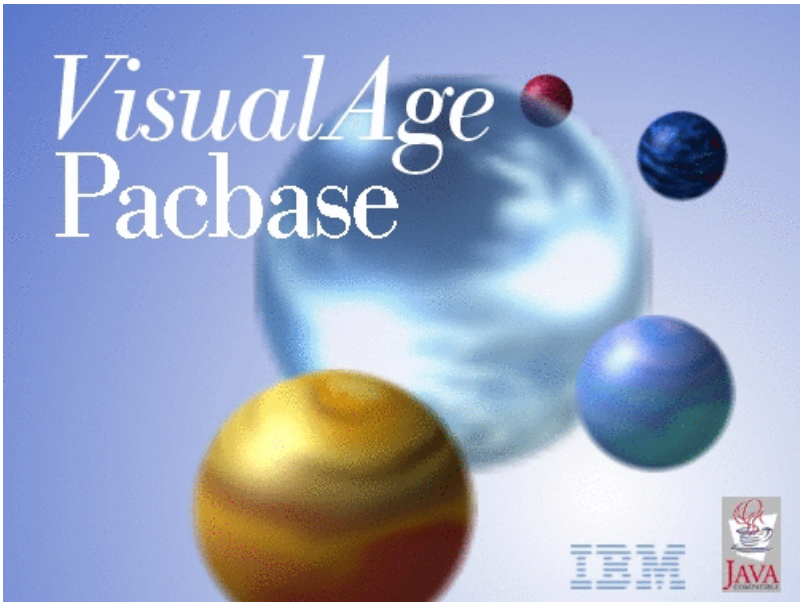
VisualAge Pacbase

Structured Code

Version 3.5



VisualAge Pacbase

Structured Code

Version 3.5

Note

Before using this document, read the general information under “Notices” on page v.

You may consult or download the complete up-to-date collection of the VisualAge Pacbase documentation from the VisualAge Pacbase Support Center at:

http://www.ibm.com/support/docview.wss?rs=37&context=SSEP67&uid=swg27005477

Consult the Catalog section in the Documentation home page to make sure you have the most recent edition of this document.

Second Edition (November 2006)

This edition applies to the following licensed programs:
- VisualAge Pacbase Version 3.5

Comments on publications (including document reference number) should be sent electronically through the Support Center Web site at: http://www.ibm.com/software/awdtools/vapacbase/support.html or to the following postal address:

IBM Paris Laboratory
1, place Jean–Baptiste Clément
93881 Noisy-le-Grand, France.

Contents

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk NY 10504–1785, U.S.A.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Paris Laboratory, SMC Department, 1 place J.B.Clément, 93881 Noisy-Le-Grand Cedex. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

IBM may change this publication, the product described herein, or both.

Trademarks

IBM is a trademark of International Business Machines Corporation, Inc. AIX, AS/400, CICS, CICS/MVS, CICS/VSE, COBOL/2, DB2, IMS, MQSeries, OS/2, PACBASE, RACF, RS/6000, SQL/DS, TeamConnection, and VisualAge are trademarks of International Business Machines Corporation, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

All other company, product, and service names may be trademarks of their respective owners.

Chapter 1. Introduction

Purpose of the Manual

This manual describes the specifications of the Structured Code function which may be used independently from the complementary functions: the Pacbench C/S, the On-Line Systems Development (OLSD) and the Batch Systems Development (BSD) functions.

Some information concerning these functions is also related to the Structured Code function. There are instances where the descriptions are entered in this manual only in order to avoid redundancy.

PREREQUISITES

In order to understand the content of this manual, you should have taken the VisualAge Pacbase Concepts and Facilities course and have an understanding of Structured Code concepts.

Also, you are expected to have thoroughly read the following manuals:

. The 'Character Mode User Interface' guide,

. The 'Data Dictionary' manual.

Description Principles

In this manual, the entities and screens managed by VisualAge Pacbase are described in two parts:

- An introductory comment explaining the purpose and the general characteristics of the entity or screen,
- A detailed description of each screen, including the input fields for on-line screens data entry into the Database.

For the description of batch input, refer to the 'Developer's Procedures' manual.

All on-line fields described in this manual are assigned an order number. These numbers are displayed on the screen examples which appear before the input field descriptions and allow for easy identification of a given field.

NOTE: If you use Developer workbench, refer to the on-line Help.

NOTE: If you use the VisualAge Pacbase WorkStation, refer to the 'WorkStation User Interface' guide which documents the corresponding windows.

Introduction to Structured Code

The Structured Code allows programmation teams to realize management programs. Linked with the Specifications Dictionary, it offers the following possibilities:

- Definition of Programs, using a Definition screen which contains the general characteristics of a Program (Program name, type, explicit keywords, etc.),
- Calling Data Structures already described in the Specifications Dictionary. They can be called as many times as necessary in one or several Programs.
- Description of work areas, either by calling existing Data Structures or describing Data Structures specific to the Program.
- Description of procedures for a given Program.
- Definition of Macro-Structures, a Macro-Structure being a parameterizable set of Structured Code lines that can be called in different Programs.
- Calling Macro-Structures in a Program. The same Macro-Structure may be invoked several times by the same Program.

WRITING PROCEDURAL CODE

Automatic functions generated for a Dialogue give standard solutions to standard problems, but do not always respond to all of the processing needs.

You can determine, at the development stage of an application, the way in which you wish to solve the problem. It is recommended to use the standard solutions provided by automatic functions as often as possible.

However, it is not recommended to try to fit a very complex procedure into the automatic functions if it is going to complicate the clear structure of the program. It is very important to completely think through the specifics of the required procedures when a program is being developed.

If you have special needs, you may complete or partially replace standard procedures using procedural code.

The Procedural Code (-P) is used to enter detailed specific procedures. Its use ensures the structuring of procedures, as well as the readability and convertibility of programs.

ADVANTAGES OF USING THE PROCEDURAL CODE

The procedural code is a high-level language used by programming teams in order to develop and implement any business-oriented programs. Procedural Code, which must be used in conjunction with the Specifications Dictionary, presents the following advantages:

- Improved conciseness compared to COBOL, due to the simplification of COBOL commands and syntax;
- Hierarchical organization of procedures, which does not exist in COBOL;
- Structured programming using the following types of structures: BLOCK (BL), IF THEN (IT), ELSE (EL), DO WHILE (DW), DO UNTIL (DU), CASE OF (CO), etc.
- Improved portability due to independence from any specific hardware, and the ability to generate in a COBOL code adapted to each computer system.

CROSS-REFERENCES

Since the Structured Code must be used in conjunction with the Specifications Dictionary, all the facilities of the Specifications Dictionary function are available.

Specifically, the user can establish cross-references between Data Elements, Data Structures and Programs written with the Structured Code function.

These cross-references, which are extremely useful during program development, become a valuable tool during program maintenance since they allow the user to immediately evaluate the consequences of any changes made.

TYPES OF PROGRAMS THAT CAN BE DEVELOPED

Both batch and on-line programs can be written in Structured Code using the various features offered by parameterized Macro-Structures, i.e, technical procedures associated with on-line Programs or Database Management Systems.

The Structured Code function does not allow for the automatic Description of Screens used in on-line Programs.

It does, however, provide the following:

- The generation of the COBOL source code, ready to be compiled and adapted to the operating system.
- Improved portability. Input in a single field enables the user to specify to which operating system a program should be adapted.
- Consistency between the described data and the generated COBOL code. Both originate from one common source: the Database.

- Customizing the automatically generated functions provided via the Batch or On-Line Systems Development functions.

Managed Entities/Associated Screens

The Structured Code function manages a single entity, the Program entity, which is defined and described on the following screens:

- Program Definition
- Call of Parameterized Macro-Structures (-CP),
- Call of Data Structures (-CD),
- Beginning Insertions (-B),
- Work Areas (-W),
- Procedural Code (-P).

The Program Definition (-P) screen is used to define the Program code, the Program name, and its general characteristics.

The Call of P.M.S.'s (-CP) screen is used to include lines described in other programs. This is done by replacing the parameters indicated with specific values.

The Data Structures (-CD) screen is used to get the description of I/O fields from the Program.

The Beginning Insertions (-B) screen is used to modify the ENVIRONMENT DIVISION statements that are generated automatically as a result of Data Structure calls.

The Work Areas (-W) screen is used to modify fy the WORKING-STORAGE and/or LINKAGE SECTIONs, supplementing the descriptions obtained automatically.

The Procedural Code (-P) screen is used to write sequences of instructions in a portable, structured, and hierarchical format.

&2NOTE& For information on Batch access lines, refer to the 'Developer's Procedures' manual.

As all entities, programs can be documented by Comment lines, by text assignment (see the 'Data Dictionary' manual).

REVERSE ENGINEERED PROGRAMS

Programs that have been ″reverse engineered″ include only the following:

- Work Area (-W) lines,
- Source Code (-SC) lines (COBOL source code).

It is possible to add Structured Code (-W and -P lines) and Calls of Marcro-Structures (-CP lines) to these programs, and then regenerate them. Call of Data Structures (-CD) and Beginning Insertions (-B) lines are ignored.

Chapter 2. Parameterized Macro-Structures

Overview

INTRODUCTION

The purpose of a Parameterized Macro-Structure is to standardize sequences of procedural code, with possible variations, in order to use them:

- One or more times in one program,
- In several different programs.

DEFINITION

A Parameterized Macro-Structure (P.M.S) is defined on a Program Definition (P) screen. It is a set of Beginning Insertions (-B), Work Areas (-W) and Procedural code (-P) lines which produces one or more sequences of statements which can be used in one or more programs.

A P.M.S. is not a sub-program. A sub-program can only contain consecutive statements. A P.M.S. can contain non-consecutive statements. It is possible, however, to call a sub-program from a P.M.S.

A PMS can also be defined from a program imported via the Reverse Engineering function. It would contain '-W', '-SC' (Source Code lines from the "reversed" program), and '-P' lines (if "reversed" program's PROCEDURE DIVISION has been modified). In this case, this PMS in only taken into account in "reversed" programs (TYPE AND STRUCTURE OF PROGRAM = 'S').

PRINCIPLES

When a P.M.S. is called into a Program, the request for a description of the Program (DCP, DCO) and for its generation (GCP, GCO) will produce the Macro-Structure(s) interspersed within the Program according to the key (Function/Sub-Function code etc.). Parameters (if any) are resolved.

As a result, P.M.S. instructions are part of the Program.

TYPES OF MACRO-STRUCTURES

Generally speaking, Parameterized Macro-Structures are used to describe functions that are common to several Programs, or to several procedures of the same Program.

There are six basic types of Macro-Structures:

- A 'general program outline type' used to give a Program a standard structure which takes into account all the Program development standard followed at the user site. This type of PMS is also used to describe a body of technical (system-oriented) procedures that are linked to the use of a TP Monitor or a DBMS;
- A 'technical (system-oriented) functions type' used to standardize specific commands, such as the input/output procedures of a DBMS (Read, Modify, Suppress, etc.);
- A 'complex technical functions type' used to resolve all the complicated procedures involved in a DBMS, whether or not in an on-line environment, such as validation management, the sequential read of a file, the complete technical procedures for Database update, the particular access path used in a Database, etc. In general, this type is a combination of elementary technical functions that the user has to rewrite in order to minimize the task of connecting elementary P.M.S.'s to one another;
- A 'general function type' used to resolve certain procedures that are common to a set of applications, such as date validation, date transformation, or "shop" standards for on-line error message handling. The user should keep in mind that this type of PMS is independent of the computer system, the TP Monitor and the DBMS used;
- A 'specific function type' used to 'harmonize' the development of programs that make up a system. For example, to standardize the presentation of certain reports or screens by using common procedures defined in a P.M.S.
- A type used to create cross-references; for example, if a P.M.S. calls a sub-program, you can automatically find out what Programs use that sub-program.

DIFFERENCE BETWEEN P.M.S.'S AND SUB-PROGRAMS

The user must often decide whether to use a sub-program or P.M.S. to consolidate all the procedures that are common to several Programs. In order to find the answer, the user must ask several questions:

. Are the common procedures consecutive?

. Is the position of these procedures defined?

. Is the number of parameters used important?

. Are these procedures executed as a general rule?

Answers to these questions will help determine which procedures should be executed in a sub-program and which should be executed in a P.M.S.

- If they are not consecutive ==> P.M.S.
- If the position is already defined ==> P.M.S.
- If the number of parameters is important ==> SP.
- If the procedures are not executed as a general rule ==> SP.

PARAMETERIZING A PROCEDURAL CODE (-P) SCREEN KEY

The System allows the user to parameterize the major part of the Procedural Code (-P) screen keys (Function code, sub-Function code, and the first two characters of the LINE NUMBER). As a recommendation, before writing a Macro-Structure, try to structure the procedure to be written. Try to minimize the number of parameters in the key, so as to:

. Facilitate usage of the P.M.S.,

. Obtain Programs with a homogenous structure.

. Minimize the resolution time of a P.M.S.

As a general rule, it is not recommended to use a P.M.S. instead of a simple line or two of Procedural Code. The latter solution may be more efficient with respect to performance at generation time and also to limit the number of necessary P.M.S. calls.

DOCUMENTATION OF A P.M.S.

Good documentation of a P.M.S. is important. It gives the user the information needed to use the P.M.S. properly: what each parameter means, which functions and/or sub-functions are used, etc.

A P.M.S. can be documented in two different ways:

- In the same way as any program, via the Comments (-GC) screen,
- On the P.M.S.'s X-Reference to Programs (-XP) or to On-Line screens (-XO) screen.

Note that Comment lines entered on the General Documentation screen will not appear in sub-reports of a Program Description (DCP, GCP; DCO, GCO) whereas the cross-reference lines will. This may be the most effective way to document the meaning of the parameters, however, since the lines will reappear each time the Macro is called, brevity is advisable.

OVERRIDE OF A P.M.S. LINE

Given the same key, Procedural Code (-P), Work Areas (-W) and Beginning Insertions (-B) lines of a Program override PMS lines. It is better to design Parameterized Macro-Structures so that as few lines as possible will be overridden by the Programs.

Each overridden P.M.S. line will appear in the Program Description (DCP, GCP; DCO, GCO) preceded by an asterisk. This can make a Program harder to read. It is preferable to include as few lines of this type as possible in a P.M.S.

If there is a Macro-Structure line key with a matching key in another Macro-Structure called into the same Program, neither of the lines are considered for processing. These lines will be identified with an asterisk in the Program Description.

In cases where the identical key appears several times, the maximum number of comment lines (with an asterisk) that will appear in the Program Description is ten. These lines will not appear in the generated code.

CONSISTENCY OF PARAMETERS

It happens frequently that one Program calls several P.M.S.'s. The user should check that the parameters are used consistently. For example, if two different P.M.S.'s are called into the same Program, and both use a Data Structure code as a parameter, both P.M.S.'s would ideally have that code in the same position. This has a twofold advantage of being easier for the programmer, and of presenting the same type of information in the same order in a Program.

While this is not always possible, it would be wise to consider the placement of parameters in existing P.M.S.'s prior to designing a P.M.S.

NOTE:: Any Program already defined in the Database can be used as a non-Parameterized Macro-Structure.

REMINDERS

The purpose of a Parameterized Macro-Structure (P.M.S.) is to standardize functions common to several Programs. A called P.M.S. is a complement to the generation possibilities of the System.

Usually, a P.M.S. appears in a Program Description as if its lines had been directly entered by a programmer.

PURPOSE OF NON-EXPANDED MACRO-STRUCTURES

Non-expanded P.M.S.'s are reserved for batch Programs only.

Some P.M.S.'s are called many times in several Programs, and the programmer considers them as part of his/her own standard environment. In this case, there is no need to see the actual lines of these P.M.S.'s in the Program Description.

However, these lines are taken into account when the Program is being generated.

ADVANTAGES AND DISADVANTAGES

When a P.M.S. is called in a Program, an index, called an 'Expansion Index,' is created for each P.M.S. description line.

The system creates these indexes in order to display P.M.S. lines in the description of the calling Program.

A P.M.S. call in a Program may have the following disadvantages:

- Slow response time during update of the P.M.S. to be expanded for all users (serialization of updates);
- Disorganization of the Index File (AN) by mass insertion of keys that are often contiguous;
- Large increase in the number of records in the Index File, thus lengthening execution time of the batch save, restore and reorganization procedures.

Using non-expanded P.M.S.'s can make improvements in these three areas provided the user does not need to view the P.M.S. to update it on-line. Response time is thus improved because non-expanded P.M.S.'s do not create extra records in the Index File (AN).

However, using non-expanded P.M.S.'s may have the following disadvantages:

- For a non-expanded P.M.S. which is not displayed on-line, writing and maintaining the Program may be more difficult;
- For a P.M.S. expansion which occurs during a Program extraction for generation and printing, the execution time of the GPRT utility procedure is increased.

On the Call of P.M.S.'s screen (-CP), non-expanded P.M.S.'s are indicated with an 'N' in the Expansion ('E') field.

The Program Entity

The purpose of the Program entity with respect to the Structured Code function is to define Parameterized Macro-Structures.

GENERAL CHARACTERISTICS

Although the primary focus in this manual is to provide the Structured Code meaning of the Program entity screens and their fields, Descriptions have been included in their entirety. This is due to the fact that the Program entity is also used to write batch programs. This means that the user may easily convert a suitable Program into a Macro.

NOTE: Macro-Structures do not take Call of Data Structures lines into account, but Programs do.

The Program entity contains:

- A required Definition screen (P), giving general characteristics (Program code, keywords, etc.),
- Comment lines entered on the General Documentation screen or X-reference to Programs / On-line screens, to provide useful and/or necessary information,
- Several types of description lines:
  - Beginning Insertions (-B) lines which enable the user to modify the IDENTIFICATION DIVISION, and the ENVIRONMENT DIVISION that is generated, up to and including the 'DATA DIVISION' and 'FILE SECTION' statements.
  - Work Areas (-W) lines supplement the DATA DIVISION in the generated Program,
  - Procedural Code (-P) lines customize the PROCEDURE DIVISION in the generated Program,
  - Source Code (-SC) lines ("reversed" program only).

INPUT SPECIFICATIONS

The program classification code of a non-expanded P.M.S. is 'N' (instead of 'M' for a regular P.M.S.) and is entered in the PROGRAM CLASSIFICATION CODE field on the Definition screen (P) of the P.M.S. (This code has a documentary value in the sense that it does not affect the generated code).

