICATION BUILDING AND MAINTENANCE
REPORT GENERATOR
QUERY LANGUAGE
4 G. LANGUAGE
USER ENVIRONMENT
DICTIONARY

DIALOGUE

DIALOGUE	DIALOGUE is Norsk Data's total concept in database management. It has the complete set of tools and utilities for: ● high performance, easy expansion, and redefinition of a data base; ● creating a tailored user interface; ● creating and maintaining applications easily and efficiently; ● generating advanced reports; ● common data dictionary information for easy coordination and maintenance of the database and applications.

The modules of DIALOGUE are described below:

USER ENVIRONMENT	UE is an integrated part of the SINTRAN operating system. It can be used together with DIALOGUE to create a tailormade, individual interface for the ND system.
4TH GENERATION LANGUAGE	UNIQUE is a tool for application development. It can be used to develop screen pictures and specify transactions directly on the screen. It saves about 90% of development time and maintenance resources.
REPORT GENERATOR	RG allows the definition of advanced reports in an easy manner by drawing the desired layout on the screen.
QUERY LANGUAGE	ACCESS is a tool which can be used to look at data base information in terms of tables. It is suitable for online use.
APPLICATION BUILDING AND MAINTENANCE	**ABM can be used to make demanding transaction systems. It is used inter-actively with simple directives. It saves about 50% of development time and 90% of maintenance resources.**
DATABASE MANAGEMENT	SIBAS is a full CODASYL database management system. Its features include high performance, as well as easy expansion and redefinition of databases. It is a flexible and a highly secure system, well suited for distributed processing environments.

CHAPTER 1 INTRODUCTION TO ABM

- INTRODUCTION TO ABM
- THE MODULES OF ABM
- THE DEPENDENCIES BETWEEN THE ABM MODULES

1 INTRODUCTION TO ABM:

ABM is a 4th generation Application Building and Maintenance system. It is built around the data dictionary concept. It is a tool for system analysts and system programmers.

ABM can be used for the fast, secure and online building and maintenance of applications for the SIBAS database. ABM is used interactively with simple menu-driven directives. Using ABM, one can typically save up to 50% of development time and 90% of maintenance resources. ABM simplifies the definition and maintenance of the SIBAS database, and the definition and maintenance of forms for your application programs. The advantages of using ABM are summarized below:

DEFINING AND MAINTAINING THE SIBAS DATABASE	With simple menu-driven commands you can define a complete SIBAS database. The database items, group items, sets, realms, system realms and os-files can be defined, together with data dictionary information. ABM also allows for easy addition to, deletion or redefinition of the database.
DEFINING AND MAINTAINING FORMS	ABM uses the FOCUS system for defining and maintaining forms. Once forms have been defined with ABM, they can be easily connected to application programs. Forms can also be changed or redefined as required.
USING APPLICATION PROGRAMS	ABM maintains variable declarations and value assignments, as defined by the data descriptions, subschemas and forms. This will relieve the programmer of having to establish these declarations again in the application programs.

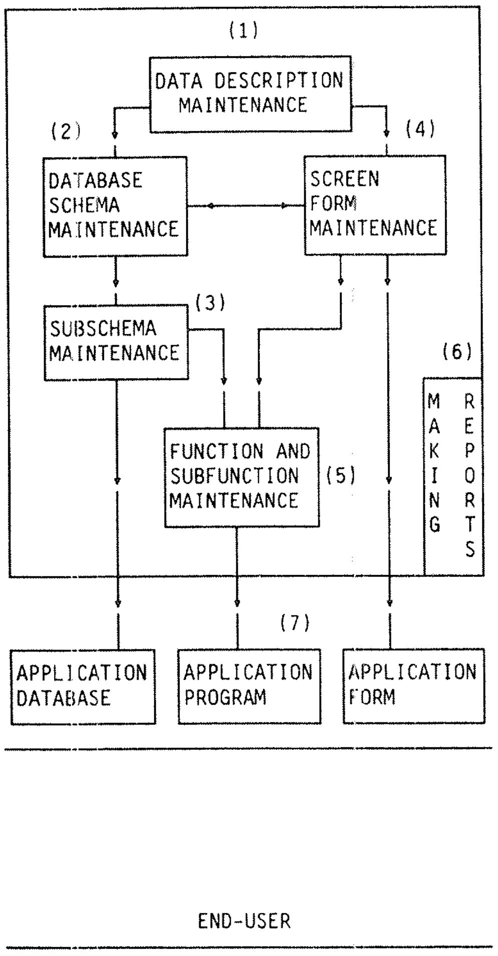
1.1 THE MODULES OF ABM

ABM is a system for creating information about applications systems that communicate with the SIBAS databases or the FOCUS screen handling system. The ABM information contains:

- data descriptions,
- descriptions of the databases (schemas, items, sets, etc.),
- descriptions of subsets of databases (subschemas),
- descriptions of the screen forms, and
- descriptions of functions and subfunctions which connect screen forms and subschemas.

The information in ABM is stored in an ordinary SIBAS database. We will refer to this ABM database as the **ABM catalog.** The features of ABM are described below.

THE MODULES OF ABM:



(1) DATA DESCRIPTION MAINTENANCE

This module allows you to describe all data items in the database, and all fields in the forms. Each data item is given a unique, easy to remember name. The data description will also contain format information, following the standard display and storage format (see appendices A and B).

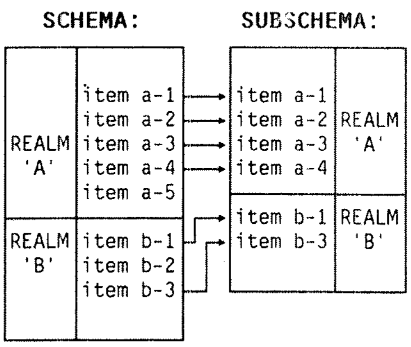
(2) DATABASE (SCHEMA) MAINTENANCE

A database schema, also called a DRL schema, is a complete description of a SIBAS database. It contains the necessary information for the automatic production of DRL input file to initiate or redefine a database.

A DRL item must have a unique, easy to remember name within a DRL database and DRL realm.

ABM may contain descriptions of several SIBAS databases.

(3) SUBSCHEMA MAINTENANCE



A subschema defines a subset of a database definition. The definition of the total database is found in the DRL schema for the database. A subschema consists of a subset of the database realms; and for each of these realms, a subset of the groups and items from the realm.

A subschema includes definitions from one, and only one, database.

A lot of different subschemas may be defined in ABM, and the size of a subschema may vary from one item to the total database.

The main purpose of the subschemas is to specify which part of the database is of interest in a particular application.

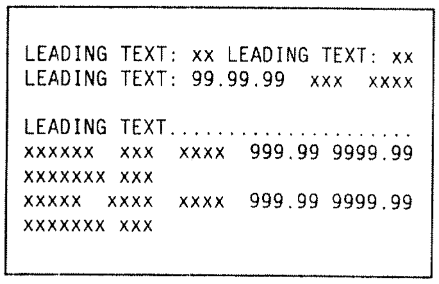
Normally, there will be a close connection between the fields of a form and the items in a subschema, used together in the same application.

A subschema may be connected to more than one subfunction (which describes a subroutine in the application).

ABM subschemas are created, changed and deleted online.

Building a subschema in ABM is done by marking the desired realms from a list of all realms. For each of the marked realms, the desired items and groups are also marked.

(4) SCREEN FORM MAINTENANCE



```
LEADING TEXT: xx LEADING TEXT: xx
LEADING TEXT: 99.99.99  xxx  xxxx

LEADING TEXT.....................
xxxxxx  xxx  xxxx  999.99 9999.99
xxxxxxx xxx
xxxxx  xxxx  xxxx  999.99 9999.99
xxxxxxx xxx
```

The organization of forms is based on the possibility of dividing the forms into logical groups of fields. A logical group may often occur several times in the same form. **Thus, one form may consist of one or more logical groups, and may have several occurrences of every logical group.** Such a logical group will be called a form record, or just a record.

Normally, there will be a close connection between a form record and a subschema record.

(5) FUNCTION AND SUBFUNCTION MAINTENANCE

The functions and subfunctions in ABM establish the connection between the programs and subroutines in the application system, and their use of database and screen forms.

To each function, one or more subfunctions are connected; and each subfunction has one subschema and/or one screen form. The connection from function to subfunction and, further, from subfunction to subschema and screen form, describes the function's use of the database and forms.

A subfunction may be connected to more than one function. Online menu commands are used to create, change and delete ABM functions and subfunctions.

(6) REPORT GENERATION

A report module, which is a part of the ABM system, makes it possible to select and extract data from the ABM catalog in an easy way. The result is formatted into readable reports which are immediately available for the user.

(7) MAKING APPLICATION PROGRAMS

A primary use of the contents in the ABM catalog is the automatic extraction of information. This information is necessary for building declarations and assignments for programs in the application system.

Information needed to run a program using SIBAS and FOCUS is **automatically generated** in the INCLUDE/COPY generating module. This relieves the programmer of establishing variable declarations and value assignments; they have already been defined in the data descriptions, subschemas and screen forms.

The application programs will generally move data between the screen forms and the application database.

The information produced about SIBAS in the INCLUDE/COPY files, consists mainly of the realms to be used. For each realm, the items and/or indexes to be used in the particular program are indicated; in other words, subschema information. Each item is declared by type (integer, character etc.) and dimension/length.

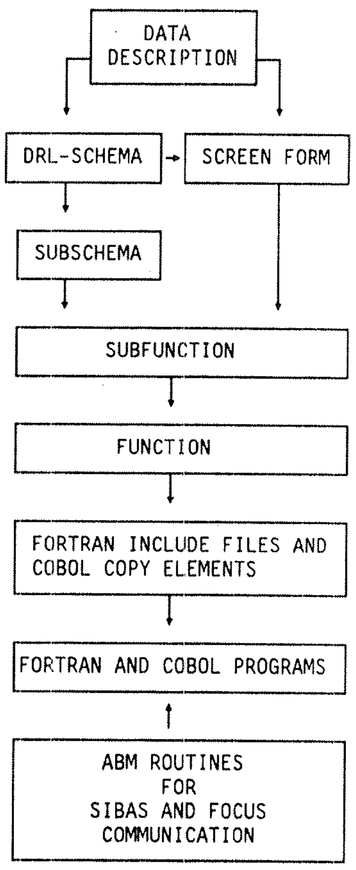
The FOCUS information is produced in the same manner as the SIBAS information. The ABM-SIBAS/ABM-FOCUS subroutine package should be used to benefit from the data structure built in the INCLUDE and COPY files.

Since all items and fields used in connection with SIBAS and FOCUS are automatically declared, the need for additional definitions of local variables in your programs is limited.

The result of the INCLUDE/COPY commands is stored in two SINTRAN files. These files should be added to your code at compilation time by using the INCLUDE and COPY statements for FORTRAN or COBOL programs.

1.2 THE DEPENDENCIES BETWEEN THE ABM MODULES

DATA DESCRIPTIONS


DATA DESCRIPTION
DRL-SCHEMA
SCREEN FORM
SUBSCHEMA
SUBFUNCTION
FUNCTION
FORTRAN INCLUDE FILES AND COBOL COPY ELEMENTS
FORTRAN AND COBOL PROGRAMS
ABM ROUTINES FOR SIBAS AND FOCUS COMMUNICATION

The data descriptions are not dependent on other modules in ABM. However, data descriptions may refer to DRL schemas or screen forms. The data descriptions cannot be deleted without deleting all the referenced items (field-items and DRL items) in screen forms and DRL schemas.

Changes in a data description may affect records in all other modules which refer to the data description, either directly or indirectly.

DRL SCHEMAS

The SIBAS items defined in the DRL schemas are dependent only on the data descriptions. The other parts of the DRL schema are not dependent on other ABM modules. The SIBAS items in the DRL schema part are connected to items in subschemas and screen fields. A DRL item may therefore not be changed or deleted without consequences for all the referenced subschemas and screen fields.

In case of a deletion of a DRL item, all the referenced items in subschemas and screen forms must be deleted first.

SCREEN FORMS

FORM:
A VDU-form described in ABM-FOCUS.

All the fields in the screen forms are dependent on the data descriptions directly, or indirectly via a DRL item. In other words, a screen field must be connected to either a data description or a DRL item. A screen form consists of many screen fields and may be connected to one or more subfunctions.

A screen form cannot be deleted without deleting all references from subfunctions.

SUBSCHEMA

SUBSCHEMA:
A collection of realms and items/keys within a database in a DRL schema described in ABM.

The subschema is dependent on the DRL schema (items, groups, realms and database) and may be connected to one or more subfunctions. Subschema information that no longer fits the DRL schema will blink in the subschema function field, and it is the user's responsibility to change the subschema contents.

> A subschema cannot be deleted without deleting all the references from subfunctions.

FUNCTION AND SUBFUNCTION

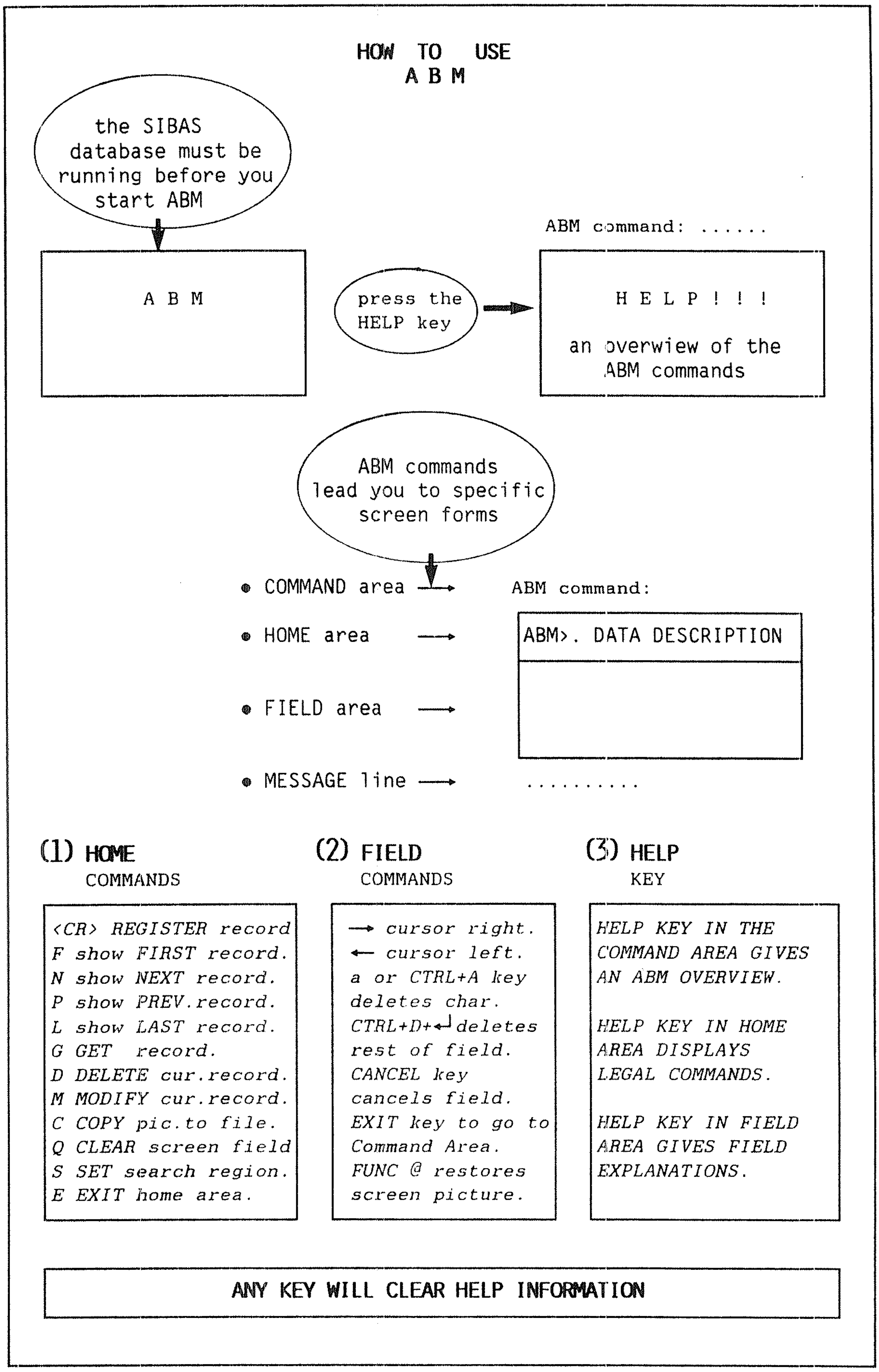
FUNCTION:
A program, i.e., a collection of one or more subfunctions.

SUBFUNCTION:
A subroutine using a screen and/or a subschema.

A function owns one or more subfunctions, and is the basis for producing INCLUDE and COPY files. A subfunction cannot be deleted before all references from functions are deleted. A change of the contents in a module will affect all other directly or indirectly dependent modules, i.e., a change in data descriptions will cause a redefinition of the DRL schemas and the screen forms. The result of this will be a redefinition of the subschemas; and further the subfunctions and the functions; and at last the INCLUDE files must be generated. The affected application programs must be recompiled and reloaded. Depending on the type of changes, adjustments in the application programs may be necessary. Some changes are so extensive (like the redefinition of a database), that they are user controlled, i.e., the whole process is performed in more than one step.

> A subfunction cannot be deleted before all references from functions are deleted.

> Before deleting any item in the database, you will have to delete the items that are interconnected to it. To find out how items in a database are connected, run the REPORT module. The report generated from this module will show interconnections between the Functions, Subfunctions, Forms, Schemas etc. You can then use this information to delete the items that are connected to a specific item, and then delete the item itself.



(1) HOME
COMMANDS

<CR> REGISTER record
F show FIRST record.
N show NEXT record.
P show PREV.record.
L show LAST record.
G GET record.
D DELETE cur.record.
M MODIFY cur.record.
C COPY pic.to file.
Q CLEAR screen field
S SET search region.
E EXIT home area.

(2) FIELD
COMMANDS

→ cursor right.
← cursor left.
a or CTRL+A key deletes char.
CTRL+D+↵ deletes rest of field.
CANCEL key cancels field.
EXIT key to go to Command Area.
FUNC @ restores screen picture.

(3) HELP
KEY

HELP KEY IN THE COMMAND AREA GIVES AN ABM OVERVIEW.

HELP KEY IN HOME AREA DISPLAYS LEGAL COMMANDS.

HELP KEY IN FIELD AREA GIVES FIELD EXPLANATIONS.

ANY KEY WILL CLEAR HELP INFORMATION

CHAPTER 2
HOW TO USE ABM

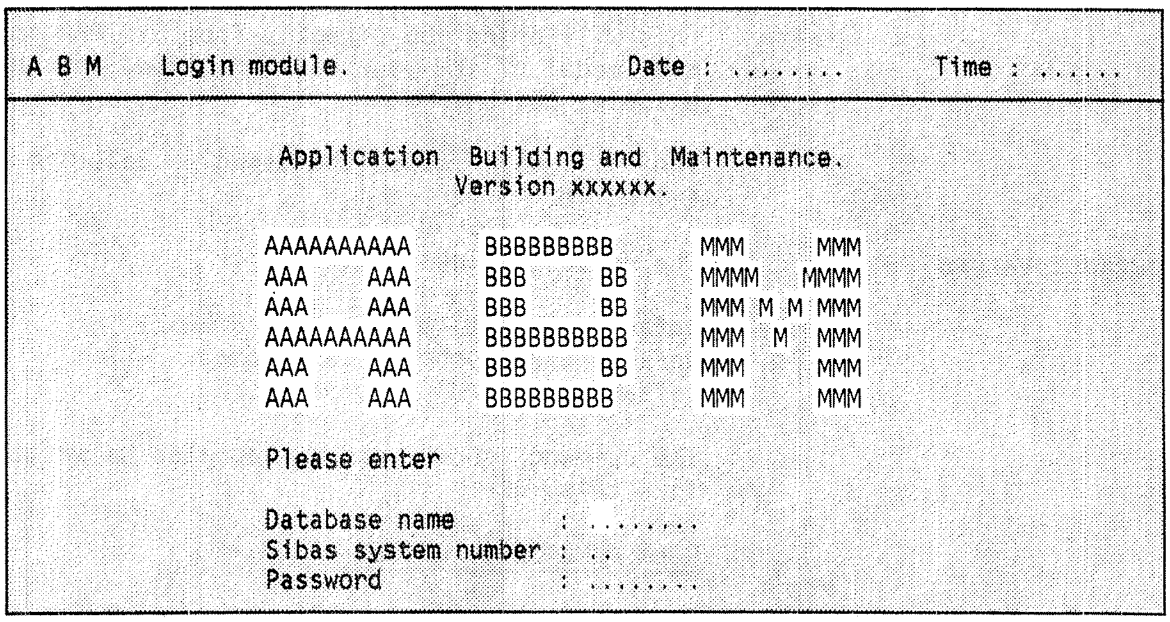
2 HOW TO USE ABM

2.1 STARTING ABM

The SIBAS database must be in a RUNNING state using BIM-log before you can start ABM. Appendix G will outline how to start and stop SIBAS in the correct way.

The maintenance functions in ABM are administered by a menu-driven command system. ABM is started by giving the command:

```
        @ABM↵
```



```
A B M    Login module.                    Date : ........      Time : ......

                 Application  Building and  Maintenance.
                              Version xxxxxx.

                AAAAAAAAAA      BBBBBBBBB       MMM      MMM
                AAA    AAA      BBB     BB      MMMM    MMMM
                AAA    AAA      BBB     BB      MMM M  M MMM
                AAAAAAAAAA      BBBBBBBBBB      MMM   M  MMM
                AAA    AAA      BBB     BB      MMM      MMM
                AAA    AAA      BBBBBBBBB       MMM      MMM

                Please enter

                Database name        : ........
                Sibas system number  : ..
                Password             : ........
```

When you start ABM, the screen picture shown above is displayed. The following input and commands are now possible:

DATABASE NAME	Give the name of the ABM catalog (ABMBASE). This name is stored as a default name, so you can press ↵. You can stop ABM at this stage by pressing the EXIT key.
SIBAS SYSTEM NUMBER	Give the number of the SIBAS process which operate the ABM catalog (default process is 1).
PASSWORD	Write the password for your ABM catalog followed by ↵. If there is no password implemented on the ABM catalog, just press ↵.

Then a new screen picture will be displayed. The ABM command line is placed at the top of the screen.

HELP

Pressing the HELP key at the ABM command line gives an overview of ABM.

ABM COMMAND

Give a command to ABM.

Alternatively, the command for starting ABM and its parameters can be written on one line. For example, if you want to start working with DBITEM, you can write:

```
@ABM,,,,DBITEM↵
```

This will bring you directly from SINTRAN into the modul of ABM you have specified, in this example: DBITEM.

The ",," will cause the default value to be filled in.

DEFAULT VALUES

The default values are:

Default ABM catalog name is ABMBASE.
Default SIBAS process is 1 (one).
Default password is no password.

The command shown above can also be written like this:

```
@ABM ABMBASE 1,,DBITEM↵
```

If you use a password to protect the ABM catalog:

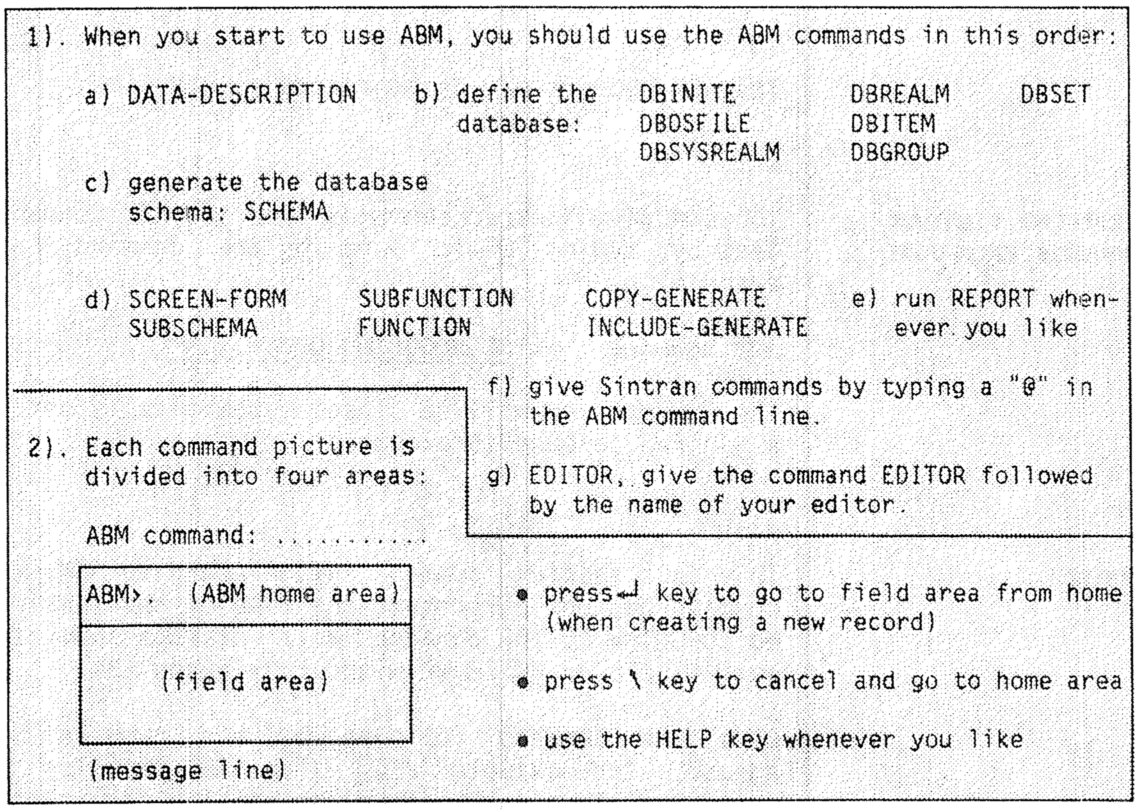
```
@ABM ABMBASE 1 password DBITEM↵
```

You may use all the combinations of the default values as you please.

> NOTE:
> The ABM catalog must
> be in a RUNNING state
> using BIM-log before
> you start ABM.

2.2 THE ABM COMMANDS

Press the HELP key in ABM command line, and the ABM HELP menu picture will be displayed. The picture shows all the commands that are available in the ABM system. Please note that, due to the dependencies between the modules in ABM when defining a <u>new</u> application database, the commands should be used in the order described below. However, when modifying an application database, the commmands can be used in any sequence.



1). When you start to use ABM, you should use the ABM commands in this order:

a) DATA-DESCRIPTION

b) define the database: DBINITE, DBOSFILE, DBSYSREALM, DBREALM, DBITEM, DBGROUP, DBSET

c) generate the database schema: SCHEMA

d) SCREEN-FORM, SUBSCHEMA, SUBFUNCTION, FUNCTION, COPY-GENERATE, INCLUDE-GENERATE

e) run REPORT whenever you like

f) give Sintran commands by typing a "@" in the ABM command line.

g) EDITOR, give the command EDITOR followed by the name of your editor.

2). Each command picture is divided into four areas:

ABM command: ....

ABM>. (ABM home area)

(field area)

(message line)

- press ↵ key to go to field area from home (when creating a new record)
- press \ key to cancel and go to home area
- use the HELP key whenever you like

THE SEQUENCE FOR USING ABM COMMANDS WHEN MAKING A NEW APPLICATION SYSTEM:

(1) For each <u>application system</u>:
1. define the DATA DESCRIPTIONS,
2. define the DRL schema with the commands

DBINITIATE	DBITEM
DBOSFILE	DBGROUP
DBSYSREALM	DBSET
DBREALM	

(2) Generate the schema for your application database by using the command SCHEMA.

You may abbreviate the commands as long as the abbreviations are unique.

(3) For each function in the application system:

1. define picture	SCREEN-FORM
2 define subschema	SUBSCHEMA
3. define subfunction	SUBFUNCTION
4. define function	FUNCTION
5. generate INCLUDE files or COPY elements	INCLUDE-GENERATE COPY-GENERATE

(No.4, define function, can be skipped.)

(4) Run SIB-DRL with the file generated in "SCHEMA" as input file.

(5) Start the application database. (See Appendix G.)

(6) Write, compile and load your application programs. (See the chapter "An example of using ABM".)

The ABM reports may be generated at any time.

EXECUTING SINTRAN COMMANDS FROM ABM

You can give Sintran commands from the ABM command line by typing "@" followed by the command. For example:

ABM command : <u>@LIST-SPOOLING-QUEUE,,,↵</u>

The ABM catalog will be closed automatically before a Sintran command is executed.

EDITOR

This is a special command with return to ABM. The command has one parameter: the name of the desired editor. The EDITOR command will start a subsystem such as PED or NOTIS-WP without stopping ABM.

Give the command followed by the name of the editor, for example PED. The ABM catalog is automatically closed before the editing can start.

Finish the editing by pressing the EXIT key. You will then return to the ABM command line. (The ABM catalog is automatically opened again.)

Below is an example using the EDITOR command:

ABM command : <u>EDITOR PED↵</u>

<edit in PED>

<press the EXIT key, and you will return to ABM>

ABM command :

PROGRAMMING LANGUAGE

In ABM, COBOL is the default language type for applications communicating with screen forms.

DEFINE-PROGRAMMING-LANGUAGE

If you want to change current language type, give the command shown below followed by the desired language; for example, FORTRAN:

```
ABM command : DEFINE-PROGRAMMING-LANGUAGE FORTRAN↵
```

GET-PROGRAMMING-LANGUAGE

If in doubt whether COBOL or FORTRAN is the current language for a screen form, you can display the current language on the screen by giving the command:

```
ABM command : GET-PROGRAMMING-LANGUAGE↵
```

AN EXAMPLE

Let's say that after having defined screen forms in ABM with the language FORTRAN, you want to write your applications in COBOL. You'll have to follow the procedure described below to update the language type and allow your screen forms to communicate with COBOL applications.

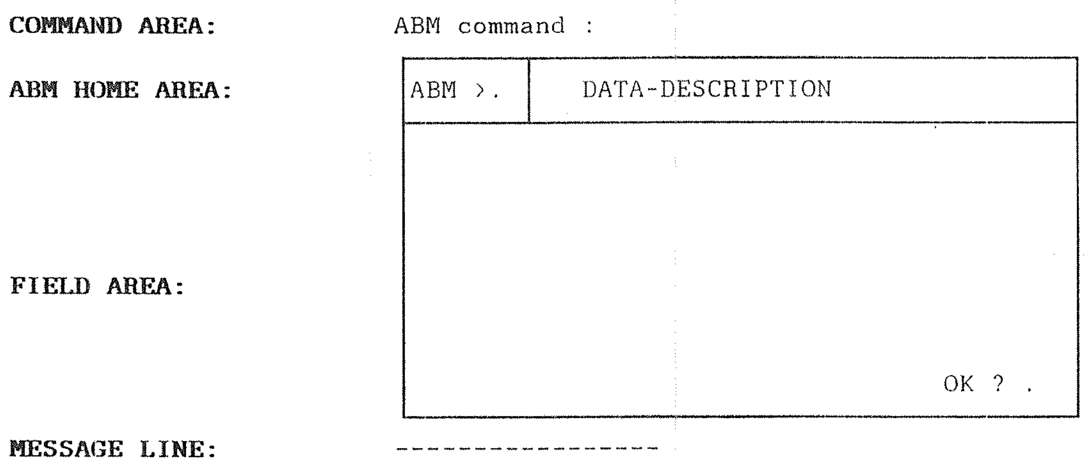
```
ABM command : DEFINE-PROGRAMMING-LANGUAGE COBOL↵
ABM command : SCREEN-FORM↵
FD> MAKE-UPTODATE-FORMS↵
```

Users of ABM are advised to use just one programming language (FORTRAN or COBOL) to communicate with one ABM catalog. But if you still want to mix COBOL and FORTRAN, you must follow the restricted naming conventions for FORTRAN.

Remember also to store the screen forms communicating with COBOL applications on a different form file from the screen forms communicating with FORTRAN applications.

2.3 NAVIGATION IN THE COMMAND PICTURES

Most commands from the HELP menu will lead you to a command picture. The command picture is divided into four different areas. Above the screen picture frame is the COMMAND area. ABM HOME area is the small area at the left-hand side, and the FIELD area is the main body of the picture. Below the screen picture frame is the MESSAGE line. This is shown in the figure below:



THE HELP KEY You can obtain information about the ABM HOME area commands and the command pictures by pressing the HELP key.

NAVIGATION From the ABM HOME area, you can go to the FIELD area by pressing the ↵ key.

HOME From the FIELD area, you can go to the ABM HOME area by pressing the \ key. You will automatically go to the HOME area when the last field in the FIELD area is filled in. **When you escape the FIELD area by pressing the \ key, the data in the form will not be stored in the ABM catalog.**

EXIT From the ABM HOME area you can go to the COMMAND area by pressing the E key or the EXIT key.

You can exit from the COMMAND area by pressing the EXIT key or by typing EXIT . You will then exit from the ABM system. The ABM catalog will be closed automatically.

EXECUTE The EXECUTE key (↳) will move the cursor directly to the OK field. You may use the EXECUTE key in all the ABM menus whenever the M (Modify) command is given.

MESSAGES

After execution of a command, a message will usually be displayed at the bottom line of the screen. Also, whenever the contents of a field are redefined or stored, a message will be displayed. The message indicates whether the operation has been successful or not.

OK?

Most of the ABM command pictures contain an OK field. The user is supposed to use this field to give the final approval or disapproval of the operation, by writing:

Y - update the ABM catalog.
N - no update of the ABM catalog,
    go to the next part of the screen picture.
R - no update, repeat data registration.
    The cursor will be positioned in the first field which is not a key value.

2.4 NAMING CONVENTIONS FOR COBOL AND FORTRAN PROGRAMS

The COPY and INCLUDE modules of ABM can be used to produce parts of standard COBOL and FORTRAN programs. However, in order to avoid compilation errors, these programs should use variable names that are unique and are easily recognizable. There is a "naming convention" for Data Descriptions, realms, items etc.

The COBOL and FORTRAN programs use slightly different naming conventions.

NAMING CONVENTIONS FOR THOSE USING COBOL

- For Data Description names referred to directly in a screen form, the first 8 characters must be unique.

Example :

```
        Employee  List

  Name: ..................
  Address: ...............
           ...............
  Function-1: ............
  Function-2: ............
```

If the field names are the same as the leading text shown in the example, then the names Function-1 and Function-2 are illegal, since the first 8 chars. are not identical.

NAMING CONVENTIONS FOR THOSE USING FORTRAN

- For the Data Description names referred to in a form, the first 5 characters must be unique.
- The first two characters in a realm name must be unique for all realm names in an application database.
- The first two characters of a realm name must also be used for all items and group item names in that realm. The next 5 characters in the item names must be unique within the realm.
- For item names which are also indexes in the the application database, characters 3, 4 and 5 must be unique in that realm.

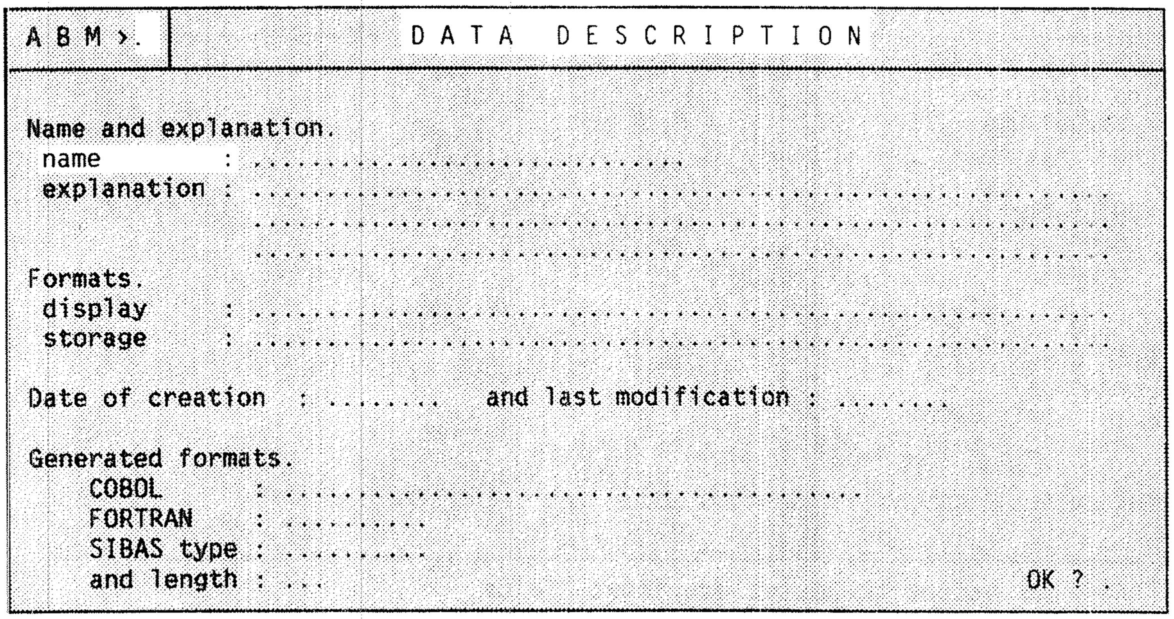
Example :

```
realm : EMPLOY
-------------------------------
items:   groups:   indexes:

EMNAME               key
EMADRES
EMFUNC1 ┐EMJOB
EMFUNC2 ┘
```

- The first 2 characters of the realm name EMPLOY are used in the names of all items, and group item(s).
- It would be illegal to make the items EMFUNC1 and EMFUNC2 index keys, since the 3rd, 4th and 5th characters (FUN) are identical.

2.5 DATA DESCRIPTION

The DATA-DESCRIPTION function is started by giving the command in full, or its abbreviation (D-D). The following picture appears on the screen:



```
A B M >.          D A T A   D E S C R I P T I O N

Name and explanation.
 name        : ..............................
 explanation : ............................................................
               ............................................................
               ............................................................
Formats.
 display     : ............................................................
 storage     : ............................................................

Date of creation  : ........   and last modification : ........

Generated formats.
    COBOL     : ........................................
    FORTRAN   : ..........
    SIBAS type : ..........
    and length : ...                                              OK ? .
```

THE ABM HOME AREA The upper left-hand corner is the ABM HOME area. In this area you can give the following commands:

↵	enter form field area when creating a record.	HELP	key for ABM command information.
F -	show FIRST record.	N -	show NEXT record.
P -	show PREVIOUS record.	L -	show LAST record.
G -	GET specific record.	D -	DELETE current record.
M -	MODIFY current record.	C -	COPY the screen picture to file.
Q -	Clear all fields on screen picture.		
S -	SET/RESET search region each 2nd time.	E -	EXIT from home area.

THE FIELD AREA

NAME: A unique name, maximum 30 chars.
EXPLANATION: Free text, maximum 180 chars.

STORAGE & DISPLAY FORMATS: See Appendix A.

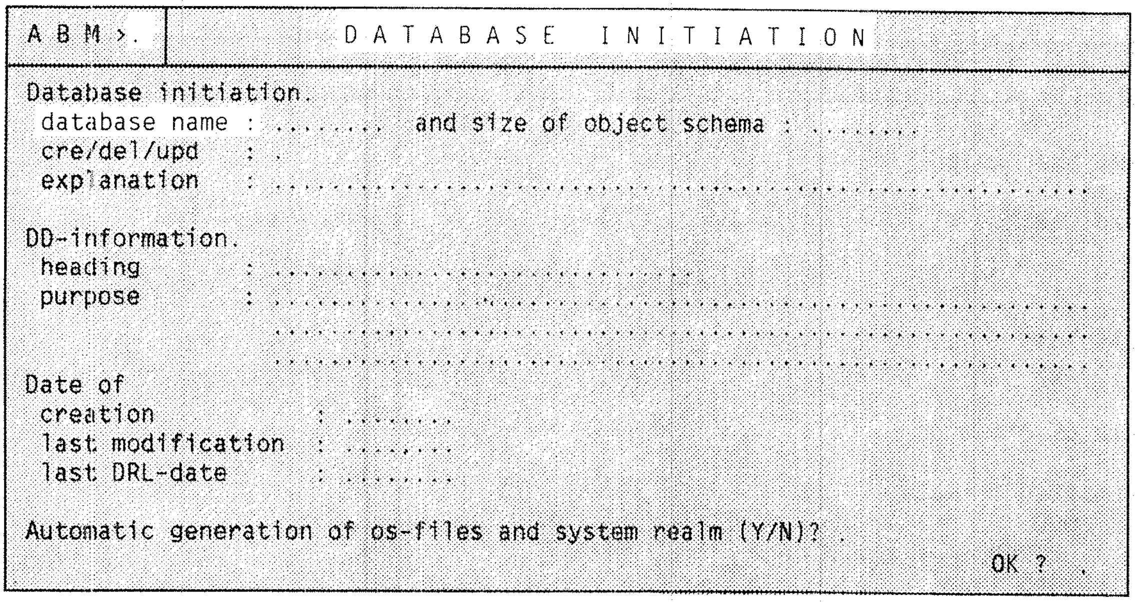
The HELP key will display the explanations of the FIELD area.

GENERATED FORMATS:

- **COBOL format:** generated from standard display & storage format. Used in item/field definitions in COPY elements.
- **FORTRAN format:** generated from standard display & storage format. Used in item/ field definitions in INCLUDE files.
- **SIBAS TYPE & LENGTH:** generated from standard storage. Used in application database.

2.6 DATABASE INITIATION

The command DBINITIATE will initiate a new application database, and cause the "Initiate Database" statement which is necessary for the DRL run.



```
A B M >.          D A T A B A S E   I N I T I A T I O N

Database initiation.
 database name : ........  and size of object schema : ........
 cre/del/upd   : .
 explanation   : ............................................................

DD-information.
 heading       : ..............................
 purpose       : ............................................................
                 ............................................................
                 ............................................................

Date of
 creation          : ........
 last modification : ........
 last DRL-date     : ........

Automatic generation of os-files and system realm (Y/N)? .
                                                            OK ? .
```

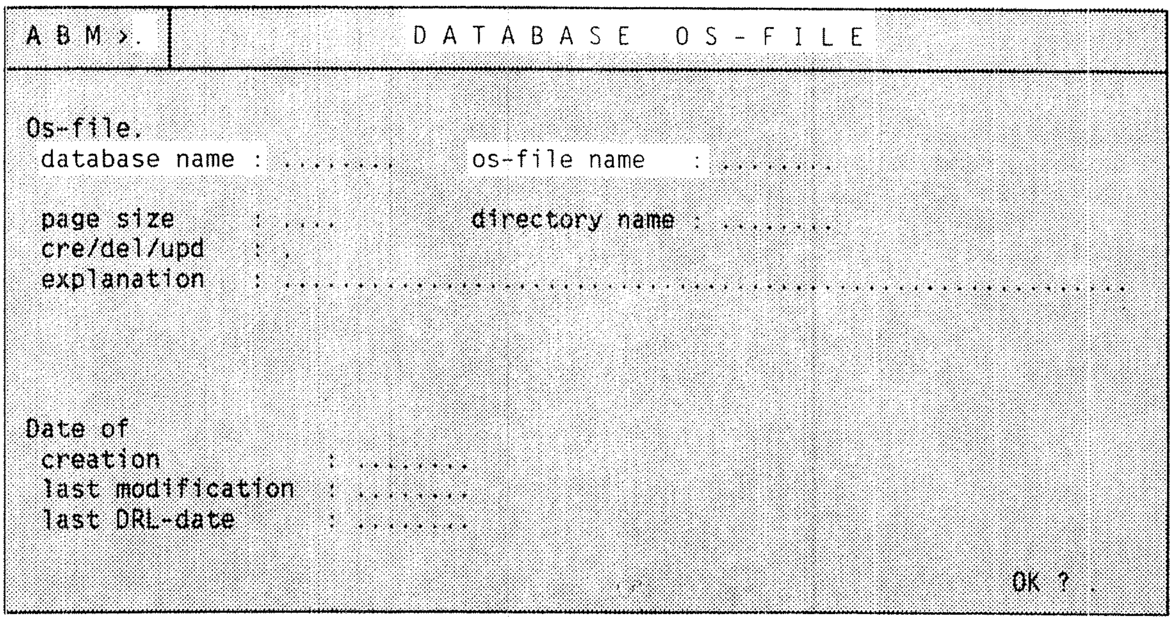
All fields except the dates are input fields.

DATABASE NAME	:Database name, unique within the ABM catalog.
SIZE OF OBJECT SCHEMA	:Size in number of 64-word blocks. To avoid problems, give a large size, at least 1000 (default is 4800).
CRE/DEL/UPD	:Create is default when defining a new application database, after the ABM command ↵ is given. If the ABM command M (Modify) is given, Update is default. Before Deleting, the user has to substitute the "U" with a "D" in the CRE/DEL/UPD field. NOTE: Deletion is only executed on a confirmed database
EXPLANATION	:60 characters describing the application database.
DD-INFORMATION	:See Appendix C.
HEADING	:Max. 30 characters.
PURPOSE	:Max. 180 characters.
DATE OF CREATION	:Date of creation of the application database.
MODIFICATION	:Date of last modification of the application database.
LAST DRL-DATE	:Date of the last run of SCHEMA's confirmation.
AUTOMATIC GEN. OF OS-FILES AND SYSTEM REALM (Y/N)?	:If Y (Yes), two os-files and one system realm will be made automatically, and the user will not have to give the commands DBOSFILE and DBSYSREALM. The generated names are made on basis of the given database name. One of the os-files will hold just the index part of the database and have "-IX" as suffix. The other os-file will contain the data and have "-DA" as suffix. The system realm will be given the same name as the os-file holding the index. Some naming examples:

Database name	: DIALOGUE	Database name	: TEST
os-file (I)	: DIALO-DA	os-file (I)	: TEST-DA
os-file (II)	: DIALO-IX	os-file (II)	: TEST-IX
system realm	: DIALO-IX	system realm	: TEST-IX

2.7 OS-FILE

The command DBOSFILE defines a new os-file (operating system file) for the application database.



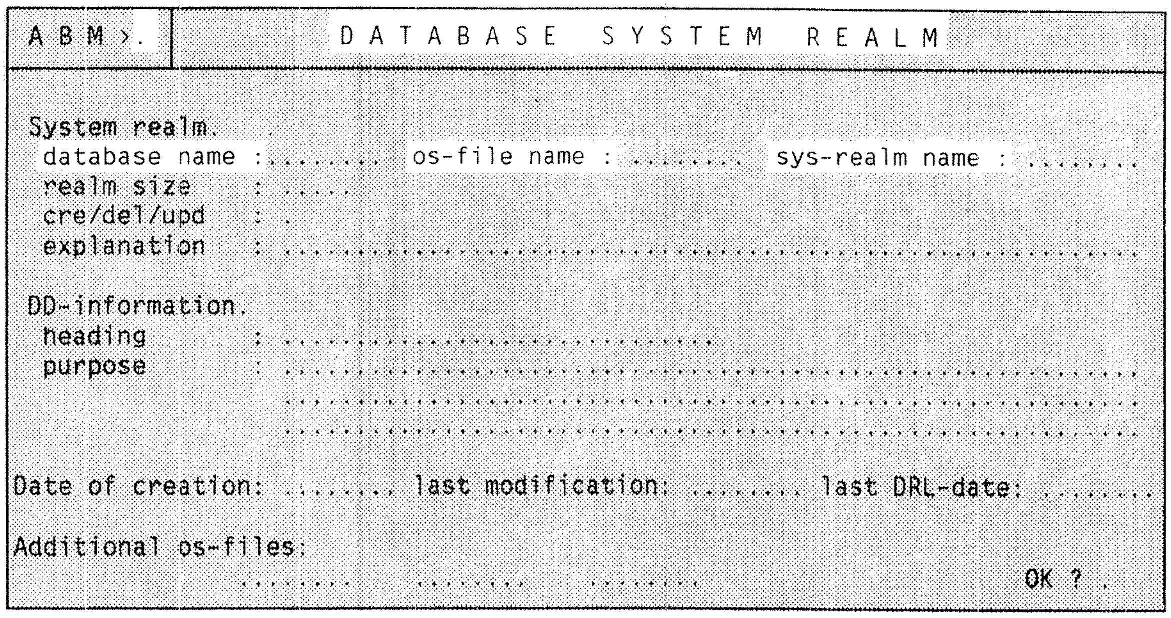
```
A B M >. |            D A T A B A S E   O S - F I L E

Os-file.
 database name : ........    os-file name   : ........

 page size     : ....        directory name : ........
 cre/del/upd   : .
 explanation   : ............................................................



Date of
 creation          : ........
 last modification : ........
 last DRL-date     : ........
                                                              OK ? .
```

All fields except the dates are input fields.

DB-NAME	:Name of the application database. Must be unique within the ABM catalog.
OS-FILE NAME	:SINTRAN III file name. Must be unique within the application database.
PAGE SIZE	:No. of words in a SIBAS page (default 512).
DIRECTORY NAME	:4 char. abbreviation of the directory where os-file is placed. If omitted, the default directory will be used.
CRE/DEL/UPD	:CREATE is the default value. This field is explained in section Database Initiation.
EXPLANATION	:60 characters free text.
DATE OF CREATION	:Date of definition of the os-file.
LAST MODIFICATION	:Date of last modification of the os-file.
LAST DRL-DATE	:Date of the last run of SCHEMA's confirmation.

2.8 SYSTEM REALM

The command DBSYSREALM will define, delete or update a system realm for the application database.



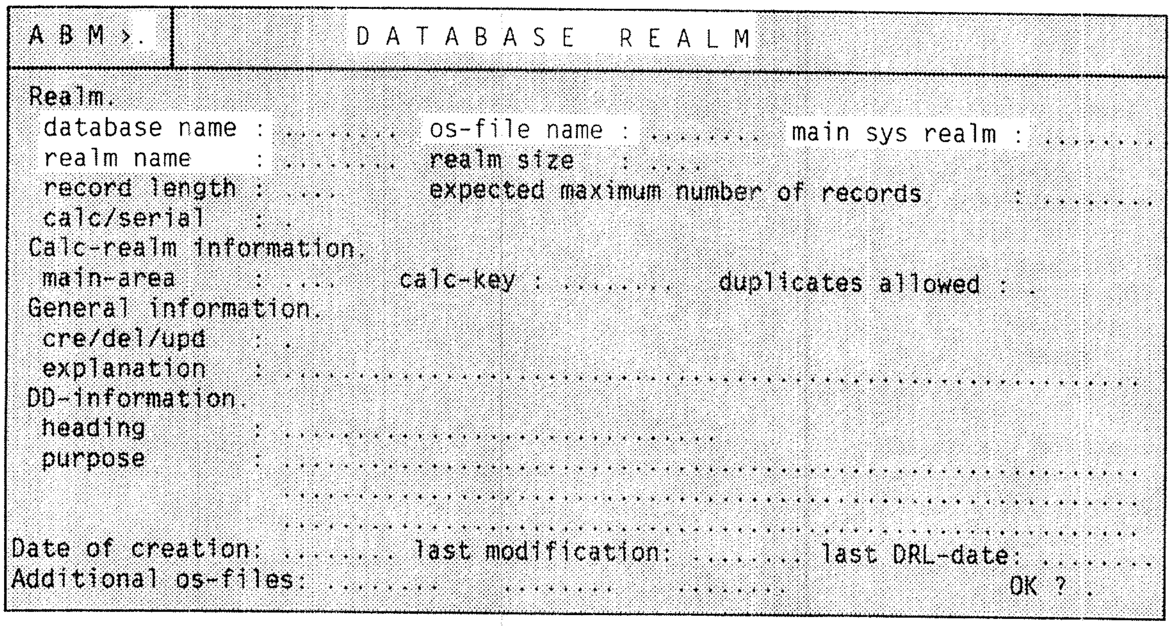
```
A B M >.    D A T A B A S E   S Y S T E M   R E A L M

 System realm.
  database name :........ os-file name : ........ sys-realm name : ........
  realm size    : .....
  cre/del/upd   : .
  explanation   : ............................................................

 DD-information.
  heading       : ..............................
  purpose       : ............................................................
                  ............................................................
                  ............................................................

Date of creation: ........ last modification: ........ last DRL-date: ........

Additional os-files:
          ........   ........   ........                               OK ? .
```

All fields except the dates are input fields.