The TYPE OF COBOL TO GENERATE for a Macro is normally 'N' so that the variant is determined by defaulting to the variant selected by the batch or on-line Program to which the Macro is attached. (This also improves portability).

PROGRAM STRUCTURE

Every program is organized as a set of successive processing steps that are performed either as a loop (in batch) or in an execution (on-line). These processing steps include:

- getting the data,
- checks,
- updates,
- printings,
- returning the output.

Each one these processing steps consists of a group of homogeneous sequences of instructions called "functions."

The Program is structured by two supplementary principles:

- Linear linking of functions in the logical order of their execution, with each executing a functional or technological task in the Program. Each function is identified by a code from 0A to 99.
- Hierarchical structuring of the processing steps in each function. A function can be broken down into sub-functions, which, in turn, can be further broken down into sub-functions, and so on.

Functions and sub-functions follow one another in the order of their codes, as determined by the EBCDIC collating sequence, with letters preceding numbers, regardless of the sorting sequence in effect for the hardware being used.

PRINCIPLES OF GENERATING A COBOL PROGRAM

Programs developed under VisualAge Pacbase are generated upon request in the COBOL variant that corresponds to the hardware and the compiler for which they are intended.

The IDENTIFICATION DIVISION of the COBOL Program is generated from the Program Definition line. This line can be modified ('-B').

The ENVIRONMENT DIVISION and the FILE SECTION are generated from Data Structure calls in the Program ('-CD'). They can be completed or modified ('-B').

The other sections of the DATA DIVISION are generated from Data Structure calls. They can be completed or modified ('-W').

The PROCEDURE DIVISION is generated from Data Structure or Segment calls and from processing descriptions in Procedural code ('-P').

Macro-structure call lines are used to call all the other pre-described Procedural code lines (see the chapter 'Macro-Structures').

CONSTANTS OF PROGRAMS

In the WORKING-STORAGE SECTION of all Programs, the System generates a PAC-CONSTANTS field in which the following are defined:

- the generation session number of the Program,
- the code of the Library in which the Program is defined,
- the generation date of the Program (MM/DD/YY if user language = 'E', or DD/MM/YY otherwise),
- the code of the Program,
- the code of the user who requested the generation,
- the generation time of the Program,
- the external name of the Program,
- the code of the Database,
- the generation date with century (MM/DD/CCYY if user language = 'E', or DD/MM/CCYY otherwise),
- the release,
- the date of the generator,
- the date of the generation skeleton.

These fields can be used in the Program execution report. They are preceded by the literal 'WORKING', which can serve as a marker in a dump in the event of an execution problem.

DEFINITION

A Program is defined on the Program Definition (P) screen. The user enters a code, a name and the main characteristics of the Program. It is accessed by entering the following input in the CHOICE field:

CH: P.....

```
                  PURCHASING MANAGEMENT SYSTEM        SG000008.LILI.CIV.1583

PROGRAM CODE                PO0001   1

PROGRAM NAME.....................: VENDOR 2EPORTS

CODE FOR SEQUENCE OF GENERATION....: PO0001 3

TYPE OF CODE TO GENERATE...........: 0      4
COBOL NUMBERING AND ALIGNMENT OPT..:        5
CONTROL CARDS IN FRONT OF PROGRAM..: B      6
CONTROL CARDS IN BACK OF PROGRAM...: B      7
COBOL PROGRAM-ID...................: PO0001 8
MODE OF PROGRAMMING................: P      9
TYPE AND STRUCTURE OF PROGRAM......: B      10
PROGRAM CLASSIFICATION CODE........: P   PRO11AM
TYPE OF PRESENCE VALIDATION........:        12
SQL INDICATORS GENERATION WITH '-' :        13

EXPLICIT KEYWORDS..:  14

UPDATED BY.........:  XXXX        ON: 06/10/2001  AT: 18:59:31    LIB: CIV

SESSION NUMBER.....: 0059        LIBRARY......: CIV    LOCK....:
O: C1 CH: Ppo0001                                            ACTION:
```

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	6		PROGRAM CODE (REQUIRED)
			Code identifying the program in the library.
2	30		PROGRAM NAME (REQUIRED IN CREAT)
			It must be as explicit as possible since the implicit keywords are created from this name.
3	6		CODE FOR SEQUENCE OF GENERATION
			Default option: PROGRAM CODE in the VisualAge Pacbase Library.
			Programs are sorted on this code in the generated program stream.
4	1		TYPE OF COBOL TO GENERATE
			Specifies the COBOL variant for the generated Program.
			The default value at creation is the value of the GENERATED LANGUAGE field in the Library Definition.
			Compatibility of Programs generated with Cobol 85, Cobol II, Cobol/370, Cobol OS/390 operates according to the value of the GENERATED LANGUAGE in the Library.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'N'	No adaptation to a language variant.
			It is used to prevent program generation.
		'0'	IBM MVS/ESA OS/390
		'1'	IBM DOS/VSE
		'3'	UNIX, WINDOWS
		'4'	BULL GCOS7
		'5'	BULL GCOS8
		'8'	UNISYS A Series
		'F'	TANDEM
		'I'	DEC/VAX VMS
		'K'	ICL 2900
		'O'	AS/400
		'U'	UNISYS 2200 Series
		'X'	IBM MVS/ESA OS/390
		'Q'	ACUCOBOL
		'C'	Extraction of COBOL Source Code. (Refer to chapter 'Appendix: Pure COBOL Source Code' in the 'Structured Code' manual).
5	1		COBOL NUMBERING AND ALIGNMENT OPTION
			This option can be used to suppress numbering or the identification of a program or to modify the justification of the generated program lines.
		blank	Numbering, justification and identification of program in accordance with the standard COBOL line (default value).
		'1'	Suppression of numbering.
		'2'	Suppression of numbering and justification of statements (columns 8 to 71 inclusive) in column 1.
		'3'	Standard numbering and justification, suppression of program identification.
		'4'	Suppression of numbering and program identification.
		'5'	Suppression of numbering and of program identification justification of instructions (columns 8 to 71 inclusive) in column 1.
6	1		Control cards in front of programs
			Enter the one-character code that identifies the job card to be inserted before the generated program.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			Default: Code entered on the Library Definition Screen
7	1		CONTROL CARDS IN BACK OF PROGRAMS
			Enter the one-character code that identifies the job card to be inserted after the generated program.
			Default: Code entered on the Library Definition Screen
8	8		COBOL PROGRAM-ID
			(Default value at generation: CODE FOR SEQUENCE OF GENERATION.)
			This code identifies the generated program:
			.in the IDENTIFICATION DIVISION,
			.in a source module library,
			.in the library of executable modules.
			This code intervenes (totally or partially) in the job control language lines generated before or after the program.
9	1		MODE OF PROGRAMMING
			Structured Code
		'P'	Default value when creating a Library. Programming in Structured Code on '-P' lines (Procedural Code).
			Cobol generator (in conjunction with the Reverse Engineering function)
		'S'	Specific procedures composed of Source Code (-SC) and Procedural Code (-P).
			With this value, the Type and structure of Program field must also be 'S'.
		'8'	Programming with '-8' type of lines.
			Used only to maintain applications written with former VisualAge Pacbase versions.
			The value entered on the Definition line of the Library is channeled down by default to the Definition line of a Program when it is created.
			At the Program level, the programming type can be modified.
			The combination of '-P' and '-8' lines called in the same Program, either directly, or via Macro-structures, is rejected.
10	1		TYPE AND STRUCTURE OF PROGRAM

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			This identifies the structure of the generated Program or the type of the Program in the Library.
		'B'	Structure of a batch Program (default option).
			It provides the general structure of an iterative program:
			.beginning of the loop (F05),
			.end of run (F20),
			.end of the loop (F9099. GO TO F05).
		'S'	Suppress automatic structure generation
			STRUCTURED CODE FUNCTION
			This type can be used to describe the TDS 'system generation', the IDS II 'schema', ...
			.suppression of COBOL divisions,
			.the program is made up of Beginning Insertions
			(-B), Work Areas (-W) and Call of Data Structures (-CD) lines.
			COBOL GENERATOR FUNCTION
			.the program is made up of '-W', -P', '-SC' and '-CP' lines.
		'T'	On-line Program structure.
			Suppression of the loop, i.e:
			.no beginning of loop (F05),
			.no end of run (F20),
			.no end of loop (F9099. GO TO F05).
		'C'	C.I.C.S. on-line Program structure.
			Suppression of the loop, i.e:
			.no beginning of loop (F05),
			.no end of run (F20),
			.no end of loop (F9099. GO TO F05).
			Same as 'T' but also with:
			.generation, at the beginning of the PROCEDURE DIVISION, of the line: MOVE CSACDTA TO TCACBAR,
			.generation in F9099 of: DFHPC TYPE=RETURN,
			.no line numbering in the generated program.
		'M'	Parameterized Macro-Structure type. (For documentation purposes only).

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			This is used for programs to be inserted into other programs. This type of program cannot be generated alone.
		'F'	Program composed of Call of Data Structures (-CD) and Pure COBOL Source Code (-9) lines.
			This option permits the manipulation of the Pure COBOL Source Code (-9) lines that invoke the structural description of the automatically generated D.S.'s, according to the characteristics assigned to that D.S. on the Call of Data Structures (-CD) screen.
			For more information see chapter 'Appendix: Pure COBOL Source Code' in the 'Structured Code' Manual.
		'D'	Program composed of Call of Data Structures (-CD), Beginning Insertions (-B), Work Areas (-W) and Pure COBOL Source Code (-9) lines. This option provides the automatic generation of the IDENTIFICATION, ENVIRONMENT and DATA DIVISIONS.
			The PROCEDURE DIVISION is written entirely on Pure COBOL Source Code (-9) lines.
		'P'	Program composed of Call of Data Structures (-CD), Beginning Insertions (-B), Work Areas (-W) and Procedural Code (-P) lines. This option provides the automatic generation of the IDENTIFICATION, ENVIRONMENT and DATA DIVISIONS. The PROCEDURE DIVISION is entirely written in Structured Code.
		'Y'	Program written in C LANGUAGE and composed of Work Areas (-W), Source Code (-SC) and Call of P.M.S.'s (-CP) lines.
11	1		PROGRAM CLASSIFICATION CODE
			This value is used primarily for documentation purposes. The label corresponding to the selected code will be displayed on Reports and Screens.
			It is also used to select the non-expansion option for Macro-Structures.
		'A'	TP System
		'D'	Sub-program
		'G'	Screen map
		'M'	Macro-structure
		'N'	Non-expanded Macro-Structure
		'P'	Program

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'S'	Schema
		'T'	On-line Program (Screen)
		'U'	Utility
		'V'	Sub-schema
12	1		TYPE OF PRESENCE VALIDATION
			In validation Programs, the presence of numeric Data Element will be determined according to this code:
			For numeric fields:
		blank	Field present if not blank (default value).
		'0'	Field present if not zero.
			For alphabetic and numeric fields:
		'L'	Field present if not low-value.
13	1		SQL INDICATORS GENERATION WITH '-'
			Cross-references available for the use of SQL indicators in Structured Language.
		BLANK	SQL indicators generated in the format: VXXNNCORUB:
		'-'	SQL indicators generated in the format: V-XXNN-CORUB.
14	55		EXPLICIT KEYWORDS
			This field allows you to enter additional (explicit) keywords. By default, keywords are generated from the instance's name (implicit keywords).
			Keywords must be separated by at least one space. Keywords have a maximum length of 13 characters which must be alphanumeric. However, '=' and '*' are reserved for special usage and are therefore ignored in keywords.
			Keywords are not case-sensitive: uppercase and lower-case letters are equivalent.
			NOTE: Accented and special characters can be declared as equivalent to an internal value in order to optimize the search of instances by keywords (Administrator workbench, 'Window' menu, 'Parameters browser' choice, in 'Special Characters' tab).
			A maximum of ten explicit keywords can be assigned to one entity. For more details, refer to the 'Character Mode User Interface' guide, chapter 'Search for Instances', subchapter 'Searching by Keywords'.

Call of Parameterized Macro-Structures (-CP)

The Call of PMS's (-CP) screen is used to call a previously defined Macro-Structure into a Program (batch or on-line), and to specify values to use for resolving parameters (if any).

PARAMETERIZING

Up to 20 parameters can be used in a PMS. A parameter is specified by '$n' (n=1,2,...,9,0) for the first 10 parameters; the next 10 (n=A,B,...J) have to appear on a "continuation" line with the same number as the preceding one.

Alphabetical values cannot be used to parameterize the Macro-Structure line indicatives.

Line indicatives are: Function or Sub-Function codes or line numbers.

The user supplies the replacement values of the respective parameters in the PARAMETER VALUES field in the form of character strings (with delimiters). Each occurrence of the parameter in the original PMS is then replaced by the value entered for this particular Program.

All parameter values (including delimiters) must be written on a maximum of two lines.

The number of characters used for each parameter value must correspond directly to the appropriate field length for the entity being parameterized. For example, if $1 is being used as a Function code, the value must be two characters.

ENTITY INSTANCES USED AS PARAMETERS

You can use instances of the Data Element, Data Structure and Segment entities as parameters.

When such an instance is called via a parameter, no cross-reference is created if the instance code is declared as a simple character string.

This type of cross-reference is established by specifying that the parameter's value is a Data Element, Data Structure or Segment code. This is done by keying in:

/E=DELCO/ or /D=DD/ or /S=SEGT/

At the time of transformation, the parameter is replaced by DELCO, DD or SEGT and cross-references are set up.

NON-EXPANDED MACRO-STRUCTURES

An 'N' in the Expansion ('E') field indicates a non-expanded PMS.

ENTERING COMMENTS

Comment lines entered on the -CP screen are displayed only from the calling program.

The number of the line from which the display must begin can be entered after the Macro-Structure code.

When comments are entered on the -XP screen of the PMS, they are displayed when the PMS is called on this screen.

Entering comments on the -XP screen makes it easier to enter parameters on PMS calls.

```
                  PURCHASING MANAGEMENT SYSTEM        SG000008.LILI.CIV.1583

PROGRAM CALL OF P.M.S.............:     1 VRPREP VENDOR RATING PREPARATION

2   3      4  5   6                                                    7
A   MACRO  LN C : COMMENTS OR PARAMETER VALUES                        D  E
    AADA30      :
    AASO30      : CO/SO/
                :
                :
                :
                :
                :
                :
                :
                :
                :
                :
                :
                :
                :
                :
                :

O: C1 CH: -CP
```

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	6		PROGRAM CODE OR SCREEN CODE (REQUIRED)
			This field contains the six-character program or on- line screen code.
2	1		ACTION CODE
		'C'	Creation of the line
		'M'	Modification of the line
		'D' or 'A'	Deletion of the line
		'T'	Transfer of the line
		'B'	Beginning of multiple deletion
		'G'	Multiple transfer
		'?'	Request for HELP documentation
		'E' or '-'	Inhibit implicit update
		'X'	Implicit update without upper/lowercase processing
3	6		MACRO-STRUCTURE CODE (REQUIRED)

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			A Macro-Structure is a set of Beginning Insertions (-B), Work Areas (-W'), Procedural Code (-P) or Source Code (from Reverse Engineering) lines.
			It is not a sub-program but a sequence of procedural code inserted in the programs before generation. During the insertion, if a line of the Macro-Structure has the same key as a line of the program, it will not be taken into account and will be identified in the calling Program by the '*' character in the ACTION CODE field.
			If lines of two Macro-Structures have the same key, they will both be ignored during the generation of the calling Program.
			The lines of the Macro-Structure replaced by lines of the Program with the same key are considered as comments; they appear (10 maximum) in the Description of the calling Program.
			A non-Parameterized Macro-Structure can contain Data Structures call lines (not inserted in the Programs).
			The call line is associated with the Macro-Structure: the deletion of the Macro-Structure triggers the deletion of the call line, the deletion of the Program does not trigger the deletion of the call lines that concern it.
4	2		LINE NUMBER
			PURE NUMERIC FIELD BLANKS EQUIVALENT TO ZERO
		'0-99'	This is used to define several documentation lines for one macro-structure or, when a macro-structure is parameterized, this can be used to call it into the same program several times.
5	1		CONTINUATION
		*	The asterisk creates a continuation line
6	50		PARAMETER VALUES
			Call of P.M.S.'s (-CP): If the line contains the '/' character, the values are those of the parameters for a P.M.S. Otherwise, they are comments on the Macro-Structure call.
			X-References to Programs/Screens (-XP/-XO): On lines with CALL TYPE = 'O' or 'P': values of the parameters for a P.M.S.
			On lines with CALL TYPE = 'C': comments on the Macro-Structure.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			The values for the parameters must be indicated one after the other, each one ending with a delimiter. The values are entered in the sequence of the assigned parameter number.
			The maximum number of parameters is 20, but limited to 10 per line. The values indicated on the first line correspond to the parameters $1 to $0, and those on the second line correspond to parameters $A to $J.
			The parameterization of the placement of the P.M.S. in the program, such as the (sub-)function code, line number, or WSS prefix, must be specified on the first line only.
			In order to nullify the value of a parameter, the following technique can be used:
			For the first parameter, enter the delimiter in the first column of this field.
			For subsequent parameters, the system will understand two consecutive delimiter characters to mean ignore the next sequential parameter.
			For example, using '/' as the delimiter, /BB/CC/ will resolve parameters $2 and $3 with BB and CC respectively. XX//ZZ/ will resolve $1 with XX, and $3 with ZZ.
			To specify a blank as a parameter value, enter it between the delimiters as with any other value.
			In order to establish cross-references when a Data Element, a Data Structure or a Segment is used as a parameter, the value should be respectively coded:
			/E=DELCO/ (DELCO = Data Element code),
			or /D=DD/ (DD = Data Structure code),
			or /S=SEGT/ (SEGT = Segment code).
			The parameter is then replaced by the DELCO, DD or SEGT value and cross-references to the Data Element, Data Structure or Segment are established.
7	1		DELIMITER OF PARAMETERIZED VALUES
		'/'	This character is used to separate the different parameter values. Default value.

X-References to Programs/Screens (-XP/-XO)

The X-References to Programs (-XP) and On-Line Screens (-XO) screens are used for entering comments the user wants to appear in the sub-reports that are produced with the Generation and Print Commands 'DCP', 'GCP', 'DCO' and 'GCO' (GPRT Procedure).

These comments may be assigned by entering 'C' for the CALL TYPE, a LINE NUMBER value, and the comment desired.