DB-NAME	:Name of the application database. Must be unique within the ABM catalog.
OS-FILE NAME	:SINTRAN III file name. Must be unique within the application database.
SYS-REALM NAME	:System realm (for indexes). Must be unique within the application database.
REALM SIZE	:No. of 64 word pages (default 1000).
CRE/DEL/UPD	:CREATE is the default value. This field is explained in section Database Initiation.
EXPLANATION	:60 characters describing the realm.
DD-INFORMATION HEADING PURPOSE	:Data Dictionary information, please refer to Appendix C.
DATE OF CREATION	:Date of definition of system realm.
LAST MODIFICATION	:Date of last modification of system realm.
LAST DRL-DATE	:Date of the last run of SCHEMA'a confirmation.
ADDITIONAL OS-FILES	:Possibility to choose additional os-files if you use SIBAS version F or a newer version.

2.9 DATABASE REALM

With the command DBREALM you can define, delete or modify a Serial or a Calc realm.



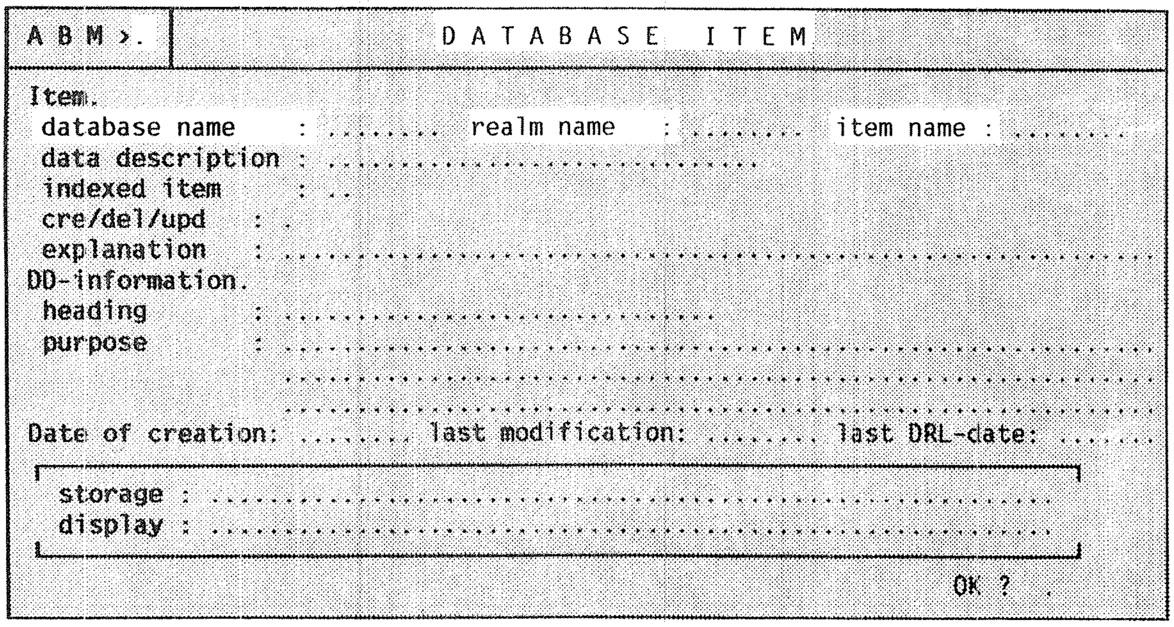
```
A B M >.        D A T A B A S E   R E A L M
Realm.
 database name : ........ os-file name : ........ main sys realm : ........
 realm name    : ........ realm size    : ....
 record length : ....     expected maximum number of records     : ........
 calc/serial   : .
Calc-realm information.
 main-area     : ....    calc-key : ........   duplicates allowed : .
General information.
 cre/del/upd   : .
 explanation   : ..............................................................
DD-information.
 heading       : ..............................
 purpose       : ..............................................................
                 ..............................................................
                 ..............................................................
Date of creation: ........ last modification: ........ last DRL-date: ........
Additional os-files: ........  ........  ........                    OK ? .
```

All fields except date/DRL-date are input fields.

DB-NAME	:The application database. Must have been defined earlier.
OS-FILE	:Must have been defined for this database.
MAIN SYS-REALM	:System realm for storing index keys for this realm. If omitted, the first defined sys-realm for this application database will be used.
REALM-NAME	:Must be unique within the DB name.
REALMSIZE	:In number of pages of os-file page size (default 100).
RECORD LENGTH	:Record length in no. of words. One word is always two bytes (default 500).
EXPECTED MAXIMUM NUMBER OF RECORDS	:Must be filled in for automatic dimensioning in the SCHEMA module.
CALC/SERIAL	:Realm type, <u>S</u>erial or <u>C</u>alc realm (default <u>S</u>).
MAIN-AREA	:No. of pages in main area. For Calc realms only.
CALC-KEY	:For Calc realms only. Refers to an item or group item which is or will be defined for the record.
DUPLICATES	:For Calc realms only. To decide whether the values of the Calc key must be unique or may have duplicates.
CRE/DEL/UPD	:CREATE is the default value. This field is explained in section Database Initiation.
HEADING	:Max. 30 chars. Text will appear as heading on screen forms and reports (see also Appendix C).
PURPOSE	:Max. 180 chars. Used for documentation and as ONLINE help (see also Appendix C).
DATE OF CREATION	:Date of definition of database realm.
LAST MODIFICATION	:Date of last modification of database realm.
LAST DRL-DATE	:Date of the last run of SCHEMA's confirmation.
ADDITIONAL OS-FILES	:Possibility to choose additional os-files if you use SIBAS version F or a newer version.

2.10 DATABASE ITEM

With the command DBITEM you can create, delete or update an item for a realm. The following screen picture appears with this command:



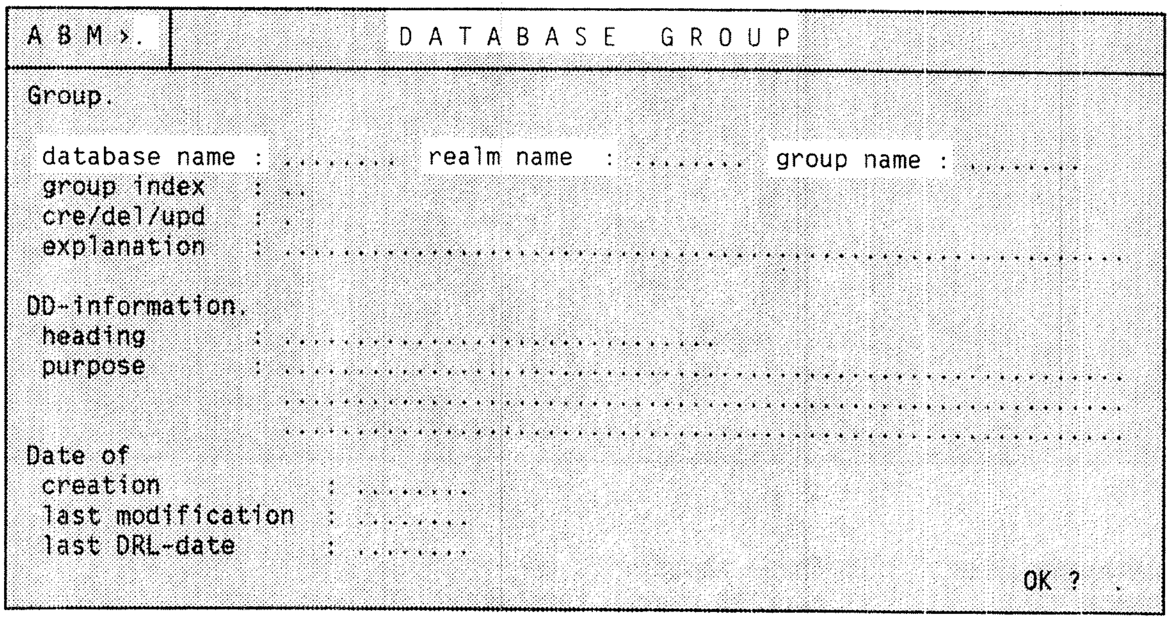
```
ABM >.              D A T A B A S E   I T E M
Item.
 database name    : ........  realm name   : ........  item name : ........
 data description : ...............................
 indexed item     : ..
 cre/del/upd      : .
 explanation      : ..............................................................
DD-information.
 heading          : ..............................
 purpose          : ..............................................................
                    ..............................................................
                    ..............................................................
Date of creation: ........ last modification: ........ last DRL-date: ........
  storage : ............................................................
  display : ............................................................
                                                                OK ? .
```

All fields except for storage, display and the dates are input fields.

DATABASE NAME	:Application database name; must have been defined earlier.
REALM NAME	:Must be unique within the application database.
ITEM NAME	:Must be unique within the realm.
DATA DESCRIPTION	:Must have been created before.
INDEXED ITEM	:AD (Automatic, Duplicates allowed). AN (Automatic, duplicates NOT allowed). MD (Manual, Duplicates allowed). MN (Manual, Duplicates NOT allowed). BLANK (No index desired).
CRE/DEL/UPD	:CREATE is the default value. This field is explained in section Database Initiation.
EXPLANATION	:60 characters describing the item.
HEADING	:Max. 30 chars. Text can appear as heading on screen forms and reports (see also Appendix C).
PURPOSE	:Max. 180 chars. Used for documentation and as ONLINE help (see also Appendix C).
DATE OF CREATION	:Date of definition of database item.
LAST MODIFICATION	:Date of last modification of database item.
LAST DRL-DATE	:Date of the last run of SCHEMA's confirmation.
STORAGE **DISPLAY**	STORAGE & DISPLAY information is fetched automatically from DATA DESCRIPTION (see Appendix A).

2.11 DBGROUP

With the command DBGROUP you can create, delete or update a group item consisting of elementary items. The items need not be contiguous in a record type. Properties such as Calc key, Index key, member set item and owner set item may be assigned to a group item in the same way as they are assigned to an elementary item. Two screen pictures are associated with the command DBGROUP. The FIRST picture is as follows:



```
A B M >.          D A T A B A S E   G R O U P

Group.

 database name : ........  realm name  : ........  group name : ........
 group index   : ..
 cre/del/upd   : .
 explanation   : ............................................................

DD-information.
 heading       : ..............................
 purpose       : ............................................................
                 ............................................................
                 ............................................................

Date of
 creation          : ........
 last modification : ........
 last DRL-date     : ........
                                                                    OK ? .
```

With this picture you primarily declare the name of the DATABASE GROUP.

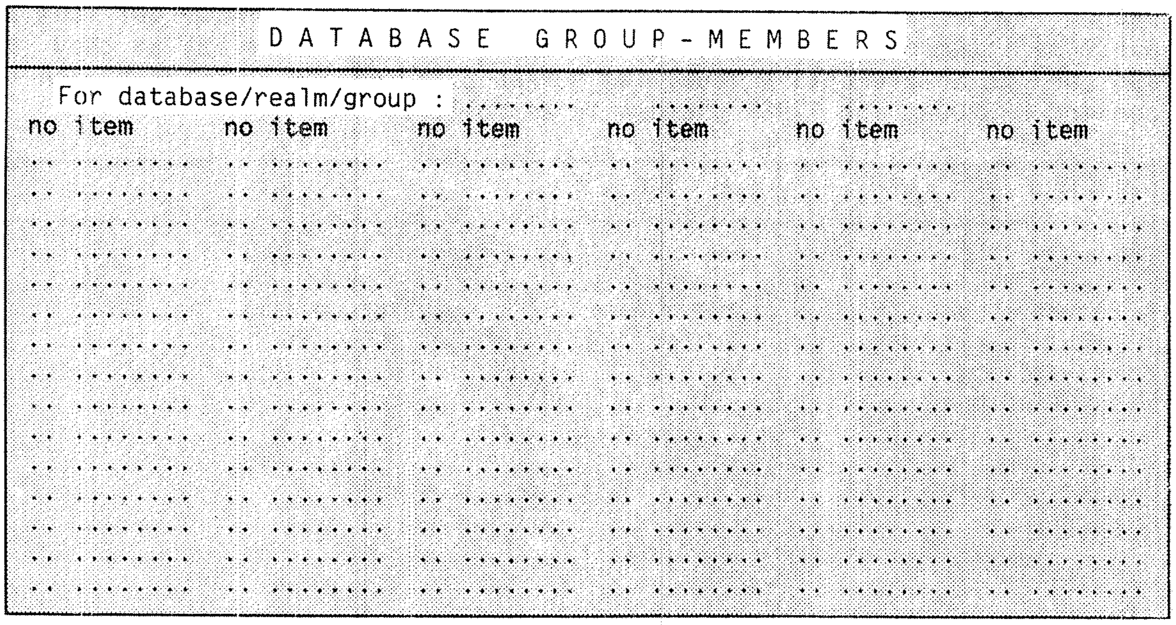
DATABASE NAME	:Database name; must have been defined earlier.
REALM NAME	:Realm name, must exist.
GROUP NAME	:Must be different from all item/group item names in the realm.
GROUP INDEX	:AD (Automatic, Duplicates allowed). AN (Automatic, Duplicates NOT allowed). MD (Manual, Duplicates allowed). MN (Manual, Duplicates NOT allowed). BLANK (No index desired).
CRE/DEL/UPD	:Create is the default when defining a new group. This field is explained in section Database Initiation.
EXPLANATION	:60 characters describing the group item.
DD-INFORMATION	:See Appendix C.
HEADING	:Max. 30 characters.
PURPOSE	:Max. 180 characters.
DATE OF CREATION	:Date of the creation of the group item.
LAST MODIFICATION	:Date of the last modification of the group item.
LAST DRL-DATE	:Date of the last run of SCHEMA's confirmation.

- In the next picture of the DBGROUP command, you can define which elementary items are to be part of the group item.

- The next picture appears when you terminate the OK field with the ↵ key or with N.

DBGROUP...(cont)

In the following picture you can specify which elementary items are to be a part of a GROUP ITEM:



```
                    D A T A B A S E   G R O U P - M E M B E R S

  For database/realm/group : ........ ........ ........
no item       no item       no item       no item       no item       no item
.. ........   .. ........   .. ........   .. ........   .. ........   .. ........
.. ........   .. ........   .. ........   .. ........   .. ........   .. ........
.. ........   .. ........   .. ........   .. ........   .. ........   .. ........
.. ........   .. ........   .. ........   .. ........   .. ........   .. ........
.. ........   .. ........   .. ........   .. ........   .. ........   .. ........
.. ........   .. ........   .. ........   .. ........   .. ........   .. ........
.. ........   .. ........   .. ........   .. ........   .. ........   .. ........
.. ........   .. ........   .. ........   .. ........   .. ........   .. ........
.. ........   .. ........   .. ........   .. ........   .. ........   .. ........
.. ........   .. ........   .. ........   .. ........   .. ........   .. ........
.. ........   .. ........   .. ........   .. ........   .. ........   .. ........
.. ........   .. ........   .. ........   .. ........   .. ........   .. ........
.. ........   .. ........   .. ........   .. ........   .. ........   .. ........
.. ........   .. ........   .. ........   .. ........   .. ........   .. ........
.. ........   .. ........   .. ........   .. ........   .. ........   .. ........
```

DATABASE NAME	!! These names will be
REALM NAME	!! carried over automatically
GROUP NAME	!! from the previous picture.
ITEM FIELDS	!!! The item fields will contain the names of items in
	!!! the specified realm of the database.

- In this picture you have to specify the items which are to be a part of the group item.

- Specify the items by giving them numbers in a sequence; otherwise you will get an error message.

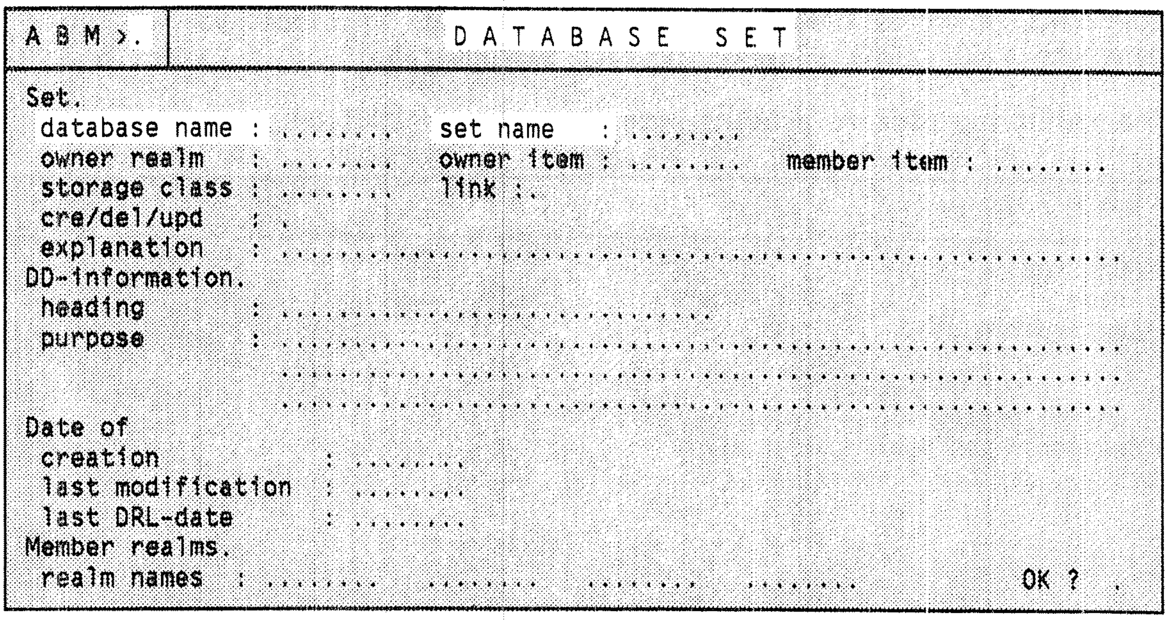
  Moving within the fields : use the keys ← and → .

  Moving from one field to another : use the keys ↓ and ↑.

- When you have finished specifying the items, you can go back to the first picture by pressing the (EXECUTE) key ↳ .

2.12 DATABASE SET

With the command DBSET, you can specify which record belongs to a particular set. The owner and member realms must be defined before using this command. The following screen picture appears with the command:



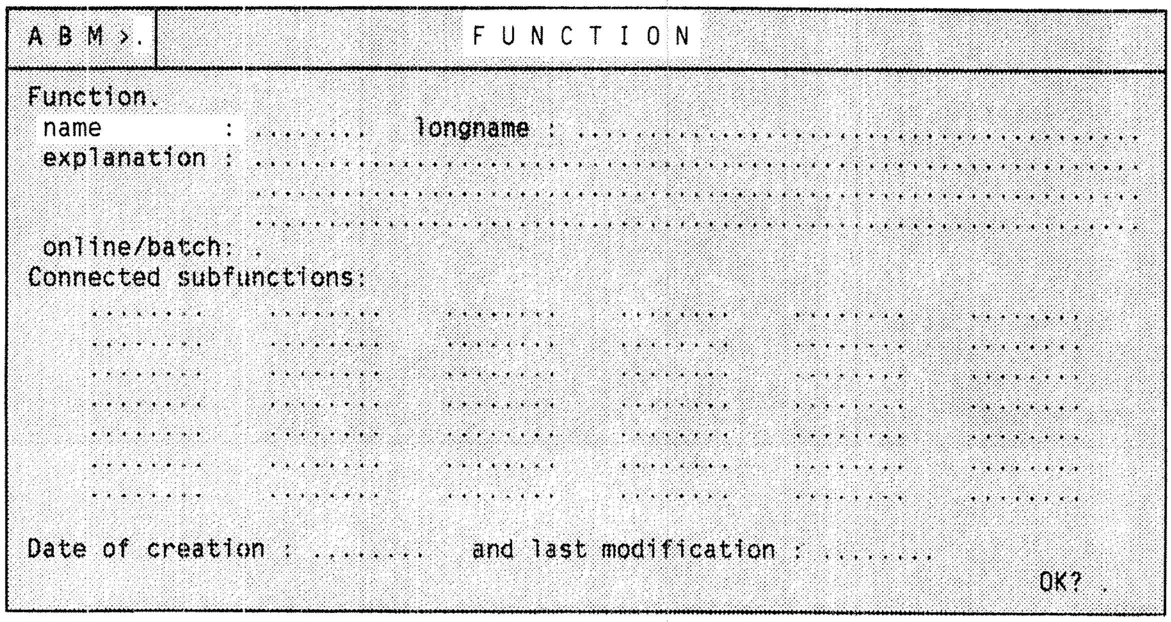
```
A B M >.              D A T A B A S E   S E T
Set.
 database name : ........   set name   : ........
 owner realm   : ........   owner item : ........   member item : ........
 storage class : ........   link :.
 cre/del/upd   : .
 explanation   : ...........................................................
DD-information.
 heading       : ...........................
 purpose       : ...........................................................
                 ...........................................................
                 ...........................................................
Date of
 creation           : ........
 last modification  : ........
 last DRL-date      : ........
Member realms.
 realm names  : ........   ........   ........   ........            OK ?  .
```

DBNAME	:Application database name; must have been defined earlier.
SETNAME	:Must be unique within the application database.
OWN-REALM	:Owner realm name. Must exist.
OWN-ITEM	:Owner set item. Must exist.
MEMB-ITEM	:Member set item. Must exist. Default: same name as in OWN-ITEM.
CLASS	:Storage class (Automatic/Manual). Default "A".
LINK	:Link (Single/Double). Default "D".
DD-INFORMATION	See Appendix C.
HEADING	
PURPOSE	
DATE	:Date of the creation or last modification in the ABM catalog.
DATE OF CREATION	:Date of the definition of database set.
LAST MODIFICATION	:Date of the last modification of database set.
LAST DRL-DATE	:Date for the last run of SCHEMA's confirmation.
MEMBER REALMS	
REALM NAMES	:Member realm name must exist. Only for users of multi-member-set. (Usually just one realm name is filled out.) Maximum four multi-member-sets allowed. Terminate all four fields with ↵.

2.13 MAINTENANCE OF FUNCTIONS

A function in ABM consists of one or more subfunctions which belong together logically. If you use a function as basis for generating COPY/INCLUDE files, you will automatically get COPY/INCLUDE files for all subfunctions which the actual function consists of. If you want to generate COPY/INCLUDE files from subfunctions instead of functions, you will have to generate a COPY/INCLUDE file for each subfunction.

The following picture appears with the command FUNCTION:



```
A B M >.              F U N C T I O N
Function.
 name        : ........   longname : ........................................
 explanation : ..............................................................
               ..............................................................
               ..............................................................
 online/batch: .
Connected subfunctions:
     ........   ........   ........   ........   ........   ........
     ........   ........   ........   ........   ........   ........
     ........   ........   ........   ........   ........   ........
     ........   ........   ........   ........   ........   ........
     ........   ........   ........   ........   ........   ........
     ........   ........   ........   ........   ........   ........
     ........   ........   ........   ........   ........   ........

Date of creation : ........   and last modification : ........
                                                                OK? .
```

FUNCTION NAME	:Must be unique within each ABM catalog. The name must not be ALL, CHANGES or STOP, as these are reserved within ABM.
ONLINE/BATCH	:O or B. Default is "O".
EXPLANATION	:Information field for the user.

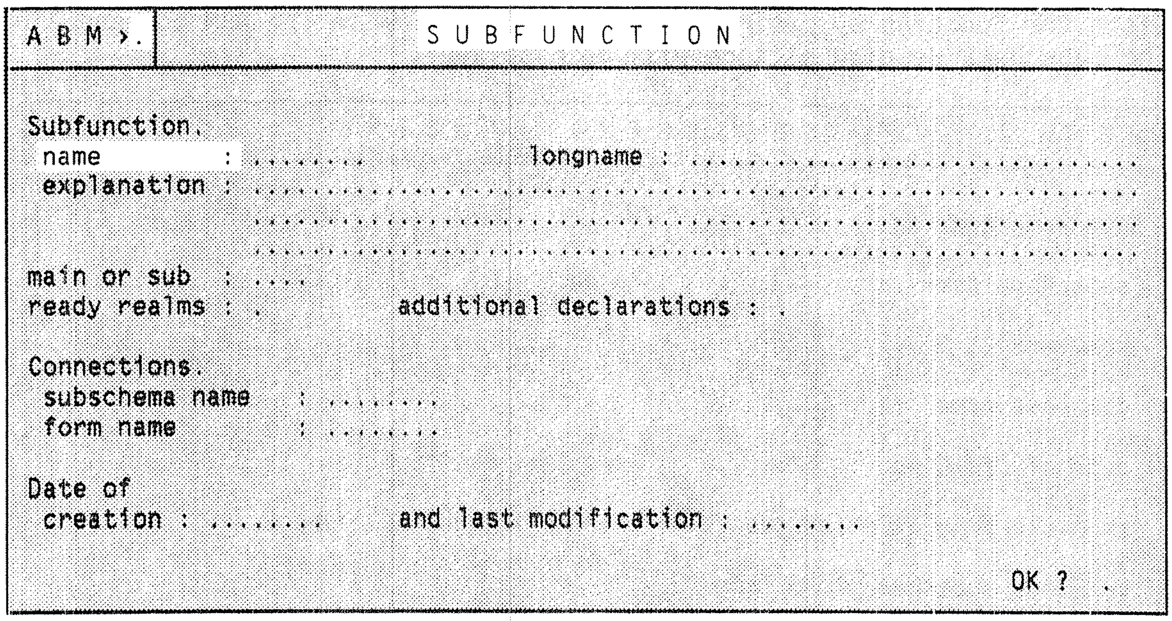
A function cannot be deleted if subfunctions are connected. A subfunction is disconnected from the function by clearing the name from the list.

2.14 MAINTENANCE OF SUBFUNCTIONS

A subfunction can be used to define parts of an application program. A subfunction may also be the basis for producing INCLUDE/COPY elements.

In order to use the ABM subroutines for application database, program and screen communication, a screen picture and/or a subschema must be referred to in the subfunction.

The command SUBFUNCTION shows the following picture:



```
A B M >.              S U B F U N C T I O N

Subfunction.
 name        : ........          longname : ..............................
 explanation : .............................................................
               .............................................................
               .............................................................
main or sub  : ....
ready realms : .        additional declarations : .

Connections.
 subschema name   : ........
 form name        : ........

Date of
 creation : ........    and last modification : ........

                                                            OK ?  .
```

SUBFUNCTION NAME	:Must be a unique name within the ABM catalog.
LONG NAME	:A more explanatory name, only used for documentation.
EXPLANATION	:Information field for the user.
MAIN/SUB ROUTINE	:Only used for documentation. M: if subfunction is a MAIN routine. S: if subfunction is a SUB routine.
READY REALM	R: if subfunction code includes preparing to call SIBAS "READY REALM" <space>: if no READY REALM.
ADD. DECLARATION	:Y or N (Yes/No), Y is default. If Y, some additional declarations will be made in the COPY/INCLUDE files. These are: DBSTATUS - Database status FCSTATUS - Focus/form status FORMFILE - Form file name MESSAGE - Message line TDBKEY - Temporary database key OPTION - Forget/remember option code TDBSRI - Temporary search region indicator OTEXT - Output from DDGTEXT and DDGMSGE
SUBSCHEMA NAME	:Name of subschema.
FORM NAME	:Form name.

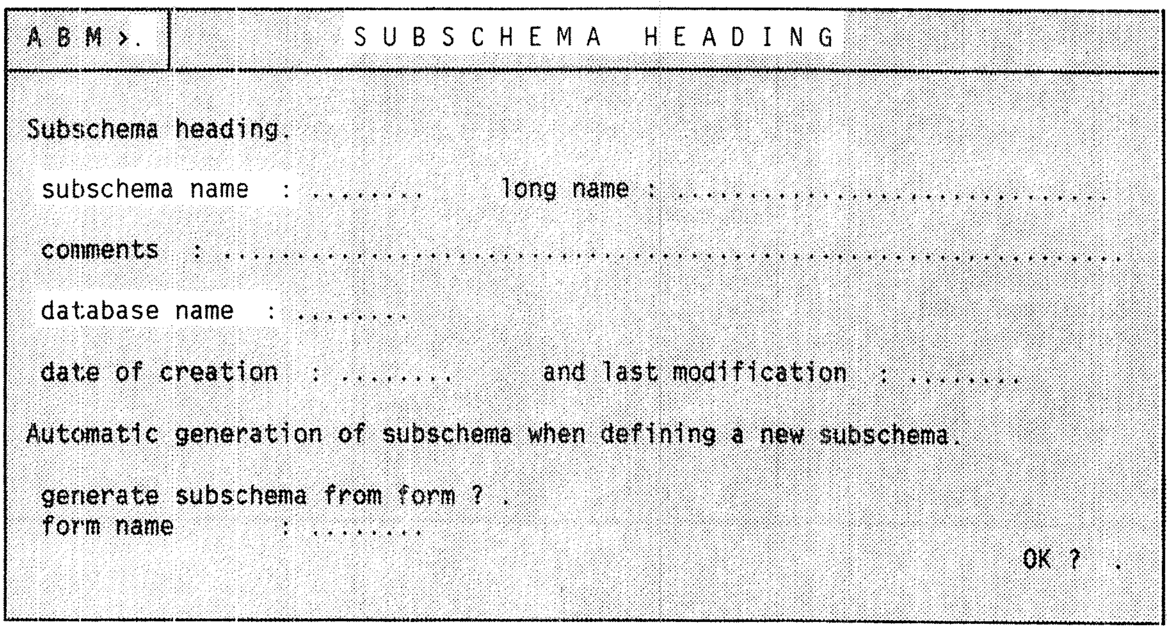
2.15 MAINTENANCE OF SUBSCHEMAS

A subschema specifies which part of the application database is of interest in a particular application.

Building up a subschema in ABM is done by marking the desired realms from a list of all realms and then marking the desired items or group items for each of the already-marked realms.

A new subschema may be generated automatically from a form.

Start the function by typing the command SUBSCHEMA in the ABM command line. The Subschema Heading picture will be displayed.



```
A B M >.            S U B S C H E M A   H E A D I N G

Subschema heading.

  subschema name   : ........     long name : ..............................

  comments  : ..............................................................

  database name   : ........

  date of creation   : ........     and last modification   : ........

Automatic generation of subschema when defining a new subschema.

  generate subschema from form ? .
  form name        : ........
                                                                  OK ?  .
```

You can move between different subschemas by using the commands F, N, P and L in the ABM home area:

F: Display First subschema
N: Display Next subschema
P: Display Previous subschema
L: Display Last subschema

When giving the command M (Modify) in the ABM home area you can change the contents of those fields in the Subschema Heading picture which only contain comments.

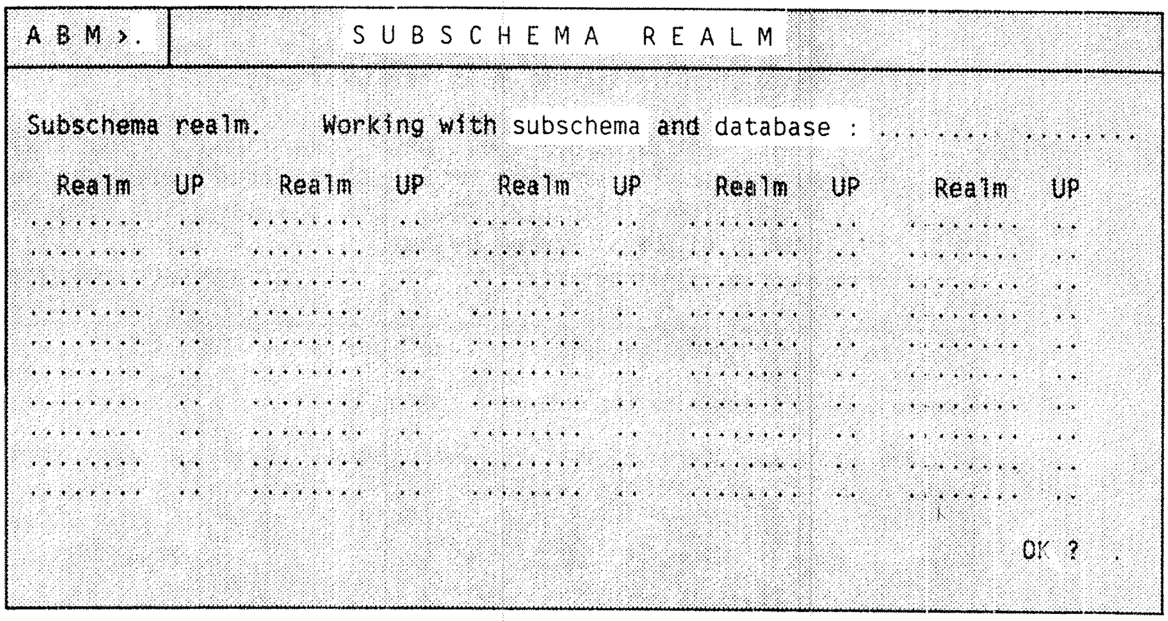
SUBSCHEMA NAME	:Name of the subschema. Subschema name must be unique within the ABM catalog.
LONG NAME	:A more explanatory name, only used for documentation.
COMMENTS	:Used for documentation.
DATABASE NAME	:Name of the database this subschema specifies.

GENERATE SUBSCHEMA FROM FORM :Y or N (Yes/No), Y is only possible if you want to generate a new subschema automatically from a form. If Y, fill in the name of the form you will make a corresponding subschema of. This will establish a subschema very quickly. The generated subschema will consist of all the indirectly defined fields in the specified form. (For further information of the fields which refer to "database", "realm" and "item", see section Generating subschema from form, page 68.)

OK? :Y - update the ABM catalog.
:N - no update of the ABM catalog;
go to the next part of the screen picture.
:R - no update, repeat data registration.

If Y in both the OK field and the "generate subschema from form" field, the subschema will be generated from the form you have specified. The subschema will be stored in the ABM catalog.

If Y in the OK field and N in the "generate subschema from form" field, the next picture, Subschema Realm, will be displayed:



```
A B M >.          S U B S C H E M A   R E A L M

Subschema realm.   Working with subschema and database : ........ ........

  Realm    UP     Realm    UP     Realm    UP     Realm    UP     Realm    UP
 ........  ..    ........  ..    ........  ..    ........  ..    ........  ..
 ........  ..    ........  ..    ........  ..    ........  ..    ........  ..
 ........  ..    ........  ..    ........  ..    ........  ..    ........  ..
 ........  ..    ........  ..    ........  ..    ........  ..    ........  ..
 ........  ..    ........  ..    ........  ..    ........  ..    ........  ..
 ........  ..    ........  ..    ........  ..    ........  ..    ........  ..
 ........  ..    ........  ..    ........  ..    ........  ..    ........  ..
 ........  ..    ........  ..    ........  ..    ........  ..    ........  ..
 ........  ..    ........  ..    ........  ..    ........  ..    ........  ..

                                                                    OK ?  .
```

MARKING REALMS All the defined realms of the given database will be displayed in this picture. Indicate the realms you want the subschema to consist of by marking the UP fields for the actual realms. You can choose what the realms will be used for, and which type of protection they shall have.

UP :Legal marking of realms of the UP field:

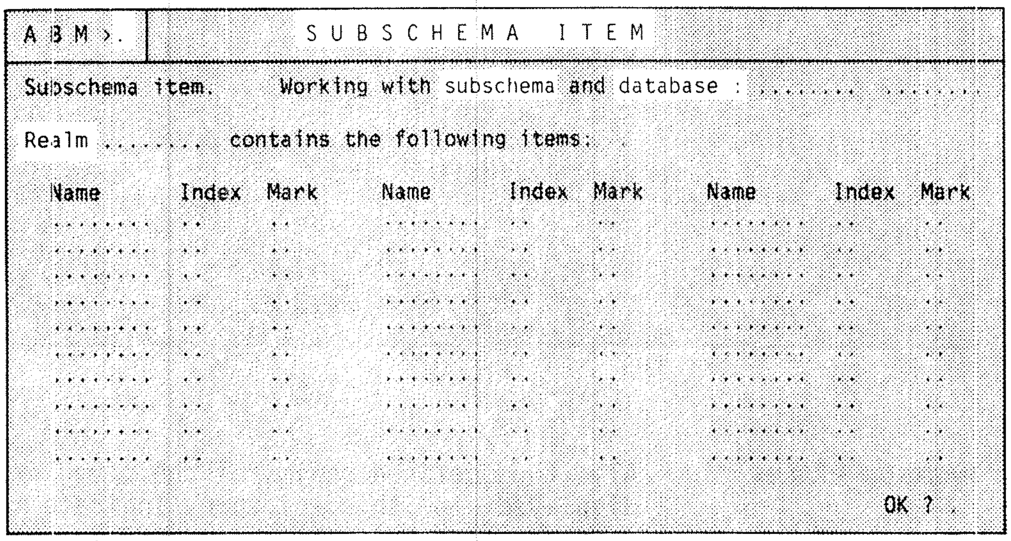
"UN" - Realms to be used for Update and with No protection.
"RN" - Realms to be used for Retrieval only and with No protection.
"UP" - Realms to be used for Update and with Protection - that is exclusive update.
" " - The realm is not of interest in this subschema.

A screen picture may contain a maximum of 50 realms at one time. To display the remaining realms, press the N (Next) key in ABM home area. (You will get a message on the screen if the database contains more than 50 realms.)

When generating a <u>new</u> subschema (even when there are more than 50 realms), you do not have to press the N key. The next screen picture holding the remaining realms will be displayed automatically.

When you have marked all the realms of interest in one picture, press the EXECUTE key. The marking of realms will be stored in the ABM catalog when you confirm the selection by typing Y in the OK field.

When generating a new subschema, you will automatically move on to Subschema Item and Subschema Group Item pictures. They look the same. A Subschema Item picture is shown below.



```
A B M > .  |            S U B S C H E M A   I T E M

Subschema item.    Working with subschema and database : ........ ........

Realm ........ contains the following items:

  Name      Index  Mark     Name      Index  Mark     Name      Index  Mark
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..

                                                                   OK ? .
```

MARKING ITEMS/ GROUP ITEMS

For each marked realm in the subschema realm picture, all items or group items from the ABM catalog will be displayed. Indicate the items/group items which are of interest to you by marking the "MARK" fields as illustrated on the next page. Then press the EXECUTE key. The items or group items of the next marked realm will be displayed, and so on until you have marked all the actual items/group items of the marked realm.

INDEX :This is key index from DRL-schema definition in ABM. The values displayed here are the values which you filled in during definition of DBITEM or DBGROUP. (See section Database Item.)

"AD"	- Automatic, Duplicates are allowed.
"AN"	- Automatic, duplicates are Not allowed.
"MD"	- Manual, Duplicates are allowed.
"MN"	- Manual, duplicates are Not allowed.
" "	- No index is desired.

MARK :You can select items/group items for the actual realm by using the following marks:

"I "	- Used as Item (not as key).
"K "	- Used as Key only.
"IK"	- Used as both Item and Key.
" "	- The item will not be used in this subschema.

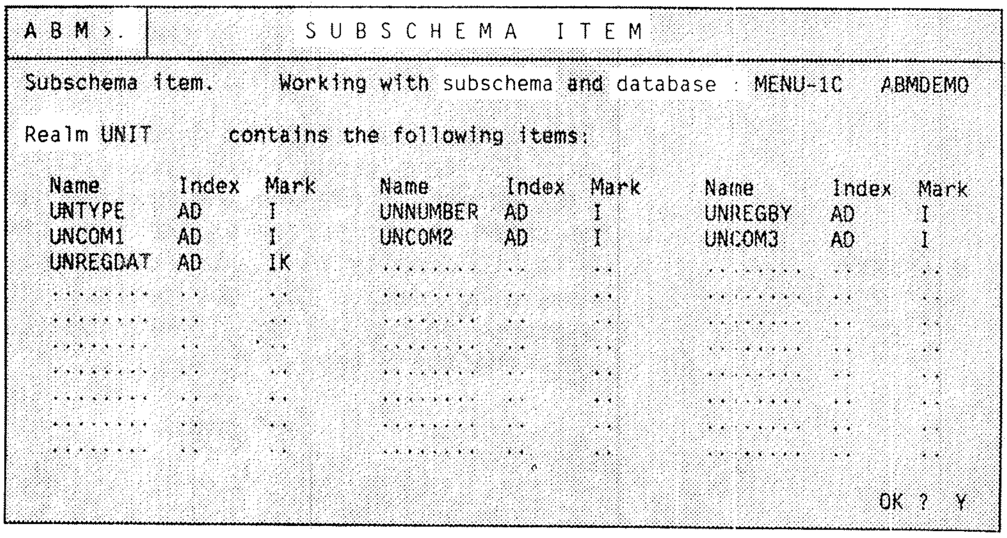
For group items the only valid marks are:

"K "	- Used as Key only.
" "	- The group will not be used in this subschema.

A screen picture may contain a maximum of 30 items or group items. If a realm contains more than 30 items or group items, press the N key in ABM home area to display the remaining items/group items.

When generating a <u>new</u> subschema (even when there are more than 30 items/group items), you do not have to press the N key. The next screen picture holding the remaining items/group items will be displayed automatially. (After you have ended the selection in one picture by pressing the EXECUTE key and confirmed the selection by typing Y in the OK field.)

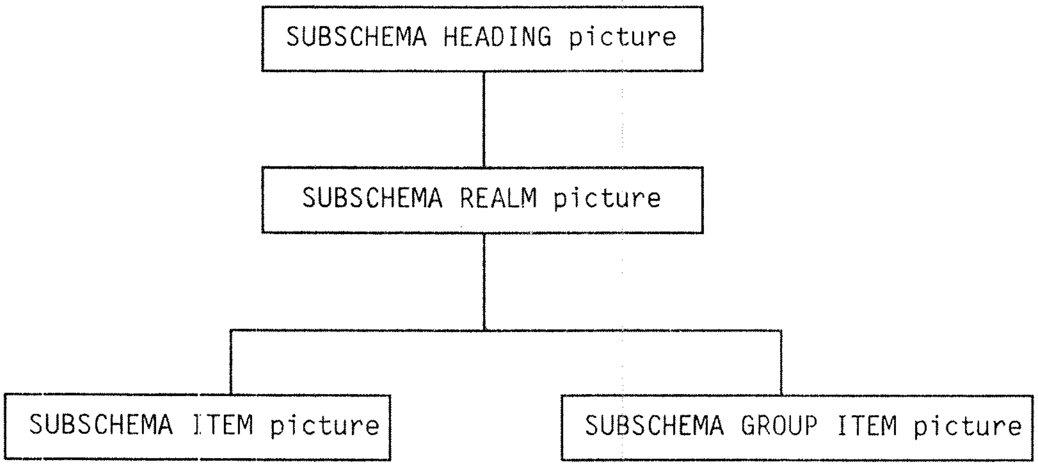
Here is an example of a filled in Subschema Item picture:



```
A B M >.        S U B S C H E M A   I T E M

Subschema item.    Working with subschema and database : MENU-1C   ABMDEMO

Realm UNIT       contains the following items:

  Name      Index  Mark     Name      Index  Mark     Name      Index  Mark
  UNTYPE    AD     I        UNNUMBER  AD     I        UNREGBY   AD     I
  UNCOM1    AD     I        UNCOM2    AD     I        UNCOM3    AD     I
  UNREGDAT  AD     IK       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..

                                                                 OK ?  Y
```

2.15.1 NAVIGATION IN SUBSCHEMA PICTURES

The SUBSCHEMA command leads to three different levels of related pictures; Heading picture, Realm picture and Item/Group Item picture, as shown below.



When NOT generating a new subschema, you can move between related pictures (forms) within the same level. You can also navigate up and down between the three levels of pictures by issuing commands in the ABM home area.

><<> Use this key to move one level of pictures down.

SHIFT + ><<> Use these keys to move up one level of pictures.

F (FIRST) When the Subschema Realm, Item or Group Item picture is displayed, the fields will contain the selections made previously. The fields may also be displayed by pressing the F key in the ABM home area (see below).

If the database contains more than 50 realms or more than 30 items/group items, you will get a message on the screen. These are the maximum numbers which can be shown on the screen at the same time. Press the

N (NEXT) N key to display the next picture which contains the remaining data. You may redisplay the first Realm picture or the first Item/Group Item picture by pressing the F key.

M (MODIFY)

Pressing the M key in ABM home area of a Subschema Realm, Item or Group Item picture, will enable you to change the selection made previously. When the desired changes are made, terminate with the EXECUTE key and confirm with Y in the OK field.

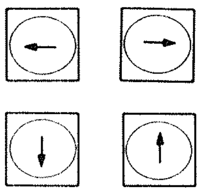
AN EXAMPLE OF NAVIGATION

Type the command SUBSCHEMA in ABM command line, and the Subschema Heading picture will be displayed. (Use the commands F, N, P, and L to move to any desirable Subschema Heading pictures.)

If you want to move one level down to a Subschema Realm picture, press the "<>" key.

Move between related Realm pictures by using the commands F and N.

ARROW KEYS



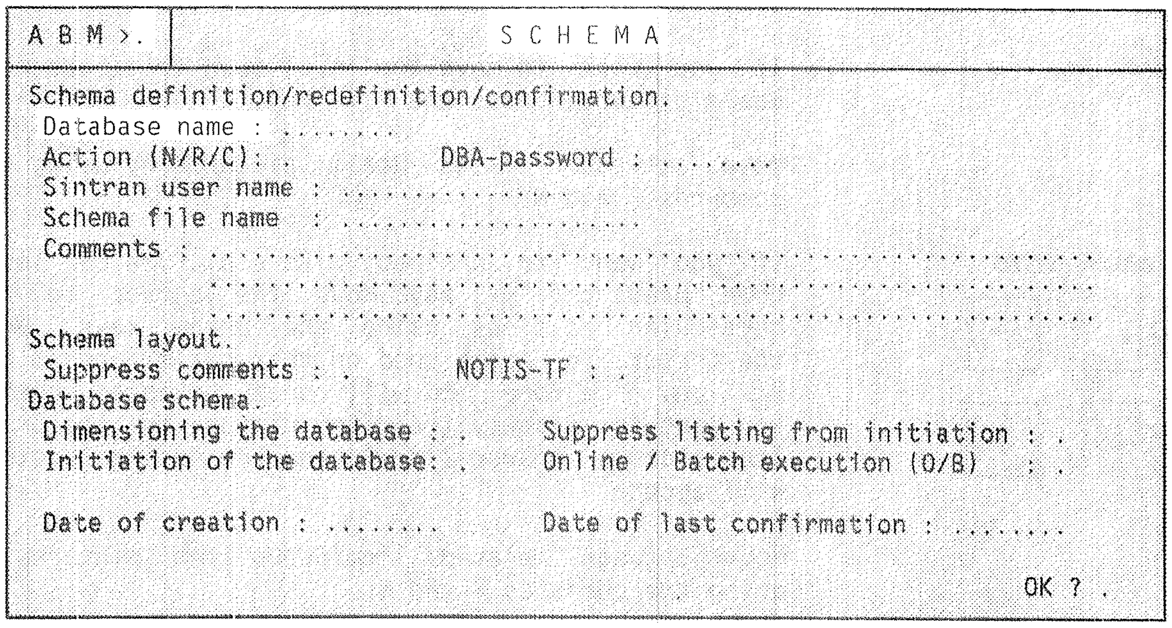
If you want to proceed with a picture one level down, press the "<>" key. Move the cursor to the particular realm by using the arrow keys. When the cursor is positioned on that realm, press the "<>" key once more. You will then be asked whether you want to display the Item picture or the Group Item picture. Answer I or G, and the desired picture will appear.

Move between related Item/Group Item pictures by using the commands F and N.

If for example, you have displayed a Subschema Item picture on the screen, and you want to navigate up to the Heading picture, continue to press SHIFT + "<>", which will lead you up level by level, until you reach the Subschema Heading picture.

2.16 GENERATING SCHEMAS

Each database has a corresponding source schema which describes the structure of the database. The source schema is translated into an internal representation of the database, the object schema, by SIBAS-DRL (Definition Redefinition Language). The ABM SCHEMA command generates complete source schemas to be processed by SIBAS-DRL.



```
A B M >.  |                      S C H E M A

Schema definition/redefinition/confirmation.
 Database name : ........
 Action (N/R/C): .            DBA-password : ........
 Sintran user name : ................
 Schema file name  : ......................
 Comments : ...............................................................
            ...............................................................
            ...............................................................
Schema layout.
 Suppress comments : .        NOTIS-TF : .
Database schema.
 Dimensioning the database : .     Suppress listing from initiation : .
 Initiation of the database: .     Online / Batch execution (O/B)   : .

 Date of creation : ........       Date of last confirmation : ........

                                                                 OK ? .
```

The values from the last Definition/Redefinition/Confirmation run (see below) for a database are stored in the ABM catalog as a "Schema entry". When the SCHEMA command has been issued, these entries can be examined using the normal commands in the home area (e.g., F(First), N(Next)......). When generating the Schema for a new database, the ↵ key is used for creating a new Schema entry. The following SCHEMA-runs for this database will have to use the M(Modify) command to change the Schema entry. If a Schema entry is deleted (using the D command), the corresponding DRL-files for Definition and Redefinition will NOT be deleted.

DATABASE NAME	:Name of an application database in the ABM catalog.
ACTION: **N (New)**	:A DRL-program (source schema) for defining the given database is generated. The program generates definition of all elements except those marked "D" in the "cre/del/upd" field.
R (Redefine)	:A DRL-program (Redefinition statement for DRL) for redefining the given database is generated. DRL-statements are generated for all changes made to the database since last Confirmation (see below). No Redefine is allowed before the first Confirmation of the database.

Modifications made to the elements in the ABM catalog are marked in the "cre/del/upd" field. When you do an "M" (Modification) in an ABM picture (after a Confirmation), the "cre/del/upd" field will automatically be set to "U".

To delete an element (after a Confirmation), do an "M" and explicitly mark the "cre/del/upd" field with a "D".

Setting these flags on modified elements enables ABM to record that an element has been modified, but not which fields have been changed within the element. ABM will therefore generate redefinition clauses for all the fields of the element.

For example, if you change the REALM-SIZE of a serial realm, a CHANGE SERIAL-REALM statement will be generated. The statement will contain redefinition of REALM-SIZE, RECORD LENGTH, HEADING and PURPOSE. Because of this , most CHANGE-statements in a Redefinition-file will contain several unnessecary clauses, but they will only have the effect of "resetting" the fields to their present value.