It is also possible to use these lines to specify parameter values, as on the Call of P.M.S.'s (-CP) screen.

A line number may be entered after the code of the screen or program that will begin the display. This line number corresponds to the macro-structure call.

RECOMMENDATION

Since these lines reappear for each call of the Macro, and since Macros may be called many times into the same Program, it is suggested that these comments be brief and contain only essential information, like the meaning of the parameters.

```
                  PURCHASING MANAGEMENT SYSTEM         SG000008.LILI.CIV.1583

PROGRAM CROSS-REFERENCES                1 AASO30 SORT INPUT PROCEDURE
2 3 4      5  6    7                                        >           8
A T PG/SC  LN C  : COMMENTS OR PARAMETER VALUES                        D E
  P VRPREP       : CO/SO/
                 :
                 :
                 :
                 :
                 :
                 :
                 :
                 :
                 :
                 :
                 :
                 :
                 :
                 :
                 :
                 :

O: C1 CH: Paaso30XP
```

```
                  PURCHASING MANAGEMENT SYSTEM        SG000008.LILI.CIV.1583

 PROGRAM CROSS-REFERENCES              1  AAPOJ1 SORT INPUT PROCEDURE
 2 3 4      5  6   7                                                  8
 A T PG/SC  LN C : COMMENTS OR PARAMETER VALUES                      D E
   O POJB01      : 020/A/
   O POJB02      : 050/A/
   O POJB03      : 100/A/
   O POJC10      : 010/A/
                 :
                 :
                 :
                 :
                 :
                 :
                 :
                 :
                 :
                 :
                 :
                 :
                 :

 O: C1 CH: Paapoj1XO
```

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	6		MACRO-STRUCTURE CODE
			A Macro-Structure is a set of Beginning Insertions (-B), Work Areas (-W'), Procedural Code (-P) or Source Code (from Reverse Engineering) lines.
			It is not a sub-program but a sequence of procedural code inserted in the programs before generation. During the insertion, if a line of the Macro-Structure has the same key as a line of the program, it will not be taken into account and will be identified in the calling Program by the '*' character in the ACTION CODE field.
			If lines of two Macro-Structures have the same key, they will both be ignored during the generation of the calling Program.
			The lines of the Macro-Structure replaced by lines of the Program with the same key are considered as comments; they appear (10 maximum) in the Description of the calling Program.
			A non-Parameterized Macro-Structure can contain Data Structures call lines (not inserted in the Programs).

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			The call line is associated with the Macro-Structure: the deletion of the Macro-Structure triggers the deletion of the call line, the deletion of the Program does not trigger the deletion of the call lines that concern it.
2	1		ACTION CODE
		'C'	Creation of the line
		'M'	Modification of the line
		'D' or 'A'	Deletion of the line
		'T'	Transfer of the line
		'B'	Beginning of multiple deletion
		'G'	Multiple transfer
		'?'	Request for HELP documentation
		'E' or '-'	Inhibit implicit update
		'X'	Implicit update without upper/lowercase processing
3	1		CALL TYPE
			This field contains the type of call used for each line. The values are as follows:
		'C'	Comments on the macro-structure.
		'O'	Call of the macro-structure in a screen.
		'P'	Call of the macro-structure in a program.
4	6		PROGRAM CODE OR SCREEN CODE
			This field contains the six-character program or on- line screen code.
5	2		LINE NUMBER
			PURE NUMERIC FIELD BLANKS EQUIVALENT TO ZERO
		'0-99'	This is used to define several documentation lines for one macro-structure or, when a macro-structure is parameterized, this can be used to call it into the same program several times.
6	1		CONTINUATION
		*	The asterisk creates a continuation line
7	50		PARAMETER VALUES
			Call of P.M.S.'s (-CP): If the line contains the '/' character, the values are those of the parameters for a P.M.S. Otherwise, they are comments on the Macro-Structure call.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			X-References to Programs/Screens (-XP/-XO): On lines with CALL TYPE = 'O' or 'P': values of the parameters for a P.M.S.
			On lines with CALL TYPE = 'C': comments on the Macro-Structure.
			The values for the parameters must be indicated one after the other, each one ending with a delimiter. The values are entered in the sequence of the assigned parameter number.
			The maximum number of parameters is 20, but limited to 10 per line. The values indicated on the first line correspond to the parameters $1 to $0, and those on the second line correspond to parameters $A to $J.
			The parameterization of the placement of the P.M.S. in the program, such as the (sub-)function code, line number, or WSS prefix, must be specified on the first line only.
			In order to nullify the value of a parameter, the following technique can be used:
			For the first parameter, enter the delimiter in the first column of this field.
			For subsequent parameters, the system will understand two consecutive delimiter characters to mean ignore the next sequential parameter.
			For example, using '/' as the delimiter, /BB/CC/ will resolve parameters $2 and $3 with BB and CC respectively. XX//ZZ/ will resolve $1 with XX, and $3 with ZZ.
			To specify a blank as a parameter value, enter it between the delimiters as with any other value.
			In order to establish cross-references when a Data Element, a Data Structure or a Segment is used as a parameter, the value should be respectively coded:
			/E=DELCO/ (DELCO = Data Element code),
			or /D=DD/ (DD = Data Structure code),
			or /S=SEGT/ (SEGT = Segment code).
			The parameter is then replaced by the DELCO, DD or SEGT value and cross-references to the Data Element, Data Structure or Segment are established.
8	1		DELIMITER OF PARAMETERIZED VALUES
		'/'	This character is used to separate the different parameter values. Default value.

Chapter 3. Modifying the Identification/Environment Div. (-B)

You can complete or modify the beginning of the generated Program with the Beginning Insertions (-B) screen. This applies to the IDENTIFICATION DIVISION, the ENVIRONMENT DIVISION, and also the 'DATA DIVISION' and 'FILE SECTION' statements.

GENERAL CHARACTERISTICS

The use of the Beginning Insertions (-B) screen is exceptional with most hardware types.

Examples of its usage are:

- For the SELECT clause for relative access files,
- Bull Gcos7 case: COPY SELECT and COPY FD clauses in a TPR TDS.

SUPPRESSION OF GENERATED LINES

If the program includes Beginning Insertions (-B) lines containing the code of a single section or paragraph, then automatically generated lines for this section or paragraph will be suppressed.

'-B' lines are ignored in a "reversed" Program.

TRANSFER OF LINES TO ANOTHER ENTITY

Lines from one entity may be copied directly to another entity. At the top of the screen, an ENTITY TYPE field with a value of 'O' or 'P', followed by the appropriate PROGRAM CODE OR SCREEN CODE, allows the user to take the lines already entered on a screen and attached to one entity, copy them, and attach them to another entity. This is not a MOVE. A duplicate set of lines is created in another entity.

EXAMPLE:

In order to copy entity lines from screen 'SCREE1' into program 'PGM001', 'O' and 'SCREE1' should be overtyped with:

'P' and 'PGM001' respectively.

```
                  PURCHASING MANAGEMENT SYSTEM        SG000008.LILI.CIV.1583

PROGRAM BEGINNING INSERTIONS : P TES001 TEST FOR POJ
                               1 2

3 4  5  6   7
A SE PA LIN INSTRUCTION TO BE INSERTED
* 60    000 DATA DIVISION
* 70    000 SUB SCHEMA SECTION
* 80    000 FILE SECTION
* 99 99 000 SUPPRESSED









O: C1 CH: -B
```

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		ENTITY TYPE
			This field is used to identify the entity to which these lines are attached:
		'O'	On-line Screen
		'P'	Program
			The user may keyboard this field in order to copy lines attached to a Screen into a Program and vice-versa.
2	6		PROGRAM CODE OR SCREEN CODE
			This field contains the six-character program or on- line screen code.
3	1		ACTION CODE
		'C'	Creation of the line
		'M'	Modification of the line
		'D' or 'A'	Deletion of the line
		'T'	Transfer of the line
		'B'	Beginning of multiple deletion

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'G'	Multiple transfer
		'?'	Request for HELP documentation
		'E' or '-'	Inhibit implicit update
		'X'	Implicit update without upper/lowercase processing
4	2		SECTION TO GENERATE
			The following section codes may be used in combination with the values entered in the PARAGRAPH TO GENERATE field:
		blank	IDENTIFICATION DIVISION.
		'00'	CONFIGURATION SECTION.
		'01'	INPUT-OUTPUT SECTION. FILE CONTROL. With PARAGRAPH TO GENERATE specifying a DATA STRUCTURE CODE, the INSTRUCTION TO BE INSERTED lines will appear after the FILE CONTROL statement.
			With PARAGRAPH TO GENERATE = blank, the data entered in the INSTRUCTION TO BE INSERTED field will override both the INPUT-OUTPUT SECTION and the FILE CONTROL statements.
		'60'	Is reserved for the DATA DIVISION.
		'99'	Is reserved for the FILE SECTION.
		'$n'	In a macro-structure, the SECTION TO GENERATE can be parameterized.
		'9*'	With PARAGRAPH TO GENERATE = ' FF': Rewriting of FD clause for FF file.
5	2		PARAGRAPH TO GENERATE
			The following codes are used to identify the COBOL statements listed below.
			With SECTION TO GENERATE = blank:
		'01'	API COBOL comments.
		'05'	PAF comments.
		'10'	PROGRAM-ID.
		'20'	AUTHOR.
		'30'	DATE-COMPILED.
		'40'	ENVIRONMENT DIVISION.
			With SECTION TO GENERATE = '00':
		'00'	SOURCE-COMPUTER.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'10'	OBJECT-COMPUTER.
		'20'	SPECIAL-NAMES.
			With SECTION TO GENERATE = '01':
		'FF'	SELECT FF-FILE
		'00'	I-O-CONTROL.
			This paragraph is not automatically generated, except in certain cases; with COBOL variant 6: BULL H66 BCD.
		'90'	With SECTION TO GENERATE = '99': RECEIVE-CONTROL (TANDEM).
		'99'	FILE SECTION.
		'$n'	In a Macro-Structure, the PARAGRAPH TO GENERATE can be parameterized.
			With SECTION TO GENERATE = '01':
6	3		LINE NUMBER
			PARAMETERIZABLE NUMERIC FIELD
		'0-999'	As a recommendation, number the lines starting with 10 by intervals of 10, thus facilitating future insertion insertions.
		$n0 to $n9	In a Macro-Structure, only the first two characters of the LINE NUMBER can be parameterized.
7	66		INSTRUCTION TO BE INSERTED
			If the INSTRUCTION TO BE INSERTED should be in Margin A of the COBOL program, begin it in the second column of this field. If it should be in Margin B, begin it in the sixth column of this field.
			Otherwise, the INSTRUCTION TO BE INSERTED will be justified in column 7 (continuation/comments) of the generated COBOL program.
		'S'	(or any word beginning with an 'S') The value 'S' in the first column of this field causes the suppression of the section or paragraph generated.
		'$n'	In a macro-structure, this field can be parameterized.

Chapter 4. Modifying the Working Storage/Linkage Section

Data Structure Calls (-CD)

The purpose of the Call of Data Structures is to identify all Data Structures used in a Program, specifying their physical characteristics as well as the way these files are to be used in the Program.

The Call of Data Structures screen is accessed by entering '-CD' in the CHOICE field from any screen within the Program entity's network.

GENERAL CHARACTERISTICS

Each Data Structure may be described on as many continuation lines as needed. Certain information must be entered on the first line of the call, as opposed to being entered on a continuation line, and vice versa.

The system assigns default values to required information areas of the Data Structure call line. By default, a Data Structure will look like a sequential file with fixed-length records. The Data Structure Description will contain all of the Data Structure records, with the Data Elements in internal format, without the optional Data Elements.

ORGANIZATION

Data Structures are 'organized' into three basic types:

. Standard Files,

. Database Blocks,

. Work Areas or Linkage Areas.

The descriptions of the latter category may involve specifying Data Structures and/or Data Elements.

It is preferable to define the WORK or LINKAGE fields on the screen provided for this purpose (-W). If the Program is a Macro-Structure (P.M.S.), the '-W' is generated in the calling Program, not the '-CD'.

NOTE: A Data Structure call in the -W screen does not allow for the creation of continuation lines (which limits the number of Segment selections to four Segments, for example).

Also, utilization, control breaks, and file matching cannot be specified on -W lines.

COMPOSITE DATA STRUCTURES

It is possible at the Program level to build a Data Structure with Segments belonging to different Data Structures.

This is accomplished by assigning the same DATA STRUCTURE CODE IN THE PROGRAM to different Data Structures, and selecting the desired Segments from each.

The common part will be made of the code of the Data Structure called on the first line.

In order to call in a Program Data Structure two or more Segments which have the same two-character SEGMENT CODE or the same LAST CHARACTER OF THE REPORT CODE, but are extracted from different Data Structures in the Library, it is necessary to change the code of one of them in the Program, in the SELECTION field.

```
                PURCHASING MANAGEMENT SYSTEM        SG000008.LILI.CIV.1583

DATA STRUCTURES USED IN PROGRAM : 1  VRPREP VENDOR RATING PREPARATION

2 3  4    5  6      7 9 11      13  14  16   18 19     21          23  25  27
                     8 10 12          15  17      20     22          24  26
A DP CO : DL EXTERN OARFU BLOCK T   B M U RE SE L UNIT C SELECTION F E R L PL
  CO    : CO PMSCO  SSFOU     0 R       D                          I     1
        :    STAT.FLD: 28           ACC. KEY:  29        RECTYPEL 30
  OI    : OI PMSPOF VSFID     0 R   1   C                          I     1
        :    STAT.FLD:              ACC. KEY: POKEY      RECTYPEL
  SO    : CO SORT   SSFTU     0 R       D                          I     1
        :    STAT.FLD:              ACC. KEY:            RECTYPEL
  WO    : CO WORK.  WSFOU     0 R       D                          I     1
        :    STAT.FLD:              ACC. KEY:            RECTYPEL
        :
        :    STAT.FLD:              ACC. KEY:            RECTYPEL
        :
        :    STAT.FLD:              ACC. KEY:            RECTYPEL
        :
        :    STAT.FLD:              ACC. KEY:            RECTYPEL
        :
        :    STAT.FLD:              ACC. KEY:            RECTYPEL

O: C1 CH: -CD
```

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	6		PROGRAM CODE (REQUIRED)
			Code identifying the program in the library.
2	1		ACTION CODE
		'C'	Creation of the line
		'M'	Modification of the line
		'D' or 'A'	Deletion of the line
		'T'	Transfer of the line
		'B'	Beginning of multiple deletion
		'G'	Multiple transfer
		'?'	Request for HELP documentation
		'E' or '-'	Inhibit implicit update
		'X'	Implicit update without upper/lowercase processing
3	2		DATA STRUCTURE CODE IN THE PROGRAM (REQUIRED)
			This code establishes the sequence in which the Data Structure will be processed in the Program.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			The first character must be alphabetic but the second one can be numeric or alphabetic.
			It is recommended to keep the same DATA STRUCTURE CODE IN THE PROGRAM and IN THE LIBRARY when the Data Structure described in the Library is used only once in the Program.
4	2	ALPHA.	Continuation of D.S. Description
		blank	First line of a Data Structure description. This line must contain all information defining the input-output characteristics, all technical characteristics and the description of the Data Structure.
			Two-letter code indicating a continuation line.
			The continuation lines are used to select the records of the different Data Structures in the Library and to request their description in a specified position.
5	2		DATA STRUCTURE CODE
			This code is made up of two alphanumeric characters. This is a logical code internal to the Database and therefore independent of the names used in Database Blocks and Programs.
6	6		EXTERNAL NAME OF THE FILE
			(Default option: DATA STRUCTURE CODE IN THE PROGRAM.)
			(NOTE: In this discussion, the term 'COBOL Variant' = the value in the TYPE OF COBOL TO GENERATE field)
			FOR the 'Y' ORGANIZATION:
			This field must contain the code of the COBOL COPY clause which represents the communication area of the Pacbench C/S Application Component which accesses the Logical View. For more details, refer to the 'Pacbench C/S Applications - Business Logic' Manual.
			FOR SQL ORGANIZATIONS:
			This field must contain the VisualAge Pacbase code of the SQL block.
			For explanations, refer to the 'Structured Code'
			manual, chapter 'Modifying the Procedure Division', subchapter 'Procedural Code Screen (-P)', and to the 'SQL Databases' manual, chapter 'SQL Accesses', Sub-chapter 'Customized SQL Accesses'.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			FOR ALL THE OTHER ORGANIZATIONS:
			IBM OS/390 (variant X): DDNAME in 1 to 6 positions.
			COBOL II IBM VS2 (Variant X): The ASSIGN clause (for sequential files, 'S' organization) with SYSnnn as external name is generated in the following form:
			SYSnnn-UT-....-S-SYSnnn
			IBM DOS (COBOL Variant 1), three forms:
			.SYSnnn Symbolic unit name.
			.xxxnnn Specifies at the same time the symbolic unit name and the external name of the Data Structure.
			.xxxxxx External name. The symbolic unit is generated with SYSnnn, nnn being incremented by one for each Data Structure starting with SYS010.
			BULL Gcos7 (COBOL Variant 4):
			.INTERNAL-FILE-NAME in 1 to 6 position.
			BULL Gcos8 (COBOL Variant 5):
			.File code (2 characters). UNISYS A Series (COBOL Variant 8):
			.nnppp numeric, generate AREA nn, AREASIZE pppp.
			TANDEM (Variant F): external name in 1 to 6 positions.
			DEC/VMS (COBOL Variant I): external name in 1 to 6 positions.
			PHYSICAL CHARACTERISTICS OF FILE
7	1		ORGANIZATION
		'S'	Sequential (Default value).
		'I'	Indexed sequential
		'V'	VSAM (IBM), UFAS (BULL), etc.
			Generates the STATUS KEY IS clause and the corresponding field is declared in the STATUS FIELD: VSAM FILE INDICATOR field.
			The file is considered sequential if the name of the key in the record is absent; it is considered indexed if the key name is entered.
		'W'	File descriptions are generated in WORKING-STORAGE before the constant 'WSS-BEGIN'.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			A Data Structure thus described will be used like a work area or processed through a function of a generalized management system (Database in particular).
		'L'	Identical to 'W' except that the user may choose the description location (See CODE FOR COBOL PLACEMENT).
		'X'	Data Structure used as a comment, not used for generation.
		'G'	Table description.
			Generates the communication area with the access module.
		'Y'	Call of the COPY clause which corresponds to the communication area between the client and the server (Pacbench C/S Business Components only).
			For details, refer to the 'Pacbench C/S Applications - Business Logic' Manual.
			DATABASES
			The values of the following codes are reserved for Database Descriptions when the Database Description function is not used. These values are taken into account by application programs.
		'D'	Reserved for the Description of Segments or records of the different Databases, IMS (DL/1), IDS II, (according to the TYPE OF COBOL TO GENERATE selected), in the generation of DBD, SYSGEN, schemas or application Programs (according to the TYPE AND STRUCTURE OF PROGRAM selected).
		'B'	Reserved for the description of records for an IDMS Database in the sub-schemas or application programs.
		'A'	Reserved for an ADABAS file description in the definition programs or usage programs of the Database.
		'T'	Reserved for the description of 'TOTAL' files in the definition programs or the usage programs of the Database.
		'Q'	Reserved for the description of SQL/DS, DB2/2 or DB2/6000 Databases (IBM), or
			ALLBASE/SQL Databases (HP3000), or
			DB2/2 or DB2/600 Databases (MICROFOCUS).
		'2'	Generation-Description of a DB2 or VAX/SQL Segment. Only physical accesses are not generated. The structure of variable indicators corresponding to the columns of the DB2 or VAX/SQL table is always generated.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'C'	Reserved for the description of an INTEREL RDBC, RFM Database Structure.
		'O'	Reserved for the description of an ORACLE (< V6) Database Structure.
		'P'	Reserved for the description of an ORACLE (V6 and V7) Database Structure.
		'R'	Reserved for the description of an RDMS Database Structure.
		'4'	Reserved for the description of a DB2/400 Database Structure.
		'N'	Reserved for the description of a NONSTOP SQL Database Structure.
		'M'	Reserved for the description of a DATACOM DB Database Structure.
		'9'	Reserved for the description of an INFORMIX, SYBASE, INGRES/SQL, and SQL SERVER Database Structure.
			The use of the System with the different DBMS's is documented in specific 'Database Description' manuals.
8	1		Access mode
		'S'	Sequential (default option).
		'R'	Random - Direct (indexed sequential organization only).
		'D'	Dynamic (VSAM files only - 'V' organization)
9	1		RECORDING MODE
		'C'	For 'P'-type organizations (Oracle V6 and V7) and '9'-type organizations (Sybase): Automatic generation of CONNECT AT Database, DECLARE Database and access SQL AT Database.
		'F'	Fixed (default option).
			At generation time, the lengths of the different records are aligned with the length of the longest record.
		'V'	Variable.
		'U'	Undefined.
		'S'	Spanned (Reserved for IBM MVS and DOS variants).
10	1		FILE TYPE - INPUT / OUTPUT
		'I'	Input file - Default option with the following values of USAGE OF DATA STRUCTURE: 'C', 'T', 'X', 'M', 'N' 'P'. This value is prohibited with all other USAGEs.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'O'	Output file - Default option with the following values of USAGE OF DATA STRUCTURE: 'D', 'S', 'R', 'E', 'I' and 'J'. This value is prohibited for all other USAGEs.
		'E'	Output file. Generation of an OPEN EXTEND clause
		'T'	Sort (on Input or Output, depending on the USAGE OF DATA STRUCTURE value).
		'R'	Input-Output (direct access Data Structures only).
11	1		UNIT TYPE
		'U'	Magnetic storage with sequential access.
			Default value.
		'D'	Magnetic memory with selective access.
			Direct access device.
		'R'	Slow peripherals (Card punch reader, printer).
			This parameter is important for the TYPEs OF COBOL TO GENERATE variant for which the "ASSIGN" clause, the FD level or the WRITE statements depend on the UNIT TYPE.
12	5	NUMER.	BLOCK SIZE SPACES AND ZEROES ARE EQUIVALENT
			PURE NUMERIC FIELD
			(Note: In this discussion the term 'COBOL Variant' = the value in the TYPE OF COBOL TO GENERATE field)
		0	Default value.
		2	The blocking factor can be zero in the following cases:
			. IBM OS (COBOL variant 0) except for indexed organization files.
			. IBM MVS. The BLOCK CONTAINS clause is generated for a VSAM file only if the library is in COBOL II.
			The corresponding COBOL clause (BLOCK CONTAINS) is not generated in the following cases:
			.sort file,
			.disk Data Structure (file stored on a disk) if no number is mentioned,
			.file with UNIT TYPE = 'R' in IBM DOS (COBOL variant 1)
			.Block 0 for UNISYS A Series (COBOL Variant 8) and AS 400 (COBOL Variant O).
			.Block 0 for IBM VSE COBOL II and file with UNIT TYPE = 'N'.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
13	1		BLOCK SIZE UNIT TYPE
		'R'	Records (default value).
		'C'	Characters.
		'N'	The BLOCK CONTAINS clause is not generated.
14	1	NUMER.	NUMBER OF CONTROL BREAKS
			(BATCH SYSTEMS DEVELOPMENT Function) All spaces are replaced with zeroes.
			For sequentially accessed, sorted files: Enter the number of Elements (elementary or group) on which there is to be control break processing for the Data Structure.
		'0'	Default.
		'1 to 9'	1 to 9 levels, according to the number of Elements to be used for control break processing. These Elements are identified as the SORT KEYs for this Data Structure.
			When there is control break processing on one or more Data Structures, two indicators keep track of the status of the records being processed:
			Note: The term 'nth key Data Element' includes all key Data Elements up to and including the nth level.
			.dd-IBn = '1': the nth key Data Element of the current record of Data Structure dd contains a new value,
			.dd-FBn = '1': the nth key Data Element of the current record of Data Structure dd contains the last occurrence of the present value.
			When these files are synchronized with others, (see FILE MATCHING LEVEL NUMBER) the control breaks are kept synchronized via two additional switches:
			.ITBn = '1': a new value in the nth key Data Element has been detected. This signals beginning processing on all synchronized D.S's.
			.FTBn = '1': the present value of the nth key Data is occurring for the last time. This signals end processing for the records in this iteration for all synchronized D.S's.
			For output files (USAGE OF DATA STRUCTURE value 'D'):
			A non-zero value will create a duplicate file layout to be generated in the WORKING-STORAGE area identifiable by a prefix of '1-'.
			Note however a preferable procedure to accomplish this is via the Work Areas (-W) Screen.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
15	1	NUMER.	FILE MATCHING LEVEL NUMBER
			BLANKS REPLACED BY ZEROES.
			For sequentially accessed files:
			Used to establish the synchronization of two or more files.
		'0'	Default.
		'1 to 9'	Enter the number of Elements (Elementary or Group) on which file matching is to be synchronized for this Data Structure. This number identifies the number of the key fields (identified in the SORT KEY/ field) that are involved in the synchronization.
			For an automatic file matching, the following conditions must be met:
			. The Data Structure control break level must be equal to the file matching level - 1, except for a transaction Data Structure, whose control break level must be equal or superior to the file matching level.
			. The Data Element(s) which constitute(s) the sort keys of a Data Structure must be sorted in ascending order.
			. The Data Element(s) which constitute(s) the sort keys of a Data Structure must have the same length for the same level.
			. These Data Elements must have a display format (if they are numeric, they must be whole numbers and unsigned).
			Switches generated to control the file matching are:
			.dd-CFn: which indicates whether a file should be processed or bypassed in this iteration, ('1' = process, '0' = bypass).
			.dd-OCn: which indicates the status of processing on a record of a principal file (USAGE OF DATA STRUCTURE = 'P').
			For sequentially accessed files:
			'1' = WRITE to the principal file
			'0' = do not WRITE.
			For direct access files:
			'1' = CREATE or REWRITE
			'0' = DELETE
16	1		USAGE OF DATA STRUCTURE

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			This code defines the role of the Data Structure in the Program and determines the generated functions.
		'C'	Consult
			Any input file (Data Structure).
		'D'	Direct
			Any output file (default).
		'T'	Table
			A file to be fully stored in memory. The table is generated according to the number of repetitions indicated on each Segment Definition. (See OCCURRENCES OF SEGMENT IN TABLE).
			The maximum number of selected Segments per D.S. = 50.
		'X'	Table
			A file to be partially stored in memory. Only Data Elements other than FILLER are loaded.
			Elementary Data Elements other than FILLER are limited to 10 (in addition to the RECORD TYPE ELEMENT) for the '00' Segment and to 29 for each specific non-00 Segment.
		'S'	Selected
			Output file extracted from another file.
			It differs from USAGE value 'D' since the generated description in the output area is not detailed. For Data Elements with an 'OCCURS DEPENDING ON' clause, the USAGE OF DATA STRUCTURE must be 'D'.
			The following values are specific to the Batch Systems Development function:
		'P'	Principal
			Input file, likely to be updated (by a transaction file - usage value 'M' or 'N').
		'R'	Result
			Updated principal file in sequential access mode. (When the Data Structure contains an 'OCCURS DEPENDING ON' clause, the output/result D.S must be declared as 'D').
		'M'	Transactions to be validated:
			Input file to be validated which may update other file(s). The generated functions range from 30 to 76.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			Note: Only one 'M' or 'N' Data Structure is allowed per Program.
		'N'	Transactions not to be validated:
			Input file which can update other files.
			The generated functions are: 30, 33, 39, 70 to 76.
			Note: Only one 'M' or 'N' D.S. is allowed per Program.
		'E'	Transaction file with errors detected:
			Output transaction file containing a field identifying records with errors. The system will generate the field(s) to track the erroneous Elements, erroneous Segments and user defined errors using the reserved Data Elements ENPR, GRPR and ERUT. (The option is selected in the RESERVED ERROR CODES IN TRANS. FILE field). Selected or not, the descriptions of these Elements are generated (using the Data Elements DE-ERR and ER-PRR).
			These descriptions precede the descriptions of the Elements.
		'I'	Direct printing (or by SYSOUT in IBM MVS)
			At the generation level, the lines with STRUCTURE NUMBER value of '00' will be ignored.
		'J'	Indirect printing to be processed by a spool Program.
			Fields required for identifying the lines, line skips, etc. are defined in Report STRUCTURE NUMBER value 00.
17	2		RESULTING FILE DATA STRUCTURE CODE
			With USAGE OF DATA STRUCTURE value 'P', indicate the DATA STRUCTURE CODE IN THE PROGRAM of the resulting output D.S.
			For an output type USAGE OF DATA STRUCTURE (value 'R' or 'D'), indicate the DATA STRUCTURE CODE IN THE PROGRAM of the input principal D.S.
18	2		SOURCE OR ERROR DATA STRUCTURE CODE
			For a transaction file (USAGE OF DATA STRUCTURE = 'M' or 'N'), enter the DATA STRUCTURE CODE IN THE PROGRAM of the transaction file containing the error fields (USAGE OF DATA STRUCTURE = 'E') if one has been called.
			For a transaction file with the error field (USAGE OF DATA STRUCTURE'), enter the DATA STRUCTURE CODE IN THE PROGRAM of the corresponding transaction file (USAGE OF DATA STRUCTURE = 'M' or 'N').

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			For a selected file (USAGE OF DATA STRUCTURE = 'S'), enter the DATA STRUCTURE CODE IN THE PROGRAM of the input source with the corresponding Data Structure code of the selected file on the line where the source file is being called.
19	1		TRANSACTION CONTROL BREAK LEVEL
			ALL SPACES REPLACED BY ZEROS.
			Default option: NUMBER OF CONTROL BREAKS
			In a transaction file, enter the position within the SORT KEY/ of the ACTION CODE ELEMENT. For example, if the SORT KEY/ value is ABCDE and the ACTION CODE ELEMENT is 'D', enter '4' here.
			This element is the minor-most key of the sort key and the one used to differentiate one type of transaction from another of the same principal file. Duplicates are detected if any key elements below this one are found to match.
20	4		PHYSICAL UNIT TYPE
			NOTE: The term 'COBOL Variant' = the value in the TYPE OF COBOL TO GENERATE field) generates the following in the SELECT clause of some COBOL variants:
			IBM DOS (COBOL Variant 1):
			Enter the model type (examples: 2314, 3330, 2400).
			MICROFOCUS, COBOL II, IBM VISUAL SET (COBOL Variant 3)
		EXT	Generation of the EXTERNAL clause at the file FD level
		LS	Generation of the LINE SEQUENTIAL clause
		EXLS	Generation of the LINE SEQUENTIAL clause and of the EXTERNAL clause at the file FD level
			ACU COBOL (COBOL Variant Q) :
		LS	Generation of the LINE SEQUENTIAL clause
			Gcos7 (COBOL Variant 4):
		'SSF'	Option WITH SSF in the SELECT clause
		'OUT'	Option -SYSOUT suffix after the filename in the SELECT clause (WITH SSF is generated).
			Gcos8 ASCII (COBOL Variant 5):
		'PT'	Printer.
		'CR'	Card reader.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'SSF'	ORGANIZATION IS GFRC SEQUENTIAL SSF CODE SET IS IS GBCD.
		'IBM'	ORGANIZATION IS IBM-OS SEQUENTIAL.
		'xxx'	WITH xxx.
		'...V'	A 'V' in the 4th position generates the clause 'VALUE OF FILE-ID is 3-FF00-IDENT' (FF is the program Data Structure code being called).
			The field 3-FF00-IDENT must be defined in -W by the user.
			BURROUGHS large system (COBOL Variant 8) UNISYS A Series:
		DK or	
		'blank'	Disk.
		'DKS'	Sort Disk (with T opening).
		'DKM'	Merge Disk (with T opening).
		'RD'	Reader.
		'PT'	Printer.
		'PO'	File.
		'TP'	Tape.
			For the 2-character codes, a third character can specify a particular final disposition:
		'..P'	Purge.
		'..R'	Release.
		'..L'	Lock.
		'..S'	Save.
		'...V'	A 'V' in the 4th position generates the clause 'VALUE OF D.S. NAME IS 3-FF00-IDENT'.
			UNISYS 2200 (COBOL Variant U):
		'CR'	Card reader.
		'CP'	Card punch.
		'UN'	Uniservo.
		'TP'	Tape.
		'PN'	Printer with external name. If the COMPLEMENTARY PHYSICAL UNIT TAPE field contains input, the RECORDING clause is also generated.
		'PT'	Printer without external name.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'PF'	Printer with external name and:
			VALUE OF PRINTER-FORMS 3-FF00-FORMS
			LINAGE IS 3-FF00-LINES
			TOP IS 3-FF00-TOP
			BOTTOM IS 3-FF00-BOTTOM
			These 4 data-names are to be declared in Work Areas (-W) lines with their appropriate values.
			AS 400 (COBOL Variant O):
		DB	Database.
		RD	Reader.
		CP	Card Punch.
		PT	Printer.
		TP	Tape.
		DK or	
		'blank'	Disk.
21	1		COMPLEMENTARY PHYSICAL UNIT TYPE
			NOTE: The term 'COBOL Variant' = the value in the TYPE OF COBOL TO GENERATE field.
			IBM DOS (COBOL Variant 1):
		'R'	Reader.
		'P'	Punch.
			BULL Gcos8 (COBOL Variant 5):
		'S'	EBCDIC Set code.
		'C'	ASCII Set code.
			UNISYS 2200 (variant U):
		'S'	Recording followed by lock mode.
			BULL Gcos7 (COBOL Variant 4) and Gcos8 (COBOL Variant 6)
		'O'	If the value 'O' is entered in this field, the OPTIONAL option is not generated.
			Otherwise, the OPTIONAL option is generated by default.
			DEC VAX VMS (COBOL Variant I)
		'A'	File opening with option ALLOWING ALL and sequential reading with option REGARDLESS.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			IBM MVS :
		'F'	OPTIONAL parameter generated in the SELECT clause of a VSAM file.
22	9		SELECTION
			This field has three mutually exclusive uses:
			1. Composition of the sort key
			This is the group of Data Elements making up the sort key for control break processing. They are identified by the value entered in the KEY INDICATOR FOR ACCESS OR SORT field on the Segment Call of Elements (-CE) screen.
			The order of sorting these key Data Elements may be entered here using the values assigned on the Call of Elements (-CE) screen in the desired order of major to minor - left to right. If no explicit entry is made here, Elements coded with value 1 to 9 will be taken as the default.
			The Data specifying the sort order must be entered on first line of the Data Structure call. (That is on the line where the CONTINUATION OF D.S. DESCRIPTION field remains blank.)
			Note: for transaction files, include the ACTION CODE and RECORD TYPE ELEMENTs as a part of the key. The order in which these Elements are sorted will determine the sequence in which the transactions update the principal file, and the policy for duplicate record detection.
			2. Selection of Segments in a Data Structure
			Rather than having all of the Segments belonging to a Data Structure described, the user may select the ones that are needed, thus avoiding unnecessary description lines and wasted work area space. This may be significant for tables (USAGE OF DATA STRUCTURE = 'T').
			This is done by entering an '*' in the first column of this field followed by a maximum of 4 SEGMENT CODES, in addition to the common part. The Segments may come from different D.S.'s, but in this case, it is better to call these Segments into another Segment.
			When the user wishes to re-create the file matching key and select records, he/she must indicate the file matching on the first Segment Call line, and the selected records on continuation lines.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			When Segments come from different D.S.'s Descriptions, the common part of the first D.S. called is considered to be the resulting file common part. The other D.S.'s must not have a common part.
			3. Report selection: To select a particular Report, the third character in the Report code must be entered in the field. To select all Reports with the same prefix, you must leave the field blank.
			Generally, continuation lines are created if more than four Segments or nine Reports are selected.
			It is possible to rename a SEGMENT CODE or LAST CHARACTER OF REPORT CODE: one line per Segment or Report to be renamed is created. Enter the LAST CHARACTER OF REPORT CODE as known in the Library, followed by the desired code for the Program separated an ″=″ sign.
			Follow the same procedure to rename the SEGMENT CODE, but precede the old Segment code with an asterisk.
			EXAMPLE:
			1=2 Rename report code 1 report code 2
			*01=02 Rename segment code 01 segment code 02.
23	1		NON-PRINTING DATA STRUCTURE FORMAT
			This option is reserved for Data Structures with a USAGE OF DATA STRUCTURE other than 'I' or 'J'.
		'E'	Input format. (Default option with USAGE OF D.S. = 'M', 'N' or 'E').
		'I'	Internal format (Default with USAGE OF D.S. NOT= 'M', 'N' or 'E').
		'S'	Output format.
			Note: the Elements making up the Segments must not exceed 999 characters.
24	1		RESERVED ERROR CODES IN TRANS. FILE
			Indicates if reserved Data Elements (ENPR, GRPR, ERUT) contained in the Data Structure Description are to be described.
		blank	The Description is not generated.
		'V'	The Descriptions are generated for all of these Data Elements.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'W'	Same as 'V', but the Data Element ENPR represents the error vector. (Reserved for USAGE OF D.S. = 'M', 'N' or 'E'.)
		'E'	Only the 'ENPR' and 'GRPR' Descriptions are generated.
		'U'	Only the 'ERUT' Description is generated.
			In a transaction file (USAGE OF D.S.= 'M', 'N' or E'), these Data Elements must appear at the beginning of the Description and are used to carry results of validations to the update.
			.ENPR: n+1 positions for values 'V' or 'E' and m+1 positions for value 'W', where:
			n = number of elementary Data Elements in the Data Structure description.
			m = greatest number of elementary Elements in the file : that is, those in the common part Segment plus the largest non-00 Segment. The extra position is the identification error.
			It initializes the DE-ERR vector.
			.GRPR: 1 position per record + 1 for group error.
			It initializes the SE-ERR vector.
			When these Elements are used in a file other than a transaction-type file, the placement and format is at the option of the user.
		1..9,0	With the Pactables function, it specifies the number of sub-schemas desired. Refer to the 'Pactables' Reference manual.
			With an SQL utilization file, it specifies the number of the sub-schemas desired (selection of a Column in a Table).
25	1		RECORD TYPE / USE WITHIN D.S.
			This option is used to select the type of record description to be used in the COBOL Program to allow different uses of the Segment Description stored in the Library.
		blank	Redefined records (Default option). No VALUE clause is generated.
		'1'	A record set without initial values or repetitions of records. These records are presented with the Segment common part followed by the different specific parts.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			If the Data Structure Description appears in the COBOL FILE SECTION, the LEVEL NUMBER (COBOL) OF THE RECORD must be 2. With this value, the specific Segments are described without redefines, at the COBOL 02 level. Several Segment Descriptions are grouped together under the same I/O area.
		'2'	A record set with the specific initial values of the Data Element of the Segment as defined on the Call of Elements or Data Element Description screen. These values may also default to blank or zero depending on the format.
			This type of description cannot be used for a Data Structure having a number of repetitions in the common part Definition. (Use ORGANIZATION = 'W' or 'L').
			Initial values are also generated for the occursed fields if the 'Generated language' of the Library is set to 'D' (COBOL II, 85, or 370).
		'3'	A record set which incorporates the number of repetitions specified in OCCURRENCES OF SEGMENT IN TABLE on the Segment Definition Screen. No VALUE clause will be generated.
			If the description of the Data Structure appears in the COBOL FILE SECTION, the LEVEL NUMBER (COBOL) OF THE RECORD must be '2'.
		'4'	A record set which incorporates the number of repetitions specified in the OCCURRENCES OF SEGMENT IN TABLE on the Segment Definition Screen.
			The associated LEVEL NUMBER (COBOL) OF THE RECORD must be '3'.
			Comment specific to the OLSD function: For a description type of '4' and a COBOL 03 level, the index is not generated.
			A COBOL 02 level is used to access the table made up of repetitions of the same record (ddssT).
			A COBOL 01 level is used to group the whole Data Structure together - common or specific parts, whether repeated or not.
			A group level field that incorporates all occurrences is generated.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			For Data Structures that do not have a value specified for the OCCURRENCES OF SEGMENT IN TABLE, use ORGANIZATION = 'W' with USAGE OF Data Structure = 'T'.
26	1		LEVEL NUMBER (COBOL) OF THE RECORD
			This option, used in conjunction with the RECORD TYPE /USE WITHIN D.S. field, defines the COBOL level number for the descriptions of Data Structures, Segments and Elements.
			In the following descriptions, the term 'D.S. Area' is meant as the area 'dd00' (possibly 1-dd00, 2-dd00).
		'1'	COBOL 01 level for D.S. Area and Segments. (Default value).
			If the Data Structure Description appears in the COBOL FILE SECTION, the Segments must be redefined.
			If a Data Structure has no common part with a non-redefined Description, the D.S. Area will only appear when the RECORD TYPE / USE WITHIN D.S. = blank.
		'2'	COBOL 01 level for D.S. Area and Segments at 02 level.
			If the RECORD TYPE / USE WITHIN D.S. = blank, both the DS Area and the Segments will be described at the 02 level. (To define the 01 level, use ORGANIZATION = 'L' and Work Areas (-W) lines.)
		'3'	Reserved for D.S. with an ORGANIZATION = 'W' or 'L'.
			COBOL 02 level for the D.S. Area and Segments at 03 level when associated with RECORD TYPE / USE WITHIN D.S. = 1, 2, or 3.
			01 level for the D.S. Area and Segments at 03 level when associated with RECORD TYPE / USE WITHIN D.S.= 4.
			03 level for both the D.S. Area and the Segments when associated with RECORD TYPE / USE WITHIN D.S. = blank.
		'4'	Reserved for Data Structures with an 'L' ORGANIZATION and USAGE OF DATA STRUCTURE = 'D'. The 01 level is to be defined via the Work Areas Screen (-W).
			COBOL 02 level for group Data Elements or elementary Elements that are not part of a group.
			Elementary Elements that are part of a group appear. The D.S. Area and Segment levels disappear.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'5'	Reserved for Data Structures in ORGANIZATION 'L' or 'W' and with a USAGE OF DATA STRUCTURE = 'D'.
			COBOL 01 level for group Data Elements or elementary Elements that are not part of a group.
			Elementary Elements that are part of a group appear. The D.S. Area and Segment levels disappear.
		'6'	Reserved for Data Structures with an 'L' ORGANIZATION and USAGE OF DATA STRUCTURE = 'D'. The 01 level is to be defined via the Work Areas Screen (-W).
			COBOL 02 level for group Data Elements or elementary Elements that are not part of a group.
			Elementary Elements that are part of a group disappear as well as D.S. Area and Segment levels.
			For standard OLSD Screens only.
		'7'	Reserved for Data Structures in ORGANIZATION 'L' or 'W' and with a USAGE OF DATA STRUCTURE = 'D'.
			COBOL 01 level for group Data Elements or elementary Elements that are not part of a group.
			Elementary Elements that are part of a group disappear as well as D.S. Area and Segment levels.
			For standard OLSD Screens only.
27	2		CODE FOR COBOL PLACEMENT
			PSEUDO-NUMERIC FIELD, blanks replaced by zeros.
			This field concerns only the principal Description of a D.S. (ddss) and not the Descriptions preceded by a prefix (1-ddss or 2-ddss).
			This field is used to obtain a Description of a D.S. in a particular area (COMMUNICATION area with DBMS's or the LINKAGE SECTION which the user must define by a Work Areas (-W) line), or at the beginning of the WORKING-STORAGE SECTION.
			This field is reserved for D.S.'s with an 'L', 'D' or 'W' ORGANIZATION, in order to place the I/O area in WORKING STORAGE.
			To have a Data Structure described in WORKING-STORAGE it is preferable to use the Work Areas (-W) lines.
		'00'	The Description of the D.S. is inserted after all the Work Areas (-W) lines. (Default value).