C (Confirm) :Confirmation may be thought of as a way of telling ABM that the description contained in ABM of the application database now equals the actual structure of the application database. (for example, when you take your application from test phase to production.) All elements marked "D" in the "cre/del/upd" fields will now be physically deleted from the ABM catalog, and the remaining elements will have their "cre/del/upd" fields set to "S" (Set confirmation flag) to mark them as Confirmed.

A Confirmation will update the DRL-dates.

When a Redefinition is run, all changes made to the database are recorded in the "cre/del/upd" field. This enables ABM to locate all changes made since the last Confirmation run.

After a Confirmation it is impossible for ABM to sort out the modified elements, since all "cre/del/upd" fields are set to "S". Running a Confirmation means that the information about changes made to the application database is lost.

In the development phase you may want to take out a New Schema every time you make database corrections. So, do not run Confirmation before your application system is finished.

If, for example, you want to delete an item from a realm after the database has been confirmed, then:

- Enter ABM and use the command DBITEM and mark the "New/Del/Change" field with "D".
- Run SCHEMA (giving "R" for redefinition).
- Redefine the database using SIBAS-DRL.
- Enter ABM and confirm the database once again.

DBA PASSWORD :When redefining the database , a password for the optional PASSWORD-clause in the START REDEFINITION-statement may be entered. The password will not be displayed on the screen.

SINTRAN USER NAME :Output from Schema generation is written to this user area. Default user is the current user, but the output may be written to another user area if you have write access to that user. An empty user name denotes current user.

SCHEMA FILE NAME :Name of output file for Schema generation. The NEW command generates a DRL-file for database definition. The REDEFINE command generates a DRL-file for database redefintion. The CONFIRM command generates a log-file listing error messages and the elements that have been deleted from the ABM catalog. If the file does not exist, it will be created.

Default file names are:

NEW	<database-name>-SCHEMA:SYMB
REDEFINE	<database-name>-REDEF:SYMB
CONFIRM	<database-name>-CONF:SYMB

COMMENTS :Maximum 180 characters.

SUPRESS COMMENT :If Y (Yes), all comments from "Explanation" field will be omitted. Use this to obtain a smaller-sized source schema.

NOTIS-TF :If Y (Yes), NOTIS-TF commands will be added to make every realm definition start on a new page. NOTIS-TF will also generate a table of contents, which is very useful when writing system documentation.

DIMENSIONING THE DATABASE	:If Y (Yes), dimensioning of the database will be done automatically. (At present, the automatic dimentioning has no effect on CALC-REALM.) The automatic dimensioning requires you to have given "Expected maximum number of records" in the Database Realm menu.
SUPRESS LISTING FROM INITIATION	:If Y (Yes), no documentation will be produced when SIBAS-DRL is run.
INITIATION OF THE DATABASE	:If Y (Yes), SIBAS-DRL is started automatically as a batch job after the database definition/ redefinition file has been generated. This batch job will generate or redefine the application database. ABM appends a MODE-file named <database-name>-BATCH:MODE to batch processor number 1. Output from the batch job is written to the file <database-name>-BATCH:LIST. Output listing from SIBAS-DRL is written to the file <database-name>-DRL:LIST. If both Schema generation and SIBAS-DRL are to be run as batch jobs, they will share the same MODE-file. This requires, however, that JEC (Job Execution Control) be installed. All batch jobs are executed under current user (i.e., the "Sintran user name" has no effect).
ONLINE/BATCH EXECUTION	:"O" will give Schema generation while you wait. "B" will submit the Schema generation as a batch job, and return control to home area immediately. ABM appends a MODE-file named <database-name>-BATCH:MODE to batch processor number 1. Output from the batch job is written to the file <database-name>-BATCH:LIST.
DATE OF CREATION	:The system will generate the date of this Schema generation/Redefinition/Confirmation.
DATE OF LAST CONFIRMATION	:The system will generate the date of last Confirmation for this database.
OK?	:If Y (Yes), a final approval of the operation is done. If N (No), the Schema generation will will not start. If R (Repeat), repeat data registration. (A thorough explanation of the OK field is found in section Navigation in the command pictures.)

PROCEDURES FOR DIMENSIONING SIBAS DATABASES:

Dimensioning of a database can be done automatically by answering Y to automatically dimensioning in the Schema picture.

If you do not want the dimensioning to be done automatically, you have to update the necessary parameters in the DB modules in ABM. You may also edit directly on the Schema.

2.17 GENERATING COBOL COPY ELEMENTS AND FORTRAN INCLUDE FILES

The commands COPY-GEN and INCLUDE-GEN will automatically generate declarations and assignments needed to run a program using SIBAS and FOCUS. COPY/INCLUDE-GEN uses the variable declarations and the value assignments already defined in data descriptions, subschemas and screen forms. The output from the generation is separated into two SINTRAN files. The names of these two files depend on whether you use the command COPY-GEN or INCLUDE-GEN.

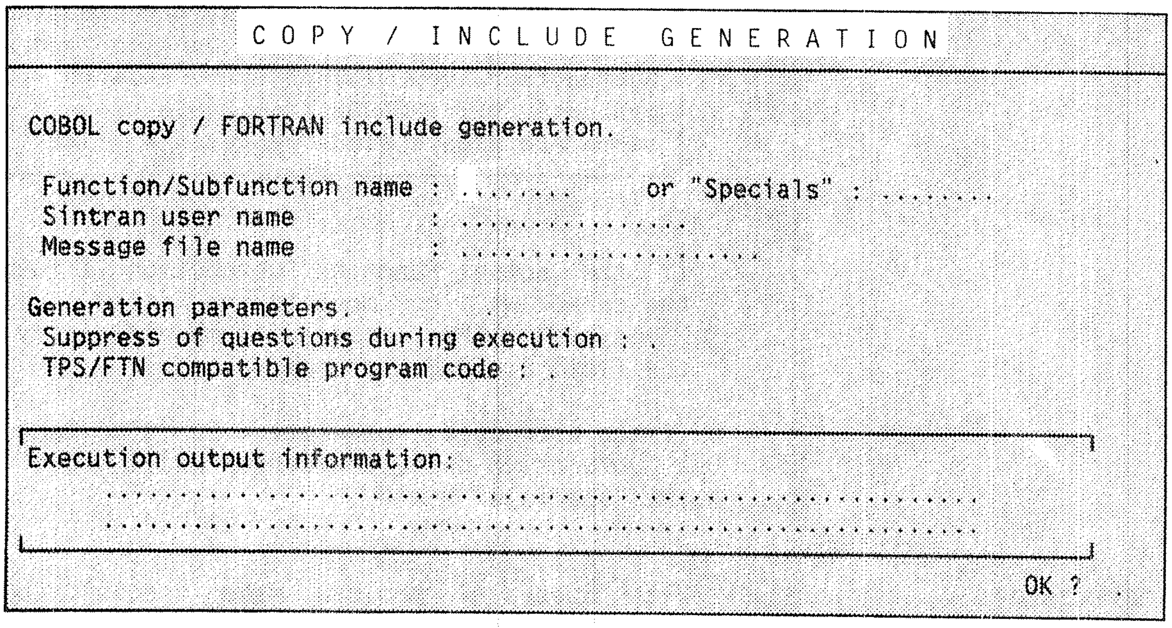
The SINTRAN files from COPY-GEN holding COBOL code are named:

DECDDC-<subfunction name>:SYMB	**for declarations**
ASSDDC-<subfunction name>:SYMB	**for assignments**

The SINTRAN files from INCLUDE-GEN holding FORTRAN code are named:

DECDDI-<subfunction name>:SYMB	**for declarations**
ASSDDI-<subfunction name>:SYMB	**for assignments**

The following picture is shown when you use either the command COPY-GEN or the command INCLUDE-GEN from the ABM command line. (The difference between these two commands will first be shown in the output from the actual generation).



```
                    C O P Y  /  I N C L U D E   G E N E R A T I O N

COBOL copy / FORTRAN include generation.

 Function/Subfunction name : ........     or "Specials" : ........
 Sintran user name         : ................
 Message file name         : ......................

Generation parameters.
 Suppress of questions during execution : .
 TPS/FTN compatible program code : .

Execution output information:
     ..............................................................
     ..............................................................
                                                             OK ? .
```

FUNCTION/SUBFUNCTION NAME	:Write the name of a Function or Subfunction. In ABM version A, you had to refer to a Function when making COPY elements or INCLUDE files. In the B version of ABM, you may refer to a Subfunction instead of a Function.

When you have written a name of a Function or Subfunction in this field, for example X, ABM will start searching for a Function named X. If a Function X exists, generation of COPY/INCLUDE will involve all the Subfunctions belonging to Function X. (There might be just one Subfunction belonging to Function X.)

If no Function X is located, ABM will start searching for a Subfunction named X. If a Subfunction X is found, the generation of COPY or INCLUDE will involve just this single Subfunction.

An error message will be shown on the screen if ABM does not find either a Function nor a Subfunction called X.

> NOTE:
> ABM will always search for a Function first. If no Function with the actual name is found, ABM will then search for a Subfunction with the same name.
>
> Do not let Functions and Subfunctions which have nothing in common, have the same name. If X exists both as a Function and a Subfunction, you will never be able to generate COPY/INCLUDE from Subfunction X, since ABM always finds Function X first.

If the actual Function/Subfunction is found, the cursor will not move to the field called "Specials". The cursor will move directly to the field SINTRAN user name.

If the user does not write a Function name or a Subfunction name, the cursor will move to the "Specials" field.

"SPECIALS" :With "Spesials" you can make special COPY/INCLUDE generations. Legal values are ALL and CHANGES.

ALL :Take care !! This generation will take quite a long time, as using ALL involves all Functions and Subfunctions defined in the ABM catalog!

CHANGES :This will make new COPY/INCLUDE files for all Functions/Subfunctions which have been changed since last Confirmation.
(See section Generating Schemas, page 38.)

SINTRAN USER NAME	:Current SINTRAN user will be shown in this field. You have to change this user name to to generate files on another SINTRAN user. (You must have write access to the user.)
MESSAGE FILE NAME	:Name of the file where messages from the COPY/ INCLUDE generation are written. Default name is ABM-MESS-<nnnnn>:DATA, where nnnnn is a five-digit number indicating the terminal from which the job is executed. You may use another message file name if you want.
SUPPRESS OF QUESTIONS DURING EXECUTION	:N (No) is default value. During generation of COPY/INCLUDE, there may appear situations where further execution seems doubtful. The user will usually get a question on the screen, asking whether the generation should continue or not. (Use the OK field to answer Y or N to these questions.) To suppress such questions and continue the generation in any case, answer Y (Yes) in this field.
TPS/FTN COMPATIBLE PROGRAM CODE	:Default value is Y (Yes). All code generated in COPY-GEN has to be TPS <1> compatible. (No use of COBOL VALUE clause.) All code generated in INCLUDE-GEN has to be both TPS and FTN <2> compatible. (Variables must not consist of more than seven significant signs, and you are not allowed to use DATA statements.) If N (No), the generated code does not have to be TPS/FTN compatible. Then the result from the generation would be just one SINTRAN file for each Subfunction. This file would contain the necessary declarations and assignments. FORTRAN DATA statements and COBOL VALUE clause could be used, and FORTRAN variable names could consist of up to 31 significant signs. (The possibility of answering N to this question is not implemented in the Version B revision 00 of ABM.)
EXECUTION OUTPUT INFORMATION	:These two lines (in addition to the usual message line) are used to give information to the user during execution of the COPY/INCLUDE generation.

<1> Tele Prosessing System
<2> TPS compatible old FORTRAN compiler

OK?

:Legal values are Y/N/R. Default is Y (Yes). Y will start generation of COPY/INCLUDE.

If N (No), execution will not start. After you have typed N and pressed ENTER (CR), you will be asked whether you want to continue generating or not. If Y, cursor will be positioned in the field "Function/Subfunction name". Then you can start a new generation.

If you do not want to continue, write N and the execution terminates.

If R (Repeat), repeat data registration. (See also explanation of the OK field in section Navigation in the command picture.)

The OK field is also used during execution. If you typed N in the "Supress of question" field, you may get some questions during execution. Answer these questions by typing Y or N in the OK field.

2.17.1 GENERATING FORTRAN INCLUDE FILES

The command INCLUDE-GEN is used to generate the FORTRAN include files:

DECDDI-<subfunction name>:SYMB for declarations,
ASSDDI-<subfunction name>:SYMB for assignments.

> If the files do not exist, the INCLUDE program will create the output files; otherwise the old files will be modified.

THE OUTPUT FILE

The output file will contain the following information:

Declarations of VARIABLES for each SIBAS item and for each picture field used in the particular subfunction. The type of the variables will be the type of the connected data descriptions.

Declarations of VALUE BUFFERS, one for each realm and one for each picture record used.

EQUIVALENCE between value buffers and the item and field variables.

ITEM-LIST and picture FIELD-NAME-LIST, one for each value buffer. The lists include length and type information for all items/fields, and some additional information about the total list. The length is the number of words, and the type is one of the following.

ITEM TYPES:

type	Description
S	Single Integer without any editing.
D	Double Integer without any editing.
1n	Equal to S but with n digits after decimal point (n = 1-5).
2n	Equal to D but with n digits after decimal point (n =1-10).
O	Character, odd number.
E	Character, even number.

This format is derived from the FOCUS field description for the corresponding data description, both for SIBAS items and picture fields.

VARIABLES, VALUEBUFFERS (low and high limit) and INDEX information for each item or group defined as a key in subschema. It is defined in a similar way as for items.

Necessary definitions for a READY REALM call, if the subfunction is marked with READY REALM.

Declaration of SUBITEM LIST. The subitem list is used by the programmer for calls to the ABM SIBAS and FOCUS libraries.

All this information about the internal formats of items etc. is produced automatically on the basis of information in the ABM catalog.

The programmer need only know the logical layout of records and item/field names so he/she can spend more time on the application proper rather than on "bit manipulation".

To become familiar with the output from COPY and INCLUDE, study the examples of COPY/ INCLUDE files (see both chapter 9 and the example floppy disk.)

EXPLANATION OF THE GENERATED STATEMENTS

Rules for the generated statements are shown on the next page. Names in parentheses are only used to ease the assignments in the INCLUDE file and should not be used by the programmer. Array names starting with "C" are character arrays; all other arrays are integer arrays.

SIBAS PART:

- Variable name	(items)	xxaaaaa	
- Value buffer	(items)	KRECxx	
- Item, (size&type) list	(items)	KITEMxx	, (CITEMxx)
- Low limit variable name	(indexes)	Lxxbbbn	
- High limit variable name	(indexes)	Hxxbbbn	
- Value buffer (low&high)	(indexes)	KVxxbbb	
- Index name information	(indexes)	KIxxbbb	, (Cxxbbb)
- Realm name list	(realms)	KREALMS	, (CREALMS)
- Usage mode list	(realms)	KUMOD	
- Protection mode list	(realms)	KPMOD	

PICTURE PART:

- Variable name	(field)	yyccccc	
- Value buffer	(field)	MRECyy	
- Field name (size&type) list	(field)	MITEMyy	, (CITEMyy)
- Reference table	(FOCUS param.)	REFTAB	

BOTH:

- Subitem list

as used in call to the ABM SIB/FC lib.	ITEMSUB
as used by programmer in assignments	CITMSUB

ITEMSUB equivalenced with CITMSUB

WHERE

xx : Realm prefix (first 2 characters in realm name).

yy : Picture record prefix (first 2 characters in record name).

aaaaa: Unique item name within the realm. Characters 3 to 7 of item name.

bbb : Unique index name within the indexes in the realm. Defined in same manner as "aaaaa", but with only the first 3 characters after realm prefix.

n : Sequence number, starting with 1, for all the member items in a group index. For an item index it is 1.

ccccc: Unique field name within the picture record. If the field is defined with reference to a SIBAS item, "ccccc" will be equal to "aaaaa" from the corresponding item. If the field is defined with reference to a data description, the first 5 characters in the data description name will be used. Used by the subroutine DDTRNSF to move values from the SIBAS value buffer (KRECxx) to the FOCUS value buffer (MRECyy) or vice versa.

See the section "Naming Conventions..." on page 20.

2.17.2 GENERATING COBOL COPY ELEMENTS

The generation of COPY elements goes mainly in the same manner as the INCLUDE-GENERATE procedure. The module is started by the command COPY-GEN. The generated COBOL copy elements are stored on two SINTRAN files:

DECDDC-<subfunction name>:SYMB	**for declarations,**
ASSDDC-<subfunction name>:SYMB	**for assignments.**

The contents of the output files are logically the same as for the INCLUDE files, but they are of course in COBOL syntax. All generated variables will have a prefix according to the table below :

COBOL GENERATED VARIABLES:

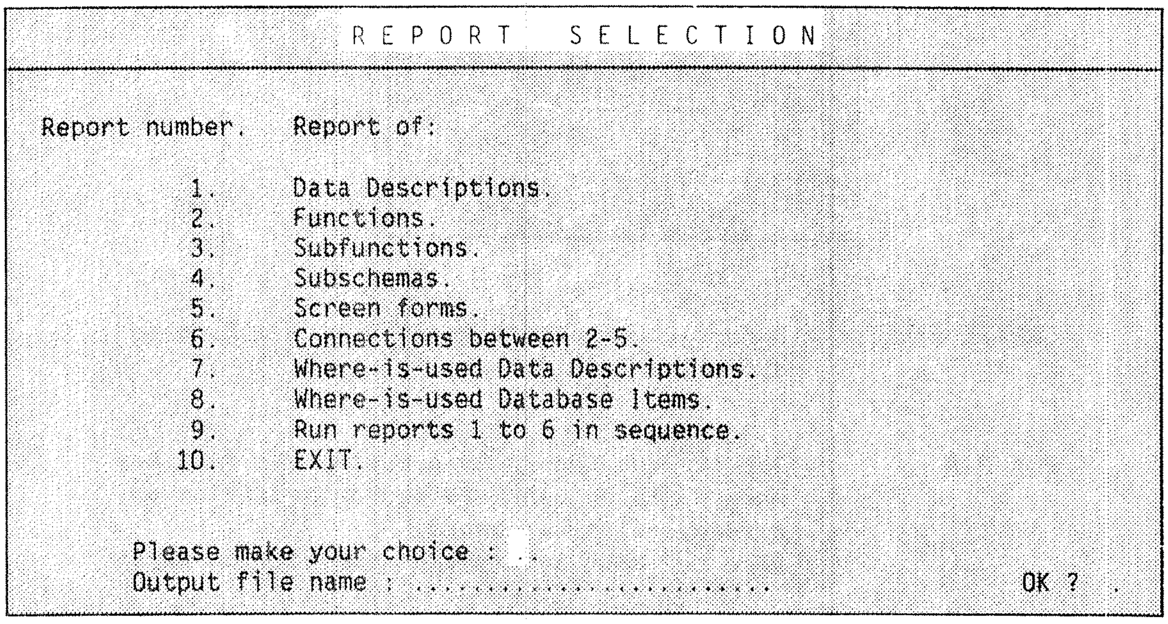
Prefix	Description
DDS-	ABM Screen variable not to be changed by user code
DDB-	ABM DataBase variable not to be changed by user code
SCV-	SCreen Value
SCC-	SCreen Command item
SCR-	SCreen Return item
DBV-	DataBase Value
DBR-	DataBase Realm
DDC-	ABM Common for Screen and Database
DBKI-	DataBase Key Item
DBKV-	DataBase Key Value

A COBOL word can consist of up to 30 significant characters (only 7 in FORTRAN) and this makes life much easier for those using ABM/COBOL rather than ABM/FORTRAN. (See the section "Name Conventions for FORTRAN and COBOL Programs".)

2.18 REPORT GENERATION

You can generate reports about the ABM catalog by using the command REPORT from the main menu. An ABM report will typically include information about all data descriptions, where they are used, and which ones are not used.

The following picture appears with the command:



```
                        R E P O R T   S E L E C T I O N

 Report number.   Report of:

          1.      Data Descriptions.
          2.      Functions.
          3.      Subfunctions.
          4.      Subschemas.
          5.      Screen forms.
          6.      Connections between 2-5.
          7.      Where-is-used Data Descriptions.
          8.      Where-is-used Database Items.
          9.      Run reports 1 to 6 in sequence.
         10.      EXIT.

       Please make your choice : ..
       Output file name : ........................             OK ? .
```

PLEASE MAKE YOUR CHOICE :Give the number of the report you wish to make.

> NOTE:
> Some reports (e.g., the Where-is-Used report) may generate a lot of output. It may also take a lot of time to generate the report.
> While a report is being generated, the status of the run will be displayed on the screen.

OUTPUT FILE NAME :Here you can give the name of an existing SINTRAN file. If the file does not exist, you can give the name of a file between quotes ("   "). The output will then automatically be written to this file.

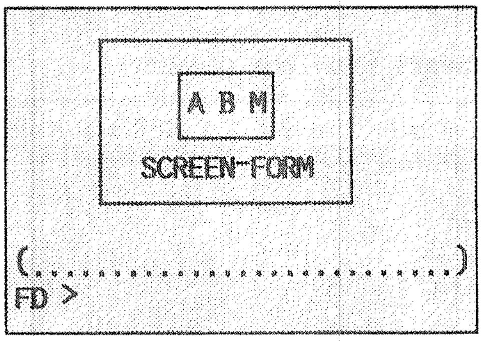
When you have selected a report number and given an output file name, SELECTION CRITERIA is displayed on the screen.

The output file will be closed when you exit the report menu.

SELECTION CRITERIA: You may choose to generate reports from selected databases, of selected Data Descriptions and so on.

HOW TO MAKE APPLICATION FORMS USING THE ABM SCREEN-FORM COMMAND.

Give command:
SCREEN-FORM↵



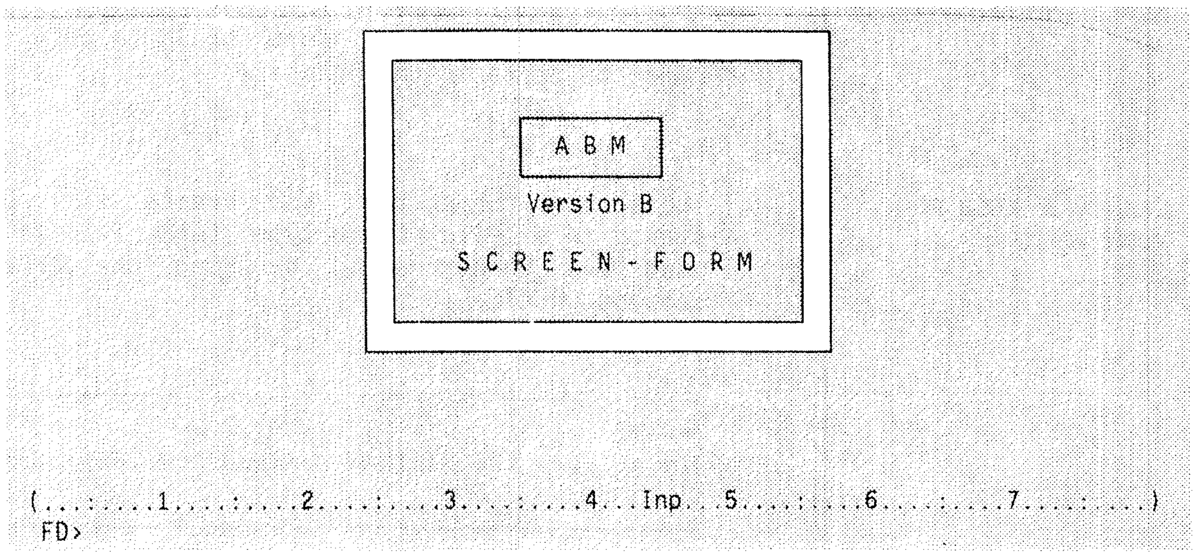
CREATE-FILE	● Before you create a form, create a file to store the form.
P A R A	● Set current record by pressing the PARA key when defining a new record or modifying an already existing record. (See the HELP information for editing keys.)
F I E L D	● Use the FIELD key to define/modify fields. (See the HELP information for editing keys.)
W R I T E	● Write your form to the file you have created.
E X I T	● EXIT when you have finished.

Remember, you may use graphics and alternate character sets and special symbols for your forms. You may also use various attributes like inverse video, high and low intensity and blinking mode. For details, use the HELP key or refer to the FOCUS Reference Manual, pages 33-34, (ND-60.137.5).

Press HELP for command information.

2.19 SCREEN-FORM

A special version of the FOCUS screen handling system is used for the online building and maintenance of applications screen forms. It is common to make a form corresponding to a database subschema, and for the application program to use this form to access the subschema. You can start this module with the command SCREEN-FORM. The following screen picture is displayed:



```
A B M
Version B
S C R E E N - F O R M

(...:....1....:....2....:....3....:....4...Inp..5....:....6....:....7....:....)
FD>
```

The following screen form commands are available.

Command	**Explanation**
<home>	Enter form definition/modification
CLEAR-DEFINITION	Clear current form definition
CREATE-FILE	Make a new form file (:FABM file)
DELETE-FORM	Delete a form from form file/ABM catalog
DESCRIBE-FORMS	Give information about a form
ENLARGE-FILE	Enlarge a form file
ENVIRONMENT	Inspect/modify environment
EXIT	Exit from SCREEN FORM
FUNC+@	Redisplay/refresh screen
GET-ENVIRONMENT	Get/modify environment specification
GET-KEY-VALUE	Get the numeric value for the key
HELP	Give help information
LIST-FILES	List files (:FABM files for this user)
LIST-FORMS	List forms in form file/ABM catalog
MAKE-UPTODATE-FORMS	Update forms after environment change
READ-FORM	Read a form from form file/ABM catalog
SAVE-ENVIRONMENT	Save environment specification
WRITE-FORM	Store form in form file/ABM catalog
@xxx	Execute the SINTRAN command xxx

For details about the commands use the HELP key
or
Refer to the FOCUS Reference Manual ND-60.137

You can give screen-form commands, together with the parameters, directly from the FD> command line.
Default values are obtained by omitting the parameter, indicated by a comma.

2.19.1 BEFORE YOU MAKE A SCREEN FORM...

The screen form that you create will normally be used by an application program to communicate with a database. So before you make your screen form ...

MAKE YOUR APPLICATION DATABASE

Make sure that you have defined your application database in the ABM catalog. The fields in your screen form should be able to refer to items in the database.

CREATE A FILE FOR YOUR FORMS

The screen forms that you create will be stored in a file of the type :FABM. This file must have been created by using the screen command CREATE-FILE.

NOTE:
Do not use the SINTRAN command CREATE-FILE!!

2.19.2 HOW TO MAKE A SCREEN FORM: AN EXAMPLE

Suppose you want to make the EMPLOY form shown below.

```
         Employee  List
 ----------------------------------
   Name: ..................
   Address: ...............
            ...............
   Function-1: ............
   Function-2: ............

                      OK.: .
```

- Create your application database with the appropriate realms and items.

  For example, you could create database EMPLIST with realm EMPLOY, and items EMNAME, EMADR1, EMADR2, EMFUNC1, and EMFUNC2.

- Define a Data Description for the OK field.

  The form will then contain 2 records. The first, the R1 record, will consist of 5 fields all referring to items in the EMPLOY realm. The second, the R2 record, will consist of the OK field which refers directly to the Data Description OK-FIELD.

- Start the screen form module with the command SCREEN-FORM↵.

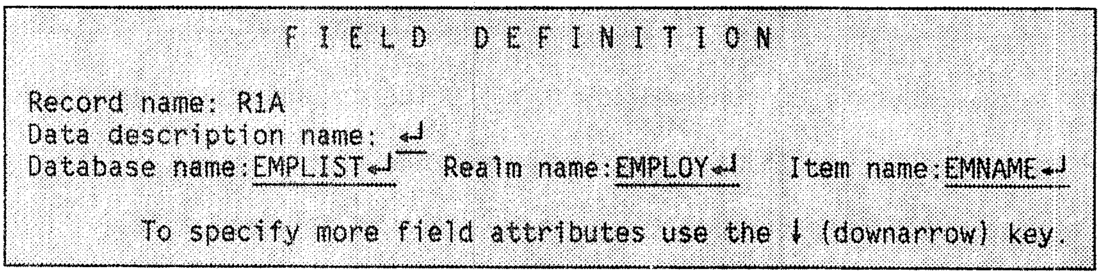
- Use the CREATE-FORM command to create a form file.

- Press the HOME key (↖) to move into the screen picture.

- Type in the form text and graphics.

- Press the PARA key with the cursor in the first position of the name field and fill in the field definition.

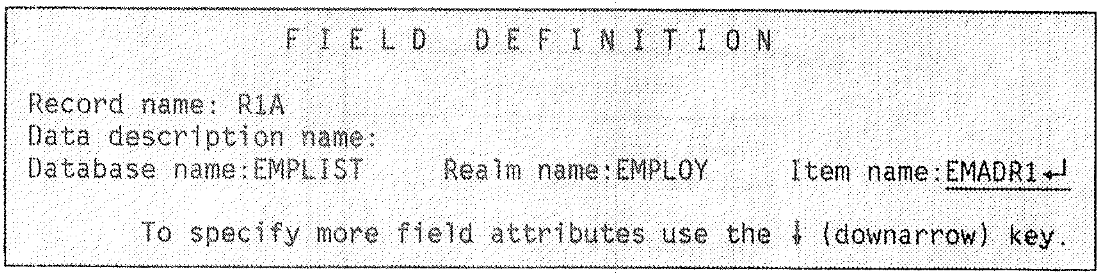
NOTE: In the following examples, the underlined values are filled in by the user; the others are system default values.



```
                 F I E L D   D E F I N I T I O N

Record name: R1A
Data description name: ↵
Database name:EMPLIST↵  Realm name:EMPLOY↵   Item name:EMNAME↵

    To specify more field attributes use the ↓ (downarrow) key.
```

(The additional field attributes are skipped because item name is terminated by ↵.)

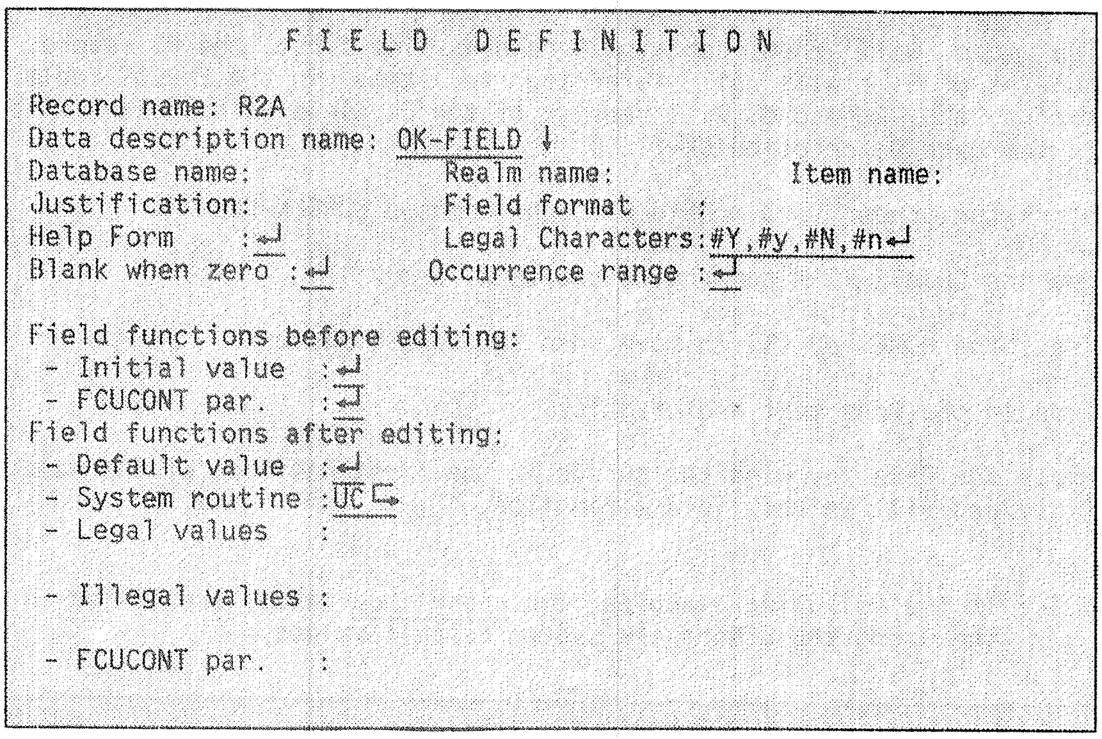
- Press the FIELD key with the cursor in the first position of the first address field and fill in the field definition.



```
                      F I E L D   D E F I N I T I O N

Record name: R1A
Data description name:
Database name:EMPLIST      Realm name:EMPLOY      Item name:EMADR1↵

      To specify more field attributes use the ↓ (downarrow) key.
```

(the default values are those of the previously defined field)

- Use the FIELD key to define the second address field and the function fields in the same way that the first address field was defined.
- Press the PARA key with the cursor in the OK field and fill in the field definition.



```
                      F I E L D   D E F I N I T I O N

Record name: R2A
Data description name: OK-FIELD ↓
Database name:             Realm name:            Item name:
Justification:             Field format    :
Help Form      :↵          Legal Characters:#Y,#y,#N,#n↵
Blank when zero :↵        Occurrence range :↵

Field functions before editing:
 - Initial value   :↵
 - FCUCONT par.    :↵
Field functions after editing:
 - Default value   :↵
 - System routine :UC ↳
 - Legal values    :

 - Illegal values :

 - FCUCONT par.    :
```

(The system routine UC converts the field value to uppercase letters.)

- Press the HOME key (↖) to move to the command line.
- Use the WRITE-FORM command to write the form to the form file.

2.19.3 DESCRIBE-FORMS: TO EXAMINE FORMS

The DESCRIBE-FORMS command can be used to examine the contents of a previously defined form. The output will contain information about form layout, field positions, field names, field lengths and types of field references.

How **field names** are built up in COBOL and FORTRAN:

COBOL : Field name is equal to item name. If there is no item, the 8 first characters in the Data Description name will be used as field name.

FORTRAN : The two first characters in record name are replaced by the two first characters in item name to create a field name.

If there is no item, the two first characters in record name + the 6 first characters in Data Description name will form the field name.

An example of output from the DESCRIBE-FORM command for the example EMPLOY form is given in the following :

```
----------------------------------------------------------------------
                          F O R M
  Form name   : EMPLOY                           Size    : 580

  Form start  : line 2   column 24
  Form end    : line 11  column 55               Written : 1986-03-20

         (ABM application programming language: FORTRAN)
----------------------------------------------------------------------


                    F O R M   L A Y O U T
                  +-----------------------------+
                  |      Employee  List         |
                  |  Name: ..................   |
                  |  Address: ...............   |
                  |           ...............   |
                  |  Function-1: ............   |
                  |  Function-2: ............   |
                  |                   OK.: .    |
                  +-----------------------------+
```

F I E L D   D E S C R I P T I O N S

Field name	Format	LD	LS	HELP	ST/JU	BWZ	Occ
R1NAME	X(18)	18	18		None	No	L
R1ADR1	X(15)	15	15		None	No	L
R1ADR2	X(15)	15	15		None	No	L
R1FUNC1	X(12)	12	12		None	No	L
R1FUNC2	X(12)	12	12		None	No	L
R2OK-FIE	X	1	1		None	No	L

- Legal char. : #Y,#y,#N,#n
- System routine : UC

F I E L D   O C C U R R E N C E S

Record	Field	Occ	Line	Col.	Data descr. or DB-name/Realm/Item
R1A	R1NAME	1	5	32	EMPLIST /EMPLOY /EMNAME
R1A	R1ADR1	1	6	35	EMPLIST /EMPLOY /EMADR1
R1A	R1ADR2	1	7	35	EMPLIST /EMPLOY /EMADR2
R1A	R1FUNC1	1	8	38	EMPLIST /EMPLOY /EMFUNC1
R1A	R1FUNC2	1	9	38	EMPLIST /EMPLOY /EMFUNC2
R2A	R2OKFIEL	1	10	48	OKFIELD

2.19.4 RULES FOR FIELDS AND RECORDS

GENERAL

- A field is always part of a record. A record consists of one or more fields.
- The record name can be up to 8 characters long. The first two characters must be unique within the form (for example an R1 record, R2 record and so on).
- The storage and display format for a field is fetched from the connected Data Description (DD). A field refers to a DD either directly (by reference to a DD name) or indirectly (via database, realm and item name).
- All fields in a record must have the same type of reference to DD names (either directly or indirectly) - not mixed.
- If a record has indirect references to DD names, they must all refer to items in the same database realm.

DEFINING RECORDS AND FIELDS

The main difference between standard FOCUS and this special version is the way in which fields, records and record occurrences are defined.

KEYS

- Press the PARA key with the cursor in a blank position to define the first field in a new record. The new record becomes the current record.

PARA

- Press the PARA key with the cursor in an existing record to modify the record. The record becomes current record.

  You may not use the PARA key in an existing record if you have copies of that record.

FIELD

- Press the FIELD key with the cursor in a blank position to define a new field in the current record.
- Press the FIELD key with the cursor in an existing field to modify the field. The record containing the field becomes the current record.

MARKING FIELDS AND RECORDS

- Press the SHIFT + PARA keys to mark a record if you want to copy, delete, or move it. All the affected fields will be shown in inverse video.

- Press the SHIFT + FIELD keys to mark a field if you want to copy, delete, or move it. The marked field will be shown in inverse video.

You may not use the SHIFT + FIELD keys to mark a field if you have copies of the record containing the field.

FIELD DEFINITION

When you press the PARA or FIELD key, you will be be prompted with the Field Definition form. Some of the fields in this form are explained below:

Record Name.
Max. 8 chars. A default record name will be assigned. If you have pressed the PARA key to display this picture, then you may change the default record name. (You are not allowed to change the default record name after having pressed the FIELD key.)

Data Description or Database Name.
If a Data Description name is given, the Database fields will be skipped. Otherwise, you must enter the Database name, Realm name, and Item name to which the form field refers.

To give additional attributes to fields, press the ↓ key in the Item name or Data Description field.

MULTI RECORD EXAMPLE

Let's say that you have defined two records, R4 and R5. The cursor is positioned in R5, and R5 is the current record.

If you want to define a new field in R4, you must first make R4 the current record by moving the cursor to any field within R4, pressing either the FIELD key or the PARA key, and then pressing the CANCEL key to get out of the Field Definition prompt. You may then define the new record by pressing the FIELD key with the cursor in a blank position.

COPYING, MOVING AND DELETING MARKED FIELDS AND RECORDS

To copy, move or delete a field or a record, you must first mark it.

DELETE

- Delete marked field or record.

COPY

- Copy marked field or record.
  Move the cursor to the new start position for the field or record, and use the COPY key.

MOVE

- Move marked field or record.
  Move the cursor to the new start position for the field or record, and use the MOVE key.

RESTRICTIONS

- You may not change a record if you have copies of that record.
- You may not copy a record if it contains copies of a field.
- You may have a maximum of 35 different records in a form.
- You may have a maximum of 35 copies of a record in a form.
- You may have a maximum of 256 fields in a form.

2.20 FIELD OCCURRENCES IN ABM VERSION B

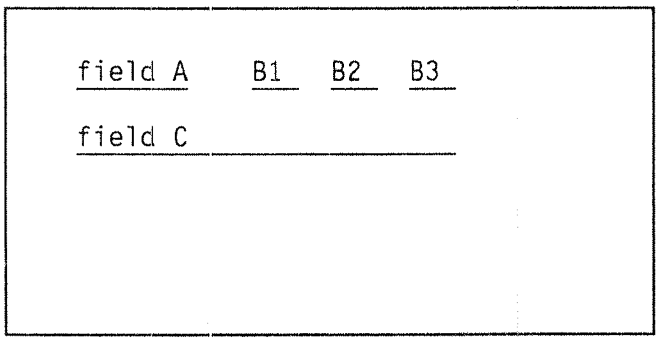
It is possible to define field occurences in screen forms. "Mark" a field by pressing the SHIFT + FIELD keys (the field will be displayed in inverse video), move the cursor to a free position and press the COPY key. Continue to move the cursor and press the COPY key if you want more copies.

The rules for field occurrences are:

- Records containing field occurrences can not be copied. That is, a field can not be both a field occurrence and a record occurrence.
- Occurrences of a field must be in sequence within the record. (The sequence is from the top down and from left to right). Otherwise a WARNING will be displayed when you store the form by the WRITE-FORM command.
- The field occurrences will make the generated code in COPY/INCLUDE files more "compact". The "occurs n times" clause is used in COBOL while an array holding all the field occurrence values is generated in FORTRAN.
- When using a record containing field occurrences in the routines in ABM-FC-LIB, you must note the following :

  When reading a whole record, split this read into separate DDRFLDS. Field occurrences must be treated separately in one read (see the following example).

An example of field occurrences in a COBOL environment:



This form contains one record, lets call it R1A. The form consists of five fields; A, B1, B2, B3 and C. The fields B1, B2 and B3 are field occurrences.

The field references (the way they refere to database items and Data Descriptions etc.) are shown in the following table:

named	record	database	realm	item	FOCUS name
A	R1A	ABMDEMO	UNIT	CUSTNO	CUSTNO
B1	R1A	ABMDEMO	UNIT	DATE	DATE.01
B2	R1A	ABMDEMO	UNIT	DATE	DATE.02
B3	R1A	ABMDEMO	UNIT	DATE	DATE.03
C	R1A	ABMDEMO	UNIT	COMMENTS	COMMENTS

A COBOL program reading the whole record will look like this:

```
READ-R1-RECORD.
    MOVE '+:CUSTNO  *'     TO DDC-SELECT.
    PERFORM READ-R1-FIELDS.
    MOVE 3                 TO SCC-RW-NO-OF-LINES.
    MOVE '+:DATE    *'     TO DDC-SELECT.
    PERFORM READ-R1-FIELDS.
    MOVE 1                 TO SCC-RW-NO-OF-LINES.
    MOVE '+:COMMENTS*'     TO DDC-SELECT.
    PERFORM READ-R1-FIELDS.

READ-R1-FIELDS.
    CALL 'DDRFLDS' USING      DDC-REF-TABLE,
                              DDS-R1-SUBSCHEMA,
                              SCV-R1,
                              FCSTATUS.
    IF FCSTATUS NOT = 0 PEFORM DD-FC-ERROR.
```

The COPY generated declarations for the R1A record value buffer would be like this:

```
03  SCV-R1.
    05  SCV-R1-CUSTNO         PIC  9(4) COMP.
    05  SCV-R1-DATE           PIC  X(8)     OCCURS 3.
    05  SCV-R1-COMMENTS       PIC  X(60).
```

2.21 DATABASE AND FORM CONNECTIONS

The parallelism between a database subschema and a form is shown below:

database subschema	:	form
realm	:	record
item	:	field

- a realm in a database subschema and a record in in the form.
- an item in a realm and a field in a record.

USE THE PARALLELISM BETWEEN A DATABASE SUBSCHEMA AND A FORM

A subschema often reflects the use of a database through a screen form. A "typical" form contains many fields organized into 2 or 3 records. Some records have fields referring directly to Data Descriptions (example: time and date). The other records refer to items in database realms.

READ A RECORD FROM A FORM AND STORE IT IN THE DATABASE

Because of the parallelism between a database subschema and a form, it is easy to read a record from the screen and store it in the database.
You could do this as follows:

```
CALL DDRFLDS   <form parameters>
CALL DDTRNSF   <from form parameters to
                database parameters>
CALL DDSTORE   <database parameters>
```

ALL PARAMETERS ARE PRODUCED IN INCLUDE/COPY FILES

This is especially easy because all the required parameters are automatically produced by INCLUDE/COPY modules.

2.22 DATABASE MAINTENANCE

You will probably never be completely finished with the development of a database system. Changes and modifications have to be made from time to time. After a change/modification is done, the affected parts of the database have to be updated to make the change available for them. The way this updating is done is dependent on whether the database has been Confirmed or not.

BEFORE CONFIRMATION

Develope a new system by programming, testing and making constatnt changes. You must know where to update after a change/modification, and you must update all the affected parts of the database.

Before the database has been confirmed, initiate the database each time you have made changes to the schema. All data in the database will be lost.

When the start of production is approaching, run a Confirmation of the database.

AFTER CONFIRMATION

After the Confirmation, you may still need to modify the database, but at this point ABM helps you with maintainance. Follow the methods described here, and ABM will update all the affected parts of the database.

The following procedures are for making changes to a database which has not been Confirmed.

CHANGING THE DATA DESCRIPTION

- When changing a data description, you must know whether the data description is used in form(s) or not. All the forms where the data description is used must be made up-to-date to make the modified data description available for them.

  Update the forms by either
  - the "make-uptodate-form" command in screen-form for each form,
  - the "make-uptodate-form" command for all forms in the form-file(s),
  - the command "read-form", modify the form and "write-form".

  When enlarging a field, be sure that the enlarged field does not overwrite an already existing field.

(The report "Where-is-used data description" will help you to find the forms using the particular data description.)

- Make new copy/include files.
- Recompile and reload your application programs.

MODIFYING A FORM

- If you modify a form (new fields, deletion of fields), remember to Modify the subschema of the form which has been updated.
- Make new copy/include files.
- Recompile and reload your application programs.

DELETING A DATA DESCRIPTION

- Data description may not be deleted if it is already in use. All the referenced items in screen forms and DRL schemas must be deleted first. Run a report to find out where the data description is used. (The data description may have been used in a group.)
- Make new copy/include files.
- Recompile and reload your application programs.
- If the data description is used in a form(s): READ-FORM, make the changes and WRITE-FORM.
- Modify the subschema of the form which has which has been updated.

The following examples show you what to do when making changes to an already confirmed database.

CHANGING THE DATA DESCRIPTION

- Make the changes in the particular data description.
- "Make-uptodate-form" if the data description is used in a form(s).
- Run a Redefinition in the schema module (menu) to generate redefinition statements for DRL.
- Run DRL.

- Make new copy/include elements by COPY-GEN or INCLUDE-GEN. Use the command CHANGES to let ABM know which files will be affected.
- Recompile and reload your application programs.

  Now the change in the data description is recognized everywhere.

UPDATE AN OLD ITEM

- Make the changes to the item. The "cre/del/upd" field will be marked "U".
- Run a Redefinition in the schema module (menu) to generate redefinition statements for DRL.
- Run SIB-DRL.
- Make new copy/include elements.
- Recompile and reload your application programs.
- Try out the modification(s).
- When everything is correct, run a new Confirmation.

INSERT A NEW ITEM

- Make the new data description if necessary.
- Create the new item with the command DBITEM.
- Run a Redefinition in the schema module (menu) to generate redefinition statements for DRL.
- Run SIB-DRL.
- Make new copy/include elements.
- Recompile and reload your application programs.
- Try out the modification(s).
- When everything is correct, run a new Confirmation.

Do all modifications within ABM, not directly in generated SCHEMAS.

Take backup of ABM catalog files, application programs and forms.

2.23 GENERATING SUBSCHEMA FROM FORM

A new subschema can be generated from an existing form. If you answer "Y" to the question "generate subschema from form?" in the Subschema Heading picture, a subschema will be generated.

Some forms, however, will not lead to a meaningful subschema. Generating subschema from a form containing only leading text and no fields, for example, will result in an "empty" subschema. Nor will records refering directly to Data Description form the basis of a subschema.

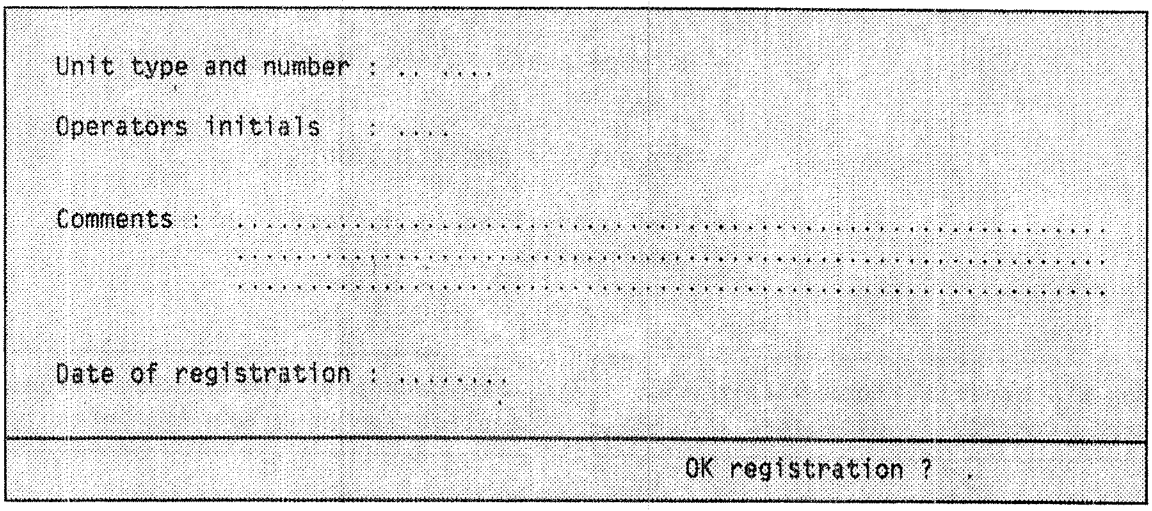
Each form is analysed before the generation of subschema starts. If the form is found "meaningful", a subschema is established; if not, a message is given on the screen and the generation is stopped.

The subschema will be generated in this way:

- The new subschema will get the same name as the form it was generated from.
- All records containing fields which have indirect referance to Data Description (i.e. those referring to a database item) will form the basis of a subschema.
- All referred realms will be marked with "UN". (Update/Non protect)
- All referred items which are not keys in the schema will be marked with an "I"; keys will be marked with "IK".

NOTE:
No group item will be marked. Marking of group items must be done by the user after the generation of subschema.

Below is an example of generating subschema based on the picture named MENU-1C:



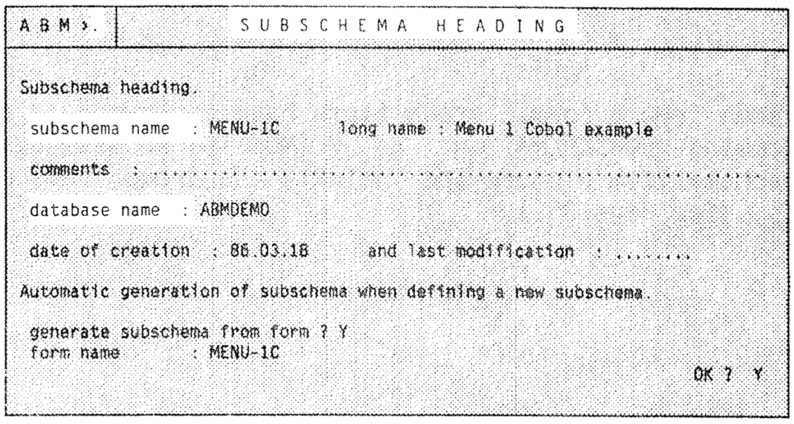
```
Unit type and number : .. ....

Operators initials   : ....


Comments : ...........................................................
           ...........................................................
           ...........................................................


Date of registration : ........

                                          OK registration ? .
```

Below is a specification of the fields found in the MENU-1C picture:

FIELDS	DATABASE	REALM	ITEM	DATA DESCRIPTION
UNIT TYPE	ABMDEMO	UNIT	UNTYPE	
UNIT NUMBER	ABMDEMO	UNIT	UNNUMBER	
OPERATOR INT.	ABMDEMO	UNIT	UNREGBY	
COMMENTS (1)	ABMDEMO	UNIT	UNCOM1	
COMMENTS (2)	ABMDEMO	UNIT	UNCOM2	
COMMENTS (3)	ABMDEMO	UNIT	UNCOM3	
DATE OF REGISTR.	ABMDEMO	UNIT	UNREGDAT	
OK REGISTRATION				OK1

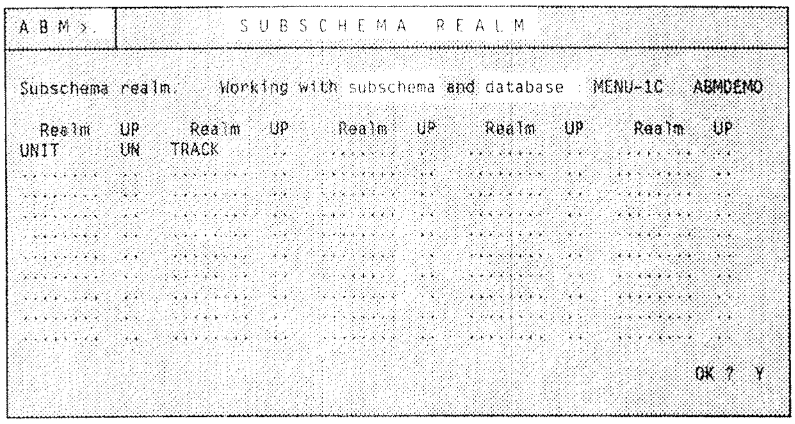
The result of the generation is a subschema named MENU-1C.