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		alphabet.	The Description of the D.S. is inserted after all the Work Areas (-W) lines whose 5-digit line number begins with this value.
			The Description and Work Areas (-W) lines are found at the beginning of the generated Program WORKING-STORAGE SECTION. These lines appear both before Data Structures with ORGANIZATION = 'W' and before those whose DATA STRUCTURE CODE IN THE PROGRAM is greater than this alphabetic code.
			(Do not use this field with a Data Structure whose ORGANIZATION = 'W'.)
		alphanum.	The Description of the D.S. is inserted after all the Work Areas (-W) lines whose 5-digit line number begins with this value. The Work Areas (-W) lines and the Description can be found in the generated Program, at the end of the WORKING-STORAGE SECTION among the user areas.
			The location is indicated on the first line of the D.S. call (CONTINUATION OF DS DESCRIPTION field = blank), and is repeated (by default) on all of its continuation lines.
			However, it is possible to attribute different locations to each record description of D.S. in a Program. This is done by entering several call lines for this D.S., specifying a record selection and a location for each one.
			Therefore, the Data Structure must have an unpacked description, whether implicit or explicit.
			WARNING: with ORACLE, you must use numeric values so that the DECLARE SECTION will be correctly generated (with data fields and indicators included in it).
28	10		STATUS FIELD - FILE INDICATOR
			(Note: In this discussion, the term 'COBOL Variant' = the value in the TYPE OF COBOL TO GENERATE field)
			Enter the DATA STRUCTURE, SEGMENT and DATA ELEMENT CODEs in the following format:
			ddsseeeeee
			(Recommendation: ss = 00).
			This field is used in one of three ways:
			For VSAM files
			.The FILE STATUS IS clause is generated using 1-ddss-eeeeee (declared as a two byte field).
			For hardware other than Gcos8 BCD and non-VSAM files

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			.The NOMINAL, SYMBOLIC or ACTUAL KEY depending on the COBOL Variant.
			The user must define the corresponding work area: 1-ddss-eeeeee.
			The positioning of this key as well as the read of the D.S. must be programmed by using Procedural Code (-P).
			For Gcos8 BCD (COBOL Variant 6)
			.Identification of the Data Structure.
			.The corresponding 'VALUE OF' clause will be generated only if it's filled in.
			.The return-code area of the input-output operations
			.The corresponding 'FILE STATUS IS' clause will be generated only if it's filled in.
29	6		Indexed Data Structure Access Key
			Required for indexed Data Structures: Enter the DATA ELEMENT CODE of the access key Element.
30	6		CODE OF RECORD TYPE ELEMENT
			Enter the code of the Data Element whose values define different record types of a Data Structure.
			Note: Must be in the common part (00 Segment).
			This code can also be specified on the Segment Definition Screen for the 00 Segment in the CODE OF RECORD TYPE ELEMENT field, and is then used as a default value at generation level.

Work Areas Screen (-W)

The Work Areas (-W) screen completes the WORKING-STORAGE SECTION, LINKAGE SECTION, and the other supplementary sections that constitute the Work Areas of the DATA DIVISION.

This screen is used to accomplish the following:

- Call in Data Structures that already exist in the Dictionary;
- Call in Data Elements that already exist in the Dictionary (with or without a Segment), in the desired format;
- Declare Data Elements that do not exist in the Dictionary;
- Write in Source languages other than COBOL, in free structure Programs (PROGRAM TYPE = 'S');

- Name additional COBOL sections.
- Generate the indexes used in a table search (with the 'SCH' OPERATOR). This is done by associating a TABLE SIZE (OCCURS CLAUSE) value to the DATA STRUCTURE and SEGMENT CODE in the WORK AREA DESCRIPTION field.

RECOMMENDATIONS

The Data Structure (-CD) call screen defines resources that are external to the Program (file, Databases, etc.). WORKING-STORAGE SECTION and LINKAGE SECTION fields are grouped together in the '-W' screen, which makes it easy to organize them.

Furthermore, it is the '-W' lines of a Macro-Structure that are incorporated into calling Programs, and not Data Structure (-CD) calls. Be sure that the Macro-Structure '-W' keys do not conflict with those of the calling Program or of other Macro-Structures.

CALLING DATA STRUCTURES

Data Structures are called by using 'F'-type lines. An input guide is used to enter the attributes of the Data Structure. (See the TYPE OF LINE or DATA ELEMENT FORMAT' field in the Screen Description.)

```
                  PURCHASING MANAGEMENT SYSTEM        SG000008.LILI.CIV.1583

WORK AREAS.....ENTITY TYPE O SA0010 *** REQUEST INPUT  ***
                                1 2
CODE FOR PLACEMENT..:  3  AB
4 5   6 7              8                                                  9
A LIN T LEVEL OR SECTION WORK AREA DESCRIPTION                           OCCURS
* 010 *             -->  MESSAGE BEFORE PROVOKED ABEND <---
* 100   01              ABEND-MESS.
* 120     05             FILLER       PIC X(24)   VALUE
* 130                    'TRANSACTION TERMINATION '.
* 150     05             ABEND-TRANS  PIC X(4).
* 170     05             FILLER       PIC X(11)   VALUE
* 180                    ' : FILE '.
* 200     05             ABEND-DDNAME PIC X(8).
* 220     05             FILLER       PIC X       VALUE SPACE.
* 240     05             ABEND-RMESS  PIC X(8).
* 260     05             FILLER       PIC X(23)   VALUE
* 270                   '. CALL EXTENSION 345.'.




O: C1 CH: -W
```

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		ENTITY TYPE
			This field is used to identify the entity to which these lines are attached:
		'O'	On-line Screen
		'P'	Program
			The user may keyboard this field in order to copy lines attached to a Screen into a Program and vice-versa.
2	6		PROGRAM CODE OR SCREEN CODE (REQUIRED)
			This field contains the six-character program or on- line screen code.
3	2		CODE FOR COBOL PLACEMENT (REQUIRED)
			PSEUDO-NUMERIC FIELD

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			Valid values for this field are alphabetic characters, (blanks replaced by zeros), numeric characters, and $n for a parameterized value in a P.M.S. This value is used to determine the placement and the sequence in which data entered on this screen will be generated in the DATA DIVISION. These characters form the first two digits of a sequencing number, with the value in the LINE NUMBER field as the last three.
			For Batch programs:
		'AA to ZZ' '0A to 0Z'	A CODE FOR COBOL PLACEMENT smaller than '00' causes the data entered on this screen to be generated at the beginning of the WORKING-STORAGE SECTION. Relatively to Data Structures called via the Call of Data Structures (-CD) screen, these data will be generated as follows:
			.before the description of Data Structures with ORGANIZATION = 'W' and whose DATA STRUCTURE CODE IN THE PROGRAM matches this prefix or is greater than it,
			.before the description of Data Structures with ORGANIZATION = 'L' or 'D' and whose CODE FOR COBOL PLACEMENT (on -CD screen) matches this prefix or is greater than it.
		'00 to 09' '1A to 19' ... '9A to 99'	A CODE FOR COBOL PLACEMENT greater than '00' causes the data entered on this screen to be generated in the WORKING-STORAGE SECTION, after all Data Structures whose CODE FOR COBOL PLACEMENT (on -CD screen) is smaller than this prefix.
			For On-Line Programs:
		'AA to 0Z'	If this value is less than '00' (from 'AA' to '0Z'), the description is generated in the WORKING-STORAGE SECTION.
		'00 to 99'	Otherwise, it is generated in the LINKAGE SECTION.
		'AA'	This value is used by the system for data generated automatically.
		'00'	This value is used by the system for data generated automatically.
			Other codes may be reserved for special usage depending upon the TP monitor type chosen for generation.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'99'	With LINE NUMBER = '999': This value is used by the system for the 'PROCEDURE DIVISION' statement. Therefore, you may use it to create a line with a sequencing number '99999', which will replace the generated statement.
		'$n'	In a Parameterized Macro-Structure, this value may be parameterized.
4	1		ACTION CODE
		'C'	Creation of the line
		'M'	Modification of the line
		'D' or 'A'	Deletion of the line
		'T'	Transfer of the line
		'B'	Beginning of multiple deletion
		'G'	Multiple transfer
		'?'	Request for HELP documentation
		'E' or '-'	Inhibit implicit update
		'X'	Implicit update without upper/lowercase processing
5	3		LINE NUMBER
			BLANKS REPLACED BY ZEROS
		'0-999'	As a recommendation, number the lines starting with 10 by intervals of 10, thus facilitating future insertions.
		$n0 to $n9	In a Macro-Structure, only the first two characters of the LINE NUMBER can be parameterized.
6	1		Type of line or Data element format
			Type of line values:
		blank	Data entered in the LEVEL AND SECTION and WORK AREA DESCRIPTION fields are to be generated as entered.
		'-'	Continuation character for a literal.
		'*'	Comment. Data entered in the LEVEL AND SECTION and WORK AREA DESCRIPTION fields contain comments to be inserted into the generated Program (Table size is not considered as comment, see line's last field).
		'$'	This value appears in column 7 of the generated COBOL and the other Elements of the WORKING line appear as it is.
		'A'	Call of an eBusiness Application. This call is fully documented in the 'COBOL API User's Guide'.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'F'	Call of a Data Structure.
			When 'F' is entered, the system responds with a formatted line which is used to facilitate data entry. The fields are the same as those used on the Call of Data Structures (-CD) screen for D.S. with ORGANIZATION = 'W' or 'L'.
			.Data Structure code in the Program.
			.Data Structure code in the Library.
			.Segment selection (enter the Segment code without an asterisk).
			(A segment code can only be renamed in batch).
			.Non-Printing Data Structure format (1 to 8).
			.Record type / Use within D.S. (I, E or S).
			.Level number (Cobol) of the record (1 to 5).
			.Organization.
			.Sub-schema number.
			Type 'F' '-W' lines are processed as Data Structure call lines (-CD) only for batch.
			If two Type 'F' '-W' lines referring to the same Data Structure (same Data Structure code in the Program) are separated, they will nevertheless be generated one after the other.
			Element format values:
		'E'	Use of the Input format of a Data Element.
		'I'	Use of the Internal format of a Data Element.
		'S'	Use of the Output format of a Data Element.
			For these format types, the presence of the Data Element in the Specifications Dictionary is checked. A cross-reference is established, which prohibits the deletion of the Data Element whenever the lines in which it is called have not been deleted themselves.
			If the Element does not exist in the Specifications Dictionary, the System sends a warning.
			When a global replacement is required (.C2), the Element is not checked but the cross-references will still be created.
			For these three format types, the entered data-name must therefore have the following format:
			W-DDSS-EEEEEE where:

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			W = a working-storage prefix,
			DDSS = a given Data Structure and Segment code,
			EEEEEE = a Data Element code which exists in the Specifications Dictionary.
			The corresponding format is automatically attributed by the System.
			For IMS sub-monitors:
		'M'	Sub-monitor; enter the code of the sub-monitor in the LEVEL OR SECTION field.
		'C'	Call of a screen into the sub-monitor named above.
			Enter the SCREEN CODE of the screen belonging to the sub-monitor in the LEVEL OR SECTION field, followed by a space and a 'D' for Dynamic call or 'S' for Static.
			Example: C OOSCRN D
			Note: Enter one SCREEN CODE per 'C'-type line.
7	17		LEVEL OR SECTION
			Enter a COBOL Level Number (example: 01, 05,....) or a section name (example: LINKAGE SECTION).
		'$n'	In a macro-structure, this value can be parameterized.
8	48		WORK AREA DESCRIPTION
			The user should always use data-names that conform to the standards recognized by the System.
			The structure of these names is 'w-ddss-eeeeee', where:
			. w = Working-storage prefix (alpha or numeric),
			. dd = DATA STRUCTURE CODE, including the work area,
			. ss = SEGMENT CODE,
			. eeeeee = DATA ELEMENT CODE.
			If this standard is followed, the Data Element/Program cross-references will be established automatically.
			Values entered in this field appear on the same line in the generated code, as the value entered in the LEVEL OR SECTION field.
			When used in combination with the TABLE SIZE (OCCURS CLAUSE) field, the value entered in this field must be left-justified. The prefix ('w-') may be omitted, however, two spaces must then be entered to replace them.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'$n'	In a Macro-Structure, the WORK AREA DESCRIPTION value can be parameterized.
			NOTES: The period is not automatically generated after 'PICTURE', enabling the user to enter a COBOL clause (like VALUE, JUSTIFIED, etc.), which must be entered on the following line.
			The period must be explicitly indicated at the end of a declaration.
			When a Data Element is called into a WORKING STORAGE or LINKAGE SECTION field, if the Data Element code exists in the Dictionary, the data name must be entered in this field. Otherwise, the generated code must be in the following format:
			03 DDSS-DELCO PICTURE X.
9	5		TABLE SIZE (OCCURS CLAUSE)
			PURE NUMERIC FIELD
			Enter the maximum number of occurrences for the table.
			An entry in this field causes the generation of the three indices: IddssM, IddssL and IddssR.
			.IddssM: initialized to the value entered.
			.IddssL: initialized to zero.
			This index may be used to load the table. It keeps track of the actual table size.
			.IddssR: initialized to zero.
			This index may be used for table searches.
			The DATA STRUCTURE and SEGMENT CODEs are entered, prefixed with some work area prefix code in the standard VA Pac format: 'w-ddss' or 'w-ddss-eeeeee'. This value MUST be left-justified in the WORK AREAS DESCRIPTION field. The prefix may be replaced by spaces.
			Note: This can be done on a comment line ('*' as the TYPE OF LINE value).
		'$n'	In a Macro-Structure, this value may be parameterized.
			This field is not taken into account when used in a formatted line.

Work Areas Formatted Line

When a Data Structure that was previously defined is to be used as a work file, the user may call this Data Structure into the WORKING-STORAGE (or LINKAGE) SECTION by requesting a Formatted Line. This is done by entering 'F' in the TYPE OF LINE field (a LINE NUMBER value is also required). The system will respond with a line containing screen labels for input fields.

INPUT FIELDS

Only the fields that pertain to the formatted line will be described in this subchapter. The fields that pertain to the Work Areas (-W) screen as a whole are described in the previous subchapter.

The formatted line fields for the most part are a subset of the fields that appear on the Call of Data Structures (-CD) screen, and are used in a similar fashion. The exceptions to this rule are:

- The SEGMENT SELECTION field is used to select Segments within a Data Structure to be described. On the Call of Data Structures screen, the user would need to enter an asterisk prior to the SEGMENT CODE. On the Work Areas screen, no asterisk is to be entered. For on-line Programs, the common part Segment (00) must be explicitly entered. With batch Programs, it is implicitly selected (if it exists).
- The SUB-SCHEMA NUMBER field is used with the Pactables function, and is used to specify which sub-schema is to be described.
- The LINE SEQUENCE field does not have a screen label. Its physical location on this line is the column directly to the right of the SUB-SCHEMA NUMBER field. This field is used only for upward compatibility, if needed.

GENERAL INFORMATION

Segments generated as a result of data entered on the formatted line are named according to the following standard: ddss.

Data elements are named: ddss-eeeeee.

```
                  PURCHASING MANAGEMENT SYSTEM        SG000008.LILI.CIV.1583

WORK AREAS.....ENTITY TYPE P TES001 TEST FOR POJ


CODE FOR PLACEMENT..: BB

A LIN T LEVEL OR SECTION WORK AREA DESCRIPTION                          OCCURS
* 020 F DP: XW  DL: XW SEL: 02______ PICT: I DESC: 2 LEV: 1 ORG: _ SS: _
* 030 F DP: XW  DL: XW SEL: 04______ PICT: I DESC: 2 LEV: 1 ORG: _ SS: _
  1   2     3       4       5             6       7      8      9    10 11









O: C1 CH: -W
```

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	3		LINE NUMBER
			BLANKS REPLACED BY ZEROS
		'0-999'	As a recommendation, number the lines starting with 10 by intervals of 10, thus facilitating future insertions.
		$n0 to $n9	In a Macro-Structure, only the first two characters of the LINE NUMBER can be parameterized.
2	1		TYPE OF LINE
		'F'	When 'F' is entered, the system responds with a formatted line which is used to facilitate data entry.
			The fields here, for the most part, are the same as those on the Call of Data Structures (-CD) screen for data structures with ORGANIZATION = 'W', 'L' or 'D'.
			.DATA STRUCTURE CODE IN THE PROGRAM.
			.DATA STRUCTURE CODE IN THE LIBRARY.
			.SEGMENT SELECTION.
			.NON-PRINTING DATA STRUCTURE FORMAT.
			.RECORD TYPE / USE WITHIN D.S.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			.LEVEL NUMBER (COBOL) OF THE RECORD.
			.ORGANIZATION.
			.SUB-SCHEMA NUMBER.
			.LINE SEQUENCE. (Note: This field has no label on the screen).
			Type 'F' '-W' lines are processed as Data Structure call lines (-CD). If two Type 'F' '-W' lines referring to the same D.S. (same DATA STRUCTURE CODE IN THE PROGRAM) are separated, they will nevertheless be generated one after the other.
3	2		DATA STRUCTURE CODE IN THE PROGRAM
			This code establishes the sequence in which the Data Structure will be processed in the Program.
			The first character must be alphabetic but the second one can be numeric or alphabetic.
			It is recommended to keep the same DATA STRUCTURE CODE IN THE PROGRAM and IN THE LIBRARY when the Data Structure described in the Library is used only once in the Program.
4	2		DATA STRUCTURE CODE
			This code is made up of two alphanumeric characters. This is a logical code internal to the Database and therefore independent of the names used in Database Blocks and Programs.
5	8		SEGMENT SELECTION
			Rather than describing all of the Segments belonging to a Data Structure, the user may select the ones that are needed, thus avoiding unnecessary description lines and wasted work area space.
			Enter the SEGMENT CODE (up to four are allowed) for the Segments to be described. Do not enter spaces between these codes.
6	1		NON-PRINTING DATA STRUCTURE FORMAT
			This option is reserved for Data Structures with a USAGE OF DATA STRUCTURE other than 'I' or 'J'.
		'E'	Input format. (Default option with USAGE OF D.S. = 'M', 'N' or 'E').
		'I'	Internal format (Default with USAGE OF D.S. NOT= 'M', 'N' or 'E').

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'S'	Output format.
			Note: the Elements making up the Segments must not exceed 999 characters.
7	1		RECORD TYPE / USE WITHIN D.S.
			This option is used to select the type of record description to be used in the COBOL Program to allow different uses of the Segment Description stored in the Library.
		blank	Redefined records (Default option). No VALUE clause is generated.
		'1'	A record set without initial values or repetitions of records. These records are presented with the Segment common part followed by the different specific parts.
			If the Data Structure Description appears in the COBOL FILE SECTION, the LEVEL NUMBER (COBOL) OF THE RECORD must be 2. With this value, the specific Segments are described without redefines, at the COBOL 02 level. Several Segment Descriptions are grouped together under the same I/O area.
		'2'	A record set with the specific initial values of the Data Element of the Segment as defined on the Call of Elements or Data Element Description screen. These values may also default to blank or zero depending on the format.
			This type of description cannot be used for a Data Structure having a number of repetitions in the common part Definition. (Use ORGANIZATION = 'W' or 'L').
			Initial values are also generated for the occursed fields if the 'Generated language' of the Library is set to 'D' (COBOL II, 85, or 370).
		'3'	A record set which incorporates the number of repetitions specified in OCCURRENCES OF SEGMENT IN TABLE on the Segment Definition Screen. No VALUE clause will be generated.
			If the description of the Data Structure appears in the COBOL FILE SECTION, the LEVEL NUMBER (COBOL) OF THE RECORD must be '2'.
		'4'	A record set which incorporates the number of repetitions specified in the OCCURRENCES OF SEGMENT IN TABLE on the Segment Definition Screen.
			The associated LEVEL NUMBER (COBOL) OF THE RECORD must be '3'.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			Comment specific to the OLSD function: For a description type of '4' and a COBOL 03 level, the index is not generated.
			A COBOL 02 level is used to access the table made up of repetitions of the same record (ddssT).
			A COBOL 01 level is used to group the whole Data Structure together - common or specific parts, whether repeated or not.
			A group level field that incorporates all occurrences is generated.
			For Data Structures that do not have a value specified for the OCCURRENCES OF SEGMENT IN TABLE, use ORGANIZATION = 'W' with USAGE OF Data Structure = 'T'.
8	1		LEVEL NUMBER (COBOL) OF THE RECORD
			This option, used in conjunction with the RECORD TYPE /USE WITHIN D.S. field, defines the COBOL level number for the descriptions of Data Structures, Segments and Elements.
			In the following descriptions, the term 'D.S. Area' is meant as the area 'dd00' (possibly 1-dd00, 2-dd00).
		'1'	COBOL 01 level for D.S. Area and Segments. (Default value).
			If the Data Structure Description appears in the COBOL FILE SECTION, the Segments must be redefined.
			If a Data Structure has no common part with a non-redefined Description, the D.S. Area will only appear when the RECORD TYPE / USE WITHIN D.S. = blank.
		'2'	COBOL 01 level for D.S. Area and Segments at 02 level.
			If the RECORD TYPE / USE WITHIN D.S. = blank, both the DS Area and the Segments will be described at the 02 level. (To define the 01 level, use ORGANIZATION = 'L' and Work Areas (-W) lines.)
		'3'	Reserved for D.S. with an ORGANIZATION = 'W' or 'L'.
			COBOL 02 level for the D.S. Area and Segments at 03 level when associated with RECORD TYPE / USE WITHIN D.S. = 1, 2, or 3.
			01 level for the D.S. Area and Segments at 03 level when associated with RECORD TYPE / USE WITHIN D.S.= 4.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			03 level for both the D.S. Area and the Segments when associated with RECORD TYPE / USE WITHIN D.S. = blank.
		'4'	Reserved for Data Structures with an 'L' ORGANIZATION and USAGE OF DATA STRUCTURE = 'D'. The 01 level is to be defined via the Work Areas Screen (-W).
			COBOL 02 level for group Data Elements or elementary Elements that are not part of a group.
			Elementary Elements that are part of a group appear. The D.S. Area and Segment levels disappear.
		'5'	Reserved for Data Structures in ORGANIZATION 'L' or 'W' and with a USAGE OF DATA STRUCTURE = 'D'.
			COBOL 01 level for group Data Elements or elementary Elements that are not part of a group.
			Elementary Elements that are part of a group appear. The D.S. Area and Segment levels disappear.
		'6'	Reserved for Data Structures with an 'L' ORGANIZATION and USAGE OF DATA STRUCTURE = 'D'. The 01 level is to be defined via the Work Areas Screen (-W).
			COBOL 02 level for group Data Elements or elementary Elements that are not part of a group.
			Elementary Elements that are part of a group disappear as well as D.S. Area and Segment levels.
			For standard OLSD Screens only.
		'7'	Reserved for Data Structures in ORGANIZATION 'L' or 'W' and with a USAGE OF DATA STRUCTURE = 'D'.
			COBOL 01 level for group Data Elements or elementary Elements that are not part of a group.
			Elementary Elements that are part of a group disappear as well as D.S. Area and Segment levels.
			For standard OLSD Screens only.
9	1		ORGANIZATION
		'G'	Table description (Pactables).
			Causes the communication area with the access module to be generated. See the 'Pactables' Reference manual.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'D'	Reserved for the Description of Segments or records of the different Databases, IMS (DL/1), IDS I, IDS II, (according to the TYPE OF COBOL TO GENERATE selected), in the generation of DBD, SYSGEN, schemas or application Programs (according to the TYPE AND STRUCTURE OF PROGRAM selected).
		'blank'	Defaults to ORGANIZATION of 'L' or 'W' depending on the value of the CODE FOR COBOL PLACEMENT.
		'A'	Reserved for an ADABAS file description in the definition Programs or usage Programs of the Database.
		'T'	Reserved for the description of 'TOTAL' files in the Definition Programs or the usage Programs of the Database.
		'Q'	Reserved for the Description of SQL/DS, DB2/2 or DB2/6000 Databases (IBM), or ALLBASE/SQL Databases (HP3000), or DB2/2 or DB2/600 Databases(MICROFOCUS).
		'2'	Generation-Description of a DB2 or VAX/SQL Segment.
			Only physical accesses are not generated. The structure of variable indicators corresponding to to the columns of the DB2 or VAX/SQL table is always generated.
		'C'	Reserved for the Description of an INTEREL RDBC, RFM Database Structure.
		'O'	Reserved for the Description of an ORACLE (< V6) Database Structure.
		'P'	Reserved for the Description of an ORACLE (V6 and V7) Database Structure.
		'R'	Reserved for the Description of an RDMS Database Structure.
		'4'	Reserved for the Description of a DB2/400 Database Structure.
		'N'	Reserved for the Description of a NONSTOP SQL Database Structure.
		'M'	Reserved for the Description of a DATACOM DB Database Structure.
		'9'	Reserved for the Description of an INFORMIX, SYBASE, INGRES/SQL, and SQL SERVER Database Structure.
			The use of the System with the different DBMS's is documented in specific 'Databases' manuals.
10	1		SUB-SCHEMA NUMBER

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'0 to 9'	Indicates the sub-schema to be selected. (The value '0' corresponds to sub-schema 10.)
11	1		LINE SEQUENCE
		'*'	This value is used to generate segments from Formatted Work Areas lines according to a generation method which should NOT be used currently. The purpose of this specification is to facilitate the maintenance of programs generated with this method, i.e., before the sub-release of VisualAge Pacbase 7.1. This field has no screen label. The field is directly to the right of the SUB-SCHEMA NUMBER field.

Chapter 5. Modifying the Procedure Division

Introduction

ORGANIZATION OF THE CHAPTER

This chapter contains a discussion of the concepts of Procedural code, as well as the Preview Facility. Since the Procedures Generated (-PG) screen is closely related to the Procedural Code (-P) screen, these two will be documented in the same subchapter. The Titles Only (-TO) screen is mentioned in the 'Titles and Conditions Screen (-TC)' subchapter.

MODIFICATION OF THE PROCEDURE DIVISION

This Chapter discusses modifications to the PROCEDURE DIVISION of a program through the use of Procedural Code (-P) lines attached directly to a batch or on-line Program. The user can also use the VisualAge Pacbase Preview Facility, which includes the Procedures Generated (-PG) screen, the Titles and Conditions (-TC) screen and the Titles Only (-TO) screen.

- The Procedures Generated (-PG) screen allows the user to write specific procedures and, at the same time, view the titles of automatically generated procedures.
- The Titles and Conditions (-TC) screen allows the user to view the general structure (titles and conditions of all procedures) of a batch or on-line Program.
- The Titles Only (-TO) screen lets the user view the hierarchical organization of program functions.

TRANSFER OF PROCEDURAL CODE (-P) LINES TO ANOTHER ENTITY

Procedural Code (-P) lines may be copied directly to another entity. See paragraph 'Transfer of lines to another entity in chapter 'Modifying the Identification / Environment Division'.

Procedural Code Screen (-P)

The Procedural Code (-P) screen is used to write all Program procedures.

These Program procedures are structured into functions and sub-functions, with each function or sub-function identified as a condition or structure type. They are hierarchically set up by level. Program procedures are described using operators followed by operands.

LEVEL OF SUB-FUNCTIONS

Functions are always an 05 level. Sub-functions have a 10 level by default. However, they can be an 06 level to a 98 level.

Within a given function, a 15-level sub-function is part of the 06- to 14-level sub-functions which precede it. In other words, a sub-function of a logically lower level will have a level number that is greater (ex: a 15-level sub-function is logically dependent on, or inferior to, a 14-level sub-function).

In this way, a sub-function dependent on another sub-function (i.e., a 15-level sub-function included in a 14-level sub-function), is only executed under the conditions of execution of the logically higher level sub-function (14-level in this case).

ELEMENTARY PROCEDURES

An elementary procedure is a series of condition lines.

The '99' level is reserved for elementary procedures. It is used to write a condition without changing the sub-function code. This condition applies until the next occurrence of a '99' level or until the end of the sub-function.

A '99' level procedure is limited to 75 lines. A sub-function can contain a maximum of 98 '99' levels.

A sub-function with no title line (N in the OPERATOR field) is assumed to be an elementary procedure and automatically assigned a '99' level.

CONDITION TYPE OR S.F. STRUCTURE

A function can be only an 'IT' (IF THEN) type if its execution depends on a condition. Otherwise, it must be a 'BL' (BLOCK) type. This is indicated in the CONDITION TYPE OR S.F. STRUCTURE field on the first line of the procedure.

If the CONDITION TYPE OR S.F. STRUCTURE is not indicated, the default assumed for any one of these options is selected according to what has been entered in the CONDITION FOR EXECUTION field. If an execution condition is entered, the System defaults to the 'IF THEN' structure; if an execution condition is not entered, it defaults to the 'BLOCK' structure.

For the sub-functions and the elementary procedures, the default options are the same. However, the user can also indicate more complex types of structures.

IF THEN ('IT') & ELSE ('EL')

The sub-function type 'IT' (IF THEN) can be followed by an 'EL' (ELSE) type sub-function of the same level.

The 'ELSE' sub-function will be executed if the 'IF THEN' sub-function condition has not been met. The 'ELSE' must directly follow the 'IF THEN' sub-function.

CASE OF ('CO')

The name of the variable which conditions the different procedures following the 'CASE OF' must be included in the 'CASE OF' statement.

The 'CASE OF' structure is followed by sub-functions of the 'IT' type (IF THEN) at the next logically lower level. The variable value corresponding to the condition for execution of the sub-function is specified each time.