Generation of the subschema MENU-1C, can be illustrated by the following pictures: (This is the same procedure you would follow to generate a a subschema without using automatic generation.)



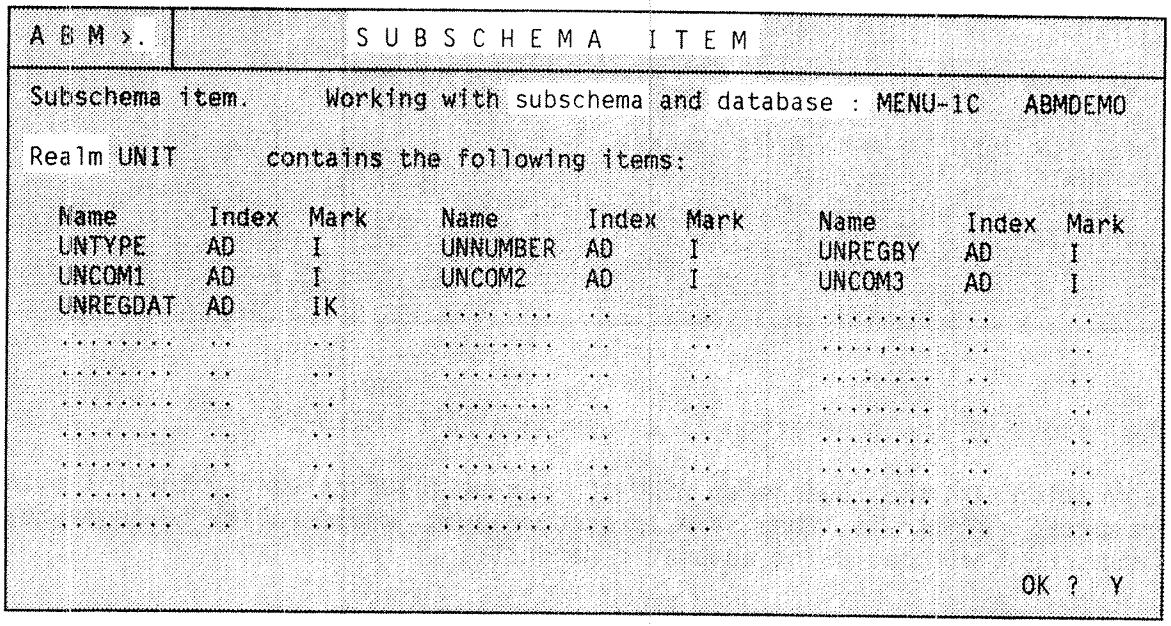
```
A B M >.   S U B S C H E M A   H E A D I N G

Subschema heading.

 subschema name  : MENU-1C      long name : Menu 1 Cobol example

 comments  : ..............................................................

 database name  : ABMDEMO

 date of creation  : 86.03.18      and last modification  : ........

Automatic generation of subschema when defining a new subschema.

 generate subschema from form ? Y
 form name        : MENU-1C
                                                              OK ?  Y
```

The picture MENU-1C refers only to the realm named UNIT; so the Subschema Realm picture establishes just one referance to this realm.



```
A B M >.   S U B S C H E M A   R E A L M

Subschema realm.   Working with subschema and database : MENU-1C   ABMDEMO

  Realm    UP     Realm    UP     Realm    UP     Realm    UP     Realm    UP
UNIT       UN   TRACK      ..   ........   ..   ........   ..   ........   ..
........   ..   ........   ..   ........   ..   ........   ..   ........   ..
........   ..   ........   ..   ........   ..   ........   ..   ........   ..
........   ..   ........   ..   ........   ..   ........   ..   ........   ..
........   ..   ........   ..   ........   ..   ........   ..   ........   ..
........   ..   ........   ..   ........   ..   ........   ..   ........   ..
........   ..   ........   ..   ........   ..   ........   ..   ........   ..
........   ..   ........   ..   ........   ..   ........   ..   ........   ..
........   ..   ........   ..   ........   ..   ........   ..   ........   ..
........   ..   ........   ..   ........   ..   ........   ..   ........   ..

                                                                   OK ?  Y
```

All items refered to by the realm UNIT are listed in the Subschema Item picture:

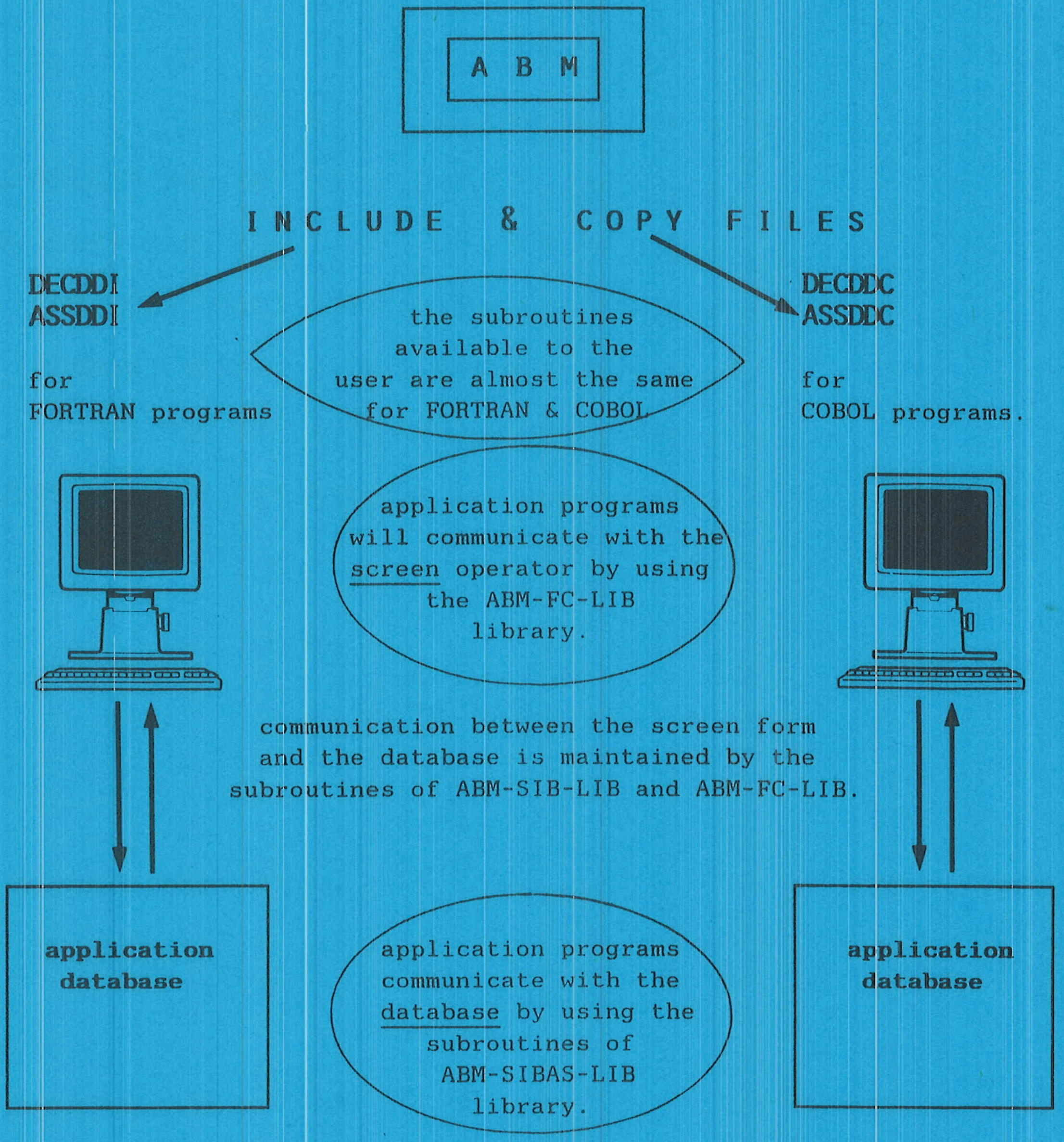


```
A B M >.            S U B S C H E M A   I T E M

Subschema item.    Working with subschema and database : MENU-1C   ABMDEMO

Realm UNIT      contains the following items:

  Name      Index  Mark     Name      Index  Mark     Name      Index  Mark
  UNTYPE    AD     I        UNNUMBER  AD     I        UNREGBY   AD     I
  UNCOM1    AD     I        UNCOM2    AD     I        UNCOM3    AD     I
  UNREGDAT  AD     IK       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..

                                                               OK ?  Y
```

After generating a subschema from a form, group items have to be marked by the user. Give the command SUBSCHEMA and do M (Modify) on subschema MENU-1C. Navigate to Subschema Item picture and indicate the group items by marking the "MARK" fields. (See page 35.)

CHAPTER 3
HOW TO WRITE PROGRAMS USING ABM

HOW TO WRITE FORTRAN & COBOL APPLICATION PROGRAMS



the subroutines available to the user are almost the same for FORTRAN & COBOL

application programs will communicate with the screen operator by using the ABM-FC-LIB library.

communication between the screen form and the database is maintained by the subroutines of ABM-SIB-LIB and ABM-FC-LIB.

application programs communicate with the database by using the subroutines of ABM-SIBAS-LIB library.

1. The application program will read data from the screen-form by using the routine DDRFLDS.

2. The data read from the screen is stored in the screen buffer.

3. From the screen buffer the data is tranferred to the database buffer by using the routine DDTRNSF.

4. From the database buffer the data is transferred to the database by using the routine DDSTORE.

3 HOW TO WRITE PROGRAMS USING ABM

3.1 WRITING PROGRAMS USING ABM

The INCLUDE and the COPY files, together with the subroutine package for communication with SIBAS/FOCUS, can be used to write bug-free programs quickly and efficiently.

ADVANTAGES OF USING INCLUDE/COPY FILES

The main advantages of using the INCLUDE and COPY files are:

- the values on items/fields are automatically available in the generated variables;
- only variable names without indexes are transferred in the SIBAS/FOCUS calls. The start position or number-of-words is not transferred;
- no computing of field numbers, position and value buffers, or item lengths is necessary.

WRITING GOOD PROGRAMS

For detailed information on writing good programs, refer to The Elements of Programming Style by B.W.Kernighan & P.J. Plauger. Below is a brief summary.

WRITING GOOD PROGRAMS:

- Say what you mean, simply and directly.
- Write clearly, not efficiently.
- Let the program do the dirty work.
- Choose easy-to-remember variable names.
- Document your program properly.
- Use proper indentation.
- Use library routines.
- Avoid temporary variables.
- Write & test a big program in small pieces
- Do not patch bad code; rewrite it.

3.2 GENERAL STRUCTURE FOR FORTRAN PROGRAMS

The following should be considered when making application programs in FORTRAN:

THE INCLUDE FILES

You can use the INCLUDE files in your application programs by the following FORTRAN statement:

```
...
$INCLUDE DECDDI-<subfunc name>

< local declarations (if necessary) >

$INCLUDE ASSDDI-<subfunc name>
< local assignments >
< program code.>
...
```

COMMUNICATION WITH THE APPLICATION DATABASE

The application programs will generally communicate with the application database by using the subroutines of the ABM-SIBAS-LIB library.

COMMUNICATION WITH THE SCREEN

Generally the application program communicates with the screen operator by using the subroutines of the ABM-FC-LIB library.

- Only read/write fields within the same logical record can be included in the same call, although one or more occurrences of that logical record may be influenced (see REFTAB parameter and DDGETRC/DDPUTRC routines).
- Standard picture initiation: one call performs all the necessary FOCUS calls.
- Data transfer between FOCUS and SIBAS buffers is done by using a call to a special routine (DDTRNSF) or by ordinary assignments.

NOTE:
When calling SIBAS/FOCUS communication routines, buffers transferred as parameters should always be transferred without a start index.

For example:
(..,KRECxx,..)
not (..,KRECxx(index),..).

3.3 USE OF SUBITEM LIST IN FORTRAN PROGRAMS

The INCLUDE files contain an item list, a field-name list and value buffers for each realm and picture record type. It is possible, by means of the subitem list, to subtract parts of an item list or field-name list.

THE SUBITEM LIST

The subitem list is declared and dimensioned in the INCLUDE file as a character array, with equivalence to an integer array. The programmer puts values into the character array. The equivalenced integer array is transferred, together with the total-item list, to the actual SIBAS/FOCUS communication routine.

In this way it is possible for one call to access items/fields in the item/field name lists.

THE RESULT-ITEM LIST

The SIBAS/FOCUS communication routines compose a result-item list from the subitem list, and the total-item/field-name list generated in the INCLUDE file. This result-item list is used in SIBAS/FOCUS calls from the corresponding SIBAS/FOCUS communication routines.

BUILDING SUBITEM LIST

```
CITMSUB(1) = 's:<item 1><item 2>...<item 3>*'
              |  |________________________| |
              |               |           end mark.
              |               |
              |           item name or
              |           field-name list.
              |        (8 chars. for each item)
 type of subitem list (+,0,-)
```

The three types of subitem lists are:

+ : Result-item list corresponds to subitem list.
0 : Result-item list corresponds to the total-item/field-name list.
- : Result-item list corresponds to those items/fields in the total-item list which are <u>not</u> in the subitem list.

NOTE:
In order to get the number of items per entry in the subitem list, check the dimension statement generated in the INCLUDE file. Maximum number of items using subitem list with the type + or - is: (number of items / 2) + 1.

3.4 GENERAL STRUCTURE FOR COBOL PROGRAMS

The programs are written as ordinary COBOL programs, with or without the ND COBOL extension using EXPORT/IMPORT (see the COBOL manual). In RT and TPS programming, no common areas are generated. Only local variables are used, and assignments are used instead of VALUE statements.

THE COPY FILES

The Copy files are included in the application program by means of the COBOL COPY statement:

```
DATA DIVISION.
  ...
WORKING-STORAGE SECTION.
  ...
01   <name>.
     COPY DECDDC-<subfunction name>.

     <additional declarations>
  ...
PROCEDURE DIVISION.
  ...
     COPY ASSDDC-<subfunction name>.
  ...
```

NOTE:
You have to define your own 01 level for the declaration part in the WORKING-STORAGE SECTION. This allows you to use the ND COBOL extension EXPORT/IMPORT on the whole area of a subfunction.

The program may be structured as usual. The only difference from "normal" programs is the way the program communicates with the screen and the application database, and the way data is transferred between database buffers and screen buffers.

3.5 USE OF SUBITEM LIST IN COBOL PROGRAMS

The item list, field-name list and value buffers are generated in the COPY files for each realm and each picture record type.

THE SUBITEM LIST

The subitem list, DDC-ITEM-LIST, is declared and dimensioned in DDC-SELECT in the COPY file. If pictures are used, DDC-SELECT is declared as a part of the DDC-REF-TABLE declaration. For database use, it is declared on the 03 level with the same name. In this way, one call may access one, some or all items/fields in the item/field name lists.

THE RESULT-ITEM LIST

The SIBAS/FOCUS communication routines compose a result-item list from the subitem list and the total-item/field-name list generated in the COPY file. This result-item list is used in SIBAS/FOCUS calls from the corresponding SIBAS/FOCUS communication routines.

```
03 DDC-SELECT.
   05 DDC-TYPE      PIC X(2).
   05 DDC-ITEM-LIST.
      07 DDC-ITEM   PIC X(8) OCCURS n.
```

There is only one SUBITEM LIST for each subfunction (COPY declaration)! The 'n' in OCCURS is computed and assigned the right value in the COPY generation module.

BUILDING SUBITEM LIST

```
MOVE 's:<item1><item2>...<item3>*' TO DDC-SELECT
      |  |_______________________||
      |             |              |
      |             |             end mark.
      |            item name or field-name list.
      |            (8 chars. for each item)
     type of subitem list (+,0,-)
```

The three types of subitem lists are:

+ : Result-item list corresponds to subitem list.

0 : Result-item list corresponds to the total-item/field-name list.

- : Result-item list corresponds to those items/fields in the total-item list which are <u>not</u> in the subitem list.

```
MOVE '+:ACCNUMB *'  TO DDC-SELECT.
CALL 'DDRFLDS' USING DDC-REF-TABLE,
                     DDS-R3-SUBSCHEMA,
                     SCV-R3,
                     FCSTATUS.
```

This reads only the R3 SUBSCHEMA field (ACCNUMB) from the screen . (The result is placed in the correct part of the SCV-R3 buffer. All the variables are generated and assigned during the COPY run.)

3.6 SIBAS/FOCUS COMMUNICATION ROUTINES

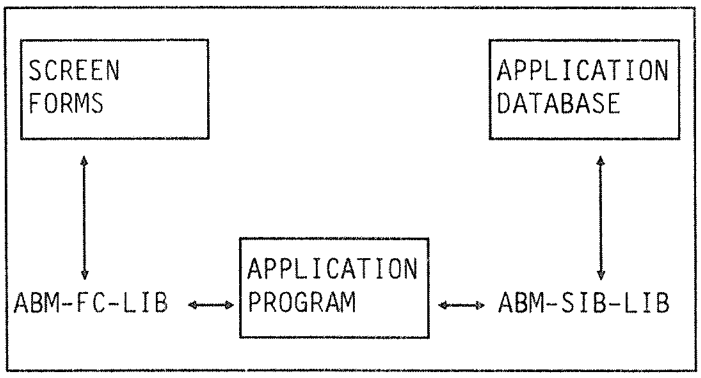
All communication routines are written in FORTRAN and placed in two library files: ABM-SIB-LIB-xm:SYMB and ABM-FC-LIB-xm:SYMB (x is the ABM version number and m is the revision number). A mode file, ABM-100-LIB-xm:MODE for ABM-100 and ABM-500-LIB-xm:MODE for ABM-500, compiles the two files as standard background libraries. (See the PD sheets)

THE COMMUNICATIONS ROUTINES

We will provide here a description of the communication routines. The name of the routines and the sequence of the parameters are exactly the same for both FORTRAN and COBOL users. The only difference is in the names of the parameters generated by INCLUDE/COPY. All such parameters are therefore described both in FORTRAN and COBOL syntax.

THE SIBAS/FOCUS COMMUNICATION

The communication between SIBAS, FOCUS and the application program is illustrated in the following figure:



Routines of the ABM-FC-LIB help communication between the application program and the screen forms.

Routines of the ABM-SIB-LIB help communication between the application program and the application database.

3.7 VALUE BUFFERS FOR ABM-FC-LIB AND FORMS

The ABM-FC-LIB routines work internally on a total screen buffer. All field and record values are read or written into this total screen buffer. Routines to get (DDGETRC) values from the total screen buffer into the local program value buffers and vice versa (DDPUTRC) are either called automatically or must be called explicitly.
Generally, the following holds:

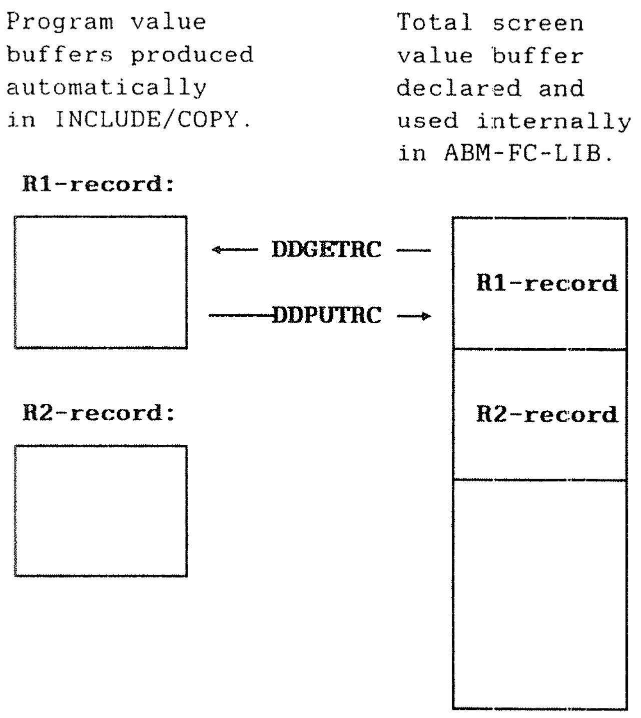
VALUE BUFFERS FOR ONE RECORD OCCURRENCE

If your form does not have more than one occurrence of any record, your automatically-generated value buffers (from INCLUDE/COPY) are exactly like the fields in the form. DDGETRC and DDPUTRC will be called automatically.

VALUE BUFFERS FOR SEVERAL RECORD OCCURRENCES

If your form has more than one occurrence of a record, then you must transfer the buffer values yourself. You could do this, for instance, by storing away the field values and by calling DDPUTRC/ DDGETRC successively).

The relationship between the program value buffers and the total screen value buffers is shown below:


Program value buffers produced automatically in INCLUDE/COPY.
Total screen value buffer declared and used internally in ABM-FC-LIB.
R1-record:
DDGETRC
DDPUTRC
R1-record
R2-record:
R2-record

CHAPTER 4
HOW TO USE ABM-SIB-LIB ROUTINES IN FORTRAN APPLICATIONS

4 HOW TO USE ABM-SIB-LIB ROUTINES IN FORTRAN APPLICATIONS

4.1 ROUTINES AND PARAMETERS IN ABM-SIBAS-LIBRARY

Description of routines in ABM-SIB-LIB

DDACCD	-	ACCUMULATION of item values.
DDFEBL	-	Find FIRST record between limits using given key.
DDFLBL	-	Find LAST record between limits using given key.
DDFORG	-	FORGET, nullify the effect of a REMEMBER call.
DDFREMB	-	FORGET old and REMEMBER a new record or a search region.
DDFTCGT	-	FIND a specific record and GET the record values.
DDFTCH	-	FIND a specific record.
DDGET	-	GET the relevant record, items or group items.
DDGETN	-	GET (read) a number of records in a search region.
DDGIXN	-	GET (read) a number of index keys.
DDINKEY	-	Reset search regions to maximum.
DDINSR	-	INSERT an index key of a record.
DDMDFY	-	MODIFY values of items or group items in a record.
DDREMO	-	REMOVE a manually maintained index key.
DDSTORE	-	STORE a (part) of a record in its realm.
DDTRNSF	-	TRANSFER of values between value buffers (for FORTRAN applications).

PARAMETERS OF THE ABM-SIB-LIB ROUTINES:

Library name: ABM-SIB-LIB-<version>.

Routines in this library		calls SIBAS routine
DDACCD	(TDBKEY,INTEG,ITEMSUB,KIxxxxx,KRECxx, DBSTATUS)	ACCID ACCDD
DDFEBL	(KIxxxxx,KVxxxxx,DBSTATUS)	SFEBL
DDFLBL	(KIxxxxx,KVxxxxx,DBSTATUS)	SFLBL
DDFORG	(TDBSRI,OPTION,DBSTATUS)	SFORG
DDFREMB	(TDBSRI,OPTION,DBSTATUS)	SFORG SREMB
DDFTCGT	(KIxxxxx,KVxxxxx,ITEMSUB,KITEMxx,KRECxx,DBSTATUS)	SFTGT
DDFTCH	(KIxxxxx,KVxxxxx,DBSTATUS)	SFTCH
DDGET	(TDBKEY,ITEMSUB,KITEMxx,KRECxx, DBSTATUS)	SGET
DDGETN	(TDBKEY,TDBSRI,N,ITEMSUB,KITEMxx,KRECxx, NOFOUND,DBSTATUS)	SGETN
DDGIXN	(TDBKEY,TDBSRI,N,KVxxxxx,NOFOUND,DBSTATUS)	SGIXN
DDINKEY	(KIxxxxx,KVxxxxx)	-----
DDINSR	(TDBKEY,KIxxxxx,DBSTATUS)	SINSR
DDMDFY	(TDBKEY,ITEMSUB,KITEMxx,KRECxx,DBSTATUS)	SMDFY
DDREMO	(TDBKEY,KIxxxxx,DBSTATUS)	SREMO
DDSTORE	(ITEMSUB,KITEMxx,KRECxx,DBSTATUS)	STORE
DDTRNSF	(KITEMxx,KRECxx,KITEMyy,KRECyy)	-----

- Other SIBAS routines are called directly from the programs.
- A direct call to SFORG/SREMB may be used instead of DDFREMB/DDREMB. Do NOT mix these two calls in the same program.
- Use DDTRNSF in FORTRAN programs only, and use DDTRNSC only in COBOL programs. <u>All</u> other routines can be called from both FORTRAN and COBOL programs.

SUBROUTINE PARAMETERS:

```
Parameter       Description
FORTRAN
----------------------------------------------------------------------
KIxxxxx         Key item information
        (1)     (not used)
        (2)     Key item length        (no. of words)
        (3- 6)  Key item  name
        (7-10)  Key realm name

KVxxxxx         Key value. Low & high limit for actual index

KITEMxx         Total-item list information
        (1)     No. of items in item list
        (2)     Total length of items (no.of words)
        (3- 6)  (not used)
        (7-10)  Realm name
        (11->)  Total-item list                (4 words pr. item)
        (N1->)  Item length                    (1 word  pr. item)
                N1 = 11 + 4*KITEMxx(1)
        (N2->)  Item type    (S, D, E, O)      (1 word  pr. item)
                N2 = 11 + 5*KITEMxx(1)

ITEMSUB         Subitem list. Type depends on start word :
                +:  => Result-item list = ITEMSUB
                -:  => Result-item list = KITEMxx minus ITEMSUB
                O:  => Result-item list = KITEMxx
                Where result-item list is used in the SIBAS call
                End subitem list by a '*'.
   Typical subitem list: citmsub(1)='+:<.ITEM.><.ITEM.>....<.ITEM.>*'

KRECxx          Value buffer (Record) for all items in KITEMxx.

TDBKEY          Temporary database key
TDBSRI          Temporary search region indicator

OPTION          =0: Forget/Remember  record
                =1: Forget/Remember  search region
                =2: Forget all records
                =3: Forget all search regions
                =4: Forget all records and search regions

DBSTATUS    DBSTATUS different from 0 or 1 indicates an error.

INTEG           Used in routine DDACCD :
                     = "  " : Call both routine ACCID and ACCDD
                     = "S " : Call ACCID
                     = "D " : Call ACCDD
```

Where xxxxx: realm prefix and datatype/item name (generated by INCLUDE)
      xx   : realm prefix (generated by INCLUDE)

If you have answered Yes to the additional declaration option in the Subfunction command, the only variables which may be declared by the user are: INTEG, N, and NOFOUND.

4.2 ABM-SIB-LIB ROUTINES FOR FORTRAN APPLICATIONS

```
DDACCD (TDBKEY,INTEG,ITEMSUB,KIxxxxx,KRECxx,DBSTATUS)
          I     I      I        I      I/O     O
```

- **ACCUMULATION of item values.**

KIxxxxx and KRECxx are found in the INCLUDE file. INTEG must be declared by the user. Build up ITEMSUB in order to specify the exact item list for the SIB-DML call. At input, the increments are put in the KRECxx buffer. Specify also the type of accumulation by giving INTEG one of the following possible values :

```
"  " - both ACCID and ACCDD are called.
"S " - ACCID is called.
"D " - ACCDD is called.
```

At output, the results are found in KRECxx.

```
DDFTCH  (KIxxxxx,KVxxxxx,DBSTATUS)
DDFEBL  (KIxxxxx,KVxxxxx,DBSTATUS)
DDFLBL  (KIxxxxx,KVxxxxx,DBSTATUS)
            I       I       O
```

- **Find a specific record.**
- **Find FIRST record between limits using given key.**
- **Find LAST record between limits using given key.**

KIxxxxx and KVxxxxx are found in the INCLUDE file. xxxxx must be the index name.

Data transfer parameters:
Give the key value by assigning a value to the variable KIxxxxx.
Give the low and high limits by assigning values to the variables Lxxxxxn and Hxxxxxn.
Use ordinary assignments.
It is only necessary to assign values for those parameters you want to be different than the default values.

Default value low limit: 00000B
Default value high limit: 77777B

Description: KIxxxxx and KVxxxxx are decomposed and the corresponding SIB-DML call is performed.

Note: At return, DDINKEY is called, which resets the low and high limits.

```
DDFTCGT (KIxxxxx,KVxxxxx,ITEMSUB,KITEMxx,KRECxx,DBSTATUS)
           I        I       I       I      O      O
```

- **Find a specific record and get the record values.**

KIxxxxx, KVxxxxx, ITEMSUB, KITEMxx and KRECxx are found in the INCLUDE file. xxxxx is the index name and xx is the realm prefix.

Give the key value by assigning the value to the variable Lxxxxx1 (Lxxxxx2, Lxxxxx3 and so on if the key is a group item.)

Build up ITEMSUB in order to specify the exact item list.

Data read from the database is available in the variables corresponding to the value buffer.

(See also description of DDFTCH and DDGET.)

```
DDTRNSF (KITEMxx,KRECxx,KITEMyy,KRECyy)
           I       I       O       O
```

- **TRANSFER of values between value buffers (for FORTRAN applications).**

The parameters are found in the INCLUDE files. xx and yy are the realm name prefix abbreviations or picture record name abbrivation.

Description: Values for items with "equal" names in the two item lists are transferred. "Equal", in this routine, means equal in the last 6 characters; that is, only different in the realm prefix. This is an important difference from the COBOL DDTRNSC routine.

Note: Because the structures in the item list and value buffers are exactly the same for the database buffers and the screen buffers, this routine may be used to transfer values from:

- one realm buffer to another;
- one picture-record buffer to another;
- a realm buffer to a picture record buffer;
- a picture record buffer to a realm buffer.

```
DDFREMB (TDBSRI, OPTION, DBSTATUS)
DDFORG  (TDBSRI, OPTION, DBSTATUS)
           I/O      I       O
```

- **FORGET old and REMEMBER a new record or a search region.**
- **FORGET, nullify the effect of a REMEMBER call.**

Description:

DDFREMB:

This routine forgets the remembered record(s)/search region(s) indicated by TDBSRI and remembers a new one. If a record/search region is not previously remembered, only remember is called.

The input value on TDBSRI indicates remembered or not:
TDBSRI = 0 : Not previously remembered
TDBSRI different from 0 : Previously remembered

Legal OPTION values 0 and 1:
0 : (forget) and remember temporary database key
1 : (forget) and remember search region indicator

DDFORG:

This routine forgets a record (0), a search region (1), all remembered records (2), all remembered search regions (3), or all remembered records and search regions (4) depending on the value in OPTION.
When forgetting one single record/search region, TDBSRI is reset to zero after the forget call. If TDBSRI = 0 at input, the forget call is skipped.
When forgetting all records/search regions, TDBSRI serves as a dummy parameter.

Note:

At the start of the program and after a FORGET-ALL call, all used temporary database keys and search region indicators should be reset to zero.

Ordinary SFORG and SREMB may be used instead of DDFREMB and DDFORG, but it is recommended that you use either SFORG/SREMB or DDFREMB/DDFORG throughout the same program.

```
DDGET  (TDBKEY,ITEMSUB,KITEMxx,KRECxx,DBSTATUS)
          I      I       I       O      O
DDMDFY (TDBKEY,ITEMSUB,KITEMxx,KRECxx,DBSTATUS)
          I      I       I       I      O
DDSTORE(ITEMSUB,KITEMxx,KRECxx,DBSTATUS)
          I        I       I       O
```

- **GET the relevant records, items or group items.**
- **MODIFY values of items or group items in a record.**
- **STORE a (part of a) record in its realm.**

KITEMxx and KRECxx are found in the INCLUDE files. xx must be the realm name.

Data transfer parameters:
Build up ITEMSUB in order to specify the exact item list for the SIB-DML call.

DDSTORE and DDMDFY: Assign values to the value buffer by assigning values to the corresponding variables, either by using the routine DDTRNSF or by ordinary assignment statements.

DDGET: Data read from the database is available in the variables corresponding to the value buffer.

Description: ITEMSUB and KITEMxx determine the result-item list. In DDMDFY and DDSTORE, the corresponding values are transferred to a local value buffer.
The corresponding SIB-DML call is performed.
In DDGET, the values returned are transferred to KRECxx.

Note: Only variables corresponding to the result-item list (i.e., the item list determined by ITEMSUB and KITEMxx) are used/changed in the call to DDGET, DDMDFY and DDSTORE.

```
DDINKEY (KIxxxxx,KVxxxxx)
             I       O
```

- **Reset search regions to maximum.**

Input : KIxxxxx
Output: KVxxxxx

Description: The value buffer KVxxxxx is reset to the default values.
Low limit : 00000B
High limit: 77777B

Use: Automatically called from the ASSDDI file for all referenced indexes.
Called from DDFTCH, DDFEBL and DDFLBL after the SIB-DML call.
May be called from application programs too, but this will normally not be necessary.

```
DDINSR (TDBKEY,KIxxxxx,DBSTATUS)
DDREMO (TDBKEY,KIxxxxx,DBSTATUS)
           I      I       O
```

- **INSERT an index key of a record.**
- **REMOVE a manually maintained index key.**

KIxxxxx is found in the INCLUDE file. xxxxx must be the index name abbreviation.

Description: Call the corresponding SIB-DML call with the index name found in KIxxxxx.

```
DDGETN (TDBKEY,TDBSRI,N,ITEMSUB,KITEMxx,KRECxx,NOFOUND,DBSTATUS)
          I        I  I     I       I      O       O       O
```

● **GET (read) a number of records in a search region.**

KITEMxx is found in the INCLUDE file. xx must be the realm name. N is the number of records desired. NOFOUND is the number found (NOFOUND .LE. N).

Description: ITEMSUB and KITEMxx determine the result-item list. The corresponding SIBAS call SGETN is performed, and the NOFOUND occurrences are transferred to KRECxx.

Note: For values of N greater than 1 (if N=1 you can also use DDGET), the KRECxx buffer must be declared by the programmer. The format for one occurrence of the specified record is the same as the generated KRECxx. You then have to repeat this N times.

```
DDGIXN (TDBKEY,TDBSRI,N,KVxxxxx,NOFOUND,DBSTATUS)
          I       I   I    O       O        O
```

● **GET (read) a number of index keys.**

The description is in the SIBAS manual. KVxxxxx is found in the INCLUDE file (for N=1).

Description: Set the maximum number of key values desired in N and call DDGIXN. The number found is returned in NOFOUND and the 'key list' is found in KVxxxxx.

Note: For values of N greater than 1, KVxxxxx must be declared by the programmer (same here as for DDGETN).

CHAPTER 5
HOW TO USE ABM-SIB-LIB ROUTINES IN COBOL APPLICATIONS

5 HOW TO USE ABM-SIB-LIB ROUTINES IN COBOL APPLICATIONS

5.1 ROUTINES AND PARAMETERS IN ABM-SIBAS-LIBRARY

Description of routines in ABM-SIB-LIB

DDACCD	-	ACCUMULATION of item values.
DDFEBL	-	Find FIRST record between limits using given key.
DDFLBL	-	Find LAST record between limits using given key.
DDFORG	-	FORGET, nullify the effect of a REMEMBER call.
DDFREMB	-	FORGET old and REMEMBER a new record or a search region.
DDFTCGT	-	FIND a specific record and GET the record values.
DDFTCH	-	FIND a specific record.
DDGET	-	GET the relevant record, items or group items.
DDGETN	-	GET (read) a number of records in a search region.
DDGIXN	-	GET (read) a number of index keys.
DDINKEY	-	Reset search regions to maximum.
DDINSR	-	INSERT an index key of a record.
DDMDFY	-	MODIFY values of items or group items in a record.
DDREMO	-	REMOVE a manually maintained index key.
DDSTORE	-	STORE a (part) of a record in its realm.
DDTRNSC	-	TRANSFER of values between value buffers (for COBOL appl.)

PARAMETERS OF THE ABM-SIB-LIB ROUTINES:

Library name: ABM-SIB-LIB-<version>.

Routines in this library			calls SIBAS routine
DDACCD	(TDBKEY,INTEG,DDC-SELECT,DDB-realm-SUBSCHEMA, DBV-realm,DBSTATUS)		ACCID ACCDD
DDFEBL	(DBKI-realm-item,DBKV-realm-item,DBSTATUS)		SFEBL
DDFLBL	(DBKI-realm-item,DBKV-realm-item,DBSTATUS)		SFLBL
DDFORG	(TDBSRI,OPTION,DBSTATUS)		SFORG
DDFREMB	(TDBSRI,OPTION,DBSTATUS)	SFORG	SREMB
DDFTCGT	(DBKI-realm-item,DBKV-realm-item,DDC-SELECT, DDB-realm-SUBSCHEMA,DBV-realm,DBSTATUS)		SFTGT
DDFTCH	(DBKI-realm-item,DBKV-realm-item,DBSTATUS)		SFTCH
DDGET	(TDBKEY,DDC-SELECT,DDB-realm-SUBSCHEMA,DBV-realm, DBSTATUS)		SGET
DDGETN	(TDBKEY,TDBSRI,N,DDC-SELECT,DDB-realm-SUBSCHEMA, DBV-realm, NOFOUND,DBSTATUS)		SGETN
DDGIXN	(TDBKEY,TDBSRI,N,DBKV-realm-item,NOFOUND,DBSTATUS)		SGIXN
DDINKEY	(DBKI-realm-item,DBKV-realm-item)		-----
DDINSR	(TDBKEY,DBKI-realm-item,DBSTATUS)		SINSR
DDMDFY	(TDBKEY,DDC-SELECT,DDB-realm-SUBSCHEMA,DBV-realm, DBSTATUS)		SMDFY
DDREMO	(TDBKEY,DBKI-realm-item,DBSTATUS)		SREMO
DDSTORE	(DDC-SELECT,DDB-realm-SUBSCHEMA,DBV-realm,DBSTATUS)		STORE
DDTRNSC	(DDx-realm1-SUBSCHEMA,xxV-realm1,DDy-realm2-SUBSCHEMA, yyV-realm2)		

- Other SIBAS routines are called directly from the programs.
- A direct call to SFORG/SREMB may be used instead of DDFREMB/DDREMB. Do NOT mix the use of these two calls in the same program.
- Use DDTRNSC only in COBOL programs.

SUBROUTINE PARAMETERS:

```
Parameter      Description
COBOL
------------------------------------------------------------------
DBKI-realm-item
    (1)     (not used)
    (2)     Key item length       (no. of words)
    (3- 6)  Key item  name
    (7-10)  Key realm name

DBKV-realm-item   Value buffers for total-item list.

DDB-realm-SUBSCHEMA
    (1)     No. of items in item list
    (2)     Total length on items (no.of words)
    (3- 6)  (not used)
    (7-10)  Realm name
    (11->)  Total-item list              (4 words pr. item)
    (N1->)  Item length                  (1 word  pr. item)
            N1 = 11 + 4*DDB-realm-SUBSCHEMA (1)
    (N2->)  Item type     (S, D, E, O)   (1 word  pr. item)
            N2 = 11 + 5*DDB-realm-SUBSCHEMA (1)

DDC-SELECT Subitem list. Type depends on start word :
           +: => Result-item list = DDC-SELECT
           -: => Result-item list = (DDB-realm-SUBSCHEMA) minus
                                    DDC-SELECT.
           O: => Result-item list = DDB-realm-SUBSCHEMA
           Where result-item list is used in the SIBAS call
           End subitem list by a '*'.
 Typical subitem list: move '+:<ITEM><ITEM>....<ITEM>*' to DDC-SELECT

DBV-realm  Only values for items in result-item list are changed/used
TDBKEY     Temporary database key
TDBSRI     Temporary search region indicator

OPTION     =0: Forget/Remember  record
           =1: Forget/Remember  search region
           =2: Forget all records
           =3: Forget all search regions
           =4: Forget all records and search regions

DBSTATUS   DBSTATUS different from 0 or 1 indicates an error.

INTEG      Used in routine DDACCD :
                = "  " : Call both routine ACCID and ACCDD
                = "S " : Call ACCID
                = "D " : Call ACCDD
```

Where realm: realm name (generated by COPY)
      item : item name (generated by COPY)

If you have answered Yes to the additional declaration option in the Subfunction command, the only variables which may be declared by the user are: INTEG, N and NOFOUND.

5.2 ABM-SIB-LIB ROUTINES FOR COBOL APPLICATIONS

```
DDFTCH ─┐                                   Input/Output
DDFEBL  ├──> USING    DBKI-realm-item,        (I)
DDFLBL ─┘             DBKV-realm-item,        (I)
                      DBSTATUS.               (O)
```

- **Find a specific record.**
- **Find FIRST record between limits using given key.**
- **Find LAST record between limits using given key.**

DBKI-realm-item and DBKV-realm-item are found in the COPY file. 'Realm' and 'item' are the realm and key item names.

Data transfer parameters:
Give the key value by assigning the value to the variable DBKV-realm-item.

Give the low and high limits by moving values to the variables DBKV-realm-item-LOW-n and DBKV-realm-item-HIGH-n.

It is only necessary to assign values to those variables you want to be different from the default values:

Default value low limit	: 00000B
Default value high limit	: 77777B

Description: The given input values are decoded and the corresponding SIB-DML call is performed. At return, DDINKEY is called.

```
                                   Input/Output
DDFTCGT    USING    DBKI-realm-item,      (I)
                    DBKV-realm-item,      (I)
                    DDC-SELECT,           (I)
                    DDB-realm-SUBSCHEMA,  (I)
                    DBV-realm,            (O)
                    DBSTATUS.             (O)
```

- **Find a specific record and get the record values.**

DBKI-realm-item, DBKV-realm-item, DDC-SELECT, DDB-realm-SUBSCHEMA and DBV-realm are found in the COPY file. 'Realm' and 'item' are the realm and key item names.

Give the key value by assigning the value to the variable DBKV-realm-item-LOW-1 (DBKV-realm-item-LOW-2 and so on if the key is a group item.)

Build up DDC-SELECT in order to specify the exact item list.

Data read from the database is available in the variables corresponding to the value buffer.

(See also description of DDFTCH and DDGET.)

```
                                    Input/Output
DDINKEY    USING     DBKI-realm-item,   (I)
                     DBKV-realm-item.   (O)
```

- **Reset search regions to maximum.**

Description: The value buffer DBKV-realm-item is reset to the default values.
Low limit : 00000B
High limit: 77777B

Use: Automatically called from the ASSDDC file for all referenced indexes.

Note: Called from DDFTCH, DDFEBL and DDFLBL, after the SIB-DML call.
May be called from application programs too, but this will normally not be necessary.

```
DDINSR ─┐                              Input/Output
DDREMO ─┴─→   USING   TDBKEY,               (I)
                      DBKI-realm-item,      (I)
                      DBSTATUS.             (O)
```

- **INSERT an index key of a record.**
- **REMOVE a manually maintained index key.**

DBKI-realm-item is found in the COPY file. 'Realm' and 'item' are the realm and key item names.

Description: Call the corresponding SIB-DML call with the index name found in the DBKI-realm-item.

```
DDGET  ──┐                              Input/Output
         ├→ USING  TDBKEY,                 (I)
DDMDFY ──┘         DDC-SELECT,             (I)
                   DDB-realm-SUBSCHEMA,    (I)
                   DBV-realm,              (I),(O for DDGET)
                   DBSTATUS.               (O)

DDSTORE     USING  DDC-SELECT,             (I)
                   DDB-realm-SUBSCHEMA,    (I)
                   DBV-realm,              (I)
                   DBSTATUS.               (O)
```

- **GET the relevant record, items or group items.**
- **MODIFY values of items or group items in a record.**
- **STORE a (part of a) record in its realm.**

DDC-SELECT, DDB-realm-SUBSCHEMA and DBV-realm are found in the COPY files. 'Realm' is the realm name.

Data transfer parameters:
Build up DDC-SELECT in order to specify the exact item list for the SIB-DML call.

DDSTORE, DDMDFY: Assign values to the value buffer by assigning values to the corresponding variables, either by using routine DDTRNSC or by ordinary assignment statements.

DDGET: Data read from the database is available in the variables corresponding to the value buffer.

Description: DDC-SELECT and DDB-realm-SUBSCHEMA determine the result-item list.
In DDMDFY and DDSTORE, the corresponding values are transferred to a local value buffer. The corresponding SIB-DML call is performed. In DDGET, the values returned are transferred to the DBV-realm area.

Note: Only variables corresponding to the result-item list ; that is, the item list determined by DDC-SELECT and DDB-realm-SUBSCHEMA , are used/changed in calls to DDGET, DDMDFY, DDSTORE.

```
                                    Input/Output
DDGETN    USING  TDBKEY,                (I)
                 TDBSRI,                (I)
                 N,                     (I)
                 DDC-SELECT,            (I)
                 DDB-realm-SUBSCHEMA,   (I)
                 DBV-realm,             (I)
                 NOFOUND,               (O)
                 DBSTATUS.              (O)
```

● **GET (read) a number of records in a search region.**

The prefixed names are found in the COPY file. 'Realm' is the realm name. N is number of records desired. NOFOUND is the number found (NOFOUND ≤ N).

Description: DDC-SELECT and DDB-realm-SUBSCHEMA determine the result-item list. The corresponding SIBAS call SGETN is performed, and the NOFOUND occurrences are transferred to DBV-realm.

Note: For values of N greater than 1 (if N=1 you can also use DDGET), the DBV-realm buffer must be declared by the programmer. The format for one occurrence of the specified record is the same as for the generated one. You then have to repeat this N times.

```
                                Input/Output
DDGIXN    USING  TDBKEY,            (I)
                 TDBSRI,            (I)
                 N,                 (I)
                 DBKV-realm-item,   (O)
                 NOFOUND,           (O)
                 DBSTATUS.          (O)
```

● **GET (read) a number of index keys.**

The parameters are described in the SIBAS manual (ND-60.127.03). DBKV-realm-item is found in the COPY file (for N=1). 'Realm' and 'item' are realm and key item names.

Description: Set the maximum number of key values desired in N and call DDGIXN. The number found is returned in NOFOUND and the 'key list' is found in the 'low' part of DBKV-realm-item.

Note: For values of N greater than 1, the DBKV-realm-item must be declared by the programmer (like in the call for DDGETN).

```
                                  Input/Output
DDACCD     USING  TDBKEY,               (I)
                  INTEG,                (I)
                  DDC-SELECT,           (I)
                  DDB-realm-SUBSCHEMA,  (I)
                  DBV-realm,           (I/O)
                  DBSTATUS.             (O)
```

● **ACCUMULATION of item values.**

DDC-SELECT, DDB-realm-SUBSCHEMA and DBV-realm are found in the COPY file. 'Realm' is the realm name.

Build up DDC-SELECT in order to specify the exact item list for the SIB-DML call. At input, the increments are put in the DBV-realm buffer. Specify also the type of accumulation by giving INTEG one of the possible values:

```
"  " - both ACCID and ACCDD are called.
"S " - ACCID is called.
"D " - ACCDD is called.
```

At output the results are found in DBV-realm.

```
                                       Input/Output
DDTRNSC    USING  DDx-realm1-SUBSCHEMA,      (I)
                  xxV-realm1,                (I)
                  DDy-realm2-SUBSCHEMA,      (O)
                  yyV-realm2.                (O)
```

● **TRANSFER of values between value buffers (for COBOL appl.)**

Parameters are found in the COPY files. 'Realm1' and 'realm2' are the realm names. x and y are either S (for Screen) or B (for dataBase). xx and yy are either SC (giving SCV - SCreen Value) or DB (giving DBV - DataBase Value). 'Realm1' must be different from 'realm2'.

Description: Values for items with "equal" names in the two item lists are transferred from 'realm1' to 'realm2'. "Equal" in this routine means equal in all characters. Note the important difference from DDTRNSF.

Note: Because the structures in the item list and value buffers are exactly the same for the database buffers and the screen buffers, this routine may be used to transfer values from:

- one realm buffer to another;
- one picture record buffer to another;
- a realm buffer to a picture record buffer;
- a picture record buffer to a realm buffer.

```
DDFREMB ──┐            Input/Output
DDFORG  ──┴──→ USING  TDBSRI,   (I/O)
                      OPTION,   (I)
                      DBSTATUS. (O)
```

- **FORGET old and REMEMBER a new record or a search region.**
- **FORGET, nullify the effect of a REMEMBER call.**

Description:

DDFREMB:
This routine forgets the remembered record/search region indicated by TDBSRI, and remembers a new one.
If a record/search region is not previously remembered, only remember is called.
The input value on TDBSRI indicates remembered or not:
TDBSRI = 0 : Not previously remembered.
TDBSRI different from 0 : Previously remembered.
Legal OPTION values 0, 1:
=0 : (forget) and remember temporary database key.
=1 : (forget) and remember search region indicator.

DDFORG:
This routine forgets a record (0), a search region (1), all remembered records (2), all remembered search regions (3), or all remembered records and search regions (4), depending on the value in OPTION.
When forgetting one single record/search region, TDBSRI is reset to zero after the forget call. If TDBSRI = 0 at input, the forget call is skipped. When forgetting all records/search regions, TDBSRI serves as a dummy parameter.

Note:

At the start of the program and after a FORGET-ALL call, all used temporary database keys and search region indicators should be reset to zero.

Ordinary SFORG and SREMB may be used instead of DDFREMB and DDFORG, but it is recommended that you use either SFORG/SREMB or DDFREMB/DDFORG throughout the same program.

CHAPTER 6
HOW TO USE ABM-FOCUS-LIBRARY IN FORTRAN APPLICATIONS

6 HOW TO USE ABM-FOCUS-LIBRARY IN FORTRAN APPLICATIONS

6.1 ROUTINES AND PARAMETERS IN ABM-FOCUS-LIBRARY

The ABM-FC-LIBRARY contains the following "user available" routines:

DDCFLDS	-	Clears fields/records or parts of records.
DDCLAT	-	Clears attributes.
DDCLFI	-	Closes an opened file.
DDCLMR	-	Clears "must-read" for fields/records.
DDCMSGE	-	Clears a message line.
DDCOPTF	-	Copies a displayed picture to file.
DDERROR	-	Decodes the error status and returns an error text.
DDGETRC	-	Gets field values from the total picture buffer.
DDGMSGE	-	Writes a message to a message line and reads the answer.
DDGTEXT	-	Writes a message in a given line and column and reads the answer.
DDGTPIC	-	Gets a picture from a file, displays and makes it ready.
DDINITE	-	Initiates and terminates the SCREEN part of a program.
DDOPFI	-	Opens a SINTRAN file for Write, Append access.
DDPUTRC	-	Puts field values into the total picture buffer.
DDRFLDS	-	Reads fields/records or parts of records.
DDSETAT	-	Sets attributes.
DDSETMR	-	Sets "must-read" for fields/records.
DDWFLDS	-	Writes fields/records or parts of records.
DDWMSGE	-	Writes a message to the message line.