The 'IF THEN' ('IT') sub-functions that depend on a 'CASE OF' sub-function must all be on the same level. They can be broken down into logically lower level sub-functions.

The last sub-function which depends on the 'CASE OF' sub-function can be a 'BLOCK' ('BL') type sub-function (non-conditioned). This last sub-function must be on the same level as the 'IF THEN' sub-functions. It will be executed when none of the 'IF THEN' conditions have been met.

If the last sub-function is not a 'BLOCK' type and none of the 'IF THEN' conditions are met within the 'CO' structure type, processing continues with the first sub-function at a higher logical level than the 'IT' sub-functions.

```
15     CO  ddss-eeeeee
  16   IT  value1
  16   IT  value2
  16   IT  value3
  16   BL
```

LOOPS

There are three types of 'LOOP' structures:

DO WHILE ('DW'), DO UNTIL ('DU') and DO ('DO').

A 'DW' ('DO WHILE') sub-function is only executed 'while' the indicated condition is true.

A 'DU' ('DO UNTIL') sub-function is executed at least once and 'until' the indicated condition is met.

A 'DO' ('DO') sub-function is executed as many times as indicated in the condition.

The user must be careful to correctly specify the conditions to be met in the first two types of sub-functions in order to avoid an infinite loop.

'WARNING' TYPE ERROR MESSAGE

When a 'WARNING' type error message is displayed, the character 'W' appears in the ACTION CODE field. The user can ignore the message by pressing Enter again.

CONDITION FOR EXECUTION

The construction of the lines of the Procedural Code (-P) screen separates the CONDITION FOR EXECUTION of a procedure from the procedure itself. That is, the left part of the screen (the OPERAND FIELD) is used for the statement and the right part for the CONDITION FOR EXECUTION, if any.

Writing a CONDITION FOR EXECUTION of a function or sub-function begins on the first line of that function or sub-function and continues onto as many lines as necessary, up to a limit of 24 lines (23 lines in case of 'Do Until').

These lines may or may not include processing statements.

However, they will be executed under the global conditions set.

Note on DATE PROCESSING OPERATORS of the On-Line Systems Development function:

When the condition is entered on several lines, the continuation lines may not contain operands. The operands must be entered before the condition continuation.

In order to facilitate the writing of a condition, the CONDITION TYPE OR S.F. STRUCTURE field must be used to indicate the 'AN' (AND) and/or 'OR' (OR) relationships within these conditions.

Parentheses, if needed, must be indicated.

PROCEDURES - OPERATORS AND OPERANDS

Procedures written in Procedural Code are written with OPERATORS followed by OPERANDS.

This makes programs easy to read by isolating the 'verbs' from the manipulated data.

OPERATORS are translated into COBOL and take into account the information provided for the different files and the features of each compiler.

An OPERATOR is indicated only once, even if the OPERANDS continue onto several lines. The one exception to this rule is the '*' (comments) OPERATOR, which must be repeated on each comment line.

TRANSFER 'GO-TO' TYPE BRANCHING

The structure of a program must remain linear. Skipping from one function to another can only be done in sequence.

Branching from one function or sub-function to a preceding function or sub-function breaks the linear flow and is not permitted.

Thus, the only legitimate TRANSFER 'GO TO' TYPE branch is one which branches to the end of the current (sub-)function.

Specific OPERATORS are used for all of the TRANSFER 'GO TO' TYPE branches.

Some OPERATORS and types of functions can be used only with the On-Line Systems Development function (see the OPERATORS field).

NON-STANDARD OPERATORS

The user may specify a paragraph label and a PERFORM instruction for a user-defined function F80. This is done by using the 'Yaa' and 'Xaa' OPERATORs (the 'aa' to be replaced by the user). When this occurs, the system will display a warning message at the bottom of the screen to inform the user that this is a non-standard operator. The letter 'W' will appear in the ACTION CODE field. If the user presses the ENTER key, the system will accept the operator.

FIELD ALIGNMENT

In a release prior to VisualAge Pacbase, the OPERANDS and CONDITION FOR EXECUTION fields were larger and not completely displayed on-line. This no longer applies to the current release (see the JUSTIFICATION OF OPERANDS and the JUSTIFICATION OF CONDITION FIELD fields).

ON-LINE PREVIEW OF THE PROCEDURES

The user can preview a program, via the Procedures Generated (-PG) screen, to see how Procedural Code is integrated with automatically generated functions.

USE OF THE PROCEDURES GENERATED (-PG) SCREEN

The Procedures Generated (-PG) screen allows the user to write specific procedures and visualize simultaneously the titles of generated procedures.

The Procedures Generated (-PG) screen is accessed by entering the following in the CHOICE field:

CH: PppppppPG

Specific procedures written on the Procedures Generated (-PG) screen are described according to the same rules which apply to the Procedural Code (-P) screen.

```
                  PURCHASING MANAGEMENT SYSTEM         SG000008.LILI.CIV.1583

PROCEDURAL CODE       P PO0001 VENDOR REPORTS                FUNCTION: 02
                      1 2                                              3

6 7  8   9   10                                 11 12 13
A SF LIN OPE OPERANDS        4                  LVTY CONDITION   5
  AA     N   GET CURRENT DATE                   10BL
  AA  10 ADT DATOR
- -- --- --- -------------------------------- ---- --------------------------
*** END ***
O: C1 CH: -P
```

```
                  PURCHASING MANAGEMENT SYSTEM        SG000008.LILI.CIV.1583

PROCEDURAL CODE       P BBINIT GENERAL PROCESSING             FUNCTION: 02
                        2                                               3

6 7  8   9   10                                 11 12 13
A SF LIN OPE OPERANDS                            LVTY CONDITION
  BB     N   MONITOR INITIALIZATION             10BL
  BB 100 M   PROGE        K-S$1-PROGE
- -- --- --- ---------------------------------- ---- --------------------------
  CC     N   DISPLAY FIRST RUN                  10IT ICF       =   ZERO
- -- --- --- ---------------------------------- ---- --------------------------
  DD     N   ACQUIRING DATE OF THE DAY          10BL
  DD 100 AD8
- -- --- --- ---------------------------------- ---- --------------------------

THIS SCREEN DISPLAYS GENERATED FUNCTIONS
O: C1 CH: -PG
```

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	1		ENTITY TYPE
			This field is used to identify the entity to which these lines are attached:
		'O'	On-line Screen
		'P'	Program
			The user may keyboard this field in order to copy lines attached to a Screen into a Program and vice-versa.
2	6		PROGRAM CODE OR SCREEN CODE
			This field contains the six-character program or on- line screen code.
3	2		FUNCTION CODE (REQUIRED)
		'AA to 99'	This code determines the placement of the Procedural Code lines in the sequence of functions. This is particularly important when used with the On-Line and Batch Systems Development functions in which automatic functions have pre-determined codes.
		' $n'	In a Macro-Structure, the FUNCTION CODE can be parameterized.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
4	1		JUSTIFICATION OF OPERANDS
			This field has been maintained for a prior release of VisualAge Pacbase.
			If you are using release 7.0 or later, this field is not used.
			Used to left-justify the operands. In other words, to display on the screen the right-hand part of the operands.
		'blank'	Left-justification of the operands field.
		' n'	(Any value other than blank):
			Right-justification of the operands field.
			This option prohibits all updates.
5	1		JUSTIFICATION OF CONDITION FIELD
			This field has been maintained for a prior release of VisualAge Pacbases. If you are using release 7.0 or later, this field is not used.
			Used to left-justify the condition. In other words, to display on the screen the right-hand part of the CONDITION FOR EXECUTION.
		'blank'	Left-justification of the condition field.
		' n'	(Any value other than blank): Right-justification of the condition field. This option prohibits all updates.
6	1		ACTION CODE
		'C'	Creation of the line
		'M'	Modification of the line
		'D' or 'A'	Deletion of the line
		'T'	Transfer of the line
		'B'	Beginning of multiple deletion
		'G'	Multiple transfer
		'?'	Request for HELP documentation
		'E' or '-'	Inhibit implicit update
		'X'	Implicit update without upper/lowercase processing
7	2		SUB-FUNCTION CODE
			Made up of numeric or alphabetic characters.
			This code determines the placement of the Procedural Code within the function.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		' $n'	In a macro-structure, the SUB-FUNCTION CODE can be parameterized.
8	3		LINE NUMBER
			PARAMETERIZABLE NUMERIC FIELD
		'0-999'	As a recommendation, number the lines starting with 10 by intervals of 10, thus facilitating future insertion insertions.
		$n0 to $n9	In a Macro-Structure, only the first two characters of the LINE NUMBER can be parameterized.
			PROCEDURE
			Procedures are written in the format of an operator followed by corresponding operands.
			Operands may be continued onto several lines. When this occurs, the OPERATOR is entered once only: on the first line.
			Normally, VisualAge Pacbase manages the punctuation. This is done according to the hierarchical relationship between the functions and sub-functions. The user can customize the punctuation as needed.
		$n	In a parameterized Macro-Structure, the OPERATOR can be parameterized.
9	3		OPERATORS
			STANDARD PROCESSING OPERATORS
		'N'	Title of function or sub-function (required). The title is to be entered on the first line of a function or sub-function, on a single line.
			Depending on whether or not an entry is made in the CONDITION FOR EXECUTION field, and whether it is a function or a sub-function, the system can assign the default values for LEVEL NUMBER and CONDITION TYPE OR S.F. STRUCTURE. (These may be modified in the normal way by the user.)
		'*'	Comment line. This operator must be repeated for each line of comments.
			See also the COMMENTS INSERTION OPTION field in the Library Definition screen.
		'M'	MOVE
			.The first operand is the source of the MOVE; subsequent operands are the targets.
		'MA'	MOVE ALL

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			.The first operand is the source, followed by the target operands.
		'MC'	MOVE CORRESPONDING
			.The first operand is the source, followed by the target operand.
		'MCI'	MOVE of identical dependent Data Elements.
			.The first operand is the source group field ; the second is the target group field.
			.The transfer is carried out for any dependent Data Element whose code is identical in both groups.
			This operator is effective if you run the GPMC procedure (see the 'Developer's Procedures' manual).
		'MF'	MOVE FUNCTION
			.The first operand is the function, one of COBOL intrinsic functions (ex: UPPER-CASE). Refer to the COBOL documentation for the list of these functions and their syntax.
		'P'	PERFORM
			.Branch to the function or sub-function indicated as the first operand, and return to sequence after executing the (sub-)function 'EXIT'.
			.If a second operand is entered, return to the sequence following the paragraph indicated. (Example: PERFORM F23BB THRU F24CC-FN.)
		'C'	COMPUTE
			.Calculation. The result field must be entered as the first operand, followed by an equal sign ('='). The fields to be computed must be separated by the necessary arithmetic operators and necessary parentheses.
		'A'	ADD
			.Addition of the first operand to the following operand(s).
		'S'	SUBTRACT
			.The first operand is subtracted from the second.
		'MP'	MULTIPLY
			.Multiplication of the first operand by the second.
		'DV'	DIVIDE
			.Division of the second operand by the first.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'MES'	DISPLAY
			.Display constants or parameters defined in operands.
		'ACC'	ACCEPT
			.Accept and transfer parameters or constants in the field defined in the operands.
		'CAL'	CALL
			.Call of the Program or sub-Program defined in the operands. The USING clause must be explicitly written out if it is to be used.
			Branch type operators:
		'GT'	GO TO
			.Branch to the end of the (sub-)function to which the statement belongs. The level of the (sub-)function must be indicated by two numeric characters in the OPERAND field.
			If the (sub)-function type is 'IT' and if it used with an 'EL' type of (sub)-function at the same level, the GT operator, used at this level, branches to the beginning of the 'EL' (sub)-function.
		'GFT'	Go to the end of the iteration.
			When this command is entered from a sub-function with a hierarchically inferior LEVEL NUMBER, the branch is to the end of the processing loop for the sub-function with the controlling level number.
			.On-Line Programs: Branch to the end of the category processing.
			.Batch Programs: Branch to the end-of-run function (F20) and set the end-of-file processing switches.
		'GDI'	Go to the beginning of the iteration. When this
			When this command is entered from a sub-function with a hierarchically inferior LEVEL NUMBER, the branch is to the top of the processing loop for the sub-function with the controlling level number.
			.On-Line Programs: Branch to the next occurrence of the current category or to the next category.
			.Batch Programs: Branch to the top of the iteration loop (F05).
		'GB'	GO TO Ffusf-900

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			.This operator branches to the paragraph that immediately precedes the 'EXIT' for the (sub-) function whose LEVEL NUMBER is entered in the OPERAND field.
			This causes a return to the top of the loop of the (hierarchically) next higher level. If the (sub)-function type is 'IT' and if it used with an 'EL' type of (sub)-function at the same level, the GB operator, used at this level, branches to the end of the 'EL' (sub)-function.
			For all branch type operators, the generated instruction is always followed by a period ('.').
		'EXA'	Generates the COBOL 'EXAMINE' command from the field entered as the Operand, possibly followed by complementary clauses (TALLYING, etc.); refer to the COBOL syntax.
			Note: for a COBOL II generation, use the INS operator instead.
		'INS'	Generates the COBOL 'INSPECT' command from the field entered as the Operand, possibly followed by complementary clauses (TALLYING, etc.); refer to the COBOL syntax.
		'COB'	Pure COBOL: justified at the B margin of the generated program.
		'COA'	Pure COBOL: justified at the A margin of the generated program.
		'U07'	Pure COBOL: justified on column 7 of the generated program.
		'SUP'	Suppresses the generation of the automatic function or sub-function with the same code as the line with this operator.
			Note: In the case of the Batch Module, enter the SUP value on the first line of the sub-function in order to delete it.
		'SCH'	Search. Table Search in the table indicated as the first operand, for the search argument indicated as the second operand. The search assumes a sorted file and starts at the beginning of the table.
			The code of this table must include the work area prefix, if it exists (EX : 1-ddss, or 1-ddss-eeeeee if the table's data element code differs from the code of the search argument). The search argument must be coded according to the System's standards.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			This operator must be used in an elementary structure whose CONDITION TYPE OR S.F. STRUCTURE = 'BL'. (This may be implicit).
			The search is done using the standard indexes (IddssM, IddssR and IddssL). If no match is found, IddssR will be greater than IddssL. See TABLE SIZE (OCCURS CLAUSE) on the Work Areas (-W) screen.
		'SCB'	Search in sorted table. As opposed to the 'SCH' operator, the search stops as soon as the table argument is greater than the search argument.
			COBOL II PROCESSING OPERATORS
		'CON'	Generates the COBOL 'CONTINUE' command (no Operand)
		'EVA'	Generates the COBOL 'EVALUATE' command of the condition the Operand expresses
		'EVT'	Generates the COBOL 'EVALUATE TRUE' command (generally, there is no Operand)
		'EVF'	Generates the COBOL 'EVALUATE FALSE' command (generally, there is no Operand)
		'EEV'	Generates the COBOL 'END-EVALUATE' command (no Operand)
		'EIF'	Generates the COBOL 'END-IF' command (no Operand)
		'EPE'	Generates the COBOL 'END-PERFORM' command (no Operand)
		'ESE'	Generates the COBOL 'END-SEARCH' command (no Operand)
		'INI'	Generates the COBOL 'INITIALIZE' command from the field entered as the Operand
		'SEA'	Generates the COBOL 'SEARCH' command from the field entered as the Operand
		'GOB'	Generates the COBOL 'GO-BACK' command from the field entered as the Operand
		'STR'	Generates the COBOL 'STRING' command before the parameters entered in the Operand field
		'UNS'	Generates the COBOL 'UNSTRING' command before the parameters entered in the Operand field
			PACBENCH C/S AND OLSD OPERATORS
			OPERATORS FOR END-OF-PROCESSING:

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'GF'	Branch to the end of the automatic sub-function where the line is inserted.
			NOTE FOR C/S SCREEN:
			Used only in functions F20, F25, F35 and F60.
			NOTE FOR BUSINESS COMPONENT:
			With ENDV in the OPERANDS field, end of Logical View processing.
		'GFR'	Reception End Processing (branch to 'END-OF-RECEPTION' paragraph).
		'GFA'	End of display processing (branch to 'END-OF-DISPLAY' paragraph).
		'GDB'	Return to the beginning of current iteration.
			NOTE: For all operators for end-of-processing, the generated instruction is always followed by a period ('.'). This means that no 'EL' (ELSE type of conditioning should be used on a line using an operator for end-of-processing, as the generated COBOL would then be erroneous.
			CALL OF PROCEDURES (Specific BUSINESS COMPONENT)
		'XT'	Call of an elementary procedure on Logical View or Segment.
			Refer to the 'Pacbench C/S - Business Logic & TUI Clients' manual, chapter 'Business Component', subchapter 'Writing Procedural Code', section 'Operators used by Pacbench Client/Server'.
			CALL (Specific C/S Screen and SCREEN)
		'OTP'	Immediate call of the screen (external name indicated as the OPERAND). The transfer occurs without delay - processing of the loop may not have completed.
		'OSC'	Call of the screen (code) indicated as the OPERAND.
		'OSD'	Call of the screen (code) indicated as the OPERAND (deferred to the end of reception processing).
			OPERATORS FOR ACCESSING SEGMENTS
		'XR'	Read of the Segment indicated in the OPERAND.
		'XP'	Read of the first record by Dynamic Access. Whatever the system, this OPERATOR always brings up a record. The Segment code is indicated in the OPERAND.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'XRN'	Sequential Read of the Segment indicated in the OPERAND (Dynamic Access).
		'XRU'	Read for update of the Segment indicated in the OPERAND
		'XW'	Write of the Segment indicated in the OPERAND.
		'XRW'	Rewrite of the Segment indicated in the OPERAND.
		'XD'	Deletion of the Segment indicated in the OPERAND.
		'XUN'	Unlocking of the Segment indicated in the OPERAND (except for DL1).
			NOTE FOR OLSD FUNCTION
			By using these operators, the corresponding access function can be generated. When the indicated Segment is a table or an SQL view, make sure the Segment is defined in the screen with either a reception or a display use. The XP and XRN operators are reserved for Segments defined in a repetitive category with a display use.
			OPERATORS FOR ERROR POSITIONING
		'ERU'	Error positioning on the screen.
			In the OPERAND field, enter in positions:
			. 1 to 4: error code (managed by the user)
			. 5: blank
			. 6: DATA ELEMENT CODE (optional) of the erroneous Data Element.
			The error message corresponding to the error code is specified on the Error Messages-Help' screen of the Dialogue. This message will be displayed on the error message line (ERMSG), and if the DATA ELEMENT CODE has been entered, the cursor will be positioned on the Data Element, and error attributes will apply.
			This OPERATOR cannot be used on a repetitive Data Element.
		'ERR'	'Manual' Data Element error.
			In the OPERAND field, enter in positions:
			. 1: error code (alphanumeric character except for '0' or '1', which are reserved for the coding of documentary messages)
			. 2: blank