The ABM-FOCUS-LIBRARY routines and the FOCUS routines:

SUBROUTINE :	CALLS FOCUS ROUTINES:
DDCFLDS (REFTAB, MITEMxx, MRECxx, FCSTATUS)	FCCLSUB,FCCLFDS
DDCLAT (REFTAB, MITEMxx, MRECxx, FCSTATUS)	FCSETAT
DDCLFI (IUNIT, FCSTATUS)	FCCLOSE
DDCLMR (REFTAB, MITEMxx, MRECxx, FCSTATUS)	FCCLMR
DDCMSGE (FCSTATUS)	FCWTXT,FCSCRIN
DDCOPTF (IUNIT, FCSTATUS)	FCPRDOC
DDERROR (FCSTATUS, MESSAGE)	-
DDGETRC (REFTAB, MITEMxx, MRECxx, FCSTATUS)	
DDGMSGE (MESSAGE, OTEXT, FCSTATUS)	All in DDWMSGE + FCRTXT
DDGTEXT (MESSAGE, OTEXT, ILINE, ICOL, FCSTATUS)	CWTXT, FCRTXT
DDGTPIC (FORMFILE, REFTAB, FCSTATUS)	FCDECFF,FCDECFN,FCCLFDS,FCCLREC
DDINITE (MFLAG)	FCINITE
DDOPFI (IFINA, IUNIT, FCSTATUS)	FCOPEN
DDPUTRC (REFTAB, MITEMxx, MRECxx, FCSTATUS)	
DDRFLDS (REFTAB, MITEMxx, MRECxx, FCSTATUS)	FCESUB
DDSETAT (REFTAB, MITEMxx, MRECxx, IATRBT, FCSTATUS)	FCSETAT
DDSETMR (REFTAB, MITEMxx, MRECxx, FCSTATUS)	FCSETMR
DDWFLDS (REFTAB, MITEMxx, MRECxx, FCSTATUS)	FCWSUB
DDWMSGE (MESSAGE, FCSTATUS)	FCWTXT,FCPWR,FCSCRIN,FCBELL,FCRCHR

PARAMETER FORTRAN	**DESCRIPTION**
MITEMxx	Total item list information :
(1)	No. of items in item list.
(2)	Total length of items (No of 16-bit words).
(3)	No. of records of this type.
(4)	First item number in first record of this type.
(5)	First word in value buffer in first record of this type.
(6)	Not used.
(7-20)	Record type name.
(11->)	Total item list (4 words per item).
(N1->)	Item length (1 word per item).
	N1 = 11 + 4*MITEMxx(1)
(N2->)	Item type: S, D, E, O, 1n or 2n. (1 word per item).
	N2 = 11 + 5*MITEMxx(1).
ITEMSUB	Subitem list: type depends on the start word.
	+: => result-item list = ITEMSUB
	-: => result-item list = MITEMxx - ITEMSUB
	0: => result-item list = MITEMxx
	Where result-item list is used in the FOCUS call.
	* => End of subitem list.
	Typical subitem list:
	citmsub(1)='+:<.ITEM.><.ITEM.>....<.ITEM.>*'
MRECxx	Record with picture values for all items in MITEMxx Only values for items in the result-item list are changed/used.
LINE, NOLINE	Line (record) number & number of occurrences.
	LINE = 0 & NOLINE = 0: All occurrences of this rec.
	LINE > 0 & NOLINE > 0: 'NOLINE' occurrences starting from occurrence 'LINE' of the record type.
	LINE > 0 & NOLINE = 0: All occurrences starting from occurrence 'LINE' of the record type.
	(defult from INCLUDE/COPY is LINE=NOLINE=1)
MESSAGE	Message type and text.
(1:1)	Message type (not written)
(byte no.1)= +	=> Turn ON the "prohibited to overwrite message line" mechanism. The operator has to give CR before the next message is written on the message line.
= -	=> Turn OFF the "prohibited to overwrite message line" mechanism.
= 0	=> The message will be written out on the message line. There is no prevention of message.
(2:2)	Delimiter (not written).
(byte no. 2)	Must be equal to ":".
(3:80)	Text string, the text that will be displayed.
(bytes 3 through 80)	

example of use: cmessage = '+: Please give unit type and number'''

NOTE !	The routines DDRFLDS, DDGMSGE and DDGTEXT will reset flags that indicate when the operator has given a CR before the next message.
OTEXT	Output message in the DDGTEXT call and the DDGMSGE call. OTEXT must be defined as a table, and have the same length as MAXOTEXT in the ABM-FC-LIB:INCL. Default length is 40 bytes.
IUNIT	Logical unit number for the output file (DDCOPTF)
FORMFILE	Picture file name
CPNS	Picture name (8 characters)
ILINE	Line number on the screen (normally 1 - 24/25)
ICOL	Column number on the screen (normally 1 - 80)
MRMO	Read mode. ┐ Default from INCLUDE is
MWMO	Write mode.┘ MRMO=MWMO=1 ("normal" read/write).
FCSTATUS	Routine status: = 0: No error in DD<FOCUS> call. otherwise : Error situation (see DDERROR).

REFTAB Reference table instead of a long parameter list:

	FORTRAN	COBOL
(1) - (4)	CPNS	SCC-PIC-NAME
(5)	MRMO	SCC-READ-MODE
(6)	MWMO	SCC-WRITE-MODE
(7)	LINE	SCC-START-RW-LINE
(8)	NOLINE	SCR-RW-NO-OF-LINES
(11) ->	ITEMSUB	DDC-SELECT

In the routine parameters we always refer to REFTAB. Default values are initiated by ASSDDI-<subfunc>. Change the values by using the name (not the index in REFTAB).

MFLAG Input to DDINITE : 1 init
0 exit

Where xx : realm prefix (generated by INCLUDE).

The variables which are not automatically decleared are:
IUNIT, ILINE, ICOL, IFINA and IATRBT.

NOTE:
All dimensioning of the parameters should be done in the file ABM-FC-LIB:INCL. Default values of the subroutines are supported on delivery of the program, but these values can be changed by the user.

Remember the parallelism :

FOCUS		SIBAS
Picture	----	Database
Record	----	Realm
Field	----	Item

6.2 HOW TO USE THE ABM-FC-LIB ROUTINES IN FORTRAN APPLICATIONS

```
DDCFLDS (REFTAB,MITEMxx,MRECxx,FCSTATUS)
           I/O     I       O       O
```

● **Clears fields/records or parts of records.**

Description: The specified parts of the fields on the screen and the corresponding parts of the picture buffer for the whole screen are cleared.
The following possibilities exist:

- Clear all fields in a record (LINE=0, NOLINE=0).
- Clear from occurrence LINE no. of picture record type and all following occurrences of that record type (NOLINE=0)
- Clear from occurrence LINE no. of picture record type and the NOLINE following record occurrences.
  The picture record type is found in MITEMxx.

Default from INCLUDE is LINE=NOLINE=1.

```
DDCLAT  (REFTAB, MITEMxx, MRECxx, FCSTATUS)
           I/O      I        O        O
```

● **Clears attributes.**

Description: This routine will give the result-item list the "normal" attribute set.

Use: Normally used to reset the same field set that is created by the DDSETAT call.

```
DDCLFI  (IUNIT, FCSTATUS)
           I       O
```

● **Closes an opened file.**

Use: This routine is used to close files opened by the DDOPFI routine.

```
DDCLMR  (REFTAB, MITEMxx, MRECxx, FCSTATUS)
          I/O      I        O        O
```

- **Clears "must-read" for fields/records.**

Description: The corresponding "clear-must-read" FOCUS call is performed for the fields in the result-item list.

Use: Normally this is used to reset the effect of a DDSETMR call; that is, reset the "set-must-read" for the same set of fields that is specified in the DDSETMR call.

```
DDCMSGE  (FCSTATUS)
             O
```

- **Clears a message line.**

Description: This routine clears the terminal's message line.

```
DDCOPTF  (IUNIT,FCSTATUS)
            I      O
```

- **Copies a displayed picture to a file.**

Description: The routine will write a form with leading texts and field contents to a file.

Use: The file should previously be opened by using the routine DDOPFI. The returned unit number from this routine should be used as input in DDCOPTF's IUNIT parameter.

```
DDGETRC (REFTAB,MITEMxx,MRECxx,FCSTATUS)
          I/O    I       O       O
```

● **Gets field values from the total picture buffer.**

Description: Field values for all fields in the picture record occurrence given from LINE and MITEMxx are transferred from the screen picture to MRECxx.

Use: Only one occurrence of each type is available in the application program at one time, as the same value buffer is used for all occurrences of a picture record type.

In order to get access to an arbitrary occurrence (without doing an ABM-FOCUS call), the routine DDGETRC is used.

```
DDGMSGE  (MESSAGE, OTEXT, FCSTATUS)
             I       O       O
```

● **Writes a message to a message line and reads the answer.**

Description: Works exactly as DDGTEXT, but the message line is always used.

```
DDGTEXT (MESSAGE, OTEXT, ILINE, ICOL, FCSTATUS)
            I       O      I      I       O
```

● **Writes a message in a given line and column and reads the answer.**

Description: The text in MESSAGE is displayed in position (ILINE, ICOL) on the operator's screen. It waits for a message from the operator. The message is returned in the OTEXT parameters.

Use: Assign the y/x-coordinate values to ILINE/ICOL for the start position of the message on the screen. Assign MESSAGE the message to be displayed. The return message will be found in OTEXT.

```
DDGTPIC (FORMFILE,REFTAB,FCSTATUS)
             I       I/O      O
```

- **Gets a picture from a file, displays and makes it ready.**

Assign the FOCUS form-file (:FABM) name to FORMFILE. Make sure that the CPNS in REFTAB holds the correct form name.

Description: The necessary ABM-FOCUS routines to initiate a form are called, and the form is displayed on the screen.

```
DDINITE (MFLAG)
          I
```

- **Initiates and terminates the SCREEN part of a program.**

DDINITE must be performed right before the first DDGTPIC call. DDINITE must be the last call before leaving the part of the application which uses a FOCUS picture.

The parameter MFLAG must be set to 1 when initiating from a background program. The values of the parameter have the following meanings:

MFLAG > 1: the device is reserved by FOCUS.
MFLAG = 0: terminate. See also the FCINITE call in the FOCUS REFERENCE MANUAL (ND-60.137).

It should be noted that a maximum number of pictures, fields and buffer areas are initiated in the ABM-FC-LIB:INCL file.

Note: For RT/TPS programming, you have to choose another strategy for initiation; for example, BLOCK DATA initiation at load time.

```
DDOPFI  (IFINA, IUNIT, FCSTATUS)
           I      O      O
```

- **Opens a SINTRAN file for Write, Append access.**

Description: Opens the file given by the name assigned to IFINA (for example: IFINA="TEST:SYMB") for Write / Append access. The file number of the opened files is returned in IUNIT.

Use: This routine is used for a call to DDCOPTF.

```
DDPUTRC (REFTAB,MITEMxx,MRECxx,FCSTATUS)
           I/O     I       I       O
```

- **Puts field values into the total picture buffer.**

Description: This is the inverse of DDGETRC.
Field values for all fields in the picture record occurrence given from LINE and MITEMxx are transferred from MRECxx to the screen picture.

Use: In order to build up a whole screen picture (before doing an ABM-FOCUS call), the routine DDPUTRC is used.

```
DDRFLDS  (REFTAB,MITEMxx,MRECxx,FCSTATUS)
            I/O     I      O       O
```

● **Reads fields/records or parts of records.**

MITEMxx and MRECxx are found in the INCLUDE files. xx is the picture record name abbreviation.

Data transfer parameters:
Build up ITEMSUB in REFTAB in order to specify the exact fields for the ABM-FOCUS calls.
Data read from the screen is available in the variables corresponding to the value buffer.

Description: A result-item list, which consists of a specified set of fields, is determined by ITEMSUB and MITEMxx.
FCEDSUB is performed on the set of fields of the picture record occurrence.

The field values read are transferred from the picture buffer for the whole screen to MRECxx if only <u>one</u> record occurrence is read. If more than one record occurrence is read in one DDRFLDS call, you must transfer the values yourself by successive calls for DDGETRC. (Hint: loop with LINE from 1 to maximum number of occurrences each time you call DDGETRC, and store the values).

Before return, the message line is cleared.

With the same call to DDRFLDS, you can read one, some or all fields in the picture record occurrence. See the LINE and NOLINE parameter in REFTAB.

Note: The fields will always be read depending on the sequence in the subitem list. The default sequence for the total item list is from left to right and from top to bottom.

Normally, "read" is performed by first cleaning the fields and then by setting the display dots in the fields. This can be changed by setting the parameter MRMO in REFTAB. See the edit mode parameter in FCEDSUB call in the FOCUS Reference Manual, ND 60.137

The most common setting for MRMO is:

1 (default) : clear fields, display dots and read.
or
2 : read (without clear).

```
DDWFLDS (REFTAB,MITEMxx,MRECxx,FCSTATUS)
          I/O    I       I        O
```

● **Writes fields/records or parts of records.**

MITEMxx and MRECxx are found in the INCLUDE files. xx is the picture record name abbreviation.

Data transfer parameters:
Build up ITEMSUB in REFTAB in order to specify the exact fields for the ABM-FOCUS calls.
Assign values to the value buffer by assigning values to the corresponding variables; either by use of the routine DDTRNSF or by ordinary assignment statements.

Only variables corresponding to the result-item list are used in DDWFLDS calls.

Description: DDWFLDS works like DDRFLDS (only reversed):
The result-item list is composed, and field values are transferred to the screen buffer for the whole screen. Finally, the field values are written to the screen (by FCWSUB).

```
DDSETAT (REFTAB, MITEMxx, MRECxx, IATRBT, FCSTATUS)
         I/O      I        O        I       O
```

● **Sets attributes.**

Description: This routine is used to set attributes (inverse video, blink etc.) on fields of a field set.

Use: The elements in the IATRBT integer array of 8 elements must be assigned value 1 to be enabled, otherwise 0 is assigned. If no elements are set, the "normal" attribute is set. Attributes can be combined. The attributes will appear the next time the fields are displayed.

Effect: The enabeling of the different elements has the following meaning :

Element number	Effect on the fields
1	High (increased) intensity
2	Low (decreased) intensity
3	Italics
4	Underlined
5	Blink (slowly)
6	Blink (rapidly)
7	Inverse video
8	Invisible (password reading)

```
DDSETMR (REFTAB, MITEMxx, MRECxx, FCSTATUS)
          I/O      I        O       O
```

- **Sets "must-read" for fields/records.**

Description: The result-item list is made from the total-item list in REFTAB and the subitem list specified in ITEMSUB. The corresponding "set-must-read" FOCUS call is performed for the fields in question.

Use: Make your subitem list by assigning values to ITEMSUB, LINE and NOLINE (if occurrences of this record). Call DDSETMR just before a DDRFLDS call.

Effect: The DDRFLDS call following a DDSETMR call will not be left before all fields in the result-item list are filled in. Only the fields in result-item list of DDSETMR are affected.

```
DDWMSGE  (MESSAGE, FCSTATUS)
             I        O
```

- **Writes a message to the message line.**

Description: The message in MESSAGE is displayed on the terminal's message line.

Use: See how to assign value to the MESSAGE parameter in the parameter description in the beginning of this chapter.

```
DDERROR  (FCSTATUS, MESSAGE)
           I/O         O
```

- **Decodes error status and returns an error text.**

Use:

After all DD calls, the status parameters should be tested.

- If DD<FOCUS-call> then test if parameter not = 0. If so call DDERROR to decode the error situation.
- If DD<SIBAS-call> then test if parameter not = 1 or (sometimes) 0. If so call DDERROR to decode the error.

NOTE:
See also the routine DDERMSG in ABM-UTILITY-LIB described in chapter 8.

Examples of use:

DDERROR called after SIBAS call.

```
CALL  DDFTCH (KIDEPNO, KVDEPNO, DBSTATUS)
IF (DBSTATUS .EQ. 0) THEN
   <no item found, give message etc.>
ELSEIF (DBSTATUS .NE. 1) THEN
  CALL  DDERROR (DBSTATUS, MESSAGE)
  IF (DBSTATUS .EQ. 0) THEN
     <call routine to display MESSAGE>
  ELSE
     <call SDBEC>
  ENDIF
ENDIF
```

DDERROR called after FOCUS call.

```
CALL DDRFLDS (REFTAB,MITEMR1,MRECR1,FCSTATUS)
IF (FCSTATUS .NE. 0) THEN
   CALL  DDERROR (FCSTATUS, MESSAGE)
   IF (FCSTATUS .EQ. 0) THEN
       <call routine to display MESSAGE message>
   ELSE
       <call routine to display the name of
        FOCUS routine where error has occurred
        - this is also found in MESSAGE >
   ENDIF
ENDIF
```

CHAPTER 7
HOW TO USE ABM-FOCUS-LIBRARY IN COBOL APPLICATIONS

7 HOW TO USE ABM-FOCUS-LIBRARY IN COBOL APPLICATIONS

7.1 ROUTINES AND PARAMETERS IN ABM-FOCUS-LIBRARY

The ABM-FC-LIBRARY contains the following "user available" routines:

DDCFLDS	-	Clears fields/records or parts of records.
DDCLAT	-	Clears attributes.
DDCLFI	-	Closes an opened file.
DDCLMR	-	Clears "must-read" for fields/records.
DDCMSGE	-	Clears a message line.
DDCOPTF	-	Copies a displayed picture to file.
DDERROR	-	Decodes the error status and returns an error text.
DDGETRC	-	Gets field values from the total picture buffer.
DDGMSGE	-	Writes a message to a message line and reads the answer.
DDGTEXT	-	Writes a message in a given line and column and reads the answer.
DDGTPIC	-	Gets a picture from a file, displays and makes it ready.
DDINITE	-	Initiates and terminates the SCREEN part of a program.
DDOPFI	-	Opens a SINTRAN file for Write, Append access.
DDPUTRC	-	Puts field values into the total picture buffer.
DDRFLDS	-	Reads fields/records or parts of records.
DDSETAT	-	Sets attributes.
DDSETMR	-	Sets "must-read" for fields/records.
DDWFLDS	-	Writes fields/records or parts of records.
DDWMSGE	-	Writes a message to the message line.

The ABM-FOCUS-LIBRARY routines and the FOCUS routines:

SUBROUTINE :	CALLS FOCUS ROUTINES:
DDCFLDS (DDC-REF-TABLE,DDS-realm-SUBSCHEMA, SCV-realm,FCSTATUS)	FCCLSUB,FCCLFDS
DDCLAT (DDC-REF-TABLE,DDS-realm-SUBSCHEMA,SCV-realm,FCSTATUS)	FCSETAT
DDCLFI (IUNIT, FCSTATUS)	FCCLOSE
DDCLMR (DDC-REF-TABLE,DDS-realm-SUBSCHEMA,SCV-realm,FCSTATUS)	FCCLMR
DDCMSGE (FCSTATUS)	FCWTXT,FCSCRIN
DDCOPTF (IUNIT, FCSTATUS)	FCPRDOC
DDERROR (FCSTATUS, MESSAGE)	-
DDGETRC (DDC-REF-TABLE,DDS-realm-SUBSCHEMA,SCV-realm,FCSTATUS)	
DDGMSGE (MESSAGE, OTEXT, FCSTATUS)	All in DDWMSGE + FCRTXT
DDGTEXT (MESSAGE,OTEXT,ILINE,ICOL,FCSTATUS)	FCWTXT, FCRTXT
DDGTPIC (FORMFILE,DDC-REF-TABLE,FCSTATUS)	FCDECFF,FCDECFN,FCCLFDS,FCCLREC
DDINITE (MFLAG)	FCINITE
DDOPFI (IFINA, IUNIT, FCSTATUS)	FCOPEN
DDPUTRC (DDC-REF-TABLE,DDS-realm-SUBSCHEMA,SCV-realm,FCSTATUS)	
DDRFLDS (DDC-REF-TABLE,DDS-realm-SUBSCHEMA,SCV-realm,FCSTATUS)	FCESUB
DDSETAT (DDC-REF-TABLE,DDS-realm-SUBSCHEMA,SCV-realm,IATRBT, FCSTATUS)	FCSETAT
DDSETMR (DDC-REF-TABLE,DDS-realm-SUBSCHEMA,SCV-realm,FCSTATUS)	FCSETMR
DDWFLDS (DDC-REF-TABLE,DDS-realm-SUBSCHEMA,SCV-realm,FCSTATUS)	FCWSUB
DDWMSGE (MESSAGE, FCSTATUS)	FCWTXT,FCPWR,FCSCRIN,FCBELL,FCRCHR

PARAMETER COBOL | **DESCRIPTION**

DDS-realm-SUBSCHEMA Total item list information :

```
        (1)     No. of items in item list.
        (2)     Total length of items (No of 16-bit words).
        (3)     No. of records of this type.
        (4)     First item number in first record of this type.
        (5)     First word in value buffer in first record
                of this type.
        (6)     Not used.
        (7-20)  Record type name.
        (11->)  Total item list              (4 words per item).
        (N1->)  Item length                  (1 word  per item).
                N1 = 11 + 4*DDS-realm-SUBSCHEMA(1)
        (N2->)  Item type: S, D, E, O, 1n or 2n. (1 word per item).
                N2 = 11 + 5*DDS-realm-SUBSCHEMA(1).
```

DDC-SELECT Subitem list: type depends on the start word.

```
+:  => result-item list = DDC-SELECT
-:  => result-item list = (DDS-realm-SUBSCHEMA)
                          minus DDC-SELECT
0:  => result-item list = DDS-realm-SUBSCHEMA
Where result-item list is used in the FOCUS call.
*   => End of subitem list.

Typical subitem list:
move '+:<ITEM><ITEM>....<ITEM>*' to DDC-SELECT
```

SCV-realm Record with picture values for all items.

SCC-START-RW-LINE line (record) number.

SCC-RW-NO-OF-LINES number of occurrences.

```
     SCC-START-RW-LINE = 0 & SCC-RW-NO-OF-LINES = 0:
       All occurrences of this record.
     SCC-START-RW-LINE > 0 & SCC-RW-NO-OF-LINES > 0:
       'SCC-RW-NO-OF-LINES' occurrences starting from occurrence
       'SCC-START-RW-LINE' of the record type.
     SCC-START-RW-LINE > 0 & SCC-RW-NO-OF-LINES = 0:
        All occurrences starting from occurrence
        'SCC-START-RW-LINE' of the record type.
(default from COPY is SCC-START-RW-LINE = SCC-RW-NO-OF-LINES = 1)
```

MESSAGE Message type and text.
(1:1) Message type (not written).
(byte no.1)= + => Turn ON the "prohibited to overwrite message line" mechanism. The operator has to press CR before the next message is written on the message line.
= - => Turn OFF the "prohibited to overwrite message line" mechanism.
= 0 => The message will be written out on the message line. There is no prevention of message.
(2:2) Delimiter (not written).
(byte no. 2) Must be equal to ":".
(3:80) Text string, the text that will be displayed.
(bytes 3 through 80)
NOTE ! The routines DDRFLDS, DDGMSGE and DDGTEXT will reset flags that indicate when the operator has pressed CR before the next message.

OTEXT	Output message in the DDGTEXT call and the DDGMSGE call. OTEXT must be defined as a table, and have the same length as MAXOTEXT in the ABM-FC-LIB:INCL Default length is 40 bytes.
IUNIT	Logical unit number for the output file (DDCOPTF)
FORMFILE	Picture file name
SCC-PIC-NAME	Picture name (8 characters)
ILINE	Line number on the screen (normally 1 - 24/25)
ICOL	Column number on the screen (normally 1 - 80)
SCC-READ-MODE	Read mode.
SCC-WRITE-MODE	Write mode.

Default from COPY: read mode and write mode = 1.

FCSTATUS Routine status:

= 0:	No error in DD<FOCUS> call.
otherwise :	Error situation (see DDERROR).

DDC-REF-TABLE
Reference table instead of a long parameter list:

(1) - (4)	SCC-PIC-NAME
(5)	SCC-READ-MODE
(6)	SCC-WRITE-MODE
(7)	SCC-START-RW-LINE
(8)	SCC-RW-NO-OF-LINES
(11) ->	DDC-SELECT

In the routine parameters we always refer to DDC-REF-TABLE. Default values are initiated by ASSDDC-<subfunc>. Change the values by using the name (not the index in DDC-REF-TABLE).

MFLAG Input parameter to DDINITE : 1 init
0 exit

Where realm : realm name (generated by COPY).

Variables which are not automatically declared are:
IUNIT, ILINE, ICOL, IFINA and IATRBT.

NOTE:
All dimensioning of the parameters should be done in the file ABM-FC-LIB:INCL. Default values of the subroutines are supported on delivery of the program, but these values can be changed by the user.

Remember the parallelism :

FOCUS		SIBAS
Picture	----	Database
Record	----	Realm
Field	----	Item

7.2 HOW TO USE THE ABM-FC-LIB ROUTINES IN COBOL APPLICATIONS

```
                                 Input/Output
DDINITE      USING   MFLAG.         (I)
```

● **Initiates and terminates the SCREEN part of a program.**

Description:

DDINITE must be performed right before the first DDGTPIC call. DDINITE must be the last call before leaving the part of the application which uses a FOCUS picture.

The parameter MFLAG must be set to 1 when initiating from a background program. The values of the parameter have the following meanings.
MFLAG > 1: the device is reserved by FOCUS.
MFLAG = 0: terminate.
See also the FCINITE call in the FOCUS REFERENCE MANUAL (ND-60.137).

It should be noted that a maximum number of pictures, fields and buffer areas are initiated in the file ABM-FC-LIB:INCL.

- -

```
                                          Input/Output
DDCFLDS      USING   DDC-REF-TABLE,           (I/O)
                     DDS-realm-SUBSCHEMA,     (I)
                     SCV-realm,               (O)
                     FCSTATUS.                (O)
```

● **Clears fields/records or parts of records.**

Description:

The specified parts of the fields on the screen, and the corresponding parts of the picture buffer for the whole screen, are cleared.

The following possibilities exist:

- Clear all fields in a record (SCC-START-RW-LINE=0, SCC-RW-NO-OF-LINES=0).

- Clear from occurrence SCC-START-RW-LINE no. of picture record type and all following occurrences of that record type (SCC-RW-NO-OF-LINES=0)

- Clear from occurrence SCC-START-RW-LINE no. of picture record type and the SCC-RW-NO-OF-LINES following record occurrences.
  The picture record type is found in DDS-realm-SUBSCHEMA.

SCC-START-RW-LINE = SCC-RW-NO-OF-LINES = 1 is default from COPY.

```
                                    Input/Output
DDRFLDS     USING   DDC-REF-TABLE,          (I/O)
                    DDS-realm-SUBSCHEMA,     (I)
                    SCV-realm,               (O)
                    FCSTATUS.                (O)
```

- **Reads fields/records or parts of records.**

DDC-REF-TABLE, DDS-realm-SUBSCHEMA and SCV-realm are found in the COPY file. "Realm" is the realm (record) name.

Data transfer parameters:
Build up DDC-SELECT in DDC-REF-TABLE in order to specify the exact fields for the ABM-FOCUS calls.
Data read from the screen is available in the variables corresponding to the value buffer.

Description: A result-item list, which consists of a specified set of fields, is determined by DDC-SELECT and DDS-realm-SUBSCHEMA.

FCEDSUB is performed on the set of fields of the picture record occurrence.

The field values read are transferred from the picture buffer for the whole screen to SCV-REALM if only <u>one</u> record occurrence is read. If more than one record occurrence is read in one DDRFLDS call, you must transfer the values yourself by successive calls for DDGETRC. (Hint: loop with SSC-START-RW-LINE from 1 to maximum number of occurrences each time you call DDGETRC, and store the values).

Before return, the message line is cleared.

With the same call to DDRFLDS, you can read one, some or all fields in the picture record occurrence. See the SSC-START-RW-LINE and SSC-RW-NO-OF-LINES parameter in REFTAB.

Note: The fields will always be read depending on the sequence in the subitem list. The default sequence for the total item list is from left to right and from top to bottom.

Normally, "read" is performed by first cleaning the fields and then by setting the display dots in the fields. This can be changed by setting the parameter SSC-READ-MODE in REFTAB. See the edit mode parameter in FCEDSUB call in the FOCUS Reference Manual, ND 60.137

The most common setting for SSC-READ-MODE is:

1 (default) : clear fields, display dots and read.
or
2 : read (without clear).

```
                                          Input/Output
DDWFLDS      USING   DDC-REF-TABLE,           (I/O)
                     DDS-realm-SUBSCHEMA,      (I)
                     SCV-realm,                (I)
                     FCSTATUS.                 (O)
```

- **Writes fields/records or parts of records.**

DDC-REF-TABLE, DDS-realm-SUBSCHEMA and SCV-realm are found in the COPY file.

Data transfer parameters:
Build up DDC-SELECT in DDC-REF-TABLE in order to specify the exact fields for the ABM-FOCUS calls.
Assign values to the value buffer by assigning values to the corresponding variables, either by use of routine DDTRNSC or by ordinary assignment statements.

Only variables corresponding to the result-item list are used in DDWFLDS calls.

Description: DDWFLDS works like DDRFLDS (only reversed):
The result-item list is composed, and field values are transferred to the screen buffer for the whole screen. Finally, the field values are written to the screen (by FCWSUB).

```
                                  Input/Output
DDGETRC     USING   DDC-REF-TABLE,         (I/O)
                    DDS-realm-SUBSCHEMA,    (I)
                    SCV-realm,              (O)
                    FCSTATUS.               (O)
```

● **Gets field values from the total picture buffer.**

Description: Field values for all fields in the picture record occurrence given from SCC-START-RW-LINE (in DDC-REF-TABLE) and DDS-realm-SUBSCHEMA are transferred from the screen picture to SCV-realm.

Use: Only one occurrence of each type is available in the application program at one time, because the same value buffer is used for all occurrences of a picture record type.

In order to get access to an arbitrary occurrence (without doing a ABM-FOCUS call) the routine DDGETRC should be used.

```
                                  Input/Output
DDPUTRC     USING   DDC-REF-TABLE,         (I/O)
                    DDS-realm-SUBSCHEMA,    (I)
                    SCV-realm,              (I)
                    FCSTATUS.               (O)
```

● **Puts field values into the total picture buffer.**

Description: This is the inverse of DDGETRC.
Field values for all fields in the picture record occurrence given from SCC-START-RW-LINE and DDS-realm-SUBSCHEMA are transferred from SCV-realm to the screen picture.

Use: Only one occurrence of each type is available in the application program at one time, as the same value buffer is used for all occurrences of a picture record type.

In order to build up a whole screen picture (before doing an ABM-FOCUS call), the routine DDPUTRC should be used.

```
                                        Input/Output
DDSETMR    USING       DDC-REF-TABLE,        (I/O)
                       DDS-realm-SUBSCHEMA,  (I)
                       SCV-realm,            (O)
                       FCSTATUS.             (O)
```

● **Sets "must-read" for fields/records.**

Description: The result-item list is made from the total-item list in DDC-REF-TABLE and the subitem list specified in DDC-SELECT. The corresponding "set-must-read" FOCUS call is performed for the fields in question.

Use: Make your subitem list by assigning values to DDC-SELECT, SCC-START-RW-LINE and SCC-RW-NO-OF-LINES (if occurrences of this record). Call DDSETMR just before a DDRFLDS call.

Effect: The DDRFLDS call following a DDSETMR call will not be left before all fields in the result-item list are filled in. Only the fields in result-item list of DDSETMR are affected.

- -

```
                                        Input/Output
DDCLMR     USING       DDC-REF-TABLE,        (I/O)
                       DDS-realm-SUBSCHEMA,  (I)
                       SCV-realm,            (O)
                       FCSTATUS.             (O)
```

● **Clears "must-read" for fields/records.**

Description: The corresponding "clear-must-read" FOCUS call is performed for the fields in the result-item list.

Use: Normally this is used to reset the effect of a DDSETMR call. That is, reset the "set-must-read" for the same set of fields that is specified in the DDSETMR call.

```
                                  Input/Output
DDSETAT    USING   DDC-REF-TABLE,       (I/O)
                   DDS-realm-SUBSCHEMA,  (I)
                   SCV-realm,            (O)
                   IATRBT,               (I)
                   FCSTATUS.             (O)
```

- **Sets attributes.**

Description: This routine is used to set attributes (inverse video, blink etc.) on fields of a field set.

Use: The elements in the IATRBT integer array of 8 elements must be assigned value 1 to be enabled, otherwise 0 is assigned. If no elements are set, the "normal" attribute is set. Attributes can be combined. The attributes will appear the next time the fields are displayed.

Effect: The enabling of the different elements has the following meaning :

Element number	Effect on the fields
1	High (increased) intensity
2	Low (decreased) intensity
3	Italics
4	Underlined
5	Blink (slowly)
6	Blink (rapidly)
7	Inverse video
8	Invisible (password reading)

```
                                        Input/Output
DDCLAT    USING     DDC-REF-TABLE,         (I/O)
                    DDS-realm-SUBSCHEMA,   (I)
                    SCV-realm,             (O)
                    FCSTATUS.              (O)
```

- **Clears attributes.**

Description: This routine will give the result-item list the "normal" attribute set.

Use: Normally used to reset the same field set that is enabled by the DDSETAT call.

```
                            Input/Output
DDCOPTF    USING    IUNIT,         (I)
                    FCSTATUS.      (O)
```

- **Copies a displayed picture to a file.**

Description: The routine will write a form with leading texts and field contents to a file.

Use: The file should previously be opened by using the routine DDOPFI. The returned unit number from this routine should be used as input in DDCOPTF's IUNIT parameter.

```
                          Input/Output
DDOPFI   USING      IFINA,       (I)
                    IUNIT,       (O)
                    FCSTATUS.    (O)
```

- **Opens a SINTRAN file for Write, Append access.**

Description: Opens the file given by the name assigned to IFINA for Write/Append access. The file number of the opened file is returned in IUNIT.

Use: This routine is used to open for a call to DDCOPTF.

```
                   Input/Output
 DDCLFI   USING    IUNIT,      (I)
                   FCSTATUS.   (O)
```

● **Closes an opened file.**

Use: This routine is used to close files opened by the DDOPFI routine.

```
                   Input/Output
DDWMSGE   USING    MESSAGE,    (I)
                   FCSTATUS.   (O)
```

● **Writes a message to the message line.**

Description: The message in MESSAGE is displayed on the terminal's message line.

Use: See how to assign a value to the MESSAGE parameter in the parameter description in the beginning of this chapter.

```
                     Input/Output
DDCMSGE   USING    FCSTATUS.    (O)
```

● **Clears a message line.**

Description: This routine clears the terminal's message line.

```
                           Input/Output
DDGTEXT    USING    MESSAGE,   (I)
                    OTEXT,     (O)
                    ILINE,     (I)
                    ICOL,      (I)
                    FCSTATUS.  (O)
```

- **Writes a message in a given line and column and reads the answer.**

Description: The text in MESSAGE is displayed in position (ILINE, ICOL) on the operator's screen. It waits for a message from the operator. The message is returned in the OTEXT parameters.

Use: Assign the y/x-coordinate values to ILINE/ICOL for the start position of the message on the screen. Assign MESSAGE the message to be displayed. The return message will be found in OTEXT.

```
                                Input/Output
DDGTPIC    USING    FORMFILE,         (I)
                    DDC-REF-TABLE,   (I/O)
                    FCSTATUS.         (O)
```

- **Gets a picture from a file, displays and makes it ready.**

Assign the FOCUS form-file name to FORMFILE. Make sure that the SCC-PIC-NAME in DDC-REF-TABLE holds the correct form name (from the ASSDDC-<subfunction> file).

Description: The necessary ABM-FOCUS routines to initiate a form are called, and the form is displayed on the screen.

```
                          Input/Output
DDGMSGE   USING   MESSAGE,   (I)
                  OTEXT,     (O)
                  FCSTATUS.  (O)
```

● **Writes a message to a message line and reads the answer.**

Description: Works exactly as DDGTEXT, but the message line is always used.

```
                             Input/Output
DDERROR    USING   FCSTATUS,    (I/O)
                   MESSAGE.      (O)
```

● **Decodes error status and returns an error text.**

Use: After all DD calls, the status parameters should be tested.

- If DD<FOCUS-call> then test if parameter not = 0.
  If so, call DDERROR to decode the error situation.
- If DD<SIBAS-call> then test if parameter not = 1 or (sometimes) 0.
  If so, call DDERROR to decode the error.

> NOTE:
> See also the routine DDERMSG in ABM-UTILITY-LIB described in chapter 8.

Examples of use:

DDERROR
called after
SIBAS call.

```
CALL 'DDFTCH'  USING DBKI-DEPMENT-DEPNO,
                     DBKV-DEPMENT-DEPNO,
                     DBSTATUS.
IF DBSTATUS = 0 THEN
   <no item found, give message etc.>
ELSE-IF DBSTATUS NOT = 1 THEN
   CALL 'DDERROR' USING DBSTATUS,
                        TEXT-LINE
   IF DBSTATUS = 0 THEN
      <call routine to display TEXT-LINE>
   ELSE
      <call SDBEC>
   END-IF
END-IF
```

DDERROR
called after
"FOCUS"
call.

```
CALL 'DDRFLDS'  USING DDC-REF-TABLE,
                      DDS-R1-SUBSCHEMA,
                      SCV-R1,
                      FCSTATUS.
IF FCSTATUS NOT = 0 THEN
    CALL 'DDERROR'  USING FCSTATUS,
                          TEXT-LINE
    IF FCSTATUS NOT = 0 THEN
        <call routine to display TEXT-LINE>
    ELSE
        <call routine to display the .....>
    END-IF
END-IF
```

CHAPTER 8
HOW TO USE ABM-UTILITY-LIBRARY

8 HOW TO USE ABM-UTILITY-LIBRARY

8.1 ROUTINES AND PARAMETERS IN THE ABM-UTILITY-LIBRARY

Description of routines in ABM-UTILITY-LIB.

ABDBCLS	-	Closes the database.
ABDBOPN	-	Opens the database.
DDERMSG	-	Gives an error message.

The ABM-UTILITY-LIB routines:

Routines in this library	
ABDBCLS	(SIBAS status, database name)
ABDBOPN	(device number, database name)
DDERMSG	(STATUS)

These routines can be used in both FORTRAN and COBOL applications.

8.2 HOW TO USE ABM-UTILITY-LIBRARY

```
DDERMSG    (STATUS)
              0
```

● **Gives an error message.**

Description:

Use this routine from the B version of ABM, if you get an error status from a DDxxx-routine after a call to SIBAS or FOCUS.

DDERMSG uses the User-Environment, and requires some space. It is assumed that UE-ERMSG-xx-B is found on user system, or on the particular user from where the application is executed. xx indicates the language of the error message; for example, EN (English).

DDERMSG writes the error messages in the language given in User-Environment. Default language is English.

Example of use:

```
CALL "DDGET" USING TDBKEY,
                   DDC-SELECT,
                   DDB-UNIT-SUBSCHEMA,
                   DBV-UNIT,
                   DBSTATUS.
IF DBSTATUS < 0
     GO TO DD-ERROR
END-IF

- - - - -

CALL "DDRFLDS" USING DDC-REF-TABLE,
                     DDS-UNIT-SUBSCHEMA,
                     SCV-UNIT,
                     FCSTATUS.
IF FCSTATUS > 0
     GO TO DD-ERROR
END-IF

- - - - -

DD-ERROR SECTION.
IF FCSTATUS NOT = 0 THEN
     CALL 'DDERMSG' USING FCSTATUS
ELSE
     CALL 'DDERMSG' USING DBSTATUS
END-IF.
```

```
ABDBOPN   (device number, database name)
                 O             O
```

● **Opens the database.**

Description: This routine prompts for the SIBAS-system-number, database number, database name and password. ABDBOPN makes a "SETDEV", (SET DEVICE), and opens the database. Device number and database name are returned to the program called ABDBOPN.

```
ABDBCLS   (SIBAS status, database name)
                 O             O
```

● **Closes the database.**

Description: ABDBCLS closes the database when database name is given and the SIBAS status is zero.

DDINITE (0) is called.

CHAPTER 9
AN EXAMPLE OF USING ABM

9 AN EXAMPLE OF USING ABM

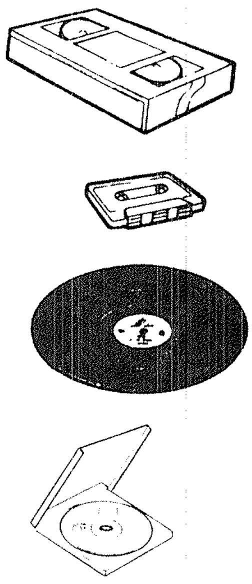
Below is an example showing how to make a user application program with ABM.

THE PROBLEM

A radio and television dealer wants a computerized system to keep track of Video Cassettes (VC), Music Cassettes (MC), Long Playing records (LP) and Compact Discs (CD).

SYSTEM SPECIFICATIONS

Start by making a loose specification of the data. Identify the following specifications:

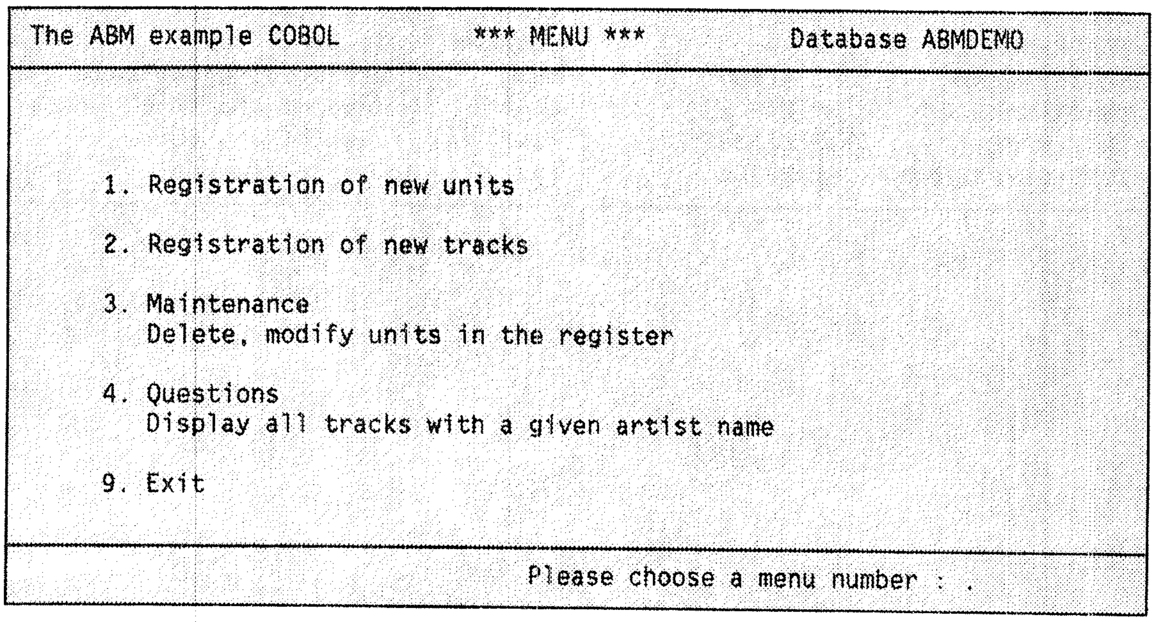


- A VC, MC , LP or a CD is called UNIT.
- Every UNIT will get a label with a TYPE and NUMBER identification.
- Each UNIT can have zero, one or more recorded TRACKS.
- A track must belong to at least one UNIT.
- Each track on a UNIT is identified by a TRACK-NUMBER and a SIDE identification.
- Each track can have an ARTIST NAME, a PERFORMANCE NAME, an ASSOCIATED NAME and a TYPE-OF-PERFORMANCE NAME.

HOW THE SYSTEM WILL BE OPERATED

Each UNIT will be registered in the system by an operator. Both operator name/initials and the date of registration will be stored.

On the following pages is an example of how to set up a system. The example contains four menus shown in this main menu:

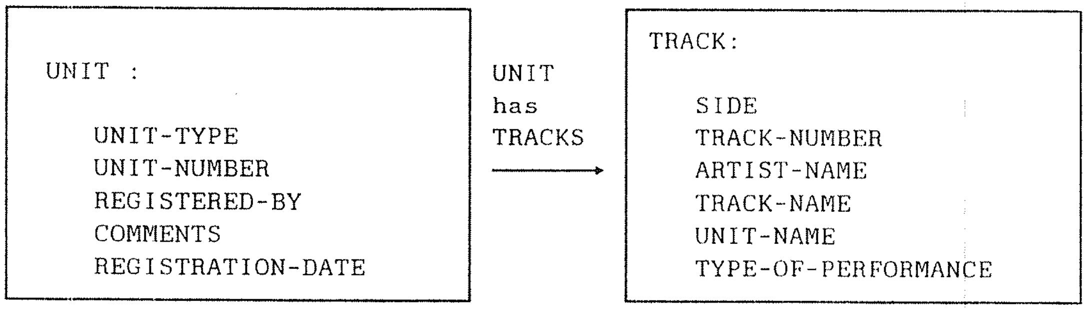


```
The ABM example COBOL          *** MENU ***          Database ABMDEMO


    1. Registration of new units

    2. Registration of new tracks

    3. Maintenance
       Delete, modify units in the register

    4. Questions
       Display all tracks with a given artist name

    9. Exit

                          Please choose a menu number : .
```

9.1 THE DATA MODEL

A data model is a formal description of data elements, made from your data specifications. You can translate the data model into a database.

UNIT specifications: **TRACK specifications:**



To represent the data items or <u>data descriptions</u> in the system we have to specify their format. The format of the data items indicate physical representation, display formats lengths etc. For our example we could use the following formats.

Data Descriptions

name	display	storage	comments
UNIT-TYPE	XX	text	LP, MC, VC or CD
UNIT-NUMBER	9999	integer	1 to 9999
REGISTERED-BY	X(4)	text	initials
COMMENTS	X(60)	text	free text
REGISTRATION-DATE	99'.'99'.'99	integer	year month day
SIDE	X	text	e.g.A,B or 1,2
TRACK-NUMBER	99	integer	1, 2, 3, ...
ARTIST-NAME	X(40)	text	artist or group name
TRACK-NAME	X(50)	text	name of the song, movie..
UNIT-NAME	X(50)	text	name of the cover
TYPE-OF-PERFORMANCE	X(30)	text	e.g. rock, pop

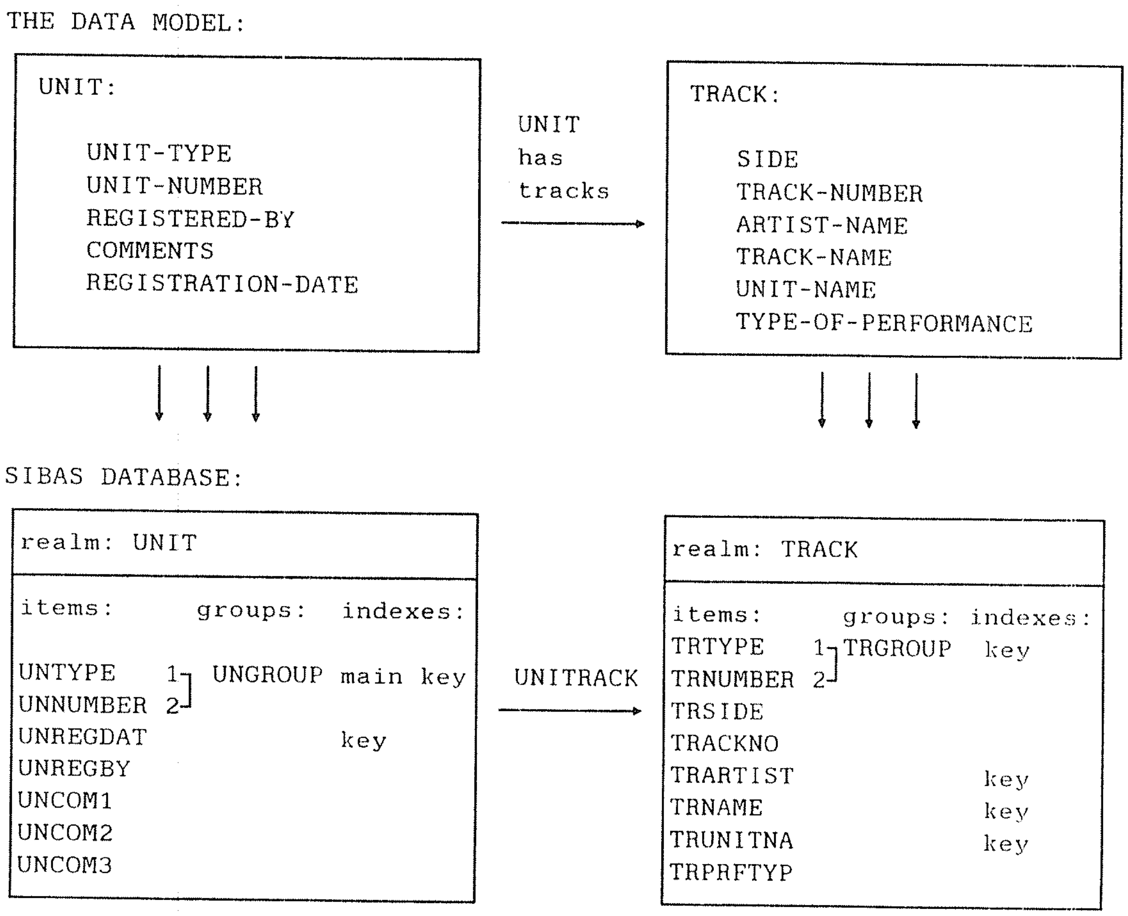
9.2 THE IMPLEMENTATION OF THE DATA MODEL

The datamodel can now be defined as a SIBAS database. The two main records UNIT and TRACK can de defined as SIBAS realms. The Data Description in each record can be used when defining SIBAS items. The relation "Unit has tracks" will become a SIBAS set.

DEFINE KEYS, INDEXES & GROUP ITEMS

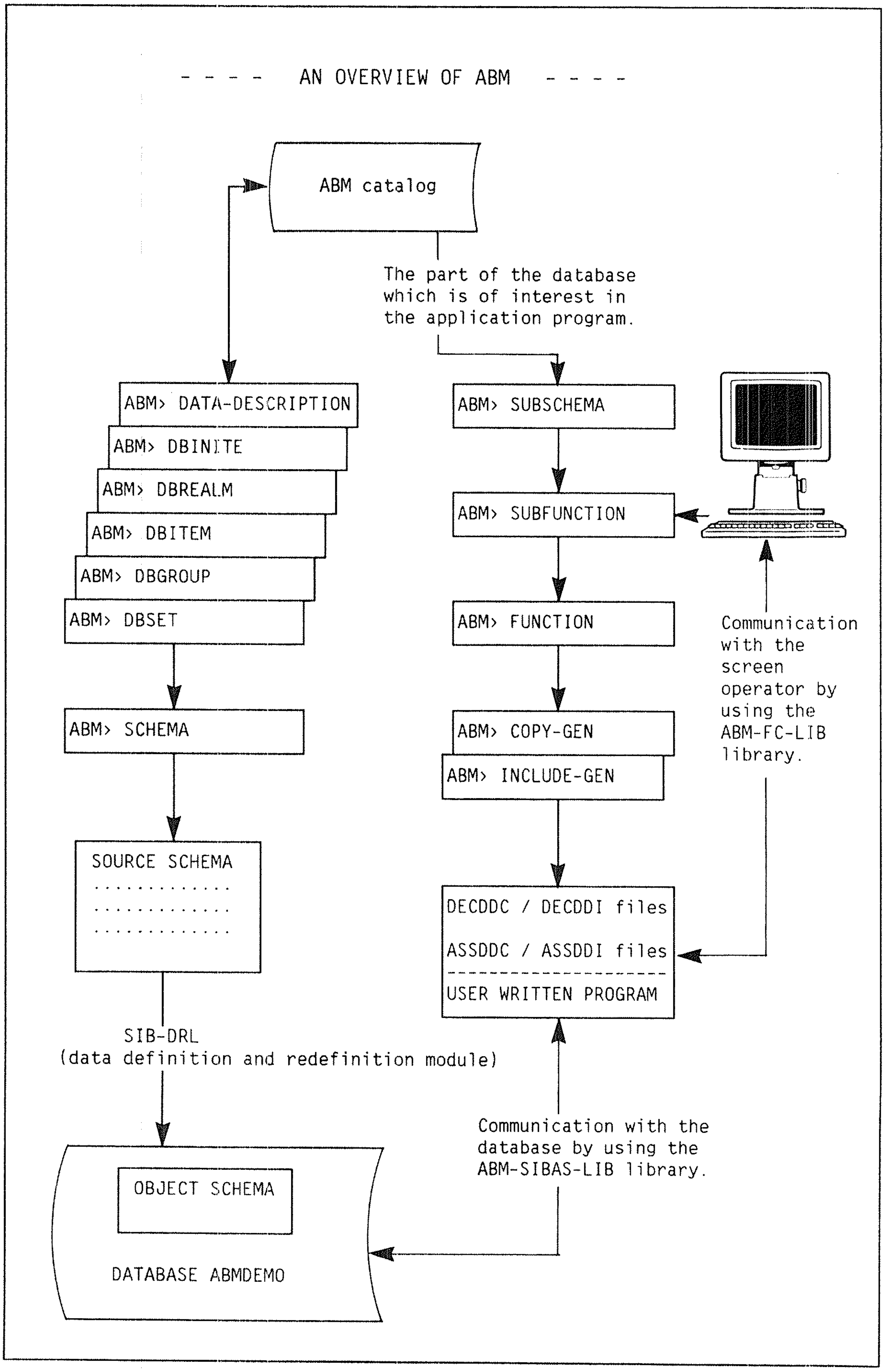
For accessing information quickly from the system, define specific items with keys or indexes. Assign a name to a collection of items in a record that have a close connection to each other, e.g. the data items UNIT-TYPE and UNIT-NUMBER define a unique Group Item for access to a specific UNIT.

The following, then, shows how the data model can be transferred to a model that can be directly specified in ABM based on SIBAS terms :



In the example shown on the previous page, the restricted naming conventions for FORTRAN are used when giving names to data descriptions, realms, items and group items. By following the naming conventions for FORTRAN, you can use the same database (ABMDEMO) for both the COBOL and FORTRAN example.

You can now define the SIBAS database using the DD- and DB- modules in ABM. When the database is defined, run the SCHEMA module. This will automatically produce a source schema for the database.

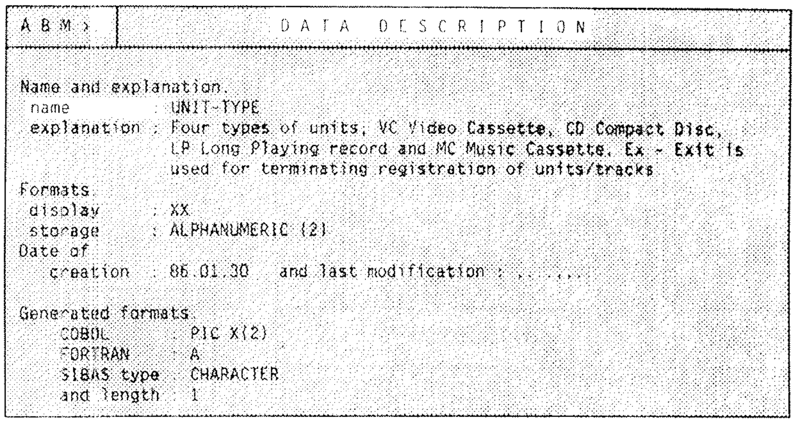

- - - - AN OVERVIEW OF ABM - - - -
ABM catalog
The part of the database which is of interest in the application program.
ABM> DATA-DESCRIPTION
ABM> DBINITE
ABM> DBREALM
ABM> DBITEM
ABM> DBGROUP
ABM> DBSET
ABM> SCHEMA
SOURCE SCHEMA
SIB-DRL
(data definition and redefinition module)
OBJECT SCHEMA
DATABASE ABMDEMO
ABM> SUBSCHEMA
ABM> SUBFUNCTION
ABM> FUNCTION
ABM> COPY-GEN
ABM> INCLUDE-GEN
DECDDC / DECDDI files
ASSDDC / ASSDDI files
USER WRITTEN PROGRAM
Communication with the screen operator by using the ABM-FC-LIB library.
Communication with the database by using the ABM-SIBAS-LIB library.

9.3 USING ABM

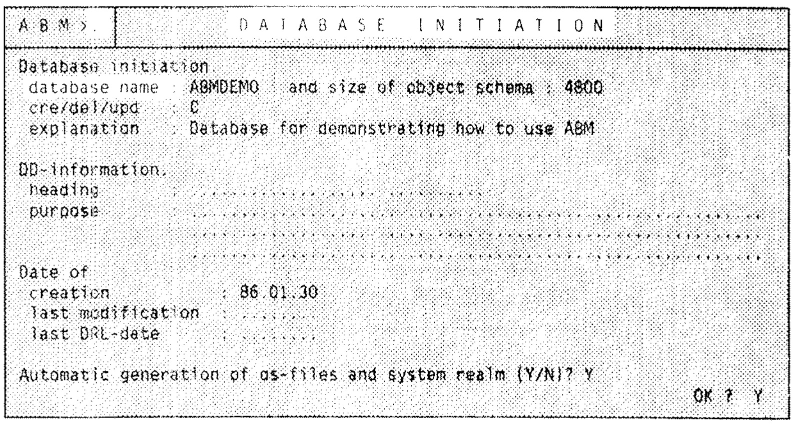
Below are copies of the screen forms used for defining the SIBAS database.

We use the ABM modules in the following order:

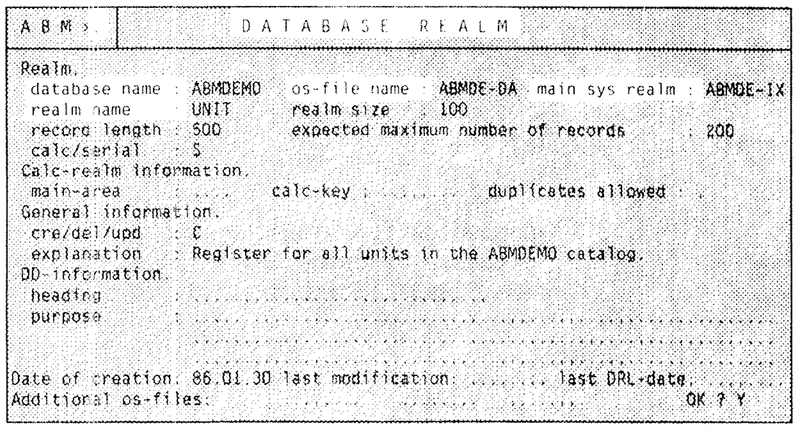
DATA-DESCRIPTIONS,
DBINITE,
DBREALM,
DBITEM,
DBGROUP,
DBSET,
SCHEMA,
SUBSCHEMA,
SUBFUNCTION,
FUNCTION
and COPY/INCLUDE.



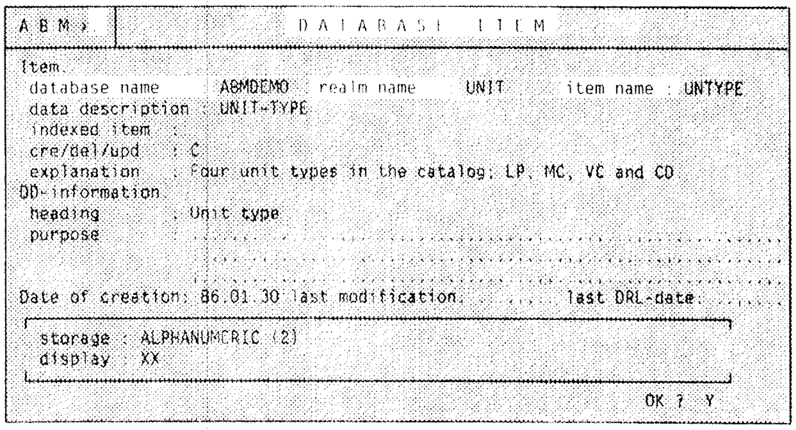
```
A B M >.          D A T A   D E S C R I P T I O N

Name and explanation.
 name          : UNIT-TYPE
 explanation : Four types of units; VC Video Cassette, CD Compact Disc,
               LP Long Playing record and MC Music Cassette. Ex - Exit is
               used for terminating registration of units/tracks
Formats.
 display      : XX
 storage      : ALPHANUMERIC (2)
Date of
   creation   : 86.01.30   and last modification : .......

Generated formats.
    COBOL        : PIC X(2)
    FORTRAN      : A
    SIBAS type   : CHARACTER
    and length   : 1
```



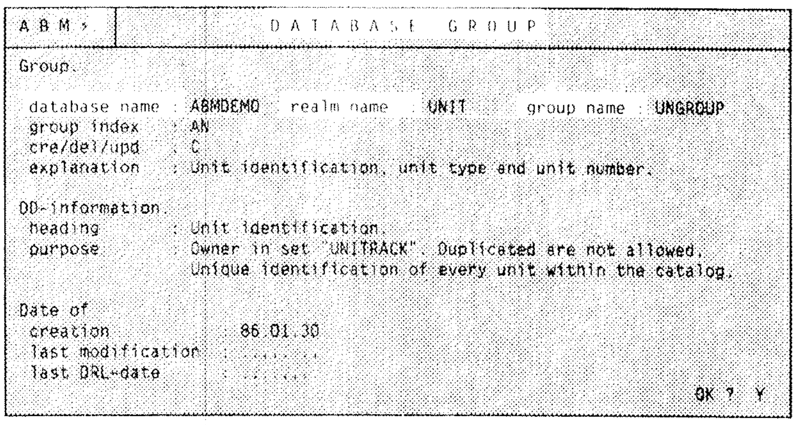
```
A B M >.          D A T A B A S E   I N I T I A T I O N
Database initiation
 database name  : ABMDEMO   and size of object schema : 4800
 cre/del/upd    : C
 explanation    : Database for demonstrating how to use ABM

DD-information.
 heading        : ...................................
 purpose        : ...........................................................
                  ...........................................................
                  ...........................................................
Date of
 creation           : 86.01.30
 last modification  : ........
 last DRL-date      : ........

Automatic generation of os-files and system realm (Y/N)? Y
                                                             OK ?  Y
```



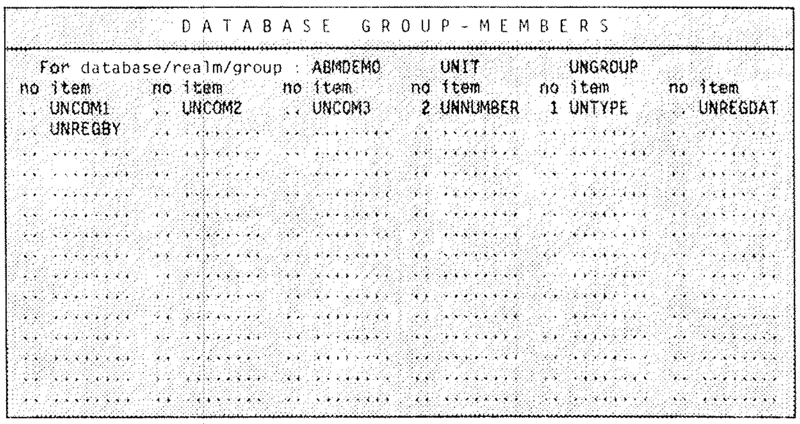
```
A B M >.          D A T A B A S E   R E A L M
Realm.
 database name : ABMDEMO  os-file name : ABMDE-DA  main sys realm : ABMDE-IX
 realm name    : UNIT     realm size   : 100
 record length : 500      expected maximum number of records      : 200
 calc/serial   : S
Calc-realm information.
 main-area     : ....   calc-key : ........   duplicates allowed : .
General information.
 cre/del/upd   : C
 explanation   : Register for all units in the ABMDEMO catalog.
DD-information.
 heading       : ...............................
 purpose       : ...........................................................
                 ...........................................................
                 ...........................................................
Date of creation: 86.01.30 last modification: ........ last DRL-date: ........
Additional os-files:        ........   ........   ........            OK ? Y
```



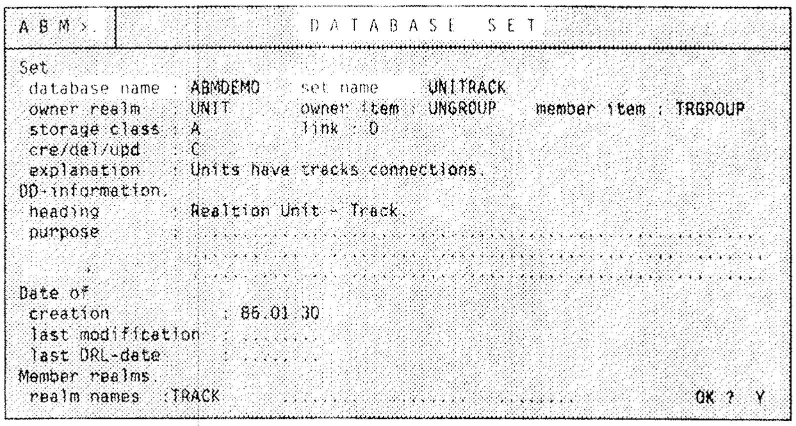
```
A B M >.                D A T A B A S E   I T E M
Item.
 database name    : ABMDEMO : realm name   : UNIT      item name : UNTYPE
 data description : UNIT-TYPE
 indexed item     :
 cre/del/upd      : C
 explanation      : Four unit types in the catalog: LP, MC, VC and CD
DD-information.
 heading          : Unit type
 purpose          : ..............................................
                    ..............................................
                    ..............................................
Date of creation: 86.01.30 last modification: ........ last DRL-date: ........
  storage : ALPHANUMERIC (2)
  display : XX
                                                              OK ?  Y
```



```
A B M >.                D A T A B A S E   G R O U P
Group.

 database name : ABMDEMO : realm name : UNIT      group name : UNGROUP
 group index   : AN
 cre/del/upd   : C
 explanation   : Unit identification, unit type and unit number.

DD-information.
 heading       : Unit identification.
 purpose       : Owner in set "UNITRACK". Duplicated are not allowed.
                 Unique identification of every unit within the catalog.

Date of
 creation          : 86.01.30
 last modification : ........
 last DRL-date     : ........
                                                              OK ?  Y
```

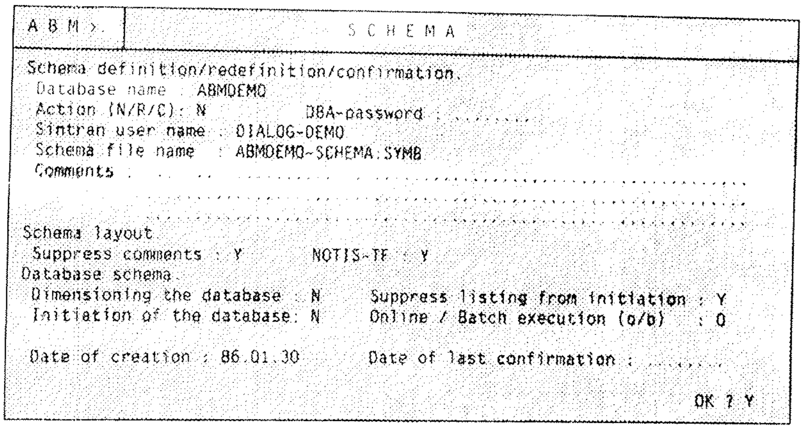


```
              D A T A B A S E   G R O U P - M E M B E R S

  For database/realm/group : ABMDEMO      UNIT        UNGROUP
no item       no item      no item      no item      no item      no item
.. UNCOM1     .. UNCOM2    .. UNCOM3     2 UNNUMBER   1 UNTYPE    .. UNREGDAT
.. UNREGBY    .. ........  .. ........  .. ........  .. ........  .. ........
.. ........   .. ........  .. ........  .. ........  .. ........  .. ........
.. ........   .. ........  .. ........  .. ........  .. ........  .. ........
.. ........   .. ........  .. ........  .. ........  .. ........  .. ........
.. ........   .. ........  .. ........  .. ........  .. ........  .. ........
.. ........   .. ........  .. ........  .. ........  .. ........  .. ........
.. ........   .. ........  .. ........  .. ........  .. ........  .. ........
.. ........   .. ........  .. ........  .. ........  .. ........  .. ........
.. ........   .. ........  .. ........  .. ........  .. ........  .. ........
.. ........   .. ........  .. ........  .. ........  .. ........  .. ........
.. ........   .. ........  .. ........  .. ........  .. ........  .. ........
.. ........   .. ........  .. ........  .. ........  .. ........  .. ........
.. ........   .. ........  .. ........  .. ........  .. ........  .. ........
.. ........   .. ........  .. ........  .. ........  .. ........  .. ........
```

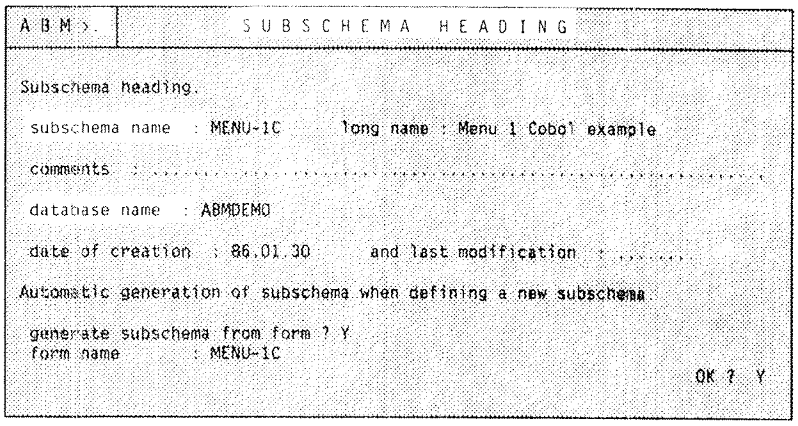


```
A B M >.                D A T A B A S E   S E T
Set.
 database name : ABMDEMO     set name   : UNITRACK
 owner realm   : UNIT        owner item : UNGROUP    member item : TRGROUP
 storage class : A           link : D
 cre/del/upd   : C
 explanation   : Units have tracks connections.
DD-information.
 heading       : Realtion Unit - Track.
 purpose       : ..............................................
                 ..............................................
               : ..............................................
Date of
 creation          : 86.01.30
 last modification : ........
 last DRL-date     : ........
Member realms.
 realm names   :TRACK   ........ ........ ........         OK ?  Y
```

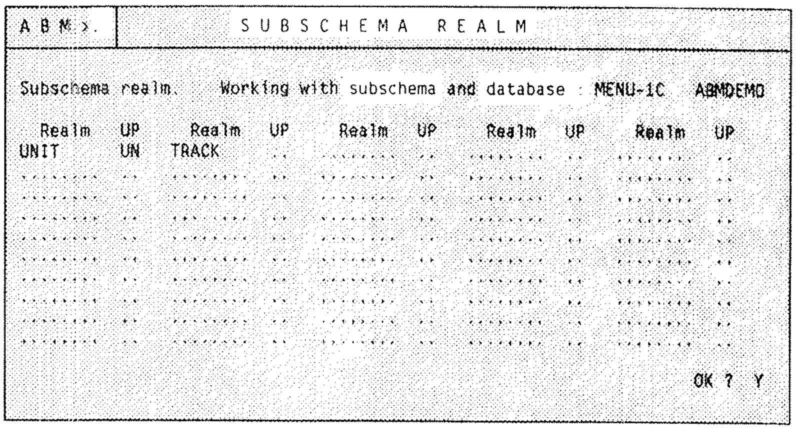
When the database is defined, run the SCHEMA module. This will produce a source schema. A copy of the source schema is shown in the next section.



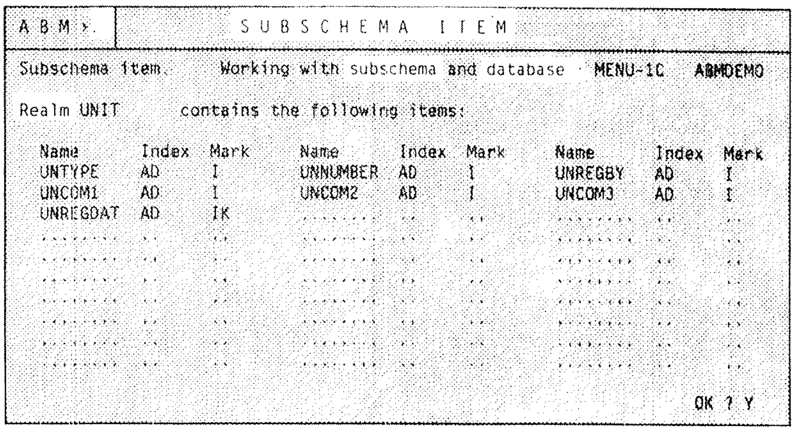
```
A B M >.                S C H E M A

Schema definition/redefinition/confirmation.
 Database name  : ABMDEMO
 Action (N/R/C): N          DBA-password : ........
 Sintran user name : DIALOG-DEMO
 Schema file name  : ABMDEMO-SCHEMA:SYMB
 Comments : ..............................................
         ..............................................
         ..............................................
Schema layout.
 Suppress comments : Y       NOTIS-TF : Y
Database schema.
 Dimensioning the database : N     Suppress listing from initiation : Y
 Initiation of the database: N     Online / Batch execution (o/b)   : O

 Date of creation : 86.01.30       Date of last confirmation : ........

                                                                  OK ? Y
```



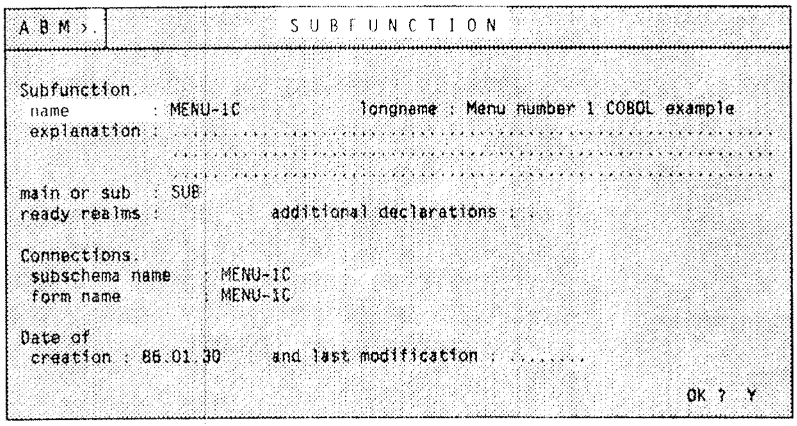
```
A B M >.        S U B S C H E M A   H E A D I N G

Subschema heading.

 subschema name   : MENU-1C      long name : Menu 1 Cobol example

 comments  : ...............................................................

 database name   : ABMDEMO

 date of creation  : 86.01.30      and last modification  : ........

Automatic generation of subschema when defining a new subschema.

 generate subschema from form ? Y
 form name        : MENU-1C
                                                                  OK ?  Y
```



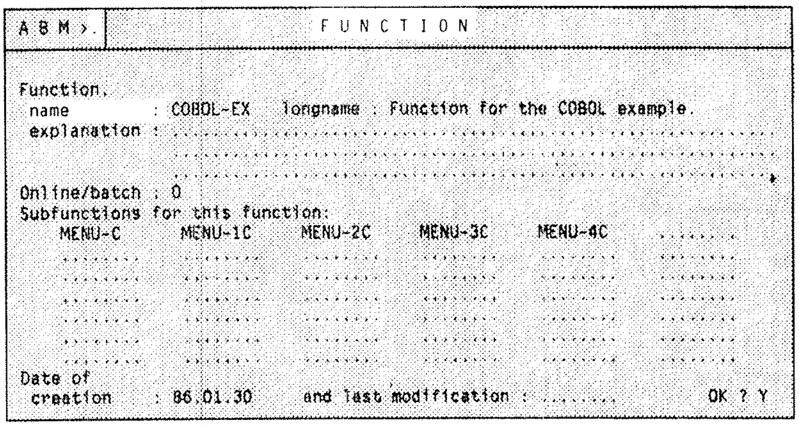
```
A B M >.        S U B S C H E M A   R E A L M

Subschema realm.   Working with subschema and database : MENU-1C   ABMDEMO

  Realm   UP    Realm   UP    Realm   UP    Realm   UP    Realm   UP
UNIT      UN   TRACK    ..   ........ ..   ........ ..   ........ ..
........  ..   ........ ..   ........ ..   ........ ..   ........ ..
........  ..   ........ ..   ........ ..   ........ ..   ........ ..
........  ..   ........ ..   ........ ..   ........ ..   ........ ..
........  ..   ........ ..   ........ ..   ........ ..   ........ ..
........  ..   ........ ..   ........ ..   ........ ..   ........ ..
........  ..   ........ ..   ........ ..   ........ ..   ........ ..
........  ..   ........ ..   ........ ..   ........ ..   ........ ..
........  ..   ........ ..   ........ ..   ........ ..   ........ ..
........  ..   ........ ..   ........ ..   ........ ..   ........ ..

                                                                  OK ?  Y
```



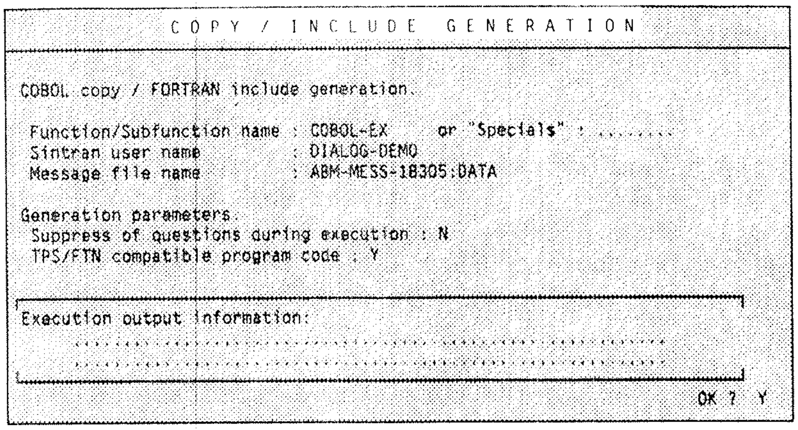
```
A B M >.        S U B S C H E M A   I T E M
Subschema item.   Working with subschema and database : MENU-1C   ABMDEMO

Realm UNIT      contains the following items:

  Name      Index  Mark     Name      Index  Mark     Name      Index  Mark
  UNTYPE    AD     I        UNNUMBER  AD     I        UNREGBY   AD     I
  UNCOM1    AD     I        UNCOM2    AD     I        UNCOM3    AD     I
  UNREGDAT  AD     IK       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..
  ........  ..     ..       ........  ..     ..       ........  ..     ..

                                                                  OK ? Y
```



```
A B M >.              S U B F U N C T I O N

Subfunction.
 name          : MENU-1C          longname : Menu number 1 COBOL example
 explanation : ..........................................................
               ..........................................................
               ..........................................................
main or sub   : SUB
ready realms :          additional declarations : .

Connections.
 subschema name    : MENU-1C
 form name         : MENU-1C

Date of
 creation : 86.01.30     and last modification : ........
                                                              OK ? Y
```



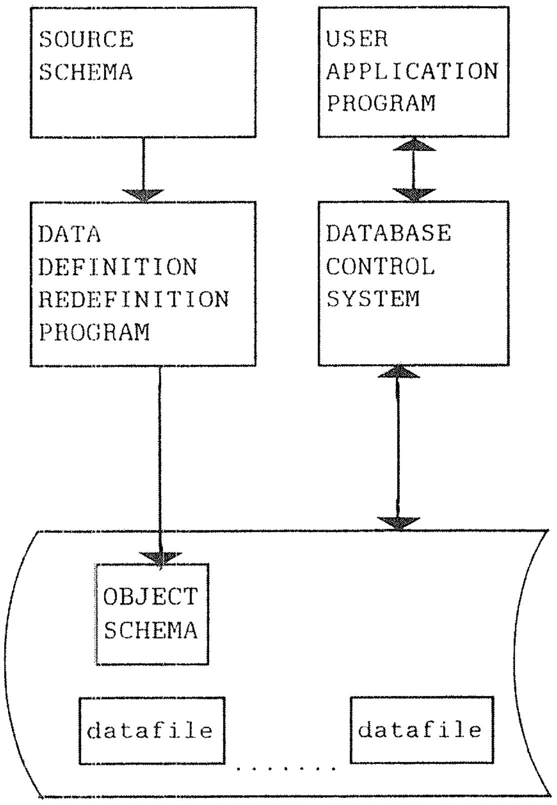
```
A B M >.              F U N C T I O N

Function.
 name          : COBOL-EX   longname : Function for the COBOL example.
 explanation : ..........................................................
               ..........................................................
               ..........................................................
Online/batch : O
Subfunctions for this function:
    MENU-C      MENU-1C     MENU-2C     MENU-3C     MENU-4C     ........
    ........    ........    ........    ........    ........    ........
    ........    ........    ........    ........    ........    ........
    ........    ........    ........    ........    ........    ........
    ........    ........    ........    ........    ........    ........
    ........    ........    ........    ........    ........    ........
    ........    ........    ........    ........    ........    ........
Date of
 creation     : 86.01.30     and last modification : ........      OK ? Y
```



```
          C O P Y  /  I N C L U D E   G E N E R A T I O N

COBOL copy / FORTRAN include generation.

 Function/Subfunction name : COBOL-EX     or "Specials" : ........
 Sintran user name         : DIALOG-DEMO
 Message file name         : ABM-MESS-18305:DATA

Generation parameters.
 Suppress of questions during execution : N
 TPS/FTN compatible program code : Y

Execution output information:
     ............................................................
     ............................................................
                                                              OK ? Y
```

9.4 SOURCE SCHEMA FOR THE SAMPLE DATABASE

A schema is a collection of all records, indexes, set types, and realms in a database.



Each database has a corresponding source schema. A source schema can be translated into an object schema (database description) by using the schema translator SIB-DRL.

The schema listing will serve as documentation for the contents of your database.

THE MAIN COMPONENTS OF A DATABASE

To deside the layout of the schema listing in the SCHEMA picture, you can answer Y (Yes) in the NOTIS-TF field. NOTIS-TF directives will then be added into the output file from SCHEMA. (You can use the output file containing NOTIS-TF directives as input to DRL.) When this output file is processed by NOTIS-TF, your schema listing will be formatted and a table of contents will be included.

A formatted listing of the source schema for our sample database is shown on the following pages.

TABLE OF CONTENTS

2

```
 *
 ***********************************************************************
 *            Schema generated by ABM.    86-04-29  10:48              *
 ***********************************************************************
 *
 START INITIATION DATABASE ABMDEMO
    SIZE  4800.

 *
 *
```

1 OS-FILE DECLARATION

```
 *
 *------------------------------------------------------------------------ *
 NEW OS-FILE ABMDE-DA  PAGESIZE   512.
 *
 NEW OS-FILE ABMDE-IX  PAGESIZE   512.
 *
 NEW SYSTEM-REALM ABMDE-IX OS-FILE ABMDE-IX  REALMSIZE  100.

 *
 *
```

3

```
 *
```

2 REALM DECLARATION

```
 *
 *
```

2.1 UNIT

```
*
*
**********************************************************************
*                                       U N I T                      *
**********************************************************************
*
NEW SERIAL-REALM   UNIT
   OS-FILE       ABMDE-DA
   REALMSIZE        100
   REC LENGTH       500
   MAIN          ABMDE-IX.

*
*------------------------------------------------------------------- *
NEW ITEM UNIT       UNCOM1     TYPE  CHARACTER    LENGTH     30
   STORAGE   "ALPHANUMERIC(60)"
   DISPLAY   "X(60)"
   HEADING   "Comments, first line.".

* *
NEW ITEM UNIT       UNCOM2     TYPE  CHARACTER    LENGTH     30
   STORAGE   "ALPHANUMERIC(60)"
   DISPLAY   "X(60)"
   HEADING   "Comments, second line.".

* *
NEW ITEM UNIT       UNCOM3     TYPE  CHARACTER    LENGTH     30
   STORAGE   "ALPHANUMERIC(60)"
   DISPLAY   "X(60)"
   HEADING   "Comments, third line.".

* *
NEW ITEM UNIT       UNNUMBER   TYPE  INTEGER      LENGTH      1
   STORAGE   "INTEGER2"
   DISPLAY   "9999"
   HEADING   "Unit number.".

* *
NEW ITEM UNIT       UNTYPE     TYPE  CHARACTER    LENGTH      1
   STORAGE   "ALPHANUMERIC(2)"
   DISPLAY   "XX"
   HEADING   "Unit type.".

* *
NEW ITEM UNIT       UNREGDAT   TYPE  INTEGER      LENGTH      2
   STORAGE   "INTEGER4"
   DISPLAY   "99'.'99'.'99"
   HEADING   "Registration date.".

* *
NEW ITEM UNIT       UNREGBY    TYPE  CHARACTER    LENGTH      2
   STORAGE   "ALPHANUMERIC(4)"
   DISPLAY   "X(4)"
   HEADING   "The operators initials.".
```

5

```
*
*-----------------------------------------------------------------
*
```

2.1.1 GROUP DECLARATION FOR UNIT

```
 * *
NEW GROUP UNIT     UNGROUP
                          UNTYPE
                          UNNUMBER
   HEADING  "Unit identification."
   PURPOSE
       "/Owner in set 'UNITRACK'. Duplicates not allowed."
       "/Unique identification of an unit in the catalog.".

 *
 *
```

2.1.2 ITEM INDEX DECLARATION FOR UNIT

```
 *
 *-----------------------------------------------------------------  *
 *
NEW INDEX UNIT     UNREGDAT
   UPDATE IS AUTOMATIC DUPLICATES ARE     ALLOWED
   SYSTEM-REALM ABMDE-IX.
 *

 *
```

2.1.3 GROUP INDEX DECLARATION FOR UNIT

```
 *
 *
NEW INDEX UNIT     UNGROUP
   UPDATE IS AUTOMATIC DUPLICATES ARE NOT ALLOWED
   SYSTEM-REALM ABMDE-IX.
 *
```

6

2.2 TRACK

```
*
*****************************************************************
*                                       T R A C K               *
*****************************************************************
*
NEW SERIAL-REALM   TRACK
   OS-FILE        ABMDE-DA
   REALMSIZE        100
   REC LENGTH       500
   MAIN           ABMDE-IX.

*
*--------------------------------------------------------------- *
NEW ITEM TRACK      TRTYPE    TYPE  CHARACTER   LENGTH       1
   STORAGE  "ALPHANUMERIC(2)"
   DISPLAY  "XX"
   HEADING  "Unit type.".

* *
NEW ITEM TRACK      TRNUMBER  TYPE  INTEGER     LENGTH       1
   STORAGE  "INTEGER2"
   DISPLAY  "9999"
   HEADING  "Unit number".

* *
NEW ITEM TRACK      TRSIDE    TYPE  CHARACTER   LENGTH       1
   STORAGE  "ALPHANUMERIC(1)"
   DISPLAY  "X"
   HEADING  "Side identification.".

* *
NEW ITEM TRACK      TRARTIST  TYPE  CHARACTER   LENGTH      20
   STORAGE  "ALPHANUMERIC(40)"
   DISPLAY  "X(40)"
   HEADING  "Artist name.".

* *
NEW ITEM TRACK      TRACKNO   TYPE  INTEGER     LENGTH       1
   STORAGE  "INTEGER2"
   DISPLAY  "99"
   HEADING  "Track number".

* *
NEW ITEM TRACK      TRPRFTYP  TYPE  CHARACTER   LENGTH      15
   STORAGE  "ALPHANUMERIC(30)"
   DISPLAY  "X(30)"
   HEADING  "Type of performance.".

* *
NEW ITEM TRACK      TRUNITNA  TYPE  CHARACTER   LENGTH      25
   STORAGE  "ALPHANUMERIC(50)"
   DISPLAY  "X(50)"
   HEADING  "Unit name on a track.".
```

```
                                                                    7
 * *
 NEW ITEM TRACK      TRNAME     TYPE  CHARACTER   LENGTH      25
    STORAGE  "ALPHANUMERIC(50)"
    DISPLAY  "X(50)"
    HEADING  "Name of performance.".

 *
 *-----------------------------------------------------------------
 *
```

2.2.1 GROUP DECLARATION FOR TRACK

```
 * *
 NEW GROUP TRACK     TRGROUP
                             TRTYPE
                             TRNUMBER
    HEADING  "Unit identification."
    PURPOSE
        "/Member in set 'UNITRACK'. Duplicates allowed.".

 *
 *
```

2.2.2 ITEM INDEX DECLARATION FOR TRACK

```
 *
 *----------------------------------------------------------------- *
 *
 NEW INDEX TRACK     TRARTIST
    UPDATE IS AUTOMATIC DUPLICATES ARE     ALLOWED
    SYSTEM-REALM ABMDE-IX.
 *
 NEW INDEX TRACK     TRUNITNA
    UPDATE IS AUTOMATIC DUPLICATES ARE     ALLOWED
    SYSTEM-REALM ABMDE-IX.
 *
 NEW INDEX TRACK     TRNAME
    UPDATE IS AUTOMATIC DUPLICATES ARE     ALLOWED
    SYSTEM-REALM ABMDE-IX.

 *
```

2.2.3 GROUP INDEX DECLARATION FOR TRACK

```
 *
 *
 NEW INDEX TRACK     TRGROUP
    UPDATE IS AUTOMATIC DUPLICATES ARE     ALLOWED
    SYSTEM-REALM ABMDE-IX.
 *
```

8

3 SET DECLARATION

```
* *
*************************************************************************
*                                   S E T   d e f i n i t i o n s   *
*************************************************************************
*
NEW SET                 UNITRACK
   LINK IS              DOUBLE
   STORAGE-CLASS IS AUTOMATIC
   OWNER                UNGROUP  UNIT
   MEMBER               TRGROUP  TRACK
   HEADING  "Relation Unit - Track.".

*END.
*************************************************************************
* End of Schema.                                                        *
*************************************************************************
```

9.5 REPORT OF THE SAMPLE DATABASE

A report of the database is generated by using the command REPORT from the main menu. An ABM report will typically include information about all data descriptions: which ones are used, where they are used, and which ones are not used.

The following is the report for our sample database.

```
 A B M - R e p o r t : DATA  DESCRIPTIONS
                                 Date : 86.03.12   Time : 12:36

     All Data Descriptions are listed in alphabetic order.

     - CREDATE  : Date of Creation.
     - MODDATE  : Date of last Modification.
     - STORAGE  : Standard Storage Format.
     - DISPLAY  : Standard Display Format.
     - COBFORM  : Cobol Format.
     - FORFORM  : Fortran Format.
     - SITYPE   : Sibas Item Type.
     - SILENG   : Sibas Item Length.
     - COMMENT  : Explanation/Text, not DD-info.


    (*)  This report is for
         all Data Descriptions
-----------------------------------------------------------------------
ABM-REPORT: DATA DESCRIPTIONS    Date : 86.03.12  Time : 12:36   Page :  1
-----------------------------------------------------------------------

Data Description Name : ARTIST-NAME                  Credate : 86.01.30
   Storage : ALPHANUMERIC(40)
   Display : X(40)
   CobForm : PIC   X(40).                           ForForm : A
   SiType  : CHARACTER                              SiLeng  :    20
   Comment : Full name of the artist and/or group name.
-----------------------------------------------------------------------
Data Description Name : COMMENTS                     Credate : 86.01.30
   Storage : ALPHANUMERIC(60)
   Display : X(60)
   CobForm : PIC   X(60).                           ForForm : A
   SiType  : CHARACTER                              SiLeng  :    30
   Comment : Free text of 60 characters used for explanation.
-----------------------------------------------------------------------
Data Description Name : NUMBER                       Credate : 86.01.30
   Storage : INTEGER2
   Display : 9
   CobForm : PIC  S9(04)     COMP.                  ForForm : I
   SiType  : INTEGER                                SiLeng  :     1
   Comment : Is used when choosing a menu number.
-----------------------------------------------------------------------
```

```
Data Description Name : OK1                          Credate : 86.01.30
   Storage : ALPHANUMERIC(1)
   Display : X
   CobForm : PIC   X(1).                           ForForm : A
   SiType  : CHARACTER                             SiLeng  :    1
   Comment : Data descriptions which is referred in picture fields only
   Comment : Must not be written like OK-1 if you use Fortran.
   Comment : The character '-' will lead to a compiler error.
--------------------------------------------------------------------------
Data Description Name : OK2                          Credate : 86.01.30
   Storage : ALPHANUMERIC(1)
   Display : X
   CobForm : PIC   X(1).                           ForForm : A
   SiType  : CHARACTER                             SiLeng  :    1
   Comment : A data description which is referred in picture fields only
   Comment : Must not be referenced twice in the same picture.
   Comment : OK2 is used in Menu-4.
--------------------------------------------------------------------------
Data Description Name : REGISTERED-BY                Credate : 86.02.06
   Storage : ALPHANUMERIC(4)
   Display : X(4)
   CobForm : PIC   X(4).                           ForForm : A
   SiType  : CHARACTER                             SiLeng  :    2
   Comment : The initials to the person who registrates a unit.
--------------------------------------------------------------------------
Data Description Name : REGISTRATION-DATE            Credate : 86.02.06
   Storage : INTEGER4
   Display : 99'.'99'.'99
   CobForm : PIC   S9(10)      COMP.               ForForm : I
   SiType  : INTEGER                               SiLeng  :    2
   Comment : The sequence is year, month and day.
--------------------------------------------------------------------------


--------------------------------------------------------------------------
ABM-REPORT: DATA  DESCRIPTIONS    Date : 86.03.12  Time : 12:36  Page :   2
--------------------------------------------------------------------------

Data Description Name : SIDE                         Credate : 86.01.30
                                                     Moddate : 86.03.12
   Storage : ALPHANUMERIC(1)
   Display : X
   CobForm : PIC   X(1).                           ForForm : A
   SiType  : CHARACTER                             SiLeng  :    1
   Comment : Side of the unit. LPs and MCs have two sides,
   Comment : CDs and VCs only one.
   Comment : It can be labeled 1 and 2, or A and B.
--------------------------------------------------------------------------
Data Description Name : TRACK-NAME                   Credate : 86.02.06
                                                     Moddate : 86.03.12
   Storage : ALPHANUMERIC(50)
   Display : X(50)
   CobForm : PIC   X(50).                          ForForm : A
   SiType  : CHARACTER                             SiLeng  :   25
   Comment : On LPs, MCs and CDs this will tyically be the name of the
   Comment : song or act performed. For VCs it can e.g. be the name of
   Comment : the act, movie, title etc.
--------------------------------------------------------------------------
```

```
Data Description Name : TRACK-NUMBER                    Credate : 86.01.30
   Storage : INTEGER2
   Display : 99
   CobForm : PIC  S9(04)     COMP.                 ForForm : I
   SiType  : INTEGER                               SiLeng  :    1
   Comment : On one SIDE of an unit you normally will find many tracks.
   Comment : The tracks are numbered 1,2,3, etc.
-------------------------------------------------------------------------
Data Description Name : TYPE-OF-PERFORMANCE             Credate : 86.01.30
                                                        Moddate : 86.02.06
   Storage : ALPHANUMERIC(30)
   Display : X(30)
   CobForm : PIC   X(30).                          ForForm : A
   SiType  : CHARACTER                             SiLeng  :   15
   Comment : The performances can be classified as e.g.
   Comment : Rock, Classic, Romantic, Horrors etc.
-------------------------------------------------------------------------
Data Description Name : UNIT-NAME                       Credate : 86.01.30
                                                        Moddate : 86.03.12
   Storage : ALPHANUMERIC(50)
   Display : X(50)
   CobForm : PIC   X(50).                          ForForm : A
   SiType  : CHARACTER                             SiLeng  :   25
   Comment : For LPs, MCs and CDs this will normally be the cover name.
   Comment : For VCs it will be the cover name, title of the serie etc.
-------------------------------------------------------------------------
Data Description Name : UNIT-NUMBER                     Credate : 86.01.30
   Storage : INTEGER2
   Display : 9999
   CobForm : PIC  S9(04)     COMP.                 ForForm : I
   SiType  : INTEGER                               SiLeng  :    1
   Comment : An unit type and number identifies uniqely an unit.
   Comment : Unit numbers are assigned from 1 and up for each unit type.
-------------------------------------------------------------------------


-------------------------------------------------------------------------
ABM-REPORT: DATA  DESCRIPTIONS    Date : 86.03.12  Time : 12:36  Page :  3
-------------------------------------------------------------------------

Data Description Name : UNIT-TYPE                       Credate : 86.01.30
                                                        Moddate : 86.03.12
   Storage : ALPHANUMERIC(2)
   Display : XX
   CobForm : PIC   X(2).                           ForForm : A
   SiType  : CHARACTER                             SiLeng  :    1
   Comment : 4 types of units. VC - Video Cassette, CD - Compact Disc,
   Comment : LP - Long Playing record and MC - Music cassette.
   Comment : EX - Exit is used for terminate registration of units/tracks
-------------------------------------------------------------------------
```

9.6 THE COBOL COPY FILE

The COBOL COPY file contains values of items and fields in the database. This makes it especially easy to make application programs. An application program need only call the COPY file (DECDDC-name:SYMB), and the values of all items and fields will be automatically available.

The following is the listing of the COPY file generated for our sample database:

THE DECLARATIONS

```
*
******************************************************************
* ABM /DECDDC-MENU-1C  /       * Generated : 86.03.11      15:33
******************************************************************
*
*
* ------------------------------* REFERANCE TABLE DECLARATION.
*
     03  DDC-REF-TABLE.
         05  SCC-PIC-NAME               PIC  X(8).
         05  SCC-READ-MODE              PIC  9(4) COMP.
         05  SCC-WRITE-MODE             PIC  9(4) COMP.
         05  SCC-START-RW-LINE          PIC  9(4) COMP.
         05  SCC-RW-NO-OF-LINES         PIC  9(4) COMP.
         05  SCR-NO-OF-LINES-READ       PIC  9(4) COMP.
         05  SCR-TERM-CHAR              PIC  9(4) COMP.
         05  DDC-SELECT.
             07  DDC-TYPE               PIC  X(2).
             07  DDC-ITEM-LIST.
                 09  DDC-ITEM           PIC  X(8) OCCURS   5.
*
* ------------------------------* R1        REALM  DECLARATION.
*
     03  DDS-R1-SUBSCHEMA.
         05  DDS-R1-NO-OF-ITEMS         PIC  9(4) COMP.
         05  DDS-R1-TOT-ITEM-LEN        PIC  9(4) COMP.
         05  DDS-R1-NO-OF-RECORDS       PIC  9(4) COMP.
         05  DDS-R1-FIRST-ITEM-NO       PIC  9(4) COMP.
         05  DDS-R1-FIRST-WORD-NO       PIC  9(4) COMP.
         05  FILLER                     PIC  9(4) COMP.
         05  DDS-R1-RECORD-NAME         PIC  X(8).
         05  DDS-R1-ITEMS.
             07  DDS-R1-ITEM-NAME       PIC X(8)       OCCURS   7.
         05  DDS-R1-ITEM-LEN            PIC 9(4) COMP OCCURS   7.
         05  DDS-R1-ITEM-TYPE           PIC X(2)       OCCURS   7.
*
*
```

```
      *
            03  SCV-R1.
                05  SCV-R1-UNTYPE               PIC   X(2).
                05  SCV-R1-UNNUMBER             PIC  S9(04)      COMP.
                05  SCV-R1-UNREGBY              PIC   X(4).
                05  SCV-R1-UNCOM1               PIC   X(60).
                05  SCV-R1-UNCOM2               PIC   X(60).
                05  SCV-R1-UNCOM3               PIC   X(60).
                05  SCV-R1-UNREGDAT             PIC  S9(10)      COMP.
      *
      * ------------------------------* R2       REALM  DECLARATION.
      *
            03  DDS-R2-SUBSCHEMA.
                05  DDS-R2-NO-OF-ITEMS          PIC  9(4) COMP.
                05  DDS-R2-TOT-ITEM-LEN         PIC  9(4) COMP.
                05  DDS-R2-NO-OF-RECORDS        PIC  9(4) COMP.
                05  DDS-R2-FIRST-ITEM-NO        PIC  9(4) COMP.
                05  DDS-R2-FIRST-WORD-NO        PIC  9(4) COMP.
                05  FILLER                      PIC  9(4) COMP.
                05  DDS-R2-RECORD-NAME          PIC  X(8).
                05  DDS-R2-ITEMS.
                    07  DDS-R2-ITEM-NAME        PIC X(8)       OCCURS   1.
                05  DDS-R2-ITEM-LEN             PIC 9(4) COMP OCCURS   1.
                05  DDS-R2-ITEM-TYPE            PIC X(2)       OCCURS   1.
      *
      *
      *
            03  SCV-R2.
                05  SCV-R2-OK1                  PIC   X(1).
                05  FILLER                      PIC   X.
      *
      * ------------------------------* UNIT     REALM  DECLARATION.
      *
            03  DDB-UNIT-SUBSCHEMA.
                05  DDB-UNIT-NO-OF-ITEMS        PIC  9(4) COMP.
                05  DDB-UNIT-TOT-ITEM-LEN       PIC  9(4) COMP.
                05  FILLER                      PIC  X(8).
                05  DDB-UNIT-RECORD-NAME        PIC  X(8).
                05  DDB-UNIT-ITEMS.
                    07  DDB-UNIT-ITEM-NAME      PIC X(8)       OCCURS   7.
                05  DDB-UNIT-ITEM-LEN           PIC 9(4) COMP OCCURS   7.
                05  DDB-UNIT-ITEM-TYPE          PIC X(2)       OCCURS   7.
      *
      *
      *
            03  DBV-UNIT.
                05  DBV-UNIT-UNNUMBER           PIC  S9(04)      COMP.
                05  DBV-UNIT-UNTYPE             PIC   X(2).
                05  DBV-UNIT-UNREGDAT           PIC  S9(10)      COMP.
                05  DBV-UNIT-UNREGBY            PIC   X(4).
                05  DBV-UNIT-UNCOM1             PIC   X(60).
                05  DBV-UNIT-UNCOM2             PIC   X(60).
                05  DBV-UNIT-UNCOM3             PIC   X(60).
      *
```

```
* ------------------------------* UNIT     INDEX DECLARATIONS.
*
     03  DBKI-UNIT-UNREGDAT.
         05  FILLER                        PIC  X(2).
         05  DBKI-UNIT-UNREGDAT-LEN        PIC  9(4) COMP.
         05  DBKI-UNIT-UNREGDAT-KEY-NAM    PIC  X(8).
         05  DBKI-UNIT-UNREGDAT-RLM-NAM    PIC  X(8).
*
     03  DBKV-UNIT-UNREGDAT.
         05  DBKV-UNIT-UNREGDAT-LOW-1      PIC  S9(10)
                                       COMP.
         05  DBKV-UNIT-UNREGDAT-HIGH-1     PIC  S9(10)
                                       COMP.
*
*
*
     03  DBKI-UNIT-UNGROUP.
         05  FILLER                        PIC  X(2).
         05  DBKI-UNIT-UNGROUP-LEN         PIC  9(4) COMP.
         05  DBKI-UNIT-UNGROUP-KEY-NAM     PIC  X(8).
         05  DBKI-UNIT-UNGROUP-RLM-NAM     PIC  X(8).
*
     03  DBKV-UNIT-UNGROUP.
         05  DBKV-UNIT-UNGROUP-LOW-1       PIC   X(2).
         05  DBKV-UNIT-UNGROUP-LOW-2       PIC  S9(04)
                                       COMP.
         05  DBKV-UNIT-UNGROUP-HIGH-1      PIC   X(2).
         05  DBKV-UNIT-UNGROUP-HIGH-2      PIC  S9(04)
                                       COMP.
*
*****************************************************************
*          END OF GENERATED DECLARATIONS.
*****************************************************************
*
```

THE ASSIGNMENTS

```
*
*****************************************************************
* ABM /ASSDDC-MENU-1C  /      * Generated : 86.03.11     15:33
*****************************************************************
*
* ------------------------------* R1       ASSIGNMENTS.
*
     MOVE   7                 TO DDS-R1-NO-OF-ITEMS.
     MOVE  96                 TO DDS-R1-TOT-ITEM-LEN.
     MOVE   1                 TO DDS-R1-NO-OF-RECORDS.
     MOVE   1                 TO DDS-R1-FIRST-ITEM-NO.
     MOVE   1                 TO DDS-R1-FIRST-WORD-NO.
*
     MOVE 'R1       '         TO DDS-R1-RECORD-NAME.
*
     MOVE 'UNTYPE   '         TO DDS-R1-ITEM-NAME(  1).
     MOVE   1                 TO DDS-R1-ITEM-LEN (  1).
     MOVE 'E '                TO DDS-R1-ITEM-TYPE(  1).
*
```

```
          MOVE 'UNNUMBER'          TO DDS-R1-ITEM-NAME(  2).
          MOVE   1                 TO DDS-R1-ITEM-LEN (  2).
          MOVE 'S '                TO DDS-R1-ITEM-TYPE(  2).
      *
          MOVE 'UNREGBY '          TO DDS-R1-ITEM-NAME(  3).
          MOVE   2                 TO DDS-R1-ITEM-LEN (  3).
          MOVE 'E '                TO DDS-R1-ITEM-TYPE(  3).
      *
          MOVE 'UNCOM1  '          TO DDS-R1-ITEM-NAME(  4).
          MOVE  30                 TO DDS-R1-ITEM-LEN (  4).
          MOVE 'E '                TO DDS-R1-ITEM-TYPE(  4).
      *
          MOVE 'UNCOM2  '          TO DDS-R1-ITEM-NAME(  5).
          MOVE  30                 TO DDS-R1-ITEM-LEN (  5).
          MOVE 'E '                TO DDS-R1-ITEM-TYPE(  5).
      *
          MOVE 'UNCOM3  '          TO DDS-R1-ITEM-NAME(  6).
          MOVE  30                 TO DDS-R1-ITEM-LEN (  6).
          MOVE 'E '                TO DDS-R1-ITEM-TYPE(  6).
      *
          MOVE 'UNREGDAT'          TO DDS-R1-ITEM-NAME(  7).
          MOVE   2                 TO DDS-R1-ITEM-LEN (  7).
          MOVE 'D '                TO DDS-R1-ITEM-TYPE(  7).
      *
      * ------------------------------* R2       ASSIGNMENTS.
      *
          MOVE   1                 TO DDS-R2-NO-OF-ITEMS.
          MOVE   1                 TO DDS-R2-TOT-ITEM-LEN.
          MOVE   1                 TO DDS-R2-NO-OF-RECORDS.
          MOVE   8                 TO DDS-R2-FIRST-ITEM-NO.
          MOVE  97                 TO DDS-R2-FIRST-WORD-NO.
      *
          MOVE 'R2      '          TO DDS-R2-RECORD-NAME.
      *
          MOVE 'OK1     '          TO DDS-R2-ITEM-NAME(  1).
          MOVE   1                 TO DDS-R2-ITEM-LEN (  1).
          MOVE 'O '                TO DDS-R2-ITEM-TYPE(  1).

      *
      * ------------------------------* UNIT     ASSIGNMENTS.
      *
          MOVE   7                 TO DDB-UNIT-NO-OF-ITEMS.
          MOVE  96                 TO DDB-UNIT-TOT-ITEM-LEN.
      *
          MOVE 'UNIT    '          TO DDB-UNIT-RECORD-NAME.
      *
          MOVE 'UNNUMBER'          TO DDB-UNIT-ITEM-NAME(  1).
          MOVE   1                 TO DDB-UNIT-ITEM-LEN (  1).
          MOVE 'S '                TO DDB-UNIT-ITEM-TYPE(  1).
      *
          MOVE 'UNTYPE  '          TO DDB-UNIT-ITEM-NAME(  2).
          MOVE   1                 TO DDB-UNIT-ITEM-LEN (  2).
          MOVE 'E '                TO DDB-UNIT-ITEM-TYPE(  2).
      *
          MOVE 'UNREGDAT'          TO DDB-UNIT-ITEM-NAME(  3).
          MOVE   2                 TO DDB-UNIT-ITEM-LEN (  3).
          MOVE 'D '                TO DDB-UNIT-ITEM-TYPE(  3).
      *
```

```
           MOVE 'UNREGBY '         TO DDB-UNIT-ITEM-NAME(  4).
           MOVE   2                TO DDB-UNIT-ITEM-LEN (  4).
           MOVE 'E '               TO DDB-UNIT-ITEM-TYPE(  4).
      *
           MOVE 'UNCOM1  '         TO DDB-UNIT-ITEM-NAME(  5).
           MOVE  30                TO DDB-UNIT-ITEM-LEN (  5).
           MOVE 'E '               TO DDB-UNIT-ITEM-TYPE(  5).
      *
           MOVE 'UNCOM2  '         TO DDB-UNIT-ITEM-NAME(  6).
           MOVE  30                TO DDB-UNIT-ITEM-LEN (  6).
           MOVE 'E '               TO DDB-UNIT-ITEM-TYPE(  6).
      *
           MOVE 'UNCOM3  '         TO DDB-UNIT-ITEM-NAME(  7).
           MOVE  30                TO DDB-UNIT-ITEM-LEN (  7).
           MOVE 'E '               TO DDB-UNIT-ITEM-TYPE(  7).
      *
      * ------------------------------* INITIATION INDEX UNREGDAT IN UNIT.
      *
           MOVE   2                TO DBKI-UNIT-UNREGDAT-LEN.
           MOVE 'UNREGDAT'         TO DBKI-UNIT-UNREGDAT-KEY-NAM.
           MOVE 'UNIT    '         TO DBKI-UNIT-UNREGDAT-RLM-NAM.
      *
           CALL 'DDINKEY' USING    DBKI-UNIT-UNREGDAT,
                                   DBKV-UNIT-UNREGDAT.
      *
      * ------------------------------* INITIATION INDEX UNGROUP IN UNIT.
      *
           MOVE   2                TO DBKI-UNIT-UNGROUP-LEN.
           MOVE 'UNGROUP '         TO DBKI-UNIT-UNGROUP-KEY-NAM.
           MOVE 'UNIT    '         TO DBKI-UNIT-UNGROUP-RLM-NAM.
      *
           CALL 'DDINKEY' USING    DBKI-UNIT-UNGROUP,
                                   DBKV-UNIT-UNGROUP.
      *
      * ------------------------------* INITIATION FORM/REALMS.
      *
           MOVE 'MENU-1C '         TO SCC-PIC-NAME.
           MOVE   1                TO SCC-READ-MODE.
           MOVE   1                TO SCC-WRITE-MODE.
           MOVE   1                TO SCC-START-RW-LINE.
           MOVE   1                TO SCC-RW-NO-OF-LINES.
      *
      *****************************************************************
      *         END OF GENERATED ASSIGNMENTS.
      *****************************************************************
      *
```

9.7 THE FORTRAN INCLUDE FILE

The FORTRAN INCLUDE file contains values of items and fields in the database. This makes it specially easy to make application programs. An application program need only call the INCLUDE file (DECDDI-name:SYMB), and the values of all items and fields will be automatically available.

The following is the listing of the INCLUDE file generated for our sample database:

THE DECLARATIONS

```
C
C*********************************************************************
C      ABM /DECDDI-MENU-1F  /      * Generated : 86.03.11     11:12
C*********************************************************************
        INTEGER*2       ITEMSUB(  44)
        CHARACTER       CITMSUB( 2)*44
        EQUIVALENCE    (ITEMSUB  ,CITMSUB)
C
C                                 * REFTAB -  REFERANCE TABLE
C
        INTEGER*2       REFTAB (  54),  MRMO,  MWMO,
     +                  LINE,  NOLINE,  NOREAD,  MTCH
C
        CHARACTER       CPNS*8
C
        EQUIVALENCE    (REFTAB(1),  CPNS),
     +                 (REFTAB(5),  MRMO),
     +                 (REFTAB(6),  MWMO),
     +                 (REFTAB(7),  LINE),
     +                 (REFTAB(8),  NOLINE),
     +                 (REFTAB(9),  NOREAD),
     +                 (REFTAB(10), MTCH),
     +                 (REFTAB(11), CITMSUB)
C
C                              * R1       -REALM : DIMENSION, VARIABLES
C
        INTEGER*2       MRECR1(  96) ,MITEMR1(  66)
        CHARACTER       CITEMR1(  8)* 8
        EQUIVALENCE    (MITEMR1( 7) ,CITEMR1( 1))
C
        INTEGER*2       R1NUMBER
C
        INTEGER*4       R1REGDAT
C
        CHARACTER       R1TYPE  *2     ,R1REGBY *4      ,R1COM1  *60    ,
     +                  R1COM2  *60    ,R1COM3  *60
C
        EQUIVALENCE    (R1TYPE   ,MRECR1(   1)),
     +                 (R1NUMBER ,MRECR1(   2)),
     +                 (R1REGBY  ,MRECR1(   3)),
     +                 (R1COM1   ,MRECR1(   5)),
     +                 (R1COM2   ,MRECR1(  35)),
     +                 (R1COM3   ,MRECR1(  65)),
     +                 (R1REGDAT ,MRECR1(  95))
```

```
C
C                              * R2       -REALM : DIMENSION, VARIABLES
C
      INTEGER*2       MRECR2(   1) ,MITEMR2(  46)
      CHARACTER       CITEMR2(  2)* 8
      EQUIVALENCE    (MITEMR2( 7) ,CITEMR2( 1))
C
      CHARACTER       R2OK1   *2
C
      EQUIVALENCE    (R2OK1    ,MRECR2(   1))
C
C                         * UNIT     -REALM : DIMENSION, VARIABLES
C
      INTEGER*2       KRECUN(  96) ,KITEMUN(  66)
      CHARACTER       CITEMUN(  8)* 8
      EQUIVALENCE    (KITEMUN( 7) ,CITEMUN( 1))
C
      INTEGER*2       UNNUMBER
C
      INTEGER*4       UNREGDAT
C
      CHARACTER       UNTYPE   *2      ,UNREGBY *4      ,UNCOM1   *60    ,
     +                UNCOM2   *60     ,UNCOM3   *60
C
      EQUIVALENCE    (UNNUMBER ,KRECUN(    1)),
     +               (UNTYPE   ,KRECUN(    2)),
     +               (UNREGDAT ,KRECUN(    3)),
     +               (UNREGBY  ,KRECUN(    5)),
     +               (UNCOM1   ,KRECUN(    7)),
     +               (UNCOM2   ,KRECUN(   37)),
     +               (UNCOM3   ,KRECUN(   67))
C
C                              * UNIT     -INDEX : DIMENSION, VARIABLES
C
      INTEGER*2       KIUNREG(10) ,KVUNREG(   4)                *UNREGDAT
      CHARACTER       CUNREG(1)*16
      EQUIVALENCE    (KIUNREG(3)  ,CUNREG(1))
C
      INTEGER*4       LUNREG1          ,HUNREG1
C
      EQUIVALENCE    (LUNREG1   ,KVUNREG(   1)),                *UNREGDA1
     +               (HUNREG1   ,KVUNREG(   3))                 *UNREGDA1
C
C                              * UNIT     -INDEX : DIMENSION, VARIABLES
C
      INTEGER*2       KIUNGRO(10) ,KVUNGRO(   4)                 *UNGROUP
      CHARACTER       CUNGRO(1)*16
      EQUIVALENCE    (KIUNGRO(3)  ,CUNGRO(1))
C
      INTEGER*2       LUNGRO2          ,HUNGRO2
C
      CHARACTER       LUNGRO1  *2      ,HUNGRO1  *2
C
      EQUIVALENCE    (LUNGRO1   ,KVUNGRO(   1)),                *UNTYPE 1
     +               (HUNGRO1   ,KVUNGRO(   3)),                *UNTYPE 1
     +               (LUNGRO2   ,KVUNGRO(   2)),                *UNNUMBE2
     +               (HUNGRO2   ,KVUNGRO(   4))                 *UNNUMBE2
C
```

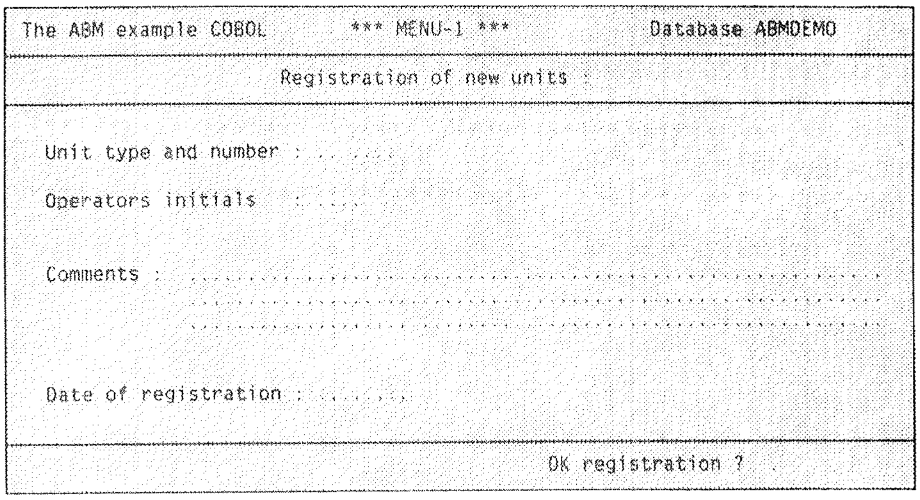
THE ASSIGNMENTS

```
C
C*********************************************************************
C      ABM /ASSDDI-MENU-1F  /      * Generated : 86.03.11      11:12
C*********************************************************************
C

C
C                                         * R1       ITEMLISTS
C
          MITEMR1( 1) =   7                                       *NO ITEM
          MITEMR1( 2) =  96                                       *LENGTH
          MITEMR1( 3) =   1                                       *NO REC.
          MITEMR1( 4) =   1                                       *1.FIELD
          MITEMR1( 5) =   1                                       *1.WORD
          MITEMR1( 6) =   7                                       *UNIQ.IT

      CITEMR1( 1) = 'R1       '
      CITEMR1(  2)='R1TYPE   '; MITEMR1( 39)=  1 ; MITEMR1(  46)="E  "*EVEN CH
      CITEMR1(  3)='R1NUMBER'; MITEMR1( 40)=  1 ; MITEMR1(  47)="S  "*SINGEL
      CITEMR1(  4)='R1REGBY  '; MITEMR1( 41)=  2 ; MITEMR1(  48)="E  "*EVEN CH
      CITEMR1(  5)='R1COM1   '; MITEMR1( 42)= 30 ; MITEMR1(  49)="E  "*EVEN CH
      CITEMR1(  6)='R1COM2   '; MITEMR1( 43)= 30 ; MITEMR1(  50)="E  "*EVEN CH
      CITEMR1(  7)='R1COM3   '; MITEMR1( 44)= 30 ; MITEMR1(  51)="E  "*EVEN CH
      CITEMR1(  8)='R1REGDAT'; MITEMR1( 45)=  2 ; MITEMR1(  52)="D  "*DOUBLE
C
C                                         * R2       ITEMLISTS
C
          MITEMR2( 1) =   1                                       *NO ITEM
          MITEMR2( 2) =   1                                       *LENGTH
          MITEMR2( 3) =   1                                       *NO REC.
          MITEMR2( 4) =   8                                       *1.FIELD
          MITEMR2( 5) =  97                                       *1.WORD
          MITEMR2( 6) =   1                                       *UNIQ.IT

          CITEMR2( 1) = 'R2       '
      CITEMR2(  2)='R2OK1    '; MITEMR2( 15)=  1 ; MITEMR2(  16)="O  "*ODD CH.
C
C                                         * UNIT     ITEMLISTS
C
          KITEMUN( 1) =   7                                       *NO ITEM
          KITEMUN( 2) =  96                                        *LENGTH

          CITEMUN( 1) = 'UNIT     '
       CITEMUN(  2)='UNNUMBER'; KITEMUN( 39)=  1 ; KITEMUN(  46)="S  "*SINGEL
       CITEMUN(  3)='UNTYPE   '; KITEMUN( 40)=  1 ; KITEMUN(  47)="E  "*EVEN CH
       CITEMUN(  4)='UNREGDAT'; KITEMUN( 41)=  2 ; KITEMUN(  48)="D  "*DOUBLE
       CITEMUN(  5)='UNREGBY  '; KITEMUN( 42)=  2 ; KITEMUN(  49)="E  "*EVEN CH
       CITEMUN(  6)='UNCOM1   '; KITEMUN( 43)= 30 ; KITEMUN(  50)="E  "*EVEN CH
       CITEMUN(  7)='UNCOM2   '; KITEMUN( 44)= 30 ; KITEMUN(  51)="E  "*EVEN CH
       CITEMUN(  8)='UNCOM3   '; KITEMUN( 45)= 30 ; KITEMUN(  52)="E  "*EVEN CH
C
```

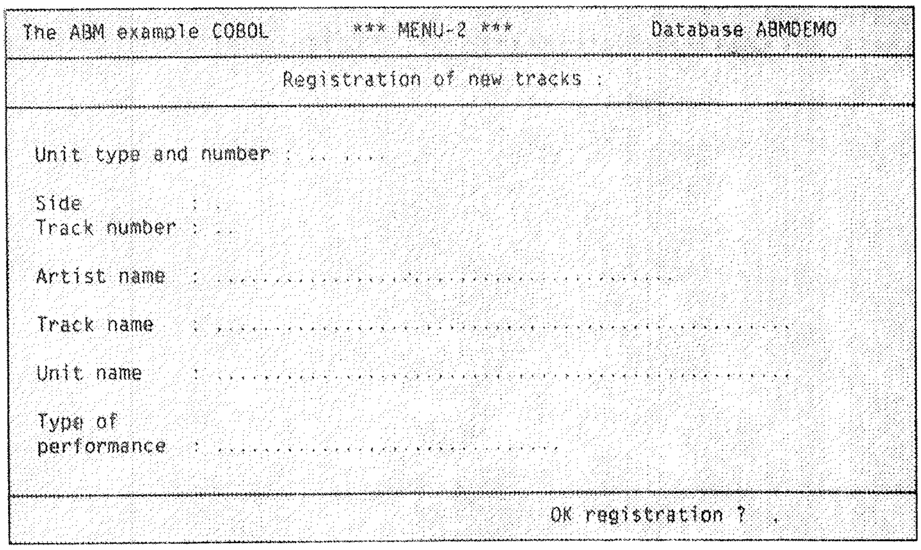
```
          KIUNREG(2)                              =                  2
                                              *LENGTH
          CUNREG(1) = 'UNREGDATUNIT     '
          CALL DDINKEY(KIUNREG,KVUNREG)

C
          KIUNGRO(2)                              =                  2
                                              *LENGTH
          CUNGRO(1) = 'UNGROUP UNIT     '
          CALL DDINKEY(KIUNGRO,KVUNGRO)
C
C**********************************************************************
C**  STANDARD INITIATION
C**********************************************************************
C
C
C
          CPNS = 'MENU-1F '
          MRMO = 1
          MWMO = 1
          LINE = 1
          NOLINE = 1
C
C**********************************************************************
C**  END OF GENERATED STATEMENTS
C**********************************************************************
C
```

9.8 A COBOL APPLICATION PROGRAM: AN EXAMPLE

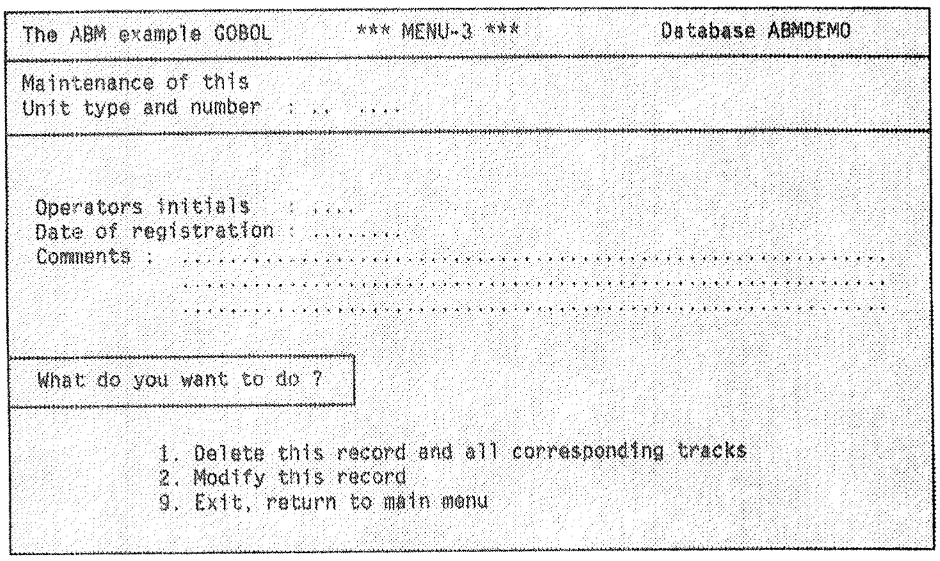
Following is an example of a COBOL program. The program uses the screen forms shown below: (The "main menu" is shown on page 141.)



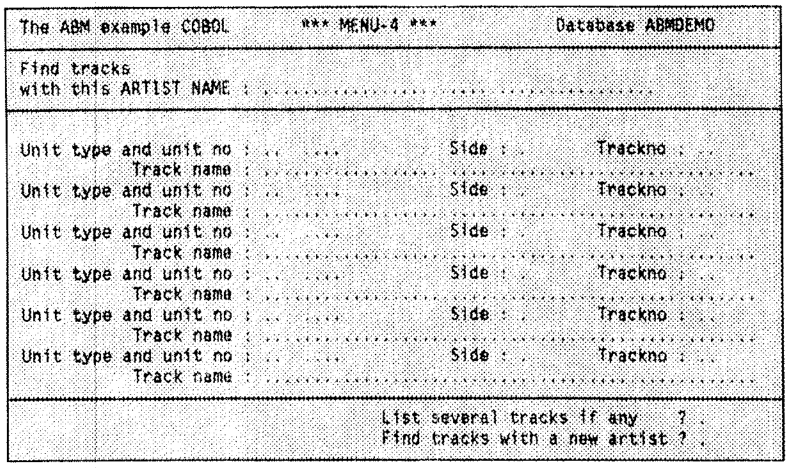
```
The ABM example COBOL        *** MENU-1 ***          Database ABMDEMO
                    Registration of new units :

  Unit type and number : .. ....

  Operators initials   : ....

  Comments : ...........................................................
             ...........................................................
             ...........................................................

  Date of registration : ........

                                          OK registration ? .
```



```
The ABM example COBOL        *** MENU-2 ***          Database ABMDEMO
                    Registration of new tracks :

  Unit type and number : .. ....

  Side         : .
  Track number : ..

  Artist name  : ..................................

  Track name   : ..................................................

  Unit name    : ..................................................

  Type of
  performance  : ..............................

                                          OK registration ? .
```



```
The ABM example COBOL        *** MENU-3 ***          Database ABMDEMO
Maintenance of this
Unit type and number  : ..  ....

  Operators initials   : ....
  Date of registration : ........
  Comments : ...........................................................
             ...........................................................
             ...........................................................

 What do you want to do ?

            1. Delete this record and all corresponding tracks
            2. Modify this record
            9. Exit, return to main menu
```



```
The ABM example COBOL       *** MENU-4 ***           Database ABMDEMO

Find tracks
with this ARTIST NAME : ..........................................

Unit type and unit no : .. ....           Side : .      Trackno : ..
           Track name : ..............................................
Unit type and unit no : .. ....           Side : .      Trackno : ..
           Track name : ..............................................
Unit type and unit no : .. ....           Side : .      Trackno : ..
           Track name : ..............................................
Unit type and unit no : .. ....           Side : .      Trackno : ..
           Track name : ..............................................
Unit type and unit no : .. ....           Side : .      Trackno : ..
           Track name : ..............................................
Unit type and unit no : .. ....           Side : .      Trackno : ..
           Track name : ..............................................

                                   List several tracks if any   ? .
                                   Find tracks with a new artist ? .
```

Please study the application program along with the comments.

```
 IDENTIFICATION DIVISION.

 Program-id. MENU-C.
 Author.     ØSÆ
 Security.   No security.
 Remarks.    ABM-example written in Cobol.

*-----------------------------------------------------------------

 ENVIRONMENT DIVISION.

*-----------------------------------------------------------------

 DATA DIVISION.
 WORKING-STORAGE SECTION.

 01 MAIN-RECORD.

     COPY DECDDC-MENU-C

*                      * Additional declarations

 01  database-name                 pic      x(8).
 01  sibas-system-number                    comp.

*-----------------------------------------------------------------

 PROCEDURE DIVISION.
 MAIN SECTION.

     perform ASSIGN-VALUE.
     perform INITIATE.

     do while scv-R1-NUMBER not = 9

        perform DISPLAY-MENU.

        if      scv-R1-NUMBER = 1    then    call "MENU-1"
        else-if scv-R1-NUMBER = 2    then    call "MENU-2"
        else-if scv-R1-NUMBER = 3    then    call "MENU-3"
        else-if scv-R1-NUMBER = 4    then    call "MENU-4" end-if.

     end-do.
```

```
           perform TERMINATE.
           STOP RUN.

      *-----------------------------------------------------------------

       ASSIGN-VALUE SECTION.

           COPY ASSDDC-MENU-C

      *-----------------------------------------------------------------

       INITIATE SECTION.

      *                        * Open database
           call 'ABDBOPN'                 using   sibas-system-number,
                                                  database-name

      *                        * Ready realm. In the subschema 'MENU-C' the
      *                          realms are marked for doing ready realms
      *                          only.
           call 'SRRLM'                   using   dbr-no-of-realms,
                                                  dbr-realm-names,
                                                  dbr-realm-usage(1),
                                                  dbr-realm-protect(1),
                                                  dbstatus
           if dbstatus < 0 or = 0         go to   DD-ERROR      end-if

      *                        * Initiate abm-focus
           move 1                         to      mflag
           call 'DDINITE'                 using   mflag

      *-----------------------------------------------------------------

       DISPLAY-MENU SECTION.

      *                        * Get the picture from the formfile
           move 'COB-EXAMPLE-BOO'         to      formfile
           call 'DDGTPIC'                 using   formfile,
                                                  ddc-ref-table,
                                                  fcstatus
           if fcstatus not = 0            go to   DD-ERROR   end-if

      *                        * Read menunumber
           move '+:NUMBER  *'             to      ddc-select
           call 'DDRFLDS'                 using   ddc-ref-table,
                                                  dds-R1-subschema,
                                                  scv-R1,
                                                  fcstatus
           if fcstatus not = 0            go to   DD-ERROR  end-if

      *-----------------------------------------------------------------

       TERMINATE SECTION.

      *                        * Normal termination. Finish realms.
           call 'SFRLM'                   using   dbr-no-of-realms,
                                                  dbr-realm-names,
                                                  dbstatus
```

```
          if dbstatus < 0               go to   DD-ERROR  end-if

      *                      * Close database
          call 'ABDBCLS'                using   dbstatus,
                                                database-name

      *-----------------------------------------------------------------

       DD-ERROR SECTION.

      *                      * Display error information
          If fcstatus not = 0  then
             call 'DDERMSG'             using   fcstatus
          else
             call 'DDERMSG'             using   dbstatus
          end-if.

      *                      * Close database
          call 'ABDBCLS'                using   0,
                                                database-name
          STOP RUN.




       IDENTIFICATION DIVISION.
       PROGRAM-ID.
       MENU-1.

      *-----------------------------------------------------------------

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       01  MAIN-RECORD.

           COPY DECDDC-MENU-1C.

      *                      * Additional declarations
       01  database-name                pic     x(8)

      *-----------------------------------------------------------------

       PROCEDURE DIVISION.

       MAIN SECTION.

          perform ASSIGN-VALUE.
          move "Y"                      to      scv-R2-OK1.
          perform DISPLAY-FORM.

      *                      * Loop as long as 'OK registration?' is not
      *                        not S(top)
          perform MENU-1-REGISTRATE-UNIT until  scv-R2-OK1="S" or ="s".
          exit program.

      *-----------------------------------------------------------------
```

```
 ASSIGN-VALUE SECTION.

     COPY ASSDDC-MENU-1C.

*-----------------------------------------------------------------

 DISPLAY-FORM.

     move 'COB-EXAMPLE-BOO'         to      formfile
     call 'DDGTPIC'                 using   formfile,
                                            ddc-ref-table,
                                            fcstatus
     if fcstatus not = 0            go to   DD-ERROR   end-if

*-----------------------------------------------------------------

 MAIN-LOGIC SECTION.

 MENU-1-REGISTRATE-UNIT.


*                        * Write message to the messageline
     move '-: Please give unit type and number '''
                                    to      message.
     call 'DDWMSGE'                 using   message,
                                            fcstatus
     if fcstatus not = 0            go to   DD-ERROR   end-if

*                        * Clear fields on the screen
     move '0:*'                     to      ddc-select
     call 'DDCFLDS'                 using   ddc-ref-table,
                                            dds-R1-subschema,
                                            scv-R1,
                                            fcstatus
     if fcstatus not = 0            go to   DD-ERROR   end-if

*                        * Read unit type and number:
     move '+:UNTYPE  UNNUMBER*'     to      ddc-select
     call 'DDRFLDS'                 using   ddc-ref-table,
                                            dds-R1-subschema,
                                            scv-R1,
                                            fcstatus
     if fcstatus not = 0            go to   DD-ERROR   end-if

*                        * Test if unit type = EX(it)
     if scv-R1-UNTYPE = 'EX' or = 'ex' then
        perform END-OF-MENU-1-REGISTRATE-UNIT
     end-if

*                        * Set low limits equal unit type and number
*                          Find the specific record

     move scv-R1-UNTYPE             to      dbkv-UNIT-UNGROUP-low-1
     move scv-R1-UNNUMBER           to      dbkv-UNIT-UNGROUP-low-2
     call 'DDFTCH'                  using   dbki-UNIT-UNGROUP,
                                            dbkv-UNIT-UNGROUP,
                                            dbstatus
```

```
           if      dbstatus > 0          then    perform UNIT-EXISTS
           else-if dbstatus < 0   then   go to   DD-ERROR
           else
       *                    * Clear the message line
             call 'DDCMSGE'              using   fcstatus
             if fcstatus not = 0         go to   DD-ERROR  end-if

       *                    * Read rest of the record
             move '-:UNTYPE  UNNUMBER*' to       ddc-select
             call 'DDRFLDS'              using   ddc-ref-table,
                                                 dds-R1-subschema,
                                                 scv-R1,
                                                 fcstatus
             if fcstatus not = 0         go to   DD-ERROR  end-if

       *                    * 'Ok registration?' Read OK field
             move '0:*'                  to      ddc-select
             call 'DDRFLDS'              using   ddc-ref-table,
                                                 dds-R2-subschema,
                                                 scv-R2,
                                                 fcstatus
             if fcstatus not = 0         go to   DD-ERROR  end-if

             if scv-R2-OK1 = "Y" or = "y"        then

       *                    * Transfer values from picture record buffer
       *                      to realm buffer
               call 'DDTRNSC'            using   dds-R1-subschema,
                                                 scv-R1,
                                                 ddb-UNIT-subschema,
                                                 dbv-UNIT
       *                    * Store record
               call 'DDSTORE'            using   ddc-select,
                                                 ddb-UNIT-subschema,
                                                 dbv-UNIT,
                                                 dbstatus
               if dbstatus not = 1       go to   DD-ERROR  end-if

             else-if scv-R2-OK1 = "N" or = "n"   then
                                         go to   MENU-1-REGISTRATE-UNIT
             end-if
           end-if

           END-OF-MENU-1-REGISTRATE-UNIT.
           exit program.

       *-------------------------------------------------------------------

        UNIT-EXISTS SECTION.

       *                    * Write message to message line
           move '+: This unit is already in register ! '''
                                         to      message
           call 'DDWMSGE'                using   message,
                                                 fcstatus
           if fcstatus not = 0           go to   DD-ERROR  end-if
```

```
*                      * Get record
     move  0                    to     tdbkey
     move  '0:*'                to     ddc-select
     call  'DDGET'              using  tdbkey,
                                       ddc-select,
                                       ddb-UNIT-subschema,
                                       dbv-UNIT,
                                       dbstatus
     if dbstatus not = 1        go to  DD-ERROR  end-if

*                      * Transfer values from realm buffer
*                        to picture record buffer
     call  'DDTRNSC'            using  ddb-UNIT-subschema,
                                       dbv-UNIT,
                                       dds-R1-subschema,
                                       scv-R1

*                      * Write record to the screen
     call  'DDWFLDS'            using  ddc-ref-table,
                                       dds-R1-subschema,
                                       scv-R1,
                                       fcstatus
     if fcstatus not = 0        go to  DD-ERROR  end-if

*-----------------------------------------------------------------

 DD-ERROR SECTION.

*                      * Display error information
     If fcstatus not = 0  then
        call 'DDERMSG'          using  fcstatus
     else
        call 'DDERMSG'          using  dbstatus
     end-if.

*                      * Close database
     call  'ABDBCLS'            using  0,
                                       database-name
     STOP RUN.




 IDENTIFICATION DIVISION.
 PROGRAM-ID.
 MENU-2.

*-----------------------------------------------------------------

 DATA DIVISION.
 WORKING-STORAGE SECTION.

 01 MAIN-RECORD.

     COPY DECDDC-MENU-2C.

*                      * Additional declarations
 01  database-name              pic    x(8)
```

```
*----------------------------------------------------------------

 PROCEDURE DIVISION.
 MAIN SECTION.

     perform ASSIGN-VALUE.
     move "Y"                        to  scv-R2-OK1
     perform DISPLAY-FORM.

*                     * Loop as long as 'OK registration?' is
*                       not S(top)
     perform MENU-2-REGISTRATE-TRACK until scv-R2-OK1 ="S" or ="s"
     exit program.

*----------------------------------------------------------------

 ASSIGN-VALUE SECTION.

     COPY ASSDDC-MENU-2C

*----------------------------------------------------------------

 DISPLAY-FORM.

     move 'COB-EXAMPLE-BOO'       to      formfile
     call 'DDGTPIC'               using   formfile,
                                          ddc-ref-table,
                                          fcstatus
     if fcstatus not = 0          go to   DD-ERROR  end-if

*----------------------------------------------------------------

 MAIN-LOGIC SECTION.

 MENU-2-REGISTRATE-TRACK.

*                     * Write message to the messageline
     move '-: Please give unit type and number '''
                                  to      message
     call 'DDWMSGE'               using   message,
                                          fcstatus
     if fcstatus not = 0          go to   DD-ERROR  end-if

*                     * Clear fields on the screen
     move '0:*'                   to      ddc-select
     call 'DDCFLDS'               using   ddc-ref-table,
                                          dds-R1-subschema,
                                          scv-R1,
                                          fcstatus
     if fcstatus not = 0          go to   DD-ERROR  end-if

*                     * Read unit type and number:
     move '+:TRTYPE  TRNUMBER*'   to      ddc-select
     call 'DDRFLDS'               using   ddc-ref-table,
                                          dds-R1-subschema,
                                          scv-R1,
                                          fcstatus
     if fcstatus not = 0          go to   DD-ERROR  end-if
```

```
*                       * Test if unit type = EX(it)
      if scv-R1-TRTYPE = 'EX' or = 'ex' then
         perform END-OF-MENU-2-REGISTRATE-TRACK
      end-if
*                       * Set low limits equal unit type and number
*                         Find the specific record
      move scv-R1-TRTYPE            to      dbkv-UNIT-UNGROUP-low-1
      move scv-R1-TRNUMBER          to      dbkv-UNIT-UNGROUP-low-2
      call 'DDFTCH'                 using   dbki-UNIT-UNGROUP,
                                            dbkv-UNIT-UNGROUP,
                                            dbstatus
      if      dbstatus = 0   then   perform UNIT-NOT-IN-REGISTER
      else-if dbstatus < 0   then   go to   DD-ERROR
      else-if dbstatus = 1   then

*                       * This unit is in the register, clear
*                         messageline
         call 'DDCMSGE'             using   message,
                                            fcstatus
         if fcstatus not = 0        go to   DD-ERROR  end-if
*                       * Read rest of the record
         move '-:TRTYPE  TRNUMBER*' to      ddc-select
         call 'DDRFLDS'             using   ddc-ref-table,
                                            dds-R1-subschema,
                                            scv-R1,
                                            fcstatus
         if fcstatus not = 0        go to   DD-ERROR  end-if
*                       * 'OK registration?' Read OK field
         move '0:*'                 to      ddc-select
         call 'DDRFLDS'             using   ddc-ref-table,
                                            dds-R2-subschema,
                                            scv-R2,
                                            fcstatus
         if fcstatus not = 0        go to   DD-ERROR  end-if

         if   scv-R2-OK1 = "N" or = "n" then
                     go to MENU-2-REGISTRATE-TRACK

         else-if scv-R2-OK1 = "Y" or = "y" then

*                       * Transfer values from picture record buffer
*                         to realm buffer
            call 'DDTRNSC'          using   dds-R1-subschema,
                                            scv-R1,
                                            ddb-TRACK-subschema,
                                            dbv-TRACK


*                       * Store record:
            call 'DDSTORE'          using   ddc-select,
                                            ddb-TRACK-subschema,
                                            dbv-TRACK,
                                            dbstatus
            if dbstatus not = 1     go to   DD-ERROR  end-if

         end-if
      end-if

      END-OF-MENU-2-REGISTRATE-TRACK.
      exit program.
```

```
*-----------------------------------------------------------------

 UNIT-NOT-IN-REGISTER SECTION.

     move '+: This unit is not in the register! Try again! '''
                                     to      message
     call 'DDWMSGE'                  using   message,
                                             fcstatus
     if fcstatus not = 0             go to   DD-ERROR  end-if

*-----------------------------------------------------------------

 DD-ERROR SECTION.

*                         * Display error information
     If fcstatus not = 0  then
        call 'DDERMSG'               using   fcstatus
     else
        call 'DDERMSG'               using   dbstatus
     end-if.

*                         * Close database
     call 'ABDBCLS'                  using   0,
                                             database-name
     STOP RUN.



 IDENTIFICATION DIVISION.
 PROGRAM-ID.
 MENU-3.

*-----------------------------------------------------------------
 DATA DIVISION.
 WORKING-STORAGE SECTION.

 01 MAIN-RECORD.

     COPY DECDDC-MENU-3.

*                         * Additional declarations.
 01  database-name                   pic     x(8).
 01  length                                  comp.
 01  iline                                   comp.
 01  icol                                    comp.

*-----------------------------------------------------------------
 PROCEDURE DIVISION.
 MAIN SECTION.

     perform ASSIGN-VALUE.
     perform DISPLAY-FORM.
     perform MENU-3-MAINTENANCE.
     exit program.

*-----------------------------------------------------------------
```

```
 ASSIGN-VALUE SECTION.

     COPY ASSDDC-MENU-3C

*---------------------------------------------------------------------

 DISPLAY-FORM.

     move 'COB-EXAMPLE-BOO'        to       formfile
     call 'DDGTPIC'                using    formfile,
                                            ddc-ref-table,
                                            fcstatus
     if fcstatus not = 0           go to    DD-ERROR   end-if

*---------------------------------------------------------------------

 MAIN-LOGIC SECTION.

 MENU-3-MAINTENANCE.
*                       * Clear fields on the screen
     move '0:*'                    to       ddc-select
     call 'DDCFLDS'                using    ddc-ref-table,
                                            dds-R1-subschema,
                                            scv-R1,
                                            fcstatus
     if fcstatus not = 0           go to    DD-ERROR   end-if

*                       * Write message to the messageline
     move '-: Please give unit type and number '''
                                   to       message
     call 'DDWMSGE'                using    message,
                                            fcstatus
     if fcstatus not = 0           go to    DD-ERROR   end-if

*                       * Read unit type and number:
     move '+:UNTYPE  UNNUMBER*'    to       ddc-select
     call 'DDRFLDS'                using    ddc-ref-table,
                                            dds-R1-subschema,
                                            scv-R1,
                                            fcstatus
     if fcstatus not = 0           go to    DD-ERROR   end-if

*                       * Set low limits equal unit type and number
*                         Find the specific record and get the
*                         record values
     move scv-R1-UNTYPE            to       dbkv-UNIT-UNGROUP-low-1
     move scv-R1-UNNUMBER          to       dbkv-UNIT-UNGROUP-low-2
     move '0:*'                    to       ddc-select
     call 'DDFTCGT'                using    dbki-UNIT-UNGROUP,
                                            dbkv-UNIT-UNGROUP,
                                            ddc-select
                                            ddb-UNIT-subschema
                                            dbv-UNIT
                                            dbstatus
```

```
          if      dbstatus = 0   then    perform UNIT-NOT-IN-REGISTER
          else-if dbstatus < 0   then    go to   DD-ERROR
          else-if dbstatus = 1   then

      *                     * Forget old and remember a new record or
      *                       a search region
             move 0                      to      option
             move 0                      to      tdbsri
             call 'DDFREMB'              using   tdbsri,
                                                 option,
                                                 dbstatus
             if dbstatus not = 1         go to   DD-ERROR  end-if

      *                     * Transfer values from realm buffer to
      *                       picture record buffer
             call 'DDTRNSC'              using   ddb-UNIT-subschema,
                                                 dbv-UNIT,
                                                 dds-R1-subschema,
                                                 scv-R1

      *                     * Write record to the screen
             call 'DDWFLDS'              using   ddc-ref-table,
                                                 dds-R1-subschema,
                                                 scv-R1,
                                                 fcstatus
             if fcstatus not = 0         go to   DD-ERROR  end-if

      *                     * Write message to the messageline and read
      *                       the answer
             move '-: Please give a menu number '''
                                         to      message
             call 'DDGMSGE'              using   message,
                                                 otext,
                                                 fcstatus
             if fcstatus not = 0         go to   DD-ERROR  end-if

             if otext = "1" then

      *                     * Write a message in the given line and column
      *                       and read the answer
                move '-: Do you really want to delete this record? '''
                                         to      message
                move  24                 to      iline
                move   1                 to      icol
                call 'DDGTEXT'           using   message,
                                                 otext,
                                                 iline,
                                                 icol,
                                                 fcstatus
                if fcstatus not = 0      go to   DD-ERROR  end-if
                if otext = "Y"  or =  "y"  then

      *                     * Sibas call. Remove the record and all
      *                       references to it if no records are connected
      *                       as members
                   move 1                to      option
                   call 'SRASE'          using   tdbkey,
                                                 option,
                                                 dbstatus
                if dbstatus not = 1  then
*                       * Write a message in the given line and column
*                         and read the answer
                   move
                   '-: Track records connected, delete although? '''
                                    to        message
                   call 'DDGTEXT'   using     message,
                                              otext,
                                              iline,
                                              icol,
                                              fcstatus
                   if fcstatus not=0 go to    DD-ERROR  end-if

                   if otext =  "y" or = "y"   then

*                       * Erase the record and all member records in
*                         the set occurrences
                      move 3          to      option
                      call 'SRASE'    using   tdbkey,
                                              option,
                                              dbstatus
                      if dbstatus not = 1     go to DD-ERROR end-if
                   end-if

*                       * Display a blank text string in given line
*                         and column
                   move '        '    to      message
                   move 44            to      length
                   call 'FCWTXT'      using   iline,
                                              icol,
                                              message,
                                              length,
                                              fcstatus
                   if fcstatus not=0 go to    DD-ERROR  end-if
                end-if
             end-if
          else-if otext = "2"         then

*                       * Write message to the messageline
*                         Press Carriage return when ready to mofify
             move '-: Modify the record! Press CR '''
                                      to      message
             call 'DDGMSGE'           using   message,
                                              otext
                                              fcstatus
             if fcstatus not = 0      go to   DD-ERROR  end-if

*                       * Read rest of the record
             move '-:UNTYPE  UNNUMBER*'
                                      to      ddc-select
             call 'DDRFLDS'           using   ddc-ref-table,
                                              dds-R1-subschema,
                                              scv-R1,
                                              fcstatus
             if fcstatus not = 0      go to   DD-ERROR  end-if
```

```
      *                     * Transfer values from picture record
      *                       buffer to realm buffer
                 call 'DDTRNSC'          using   dds-R1-subschema,
                                                 scv-R1,
                                                 ddb-UNIT-subschema,
                                                 dbv-UNIT

      *                     * Modify items in the record
                 call 'DDMDFY'           using   tdbkey,
                                                 ddc-select,
                                                 ddb-UNIT-subschema,
                                                 dbv-UNIT,
                                                 dbstatus
                 if dbstatus not = 1     go to   DD-ERROR  end-if

              else
                 perform  END-OF-MENU-3-MAINTENANCE
              end-if

           end-if

      *                     * Write message to the messageline and read
      *                       the answer
           move '-: Maintenance of several records? '''
                                         to      message
           call 'DDGMSGE'                using   message
                                                 otext,
                                                 fcstatus
           if fcstatus not = 0           go to   DD-ERROR  end-if

           if otext = "Y" or   = "y"     then
              perform  MENU-3-MAINTENANCE
           end-if

       END-OF-MENU-3-MAINTENANCE.
      *-----------------------------------------------------------------

       UNIT-NOT-IN-REGISTER SECTION.

           move '+: This unit is not in the register! Try again! '''
                                         to      message
           call 'DDWMSGE'                using   message,
                                                 fcstatus
           if fcstatus not = 0           go to   DD-ERROR  end-if
      *-----------------------------------------------------------------

       DD-ERROR SECTION.

      *                     * Display error information
           If fcstatus not = 0  then
              call 'DDERMSG'             using   fcstatus
           else
              call 'DDERMSG'             using   dbstatus
           end-if.

      *                     * Close database
           call 'ABDBCLS'                using   0,
                                                 database-name
           STOP RUN.
```

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.
 MENU-4.

*-----------------------------------------------------------------

 DATA DIVISION.
 WORKING-STORAGE SECTION.

 01 MAIN-RECORD.

     COPY DECDDC-MENU-4C.

*                          * Additional declarations.
 01  database-name                     pic     x(8).
 01  no-of-records                             comp.
 01  temporary-line                            comp.
 01  flag                                      comp.
 01  no-of-rec-written                         comp.

*-----------------------------------------------------------------

 PROCEDURE DIVISION.
 MAIN SECTION.

     perform ASSIGN-VALUE.
     perform DISPLAY-FORM.
*                          * Loop as long as 'find tracks with a new
*                            artist is not N(o)
     perform MENU-4-QUESTIONS until SCV-R4-OK2 = "N" or = "n".
     exit program.

*-----------------------------------------------------------------

 ASSIGN-VALUE SECTION.

     COPY ASSDDC-MENU-4C

*-----------------------------------------------------------------

 DISPLAY-FORM.

     move 'COB-EXAMPLE-B'          to      formfile
     call 'DDGTPIC'                using   formfile,
                                           ddc-ref-table,
                                           fcstatus
     if fcstatus not = 0           go to   DD-ERROR  end-if

*-----------------------------------------------------------------
```

```
       MENU-4-QUESTIONS SECTION.

      *                       * Write message to the messageline
           move '-: Please give artist name '''
                                        to      message
           call 'DDWMSGE'               using   message,
                                                fcstatus
           if fcstatus not = 0          go to   DD-ERROR  end-if

      *                       * Read artist name
           move '+:TRARTIST*'           to      ddc-select
           call 'DDRFLDS'               using   ddc-ref-table,
                                                dds-R1-subschema,
                                                scv-R1,
                                                fcstatus
           if fcstatus not = 0          go to   DD-ERROR  end-if

      *                       * Transfer values from picture record
      *                         buffer to realm buffer
           call 'DDTRNSC'               using   dds-R1-subschema,
                                                scv-R1,
                                                ddb-TRACK-subschema,
                                                dbv-TRACK


      *                       * Low limit = artist name read from terminal
      *                         Find first record between limits using given
      *                         key

           move scv-R1-TRARTIST         to   dbkv-TRACK-TRARTIST-low-1
           move scv-R1-TRARTIST         to   dbkv-TRACK-TRARTIST-high-1

           call 'DDFEBL'                using   dbki-TRACK-TRARTIST,
                                                dbkv-TRACK-TRARTIST,
                                                dbstatus

           if      dbstatus = 0  then   perform ARTIST-NOT-IN-REGISTER
           else-if dbstatus < 0  then   go to   DD-ERROR
           else-if dbstatus = 1  then

      *                       * Count number of records found
               move 1                   to      no-of-records

      *                       * Get the items in the record
               move '0:*'               to      ddc-select
               call 'DDGET'             using   tdbkey,
                                                ddc-select,
                                                ddb-TRACK-subschema,
                                                dbv-TRACK,
                                                dbstatus
               if dbstatus not = 1      go to   DD-ERROR  end-if

      *                       * Transfer values from realm buffer
      *                         to picture record buffer
               call 'DDTRNSC'           using   ddb-TRACK-subschema,
                                                dbv-TRACK,
                                                dds-R2-subschema,
                                                scv-R2
```

```
*                     * Put field values into the total picture
*                       buffer
          call 'DDPUTRC'           using    ddc-ref-table
                                            dds-R2-subschema,
                                            scv-R2,
                                            fcstatus
          if fcstatus not = 0      go to    DD-ERROR  end-if

*                     * Loop as long as records with given artist
*                       name is found in the database ( dbstatus=1 )
          do while dbstatus = 1

*                     * Sibas call. Find next record in search
*                       region
             call 'SRNIS'          using    tdbkey
                                            tdbsri
                                            dbstatus

             if dbstatus = 1       then
*                     * Count number of records found
                add 1              to       no-of-records

*                     * Get the items in the record
                move '0:*'         to       ddc-select
                call 'DDGET'       using    tdbkey,
                                            ddc-select,
                                            ddb-TRACK-subschema,
                                            dbv-TRACK,
                                            dbstatus
                if dbstatus not = 1  go to  DD-ERROR  end-if

*                     * Transfer values from realm buffer
*                       to picture record buffer
                call 'DDTRNSC'     using    ddb-TRACK-subschema,
                                            dbv-TRACK,
                                            dds-R2-subschema,
                                            scv-R2

*                     * The size of the total picture record buffer
*                       is limited to 600 words (16 bit). It can be
*                       changed (see appendix).
*                       Maximum of R2-records is here 600/29 = 30

                if no-of-records > 30 then
*                     * Write message to the messageline
                   move
                   '-:The total picture buffer must be increased!'''
                                   to       message
                   call 'DDWMSGE'  using    message,
                                            fcstatus
                   if fcstatus not=0 go to  DD-ERROR  end-if
                   go to LAST-PART
                end-if

*                     * Count line record number
                add 1              to       scc-start-rw-line
```

```
*                      * Put field values into the total picture
*                        buffer
              call 'DDPUTRC'      using   ddc-ref-table
                                          dds-R2-subschema,
                                          scv-R2,
                                          fcstatus
              if fcstatus not = 0 go to   DD-ERROR  end-if
           end-if
        end-do

*                      * Write records from the total picture buffer
*                        to the screen. It is only possible to list
*                        6 record occurences to the screen in this
*                        example

*                      * Loop for all record occurences
        do for scc-start-rw-line from 1 to no-of-records

*                      * Save occurence number scc-start-rw-line in
*                        the total picture buffer
           move scc-start-rw-line  to     temporary-line

*                      * Get field values from the total picture
*                        buffer
           call 'DDGETRC'         using   ddc-ref-table
                                          dds-R2-subschema
                                          scv-R2
                                          fcstatus
           if fcstatus not = 0    go to   DD-ERROR  end-if

*                      * Calculate correct occ. number
*                        scc-start-rw-line on the screen
           compute scc-start-rw-line = temporary-line - flag

*                      * Write record to the screen
           call 'DDWFLDS'         using   ddc-ref-table
                                          dds-R2-subschema
                                          scv-R2
                                          fcstatus
           if fcstatus not = 0    go to   DD-ERROR  end-if

*                      * Count number of records written to the
*                        screen
           add 1 to               no-of-rec-written

           if no-of-rec-written not < 6  then

*                      * The screen is full, reset scc-start-rw-line
*                        and set new flag value
              move 1 to           scc-start-rw-line
              add  6 to           flag

*                      * 'List several tracks, if any?' Read OK field
              move '0:*'          to      ddc-select
              call 'DDRFLDS'      using   ddc-ref-table,
                                          dds-R3-subschema,
                                          scv-R3,
                                          fcstatus
              if fcstatus not = 0 go to   DD-ERROR  end-if
```

```
                 if scv-R3-OK1 = "N" or "n"   then
*                        * Do not want to continue listing several
*                          tracks
                    go to LAST-PART

                 else

*                        * Clear all occurences of this record
                    move 0              to      scc-rw-no-of-lines
                    call 'DDCFLDS'      using   ddc-ref-table,
                                                dds-R2-subschema,
                                                scv-R2,
                                                fcstatus
                    if fcstatus not=0 go to     DD-ERROR  end-if

*                        * Reset scc-rw-no-of-lines and
*                          reset number of records written to the
*                          screen
                    move 1              to      scc-rw-no-of-lines
                    move 0              to      no-of-rec-written

                 end-if
              end-if

*                        * Reset occ number line in the tot pic buffer
              move  temporary-line    to      scc-start-rw-line
           end-do
        end-if

 LAST-PART.
*                        * Reset start-rw-line, flag and number of rec
*                          written
        move 1                          to      scc-start-rw-line
        move 0                          to      flag
        move 0                          to      no-of-rec-written

*                        * Clear field
        move '0:*'                      to      ddc-select
        call 'DDCFLDS'                  using   ddc-ref-table,
                                                dds-R3-subschema,
                                                scv-R3,
                                                fcstatus
        if fcstatus not = 0             go to   DD-ERROR  end-if

*                  * 'Find tracks with a new artist?' Read OK field
        call 'DDRFLDS'                  using   ddc-ref-table,
                                                dds-R4-subschema,
                                                scv-R4,
                                                fcstatus
        if fcstatus not = 0             go to   DD-ERROR  end-if
```

```
          if scv-R4-OK2 = "Y" or = "y"   then

      *                      * Yes, clear field
             call 'DDCFLDS'             using   ddc-ref-table,
                                                dds-R1-subschema,
                                                scv-R1,
                                                fcstatus
             if fcstatus not = 0        go to   DD-ERROR  end-if

      *                * Clear all occurences of this record
             move 0                     to      scc-rw-no-of-lines
             call 'DDCFLDS'             using   ddc-ref-table,
                                                dds-R2-subschema,
                                                scv-R2,
                                                fcstatus
             if fcstatus not = 0        go to   DD-ERROR  end-if

      *                * Reset number of occurences
             move 1                     to      scc-rw-no-of-lines

          else
                go to END-OF-MENU-4-QUESTIONS
          end-if

       END-OF-MENU-4-QUESTIONS.

      *-----------------------------------------------------------------

       ARTIST-NOT-IN-REGISTER SECTION.

          move
            '+:This artist name is not in the register! Try again! '''
                                        to      message
          call 'DDWMSGE'                using   message,
                                                fcstatus
          if fcstatus not = 0           go to   DD-ERROR  end-if

      *-----------------------------------------------------------------

       DD-ERROR SECTION.

      *                      * Display error information
          If fcstatus not = 0  then
             call 'DDERMSG'             using   fcstatus
          else
             call 'DDERMSG'             using   dbstatus
          end-if.

      *                      * Close database
          call 'ABDBCLS'                using   0,
                                                database-name
          STOP RUN.
```

9.9 A FORTRAN APPLICATION PROGRAM: AN EXAMPLE

This is an example of a FORTRAN program. The program uses the screen forms from the sample COBOL application program on page 170.

Please study the application program along with the comments.

```
* ---------------------------------------------------------------------
*      Example of an ABM FORTRAN application.
* ---------------------------------------------------------------------

      PROGRAM MENU
*                              *  Additional declarations.
      integer*2   idbname(4)

$INCLUDE DECDDI-MENU-F
$INCLUDE ASSDDI-MENU-F

*                              * Open database
      call ABDBOPN (isibasno,idbname)

*                              * Ready realm. In the subschema 'MENU-F'
*                                the realms are marked for doing ready realm
*                                only.
      call SRRLM (knrea, krealms, kumod, kpmod, dbstatus)
          if(dbstatus.lt.0) goto 888

*                              * Initiate abm-focus
      MFLAG    =  1
      call DDINITE (MFLAG)

      do while ( R1number.ne.9 )

*                              * Get the picture from the formfile
          cformfile = 'FORT-EXAMPLE-B00'
          call DDGTPIC (formfile,reftab,fcstatus)
              if(fcstatus.ne.0) goto 888

*                              * Read menunumber
          citmsub(1) = '+:R1NUMBER*'
          call DDRFLDS (reftab,mitemR1,mrecR1,fcstatus)
              if(fcstatus.ne.0) goto 888

          if (R1number.eq.1) call MENU1
          if (R1number.eq.2) call MENU2
          if (R1number.eq.3) call MENU3
          if (R1number.eq.4) call MENU4

      end do
```

```
*                              * Normal termination. Finish realms.
       call SFRLM (knrea, krealms, dbstatus)
           if(dbstatus.lt.1) goto 888

*                              * Close database.
       call ABDBCLS (0, idbname)

       goto 999
888    call ERROR(fcstatus,dbstatus)

999    stop
       end
*------------------------------------------------------------------------

       SUBROUTINE MENU1

$INCLUDE DECDDI-MENU-1F
$INCLUDE ASSDDI-MENU-1F

*                              * Get the picture from the formfile
       cformfile = 'FORT-EXAMPLE-BOO'
       call DDGTPIC (formfile, reftab, fcstatus)
           if(fcstatus.ne.0) goto 888

*                              * Loop as long as 'OK registration?' is not
*                                S(top)
       do while (R2OK1.ne.'S'.and.R2OK1.ne.'s')

*                              * Write message to the messageline
           cmessage = '-: Please give unit type and number '''
           call DDWMSGE (message, fcstatus)
               if(fcstatus.ne.0) goto 888

*                              * Clear fields on the screen
           citmsub(1) = 'O:*'
           call DDCFLDS (reftab, mitemR1, mrecR1, fcstatus)
               if(fcstatus.ne.0) goto 888

*                              * Read unit type and number:
           citmsub(1) = '+:R1TYPE  R1NUMBER*'
           call DDRFLDS (reftab, mitemR1, mrecR1, fcstatus)
               if(fcstatus.ne.0) goto 888

*                              * Test if unit type = EX(it)
           if (R1TYPE.eq.'EX'.or.R1TYPE.eq.'ex') goto 999

*                              * Set low limits equal unit type and number
*                                Find the specific record
           LUNGRO1 = R1TYPE
           LUNGRO2 = R1NUMBER
           call DDFTCH (kiUNGRO, kvUNGRO, dbstatus)

           if(dbstatus.eq.1) then

*                              * Write message to the message line
               cmessage = '+: This unit is already in register ! '''
               call DDWMSGE (message, fcstatus)
                   if(fcstatus.ne.0) goto 888
```

```
*                            * Get record
              citmsub(1) = '0:*'
              call DDGET (0, itemsub, kitemUN, krecUN, dbstatus)
                  if(dbstatus.ne.1) goto 888

*                            * Transfer values from realm buffer
*                              to picture record buffer
              call DDTRNSF (kitemUN, krecUN, mitemR1, mrecR1)

*                            * Write record to the screen
              call DDWFLDS (reftab, mitemR1, mrecR1, fcstatus)
                  if(fcstatus.ne.0) goto 888


          elseif (dbstatus.eq.0) then

*                            * Clear the message line
              call DDCMSGE (fcstatus)
                  if(fcstatus.ne.0) goto 888

*                            * Read rest of the record
              citmsub(1) = '-:R1TYPE  R1NUMBER*'
              call DDRFLDS (reftab, mitemR1, mrecR1, fcstatus)
                  if(fcstatus.ne.0) goto 888

*                            * 'OK registration?' Read OK field
              citmsub(1) = '0:*'
              call DDRFLDS (reftab, mitemR2, mrecR2, fcstatus)
                  if(fcstatus.ne.0) goto 888

              if (R2OK1.eq.'Y' .or. R2OK1.eq.'y') then

*                            * Transfer values from picture record buffer
*                              to realm buffer
                  call DDTRNSF (mitemR1, mrecR1, kitemUN, krecUN)

*                            * Store record
                  call DDSTORE (itemsub, kitemUN, krecUN, dbstatus)
                      if(dbstatus.ne.1) goto 888
              end if

          elseif (dbstatus.lt.0) then
              goto 888

          end if

      end do

      goto 999
888   call ERROR(fcstatus,dbstatus)

999   return
      end

*----------------------------------------------------------------------
```

```
      SUBROUTINE MENU2


$INCLUDE DECDDI-MENU-2F
$INCLUDE ASSDDI-MENU-2F

*                            * Get the picture from the formfile
      cformfile = 'FORT-EXAMPLE-B00'
      call DDGTPIC (formfile, reftab, fcstatus)
           if(fcstatus.ne.0) goto 888

*                            * Loop as long as 'OK registration?' is not
*                              S(top)
      do while (R2OK1.ne.'S'.and.R2OK1.ne.'s')

*                            * Write message to the messageline
           cmessage = '-: Please give unit type and number '''
           call DDWMSGE (message, fcstatus)
                if(fcstatus.ne.0) goto 888

*                            * Clear fields on the screen
           citmsub(1) = 'O:*'
           call DDCFLDS (reftab, mitemR1, mrecR1, fcstatus)
                if(fcstatus.ne.0) goto 888

*                            * Read unit type and number:
           citmsub(1) = '+:R1TYPE  R1NUMBER*'
           call DDRFLDS (reftab, mitemR1, mrecR1, fcstatus)
                if(fcstatus.ne.0) goto 888

*                            * Test if unit type = EX(it)
           if (R1TYPE.eq.'EX'.or.R1TYPE.eq.'ex') goto 999

*                            * Set low limits equal unit type and number
*                              Find the specific record
           LUNGRO1 = R1TYPE
           LUNGRO2 = R1NUMBER

           call DDFTCH (kiUNGRO, kvUNGRO, dbstatus)
           if (dbstatus.eq.1) then

*                            * This unit is in the register, read rest of
*                              the record
                citmsub(1) = '-:R1TYPE  R1NUMBER*'
                call DDRFLDS (reftab,mitemR1,mrecR1,fcstatus)
                     if (fcstatus.ne.0) goto 888

*                            * 'OK registration?' Read OK field
                citmsub(1)='O:*'
                call DDRFLDS (reftab,mitemR2,mrecR2,fcstatus)
                     if (fcstatus.ne.0) goto 888

                if (R2OK1.eq.'Y'.or.R2OK1.eq.'y') then

*                            * Transfer values from picture record buffer
*                              to realm buffer
                     call DDTRNSF (mitemR1,mrecR1,kitemTR,krecTR)
```

```
*                           * Store record:
                    call DDSTORE (itemsub,kitemTR,krecTR,dbstatus)
                        if (dbstatus.ne.1) goto 888
                end if

            elseif (dbstatus.eq.0) then

*                           * Write message to the messageline
                cmessage='+: This unit is not in the reg.! Try again!'''
                call DDWMSGE (message,fcstatus)
                    if(fcstatus.ne.0) goto 888

            else
                goto 888
            end if

        end do
        goto 999
888     call ERROR(fcstatus,dbstatus)
999     return
        end

*-------------------------------------------------------------------

      SUBROUTINE MENU3


*                           *  Additional declarations.
*                              Used as otext parameters in the calls
*                              DDGTEXT and DDGMSG.
*                              Note! The size must be declared with length
*                              40 bytes.
      integer*2   iyesno(20),menunumber(20)
      character   yesno*1 ,cmenunumber*1
      equivalence (menunumber(1),cmenunumber), (iyesno(1),yesno)

$INCLUDE DECDDI-MENU-3F
$INCLUDE ASSDDI-MENU-3F

*                           * Get the picture from the formfile
      cformfile = 'FORT-EXAMPLE-BOO'
      call DDGTPIC (formfile, reftab, fcstatus)
         if(fcstatus.ne.0) goto 888

100   continue
*                           * Clear fields on the screen
      citmsub(1) = '0:*'
      call DDCFLDS (reftab, mitemR1, mrecR1, fcstatus)
         if(fcstatus.ne.0) goto 888

*                           * Write message to the messageline
      cmessage = '-: Please give unit type and number '''
      call DDWMSGE (message, fcstatus)
         if(fcstatus.ne.0) goto 888
```

```
*                              * Read unit type and number:
      citmsub(1) = '+:R1TYPE  R1NUMBER*'
      call DDRFLDS (reftab, mitemR1, mrecR1, fcstatus)
         if(fcstatus.ne.0) goto 888

*                              * Set low limits equal unit type and number
*                                Find the specific record and get the record
*                                values
      LUNGRO1 = R1TYPE
      HUNGRO1 = R1TYPE
      LUNGRO2 = R1NUMBER
      HUNGRO2 = R1NUMBER
      citmsub(1) = '0:*'

      call DDFTCGT (kiUNGRO, kvUNGRO, itemsub, kitemUN, krecUN,
     &             dbstatus)

      if (dbstatus.eq.0) then
*                              * Write message to the messageline
         cmessage = '+: This unit is not in the register ! '''
         call DDWMSGE (message, fcstatus)
            if(fcstatus.ne.0) goto 888

      elseif (dbstatus.eq.1) then

*                              * Remember temporary database key
         option = 0
         tdbsri = 0
         call DDFREMB (tdbsri,option,dbstatus)
            if (dbstatus.ne.1) goto 888

*                              * Transfer values from realm buffer to
*                                picture record buffer
         call DDTRNSF (kitemUN, krecUN, mitemR1, mrecR1)

*                              * Write record to the screen
         call DDWFLDS (reftab, mitemR1, mrecR1, fcstatus)
            if(fcstatus.ne.0) goto 888

*                              * Write message to the messageline and read
*                                the answer
         cmessage = '+: Please give a menu number '''
         call DDGMSGE (message,menunumber,fcstatus)
            if(fcstatus.ne.0) goto 888

         if (cmenunumber.eq.'1') then

*                              * Clear the messageline
            call DDCMSGE (fcstatus)
               if(fcstatus.ne.0) goto 888
```

```
*                          * Write a message in the given line and column
*                            and read the answer
         iline=24 ; icol=1
         cmessage = '+: Do you really want to delete this record? '''
         call DDGTEXT (message,iyesno,iline,icol,fcstatus)
            if(fcstatus.ne.0) goto 888

         if (yesno.eq.'Y'.or.yesno.eq.'y') then

*                          * Sibas call. Remove the record and all
*                            references to it if no records are connected
*                            as members
            option=1
            call SRASE (tdbkey,option,dbstatus)

            if (dbstatus.ne.1) then
*                          * Write a message in the given line and column
*                            and read the answer
               cmessage='+:Track records connected, delete although?
     &'''
               call DDGTEXT (message,iyesno,iline,icol,fcstatus)
                  if(fcstatus.ne.0) goto 888

               if (yesno.eq.'Y'.or.yesno.eq.'y') then
*                          * Erase the record and all member records in
*                            the set occurrences
                  option=3
                  call SRASE (tdbkey,option,dbstatus)
                     if(dbstatus.ne.1) goto 888
               end if
            end if
         end if

*                          * Display a blank text string in given line
*                            and column
         cmessage='                                            '
         ilength = 44
         call FCWTXT (iline,icol,message,ilength,fcstatus)
            if(fcstatus.ne.0) goto 888

      elseif (cmenunumber.eq.'2') then

*                          * Write message to the messageline
*                            Press Carriage return when ready to modify
         cmessage='+: Modify the record! Press CR '''
         call DDGMSGE (message,otext,fcstatus)
            if(fcstatus.ne.0) goto 888

*                          * Read rest of the record
         citmsub(1) = '-:R1TYPE  R1NUMBER*'
         call DDRFLDS (reftab, mitemR1, mrecR1, fcstatus)
            if(fcstatus.ne.0) goto 888

*                          * Transfer values from picture record buffer
*                            to realm buffer
         call DDTRNSF (mitemR1, mrecR1, kitemUN, krecUN)
```

```
*                               * Modify items in the record
            citmsub(1) = '-:UNTYPE  UNNUMBER*'
            call DDMDFY (tdbkey,itemsub,kitemUN,krecUN,dbstatus)
               if(dbstatus.ne.1) goto 888
         end if

      elseif (dbstatus.lt.0) then
         goto 888

      end if

*                               * Write message to the messageline and read
*                                 the answer
      cmessage='+: Maintenance of several records? '''
      call DDGMSGE (message,iyesno,fcstatus)
         if(fcstatus.ne.0) goto 888

      if (yesno.eq.'Y'.or.yesno.eq.'y') goto 100

      goto 999
888   call ERROR(fcstatus,dbstatus)

999   return
      end

*----------------------------------------------------------------------

      SUBROUTINE MENU4


$INCLUDE DECDDI-MENU-4F
$INCLUDE ASSDDI-MENU-4F

*                               * Get the picture from the formfile
      cformfile='FORT-EXAMPLE-BOO'
      call DDGTPIC (formfile,reftab,fcstatus)
         if(fcstatus.ne.0) goto 888

*                               * Loop as long as 'Find tracks with a new
*                                 artist?' is not N(o).
      do while (R4OK2.ne.'N'.and.R4OK2.ne.'n')

*                               * Write message to the messageline
         cmessage = '-: Please give artist name'''
         call DDWMSGE (message,fcstatus)
            if(fcstatus.ne.0) goto 888

*                               * Read artist name
         citmsub(1) = '+:R1ARTIST*'
         call DDRFLDS (reftab,mitemR1,mrecR1,fcstatus)
            if(fcstatus.ne.0) goto 888

*                               * Transfer values from picture record buffer
*                                 to realm buffer
         call DDTRNSF (mitemR1,mrecR1,kitemTR,krecTR)

*                               * Low limits = artist name read from terminal
*                                 Find first record between limits using given
*                                 key
```

```
         LTRART1 = R1ARTIST
         HTRART1 = R1ARTIST
         call DDFEBL (kiTRART,kvTRART,dbstatus)

         if (dbstatus.lt.0) goto 888
         if (dbstatus.eq.0) then
*                           * Write message to messageline
            cmessage = '+: This artist name is not in the register'''
            call DDWMSGE (message,fcstatus)
               if (fcstatus.ne.0) goto 888

         elseif (dbstatus.eq.1) then
*                           * Count number of records found
            norecords = 1
*                           * Get the items in the record
            citmsub(1) = '0:*'
            call DDGET (tdbkey,itemsub,kitemTR,krecTR,dbstatus)
               if (dbstatus.lt.0) goto 888

*                           * Transfer values from realm buffer
*                             to picture record buffer
            call DDTRNSF (kitemTR,krecTR,mitemR2,mrecR2)

*                           * Put field values into the total picture
*                             buffer
            call DDPUTRC (reftab,mitemR2,mrecR2,fcstatus)
                  if(fcstatus.ne.0) goto 888

*                           * Loop as long as records with given artist
*                             name is found in the database (dbstatus=1)
            do while (dbstatus.eq.1)

*                           * Sibas call. Find next record in search
*                             region
               call SRNIS (tdbkey,tdbsri,dbstatus)
                  if (dbstatus.lt.0) goto 888

               if (dbstatus.eq.1) then
*                           * Count number of records found
                  norecords = norecords + 1

*                           * Get the items in the record
                  citmsub(1) = '0:*'
                  call DDGET (tdbkey,itemsub,kitemTR,krecTR,dbstatus)
                     if (dbstatus.lt.0) goto 888

*                           * Transfer values from realm buffer
*                             to picture record buffer
                  call DDTRNSF (kitemTR,krecTR,mitemR2,mrecR2)

*                           * The size of the total picture record buffer
*                             is limited to 600 words (16 bit). It can be
*                             changed (see appendix).
*                             Maximum of R2-records is here 600/29 = 30
```

```
                  if (norecords.gt.30) then
*                              * Write message to the messageline
                    cmessage='+:The total picture buffer must be increas
     &ed! '''
                    call DDWMSGE (message,fcstatus)
                       if(fcstatus.ne.0) goto 888
                    goto 100
                  end if

*                              * Count line record number
                  line = line + 1

*                              * Puts field values into the total picture
*                                buffer
                  call DDPUTRC (reftab,mitemR2,mrecR2,fcstatus)

               end if

            end do

*                              * Write records from the total picture buffer
*                                to the screen It is only possible to list 6
*                                record occurences to the screen in this
*                                example

*                              * Loop for all record occurences
            do for line = 1, norecords

*                              * Save occurence number LINE in the total
*                                picture buffer
               itemporaryline = line

*                              * Get field values from the total picture
*                                buffer
               call DDGETRC (reftab,mitemR2,mrecR2,fcstatus)
                  if(fcstatus.ne.0) goto 888

*                              * Calculate correct occ. number LINE on the
*                                screen
               line = itemporaryline - iflag

*                              * Write record to the screen
               call DDWFLDS (reftab,mitemR2,mrecR2,fcstatus)
                  if(fcstatus.ne.0) goto 888

*                              * Count number of records written to the
*                                screen
               nowritten = nowritten + 1
               if (nowritten.eq.6) then
*                              * The screen is full, reset line parameter
*                                and set new flag value
                  line = 1
                  iflag = iflag + 6

*                                'List several tracks, if any?' Read OK field
                  citmsub(1) = '0:*'
                  call DDRFLDS (reftab,mitemR3,mrecR3,fcstatus)
                     if(fcstatus.ne.0) goto 888
```

```
                 if (R3OK1.eq.'N'.or.R3OK1.eq.'n') then
*                           * Do not want to continue listing several
*                             tracks
                    goto 100

                 else

*                           * Clear all occurences of this record
*                             (noline=0)
                    noline = 0
                    call DDCFLDS (reftab,mitemR2,mrecR2,fcstatus)
                       if(fcstatus.ne.0) goto 888

*                           * Reset noline and reset number
*                             of records written to the screen
                    noline    = 1 ; nowritten = 0
                 end if
              end if
*                           * Reset occ number LINE in the tot pic buffer
              line = itemporaryline

              end do
          end if

100       continue

*                           * Reset line, flag and number of records
*                             written
          line = 1  ; iflag = 0 ; nowritten = 0
*                           * Clear field
          citmsub(1) = '0:*'
          call DDCFLDS (reftab,mitemR3,mrecR3,fcstatus)
             if(fcstatus.ne.0) goto 888

*                           * 'Find tracks with a new artist?'
*                             Read OK field
          call DDRFLDS (reftab,mitemR4,mrecR4,fcstatus)
             if(fcstatus.ne.0) goto 888

          if (R4OK2.eq.'Y'.or.R4OK2.eq.'y') then

*                           * Yes. Clear fields
             call DDCFLDS (reftab,mitemR1,mrecR1,fcstatus)
                if(fcstatus.ne.0) goto 888

*                           * Clear all occurences of this record
*                             (noline=0)
             noline = 0
             call DDCFLDS (reftab,mitemR2,mrecR2,fcstatus)
                if(fcstatus.ne.0) goto 888
*                           * Reset number of occurences
             noline = 1

           else
              goto 999
           end if

        end do
```

```
      goto 999
888   call ERROR(fcstatus, dbstatus)

999   continue
      return
      end

* -------------------------------------------------------------------------


      SUBROUTINE ERROR (fcstatus,dbstatus)

      INTEGER*2 fcstatus, dbstatus, idbname(4)

*                                  * Display error information

      if (fcstatus.ne.0) then
          call DDERMSG (fcstatus)
      else
          call DDERMSG (dbstatus)
      end if

*                                  * Close database
      call ABDBCLS (0,idbname)
      stop
      end
```

APPENDIX

APPENDIX A
DISPLAY CODE

The DISPLAY code follows closely the COBOL syntax for editing pictures. There are, however, a few extensions. These extensions can insert text strings in an item when it is printed or displayed.

EXAMPLE:

ITEM CONTENT	DISPLAY CODE	DISPLAYED AS
11 char "Bill Hansen"	xxxxxxxxxxxx	"Bill Hansen"
11 char "Bill Hansen"	xxxxxx	"Bill H"
11 char "Bill Hansen"	x(12)	"Bill Hansen "
11 char "Bill Hansen"	"Mr. "x(12)	"Mr. Bill Hansen "
5 char "10000"	zzzzz.zz	"10000.00"
5 char "10000"	+zzzzz.zz	"+10000.00"
5 char "10000"	"Kr "zzzzz.zz	"Kr 10000.00"
5 char " 200"	+zzzzz.zz	" 200.00"
5 char "10000"	99999.99-	"10000.00 "
integer -200	99999.99-	"00200.00-"
●11 char "Bill Hansen"	>x(6)	" Bill H"
●11 char "Bill Hansen"	<x(6)	"Bill H "

● **These options are only allowed in the SCREEN-FORMs.**

APPENDIX B
STORAGE CODE

The storage code indicates how data is stored on disk. The storage code is required for data computation and converting one field to another.

In many cases, a storage code can be generated automatically from a display code.

Different programming languages and software packages may not able to handle all types of storage code. Remember this when choosing a storage code for data items.

STORAGE CODE DICTIONARY SYNTAX	**COBOL**	**FORTRAN-77**	**DISPLAY CODE**
ALPHANUMERIC(12)	PIC X(12)	CHARACTER*12 equivalenced with INTEGER( )	XXXXXXXXXXXX
INTEGER2	PIC S9999 COMP	INTEGER*2	-ZZZ9
INTEGER4	PIC S9(8) COMP	INTEGER*4	-ZZZZZZZ9
UNPACKED DEC(5,2)	PIC S99999V99	●N.A.	-ZZZZZ.ZZ
PACKED DEC(5,2)	PIC SZZZZ.ZZ COMP-3	●N.A.	-ZZZZZ.ZZ
PACKED DEC(12,2)	PIC SZ(12).ZZ COMP-3	●N.A.	-Z(12).ZZ

●N.A. = No arithmetic possible on this storage code.

APPENDIX C
DATA DICTIONARY INFORMATION

The DATA DICTIONARY will contain descriptions of all data in a database. The data contained in the Data Dictionary can also be used by other programs such as Query Languages, Report Generators, Screen Handling Programs and Program Generators.

The Data Dictionary information is set up in the following commands:

```
DATABASE INITIATE
DATABASE SYSTEM-REALM
DATABASE REALM
DATABASE ITEM
DATABASE SET
DATABASE GROUP
```

DEFINING THE DATA DICTIONARY

The following is an explanation of the Dictionary parameters:

PURPOSE	This information will be used as documentation of the database unit. It can also be used as HELP information while using the database online.
HEADING	This is usually a short text indicating the context of the database unit. It can also be used, for instance, as a leading text in screen displays or as report headers.
DISPLAY **(for data items only)**	This indicates how the stored data should be edited. It can also be used by a screen handler or a report writer for formatting information.
STORAGE **(for data items only)**	This information allows programs to convert data correctly from a stored bit pattern to a readable version and vice versa.

APPENDIX D
ROUTINES IN ABM-SIB-LIBRARY

Routines for Communication with the Application Database.

Routines in ABM-SIB-LIB		COBOL	FORTRAN
DDACCD	- ACCUMULATION of item values.	99	85
DDFEBL	- Find FIRST record between limits using given key.	95	85
DDFLBL	- Find LAST record between limits using given key.	95	85
DDFORG	- FORGET, nullify the effect of a REMEMBER call.	100	87
DDFREMB	- FORGET old and REMEMBER a new record or a search region.	100	87
DDFTCGT	- FIND a specific record and GET the record value.	96	86
DDFTCH	- FIND a specific record.	95	85
DDGET	- GET the relevant record, items or group items.	97	88
DDGETN	- GET (read) a number of records in a search region	98	90
DDGIXN	- GET (read) a number of index keys.	98	90
DDINKEY	- Reset search regions to maximum.	96	89
DDINSR	- INSERT an index key of a record.	97	89
DDMDFY	- MODIFY values of items or group items in a record	97	88
DDREMO	- REMOVE a manually maintained index key.	97	89
DDSTORE	- STORE a (part) of a record in its realm.	97	88
DDTRNSF	- TRANSFER of values between value buffers (for FORTRAN applications).	99	86

APPENDIX E
ROUTINES IN ABM-FOCUS-LIBRARY

Routines for Communication with Screen Forms.

The ABM-FC-LIBRARY contains the following routines:

	COBOL	FORTRAN
DDCFLDS - Clears fields/records or parts of records.	121	106
DDCLAT - Clears attributes.	127	106
DDCLFI - Closes an opened file.	128	106
DDCLMR - Clears "must-read" for fields/records.	125	107
DDCMSGE - Clears a message line.	128	107
DDCOPTF - Copies a displayed picture to file.	127	107
DDERROR - Decodes the error status and returns an error text.	130	114
DDGETRC - Gets field values from the total picture buffer	124	108
DDGMSGE - Writes a message to a message line and reads the answer.	130	108
DDGTEXT - Writes a message in a given line or column and reads the answer.	129	108
DDGTPIC - Gets a picture from a file, displays it and makes it ready.	129	109
DDINITE - Initiates and terminates the SCREEN part of a program.	121	109
DDOPFI - Opens a SINTRAN file for Write, Append access.	127	110
DDPUTRC - Puts field values into the total picture buffer	124	110
DDRFLDS - Reads fields/records or parts of records.	122	111
DDSETAT - Sets attributes.	126	112
DDSETMR - Sets "must-read" for fields/records.	125	113
DDWFLDS - Writes fields/records or parts of records.	123	112
DDWMSGE - Writes a message to the message line.	128	113

APPENDIX F
ROUTINES IN ABM-UTILITY-LIBRARY

The ABM-UTILITY-LIBRARY contains the following routines:

BDBCLS	- Closes the database.	137
BDBOPN	- Opens the database.	137
DDERMSG	- Gives an error message.	136

APPENDIX G
OPERATING THE ABMBASE

It is extremely important when using ABM, to have a fixed operation strategy. Take orderly **backups** and use mode files (or user-environment menu) that you know are free from errors. To avoid losing data and causing inconsistent internal structures in the database, always do a recovery when the machine stops or whenever SIBAS is aborted.

This appendix is an extract from the SIBAS II OPERATOR MANUAL. It provides you with some safe operating routines so you can avoid problems running ABM. For further information, see SIBAS II OPERATOR MANUAL.

In this appendix you will find descriptions of how to:

- Start ABMBASE with BEFORE-IMAGE-LOG and ROUTINE-LOG.
- Start ABMBASE, normal procedure.
- Take backup.
- Verify the structure of the ABMBASE.
- Start ABMBASE with recovery from backup and R-LOG.

START ABMBASE WITH BEFORE-IMAGE-LOG AND ROUTINE-LOG

Before Image Log (BIM-LOG)

BIM-LOG must be initiated.
The BIM-LOG is a part of the database and resides on the schema os-file (with filename <databasename:DATA>). Create the schema file as a contigous file with a size at least equal to the maximum BIM-LOG size.

Routine-log (R-LOG)

For greater security, run Routine-log (R-LOG) as well.

The R-LOG is a contigous file (with file name <databasename:LOGG>) which contains the DML calls/commands given to SIBAS and the input/output data to/from SIBAS. (@create-file <databasename:LOGG>, size where the size depends on how many SIBAS calls are performed, and how often backup is taken.)

The R-LOG must always be on a different disk from the database files in case of a disk-crash.

The ABM database ABMBASE must be assigned to a free SIBAS process before use. This is achieved through the START command in the interactive SIBAS-SERVICE module. Enter the user RT or SYSTEM in order to START (i.e. assigned to a free SIBAS process).

NOTE: The following points should be kept in mind:

- User RT must have been created as a FRIEND (@CREATE-FRIEND RT) and given Read/Write/Append access to the database files. (@SET-FRIEND ACCESS,RT,RWA)

- The database files must be existing and closed before the START command will work. SIBAS must be in READY state before the START command is given.

- The database must be in RUNNING state before applications can use it. (Give the RUN command.)

Example:

(Everything that YOU type is shown underlined.)

```
Enter user RT or SYSTEM:


@SIBAS-SERVICE↵

S I B A S  I I , version E
SIBAS-SERVICE, revision 10, XMSG used for remote communication.

Explanation ? N↵
Status for all SIBAS-processes ? N↵
SYSTEM NUMBER (00-11=LOCAL, x00-x11=REMOTE, x=DB-machine no):1↵

SIBAS-SYS-NO: 1,  SIBAS-500 STATE: READY
>>INITIATE-LOG↵
  OWNER:DIA-SYS↵
  DATABASE-NAME:ABMBASE↵
    1: INIT-R-LOG
    2: RESET-R-LOG
    3: REMOVE-R-LOG
    4: CONNECT-R-LOG
    5: INIT-BEFORE-IMAGE-LOG
    6: REMOVE-BEFORE-IMAGE-LOG

  CODE:1↵
  LOG-DIRECTORY, 4 CHAR.:P-TW↵
    MAX-SIZE-1K-PAGES
      WHEN CODE=5: BETWEEN APPROX. 504 AND 4096
      WHEN CODE=1: BETWEEN 2 AND 30000

  SIZE:1000↵
    2: DIRECT-R-LOG     WHEN CODE=1
    3: CIRCULAR-R-LOG   WHEN CODE=1
    .: C-P TRIGGER-SIZE WHEN CODE=5

  R-LOG-TYPE or C-P TRIGGER-SIZE:2↵
INITIATION MAY TAKE TIME -WAIT-
```

```
SIBAS-SYS-NO: 1,  SIBAS-500 STATE: READY
>>INITIATE-LOG↵
  OWNER:DIA-SYS↵
  DATABASE-NAME:ABMBASE↵
    1: INIT-R-LOG
    2: RESET-R-LOG
    3: REMOVE-R-LOG
    4: CONNECT-R-LOG
    5: INIT-BEFORE-IMAGE-LOG
    6: REMOVE-BEFORE-IMAGE-LOG

  CODE:5↵
  LOG-DIRECTORY, 4 CHAR.:P-TH↵
    MAX-SIZE-1K-PAGES
      WHEN CODE=5: BETWEEN APPROX. 504 AND 4096
      WHEN CODE=1: BETWEEN 2 AND 30000

  SIZE:800↵
    2: DIRECT-R-LOG     WHEN CODE=1
    3: CIRCULAR-R-LOG   WHEN CODE=1
    .: C-P TRIGGER-SIZE WHEN CODE=5

  R-LOG-TYPE or C-P TRIGGER-SIZE:0↵

SIBAS-SYS-NO: 1,  SIBAS-500 STATE: READY
>>START↵
  OWNER:DIA-SYS↵
  DATABASE-NAME:ABMBASE↵

SIBAS-SYS-NO: 1,  SIBAS-500 STATE: DBA
>>RUN↵
  NEW RUNFLAG ?:N↵

SIBAS-SYS-NO: 1,  SIBAS-500 STATE: RUNNING
>>DATABASE-STATUS↵
 BEFORE IMAGE LOG ACTIVE
 INTERFACE * LOG ADDRESS *                         TIME
STATISTICS * BLOCK  WORD * BASIC-UNIT HOUR/MIN./SEC. DAY/MONTH/YEAR
--------------------------------------------------------------------
CURRENT    *     1   004 *       41       10:42:01     30.01. 1986
INITIATION *     1   004 *        8       10:41:58     30.01. 1986
LAST OPEN  *     0   000 *        0       00:00:00     00.00.    0
LAST CLOSE *     0   000 *        0       00:00:00     00.00.    0
LAST CHECK *     0   000 *        8       10:41:58     30.01. 1986

FILE SIZE  *  2000   LOGGED CALLS =     0 LOG-TYPE: DIRECT
DATABASE (DIA-SYS )ABMBASE  IS OPENED BY   0 USERS
TOTAL NUMBER OF SIBAS CALLS EXECUTED SINCE START:          9
RUNFLAG 000000B,OFLOG NOT ALLOWED


SIBAS-SYS-NO: 1,  SIBAS-500 STATE: RUNNING
>>EXIT↵

 - EXIT -
```

START ABMBASE, NORMAL PROCEDURE

Normal start of the ABMBASE. This procedure uses RECOVER to reprocess the database to achieve better security.

Example:

(Everything that YOU type is shown underlined.)

Enter user SYSTEM

```
@SIBAS-SERVICE↵

S I B A S  I I , version E
SIBAS-SERVICE, revision 10, XMSG used for remote communication.

Explanation ? N↵
Status for all SIBAS-processes ? N↵
SYSTEM NUMBER (00-11=LOCAL, x00-x11=REMOTE, x=DB-machine no):1↵

SIBAS-SYS-NO: 1,  SIBAS-500 STATE: READY
>>START↵
  OWNER:DIA-SYS↵
  DATABASE-NAME:ABMBASE↵

SIBAS-SYS-NO: 1,  SIBAS-500 STATE: DBA
>>RECOVER↵

SIBAS-SYS-NO: 1,  SIBAS-500 STATE: RECOVERY
>>STANDARD-REPRO↵
  ROLL-BACK TO LAST CHECKPOINT ? :YES↵
  PASSWORD:,,↵

 REPROCESSING WILL NORMALLY TAKE SOME TIME - WAIT -

 REPROCESSING/LISTING  STOPPED, REASON:
 END OF LOG FOUND

SIBAS-SYS-NO: 1,  SIBAS-500 STATE: RECOVERY
>>FINISH↵

WILL YOU REALLY FINISH RECOVERY ?Y↵

SIBAS-SYS-NO: 1,  SIBAS-500 STATE: DBA
>>FORCE-CLOSE↵
  OCTAL RUN-ID, (EXAMPLE  40106B), OR -1 IF ALL USERS:-1↵

>>RUN↵


SIBAS-SYS-NO: 1,  SIBAS-500 STATE: RUNNING
>>EXIT↵

 - EXIT -
```

TAKE BACKUP

Take regular backup of the ABM catalog and your application system.

> NOTE:
> Remember to take backup of the database files (ABMBASE:DATA, ABMOSFI:DATA, ABMBASE:LOGG) and your own form files (:FABM files).

To avoid inconsistent databases, take backup at regular intervals.

The database must always be stopped before backup is taken. (Stop the database or the machine). The database is stopped by giving the STOP command in the SIBAS-SERVICE module.

Backup procedure:

- Enter the SIBAS-SERVICE module and use the command SET-SIBAS-UNAVAILABLE.
- Broadcast a message.
- Check that the database is closed (use the command DATABASE-STATUS in the SIBAS-SERVICE module).
- Use the STOP command to stop SIBAS.
- Log in as the user who owns the database and run a mode file to VERIFY the database. If the verification indicates error in the database, perform START ABMBASE WITH RECOVERY FROM BACKUP AND R-LOG.
- If the verification shows that the database is okay, carry out the backup.
- RESET the R-LOG !!
  Use the command INITIATE-LOG with code 2.
- Open the database for update and close the database.
- Use the START command to start SIBAS and the RUN command to set SIBAS in running state.
- SET-SIBAS-AVAILABLE to make it available for users.
- Broadcast a message.

VERIFY THE STRUCTURE OF THE ABMBASE

To VERIFY a database means to check and see if the internal database structure is correct. Run a database verification before taking the backup copy to avoid making a copy of an erroneous database.

Enter RT and stop ABMBASE.

Enter user DIA-SYSTEM.

```
@SIB2-DBM↵

Explanation ? NO↵
Interactive ? NO↵
Input-file  : TERMINAL↵
List-file   : "VERIFIED:SYMB"↵
START ABMBASE.↵
READY ALL.↵
FREE-SPACE-STAT.↵
VERIFY MODE READ-ONLY.↵
VERIFY PAGE-LINK DATABASE.↵
VERIFY INDEX DATABASE.↵
VERIFY SET DATABASE.↵
EXIT.↵
```

The result is found in the file VERIFIED:SYMB

If the database verification gives error messages, reprocess the R-LOG corresponding to the backup. (Start ABMBASE with recovery from backup and R-LOG.)

NOTE:
For those using SIBAS version E or an older version:
If you have subfunctions with no connection to a form or no connection to a subschema, you will get a message when doing 'VERIFY SET DATABASE'. The message says that 'number of records read via set does not correspond to number of records read in physical order'.

The difference between members read in physical order and those read via set is equal to the number of subfunctions wich do not have connections to both a form and a subschema.

You can ignore this message.

NOTE:
Do not use the command "VERIFY MODE REGENERATE". The ABM catalog version A and B contains three manually maintained sets, which will be completely disconnected by this command.

START ABMBASE WITH RECOVERY FROM BACKUP AND R-LOG

This should only be done if the database verification indicates error in the database, or if there has been a disk crash. Use the R-LOG to update the backup files.

Example:

- Enter user DIA-SYSTEM
- Install the backup
- Enter user SYSTEM

  @SIBAS-SERVICE↵

- Perform the same commands as shown on page 221 with one exception: for "ROLL-BACK TO LAST CHECKPOINT?", answer NO↵.

APPENDIX H
DATA TRANSFER BETWEEN APPLICATION ROUTINES

In this section we will describe the possible ways of transferring data between a program and a subroutine (or between two subroutines) where both have generated INCLUDE/COPY files. In other words, we will describe data transfer between two subfunctions in ABM.

Data may be transferred as variables in a parameter list.

DATA TRANSFER IN FORTRAN:

For FORTRAN, local variables must be declared in the called routine, because variables used in an equivalence cannot be used as formal parameters in a subroutine.

Values in input parameters must be transferred from these formal parameters to the variables generated in the INCLUDE-files at the start of the subroutine.

DATA TRANSFER IN COBOL

In COBOL, this will be no problem. You can use the 01-level as a parameter in a subroutine and use COPY on the whole area. The use of EXPORT/IMPORT is also a possibility.

Data may be transferred by transferring the involved item list, field name list and value buffers as parameters in the call. The called routine gets the values by using the routine DDTRNSC from the parameters to its own lists and value buffers.

Data may be transferred through the picture. If both routines use the same picture, the routine DDGETRC may be used to transfer data from the common area storing the complete picture to the local buffers in the called routine.

APPENDIX I
COMPILATION ERRORS

Compilation errors that might occur:

Too many local variables (FORTRAN system limitation).

Too many variables generated in the INCLUDE file i.e. the picture and subschema contain too many variables.

Solution:
Split the program in two (for example, a program and a subroutine) each having its own subschema or put a few of the generated arrays into a common area.

Illegal equivalence:

The same variable name is used for two or more variables. That is, one variable is equivalenced into more than one place in the value buffer.

Normally the field definition in the picture is not unique, and the field definition for one of the fields should be changed.

Illegal COBOL declarations:

If, for example, the field definition in the picture is not unique, this will result in double (or more) defined variable names that later on can not be referred to in a unique way. The COBOL compiler will give a warning only if you try to refer to one of these.

The solution is to go back and redefine one (or more) of the fields involved.

APPENDIX J
HOW TO LOAD AN ABM APPLICATION

Below is an example loading an ABM application on ND-100:

```
@brf-linker
*prog-file "<name-of-application>"
*load ABM-FC-BLOCK                    % Start by loading ABM-FC-BLOCK which
                                      % contains some common variables used
                                      % by ABM itself.
*load <the application programs>
*load ABM-UTILITY-LIBRARY             % Contains some special routines, such
                                      % as DDERMSG.
*load ABM-SIBAS-LIBRARY               % ABM interface to Sibas.
*load SIBAS-LIBRARY
*load ABM-FOCUS-LIBRARY               % ABM interface to Focus.
*load FOCUS-CODE                      % Part of Focus library containing code.
*load FOCUS-DATA                      % Part of Focus library containing data.
*load COBOL-LIBRARY                   % If the application is written in
                                      % COBOL.
*LOAD FORTRAN-LIBRARY                 % The ABM interface is written in
                                      % FORTRAN, so this library has to be
                                      % loaded even when your application
                                      % is written in COBOL.
*exit
```

Loading an ABM application on ND-500:

Linkage Loader gives you several possibilities for loading, but the simplest way is to load everything on one domain.

```
@linkage-loader
*set-domain "example"
*open-seg "example",,,
*load ABM-FC-BLOCK
*load <the application programs>
*load ABM-UTILITY-LIBRARY
*load ABM-SIBAS-LIBRARY
*FORCE-SEGMENT-LINK ()SIBAS-LIBRARY
*FORCE-SEGMENT-LINK ()SIBAS-MESSAGE
*LOAD ABM-FC-LIBRARY
*LOAD FOCUS-CODE
*LOAD FOCUS-DATA
*LOAD COBOL-LIBRARY
*LOAD FORTRAN-LIBRARY
*EXIT
```

If you want to use TRUE <1> , you ought to compile ABM-FC-LIB with "conditional compiling", with the T flag. You can then use TRUE's broadcast system (listen to broadcast).

Do not load the ABM-FOCUS libraries before the SIBAS libraries (ABM-SIB-LIB and SIBAS-LIB) are loaded.

<1>   Transaction User Environment

APPENDIX K ERROR MESSAGES

This appendix contains error messages with standard error codes (SEC) from the ABM-FC-LIB and ABM-SIB-LIB routines. (See the FOCUS reference manual and the SIBAS user manual for other SEC's).

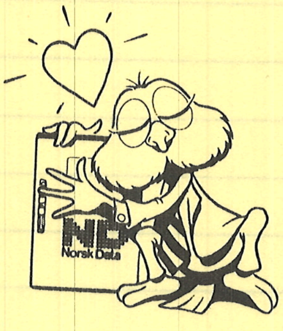
SEC OCTAL:	ERROR MESSAGE AND EXPLANATION:
16601	Error from ABM error handling routines.
	Calling DDERROR or DDERMSG with a wrong parameter or after a successfully executed DD-call.
16602	Unknown form name.
	The form name given in the reference table does not match the form name used in DDGTPIC.
16603	Trying to use too many forms at the same time.
16604	NOLINE parameter out of range.
	You have probably forgotten to reset the NOLINE parameter after the last DD-call.
16605	LINE parameter out of range.
	You have probably forgotten to reset the LINE parameter after the last DD-call.
16606	NOLINE parameter too large.
	The number of record occurences for this record is not that large.
16607	Invalid message type.
	The message has to start with the bytes "+:", "0:" or "-:".
16610	Illegal termination of message.
	The message is too long or not terminated by an "'".

16611	Too many items in result-item-list or illegal termination. You have probably specified too many items in your item-list. Try to use the "-:" variant instead of the "+:". The termination character "*" can be missing or wrongly placed.
16612	Result-item-list is empty. No match on field or item names is found. Check that you are working with the correct record.
16613	Subitem-list is not a subset of total-item-list. You have specified items/fields that are not part of the specified record. See the ASSDDI- or ASSDDC- files for the right item/field names.
16614	Illegal start word in subitem-list. The subitem list has to start with the bytes "+:", "0:" or "-:".
16615	Subitem-list is empty. The specification "+:*" or "-:*" is not allowed.
16616	Too long datarecord for result-item-list. The maximum data value buffer in ABM-SIB-LIB calls is 256 16-bit words.
16617	ICODE parameter in DDFREMB/DDFORG out of range. See the allowed parameter values in the description of the routines.
16620	Illegal or unknown type of accumulation. From the DDACCD call.
16621	Illegal values in set attribute. Unknown or illegal value in the field attribute value to the DDSETAT call.

Index

*************** SEND US YOUR COMMENTS!!! ***************

Are you frustrated because of unclear information in this manual? Do you have trouble finding things? Why don't you join the Reader's Club and send us a note? You will receive a membership card — and an answer to your comments.



Please let us know if you

* find errors
* cannot understand information
* cannot find information
* find needless information

Do you think we could improve the manual by rearranging the contents? You could also tell us if you like the manual!

************* HELP YOURSELF BY HELPING US!! *************

Manual name: ABM User Manual

Manual number: ND–60.203.2 EN

What problems do you have? (use extra pages if needed) ______________________

Do you have suggestions for improving this manual ? ______________________

Your name: ______________________ Date: __________

Company: ______________________ Position: __________

Address: ______________________

What are you using this manual for ? ______________________

NOTE!
This form is primarily for documentation errors. Software and system errors should be reported on Customer System Reports.

Send to:
Norsk Data A.S
Documentation Department
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

Norsk Data's answer will be found on reverse side

Answer from Norsk Data

Answered by _______________ Date _______

Norsk Data A.S
Documentation Department
P.O. Box 25, Bogerud
0621 Oslo6, Norway

Systems that put people first