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			. 3: code (six positions) of the variable Data Element with which the error code is associated. The cursor is positioned and the Data Element takes on the attributes defined for the Data Elements in error. In the case of a repetitive Data Element, its code is indicated, followed by the sequence number of the Data Element instance.
			The use of error messages with the On-Line Systems Development function is detailed in the corresponding Reference Manual.
			NON-STANDARD OPERATORS
		'Yaa'	Generates a COBOL paragraph label for Function F80. Followed by a Segment code (ddss) in the OPERAND field, will generate a 'F80-ddss-aa' COBOL paragraph label.
		'Y'	Operator specific to the Business Component Generation of the automatic function label where this latter has been replaced by the insertion of specific code (*R).
		'Xaa'	With 'Yaa', will generate a PERFORM of paragraph 'F80-ddss-aa'.
		'ERL'	Specific to the Business Component associated with a graphic application (Folder or single-view development).
			This operator is used to position an error on a(n) LOCK/UNLOCK request made by a graphic Client on an occurrence already locked or not, respectively.
			For more details, refer to the 'Pacbench C/S - 'Business Logic' manual.
			The use of error messages with the Structured Code function is detailed in the corresponding Reference Manual.
			BATCH PROCESSING OPERATORS
			These operators cannot be used with the OLSD function.
			The OPE, CLO, R, RN, W and RW commands ensure the opening, closing, read, write and rewrite of files with sequential or indexed-sequential organizations.
			The statements generated are adapted to the specifications indicated on the Call of Data Structure (-CD) screen, for D.S.'s entered on the first 23 line and to the appropriate COBOL variant, as selected.
			Thus, a same 'READ' operator can generate: a READ AT END, a READ INVALID KEY, a CALL GETSEQ or GETRAN, or a RETURN AT END.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			With the exception of the OPEN and CLOSE of a file, these operators will call for the intervention of the 'IK' variable which will take on a value other than zero if there is an abnormal execution of the generated instruction (End of File, Error on Key, etc.).
			You must determine the action to take, according to the value of the 'IK' variable.
			With the following Operators, enter the DATA STRUCTURE CODE IN THE PROGRAM (2 characters) as the only OPERAND.
			The OPERAND must be on the same line as the OPERATOR. If it is on a continuation line ('blank' in OPERATOR field), it will be generated in the 'INVALID KEY' clause.
		'OPE'	OPEN
		'CLO'	CLOSE
		'R'	READ
		'RN'	READ NEXT
		'DEL'	DELETE
			With the following operators, enter the SEGMENT CODE as the Operand. Other options may follow, such as FROM or AFTER.
		'W'	WRITE
		'RW'	REWRITE
		'SRT'	SORT.
			The operands are the parameters following the SORT command.
		'STA'	START
			For START, the D.S. CODE IN THE PROGRAM is entered, followed by the setting of the key:
			EXAMPLE: STA FF NOT < FF00-START generates:
			MOVE 0 TO IK START FF-FILE KEY IS NOT < FF00-START INVALID KEY MOVE 1 TO IK.
		'E'	User defined Error Message.
			The OPERAND field is coded as follows:
			Column 1: A User Error Code character.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			Note: Avoid values '0 to 5' inclusive, as they have predefined meanings.
			Recommendation: Use '6' since this is the value used in standard macros.
			Column 2 to 4: Enter a unique identifying number for this message.
			Column 5: Gravity of the error
			Column 6: Begin your error message.
			Note: The message may be continued in the CONDITION FOR EXECUTION field.
			DATE AND TIME PROCESSING OPERATORS
			DATE PROCESSING OPERATORS
		'ADT'	Call of the System Date.
			The date obtained will have the format YYMMDD in the 'DATOR' constant and in the Data Element indicated in the OPERAND field.
			For IBM hardware:
			An inversion option may be used to indicate the position of the day and month in the system date according to the value entered in the SYSTEM DATE FORMAT INDICATOR field on the Library Defenition screen.
		'ADC'	System date with century: CCYYMMDD (CC = century).
			Note: in COBOL II and COBOL 85, if the year is less than '61', the century is automatically set to '20'.
			Date formatting.
		'AD'	A date may be formatted in different ways.
			When the condition is entered on several lines, the operands must be entered before the condition continuation lines (either on lines without operand or on comment lines).
			The OPERANDS are 'XY DELCO1 DELCO2', where X and Y are each replaced by the value of one of the following codes:
			Code: I
			Generated format: 'Internal' - YYMMDD
			Code: D
			Generated format: 'Display' - MMDDYY or DDMMYY

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			according to the value entered in the DATE FORMAT IN GENERATED PROGRAMS field on the Library Definition screen.
			Code: E
			Generated format: 'Extended' - MM/DD/YY or DD/MM/YY
			according to the value entered in the DATE FORMAT IN GENERATED PROGRAMS field on the Library Definition screen.
			Code: S
			Generated format: 'Internal' - CCYYMMDD
			Code: C
			Generated format: 'Display' - DDMMCCYY or MMDDCCYY
			according to the value entered in the DATE FORMAT IN GENERATED PROGRAMS field on the Library Definition screen.
			Code: M
			Generated format: 'Extended' - DD/MM/CCYY
			according to the value entered in the DATE FORMAT IN GENERATED PROGRAMS field on the Library Definition screen.
			Code: G
			Generated format: 'Extended' - CCYY/MM/DD
			EXAMPLE
			In order to change an 'I'-formatted date into a 'D'-formatted date, 'AD' should be entered in the OPERATOR field and 'ID DELCO1 DELCO2' in the OPERAND field: DELCO1 is the Data Element containing a YY/MM/DD date format (it is possible to use the DATOR constant) and DELCO2 is the data element containing the changed date format: DD/MM/YY or MM/DD/YY.
			A SF LIN OPE OPERAND LVTY CONDITION
			BB 100 AD ID DELCO1 DELCO2
			BATCH FUNCTION: the date processing function is generated in F9520. You may change this by coding, in an 'O'-type line of the Program's -GO, the DATPRO = ffss parameter, where ffss is the specified function-subfunction code.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'AD0'	Century positioned from DAT-CTY field initialized to '19' and it can be modified by the user.
		'AD1'	Century set to '19' if System year is less than the value in DAT-CTYT field (Default='61'),'20' otherwise
		'AD2'	Century set to '20' if System year is less than the value in DAT-CTYT field (Default='61'), '19' otherwise
			DATE COMPUTING
		'DAD'	Number of days between two dates. In the OPERANDS field, you must enter:
			xy ffnn-date1 ssnn-date2
			'x' being the format of 'date1' (optional). If you do not specify it, it will be extracted from the repository, provided the Data Element has a date format.
			'y' being the format of 'date2' (optional). If you do not specify it, it will be extracted from the repository, provided the Data Element has a date format.
			The comparison result is put in the NUM-DAYS field, which is automatically declared in the Working Section
		'DAO'	Days added to or subtracted from a date. In the OPERANDS field, you must enter:
			x snumber ffnn-date1
			'x' being the date format (optional). If you do not not specify any format, it will be extracted from the repository, provided the Data Elements have a date format.
			's' being the '+' sign for an addition or the '-' sign for a subtraction
			'number' being the number to add to or subtract from the date. It can be an integer or a work area.
			ffnn-date1 being the input and output date field
			To load the result into other dates, besides date1, just indicate these other dates (same format as date1) on a continuation line, without any operator.
			DATE PROCESSING OPERATORS (OLSD)
		'AD6'	Equivalent to ADT + ADI (See below).
			Transforms the system date into a 6-character date format of MMDDYY or DDMMYY depending on the value in the DATE FORMAT IN GENERATED PROGRAMS field on the Library Definition screen. The transformed date is moved into the Data Element indicated in the OPERAND field.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'AD8'	Transforms the system date into an 8-character date with slashes, MM/DD/YY or DD/MM/YY depending on the value in the DATE FORMAT IN GENERATED PROGRAMS field on the Library Definition screen. The transformed date is moved into the Data Element indicated in the OPERAND field.
			DATE PROCESSING OPERATORS (BATCH)
		'ADI'	Date inversion.
			The first two characters are replaced by the last two characters and vice-versa.
			Both OPERANDS must have a length of 6 characters.
			The second OPERAND is optional. If absent, the modified date will be moved back into the first OPERAND.
		'ADS'	Reversal of date with century.
			Both OPERANDS must have a length of 8 characters. The second OPERAND is optional. If absent, the modified date will be moved back into the first OPERAND.
		'ADE'	Insertion of slashes in a date.
			The first OPERAND must be the field which contains the original six-character date, and the second must contain an eight-character field which will receive the reformatted date.
		'ADM'	Insertion of slashes in a date with century.
			TIME PROCESSING OPERATORS
		'TIM'	Hour display in 'HHMMSS' format from the EIBTIME field in CICS; from the TIME field with other hardware.
			EXAMPLE:
			A SF LIN OPE OPERAND LVTY CONDITION
			BB 100 TIM DELCO1
		'TIF'	'HHMMSS' format changed into 'HH:MM:SS'.
			EXAMPLE:
			A SF LIN OPE OPERAND LVTY CONDITION
			BB 100 TIF DELCO1 DELCO2
			COMMUNICATION OPERATORS
			NOTE: Non operational with the OLSD Function.
			These OPERATORS ensure the liaison between a COBOL program and the 'Communication Units' in use:

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'ENA'	ENABLE Connecting the Unit.
		'DSB'	DISABLE Disconnecting the Unit.
		'RE'	RECEIVE Receiving a Message.
		'SD'	SEND Sending a Message.
			DBMS OPERATORS
		'B..'	CODASYL OPERATORS
			The Codasyl OPERATORS are coded with a 'B' as the first character, followed by the two-character codes defined below:
		'RY'	READY
		'FH'	FINISH
		'FD'	FIND
		'G '	GET
		'ER'	ERASE
		'DT'	DISCONNECT_____FROM_____
		'CT'	CONNECT_____TO_____
		'MD'	MODIFY
		'ST'	STORE
			TP MONITOR OPERATORS
			These OPERATORS cannot be used for the screen processing descriptions with the On-Line Systems Development function.
			CICS MACRO LEVEL OPERATORS
		'D..'	These OPERATORS are coded with a 'D' as the first character, followed by the specific two-character code of the CICS Macro.
		'XX'	DFHXX TYPE =
		'BM'	DFHBMS TYPE =
			All OPERATORS must be left justified.
			The continuation character in column 72 of the generated program and the justification of the continuation lines is automatically managed.
			CICS COMMAND LEVEL OPERATORS
		'EXC'	EXEC CICS operands END-EXEC.
			COBOL API and PAF/DAF OPERATORS

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'EXS'	Call of a Folder Manager. This OPERATOR is used to write a service request on a Folder.
			For more detailed information, refer to the 'COBOL API User's Guide'.
		'EXP'	This OPERATOR is used to activate a VA Pac Database access request via the Pacbase Access Facility (PAF). It generates PAF or DAF modules.
			EXEC PAF (...request...)
			END-EXEC.
			For more detailed information, refer to the 'PAF' or 'DAF' manuals (Pacbase / DSMS Access Facility').
			SQL OPERATORS
			SQL operators are documented in the manual dedicated to 'SQL Databases' in the Developer's Documentation, chapter "SQL Accesses".
		SCC	CONNECT order or its equivalent.
		SDC	DISCONNECT order or its equivalent.
		SCO	COMMIT order.
		SRO	ROLLBACK order.
		SWH	WHENEVER order.
			The SQL operators should be used with the following syntax:
			- SCC cccccc d
			- SDC cccccc d r
			- SCO cccccc d
			- SRO cccccc d
			- SWH instruction
			cccccc = VA Pac code of the Block (6 characters long).
			d = value 2 if distributed base (ex: Oracle Sybase).
			r = value R to select DISCONNECT order with ROLLBACK order.
			Rules:
			- d and r indicators can be reversed.
			- Each order can be completed on continuation lines (with no operator). You can specify, for example, the FORCE option in a COMMIT order for ORACLE.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			Generation:
			- On the Segment Calls (-CS or -CD), if you enter an SQL organization (ORGANIZATION field) and so a Block code (EXTERNAL NAME field), this organization has priority on the Block type indicated on the Block Definition.
			- If the block is indicated in the Segment Calls as being distributed, the orders linked to this block will be generated ″distributed″.
			- Unrecognized SQL orders are ignored.
			The END-EXEC is automatically generated and, with the batch generator, it is is always followed by a period.
		SQL	For batch only, the generated SQL order is standard and can be modified in the Segment's -GG screen.
			The customization of SQL accesses is documented in the 'SQL Databases' manual.
			In order to take these modifications into account, the Program -CD must contain a Block code in the EXTERNAL NAME field and an organization in the ORGANIZATION FIELD.
			The SQL operator should be used with the following syntax:
			OPE I OPERANDS
			SQL I FFSS FLSS SO PO (1st format)
			SQL I FFSS SO PO (2nd format)
			FFSS : file-segment code in the Program
			FLSS : file-segment code in the Library
			SO : type of the standard order (two characters) or specific order described in S FLSS GG.
			PO : particular order identifier in S FLSS GG.
			The 1st format should be used if the code of the program is different from the one of the library.
			The particular order code is optional, it modifies the description resulting of the standard order.
			The implemented SQL organizations are the following:
			C (Interel)
			M (Datacom)
			N (Nonstop)

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			O and P (Oracle)
			Q (SQL-DS ALLbase)
			2 (DB2)
			3 (SQL SERVER)
			4 (DB2/400)
			9 (Informix Ingres and SYBASE)
			H (General SQL organization)
			For a number of these organizations, the reference of the Block type is required (H and 9 for exemple).
			Restrictions:
			- the RDMS orders' syntax is not implemented ('R' organization)
			- The prefixing rule is not applied. The table name is not modified, the dot is removed if there is one.
			NOTE:
			In the case of Program-Macro and Macro-Macro lines with same indicators, information entered in the S FFSS GG screen are generated although not needed.
			Relational operator:
		EXQ	EXEC SQL operands END-EXEC.
10	32		OPERANDS
			This field contains the OPERANDS required by the preceding OPERATOR to complete the procedural instruction.
			The first OPERAND must be placed on the same line as the OPERATOR.
			When an OPERATOR calls for several OPERANDS, they must be indicated one after the other in a continuous sequence, separated by at least one blank.
			The Qualification of Names using 'OF' is acceptable if the 'OF' of the first OPERAND is on the same line as the OPERATOR.
			When the first OPERAND is an alphanumeric literal that does not completely fit on one line, the continuation of the literal can be indicated between quotes on the following line. The System ensures continuity for the literal at the generation level.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			STRUCTURE
			The structure of a function or sub-function is defined by a LEVEL NUMBER and a CONDITION TYPE OR S.F. STRUCTURE, with which a condition, variable or a value can be associated, if necessary.
			Structures with levels other than 99 must be defined on the first line of a (sub-)function, while conditioning can continue onto several lines.
			Default values for level and type are taken from the title line of the (sub-)function.
			Functions are the highest level (05) structures.
11	2		LEVEL NUMBER
			PARAMETERIZABLE NUMERIC FIELD
			The LEVEL NUMBER is indicated only on the first line of a Structure.
			The CONDITION FOR EXECUTION of a Structure at a given level applies to all the logically lower level Structures which follow the initial Structure, until the next logically higher level Structure is encountered.
		'05'	This level is always assigned to functions. It is also the default level of the first line of a function Structure.
		'10'	This is the default level of the first line of a sub-function Structure.
		06 to 98	Possible levels for a sub-function.
		'99'	Defines an elementary procedure in a function or sub-function. (Maximum number of '99' levels in a sub-function = 98).
		'$n'	In a Macro-Structure the LEVEL NUMBER can be parameterized.
12	2		CONDITION TYPE OR S.F. STRUCTURE
			On the first line of a function or sub-function, the CONDITION TYPE OR S.F. STRUCTURE value indicates the 'Structure type' processing to be executed.
			In a Macro-Structure the CONDITION TYPE OR S.F. STRUCTURE cannot be parameterized.
			A (sub-)function constitutes a block of processing or structure. It's also possible to define, within a (sub-)function, elementary Structures characterized by a '99' level.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'BL'	'Block' type Structure.
			Default value for all non-conditioned Structures.
		'IT'	'IF THEN' type structure.
			Default value for all conditioned Structures. Executed if the condition is satisfied.
		'EL'	'ELSE' type Structure.
			Prohibited at the function level.
			Executed if the preceding Structure at the same level (which must be an 'IF THEN') was not executed. An 'ELSE' Structure cannot have its own condition.
		'CO'	'CASE OF' type Structure.
			Prohibited at the '05' function level and at the '99' level.
			A 'CO' Structure is used for procedures that are exclusive of each other, and that are executed depending on the possible values of a variable.
			In a 'CASE OF' Structure only the name of the variable is defined (in the condition field and on a single line). The possible values of this variable are specified in the Structures at the next lower, non-elementary, hierarchical level. In a Dialog screen, nesting of 'CASE OF' loops is not authorized within another 'CASE OF' loop.
		'DW'	'DO WHILE' Structure.
			Prohibited at the '05' function level and at the '99' level.
			This Structure is executed repeatedly, as long as its condition is satisfied.
		'DU'	'DO UNTIL' Structure.
			Prohibited at the '05' function level and at the '99' level.
			This Structure is executed repeatedly until its condition is satisfied. Thus, it is executed at least one time.
			For the 'DW' and 'DU' type Structures, the user must set up the condition status (incrementation of an index, for example).
		DO	'DO' Structure ('loop' Structure).
			Cannot be used at an '05' function level or the '99' level.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			The 'DO' Structure is executed in a repetitive way depending upon the conditioning of three variables: the first variable being the iteration number where the processing should start; the second one, the iteration number where the processing should stop, the third being the incrementing interval.
			All three variables may be either numbers or Data Elements. The increment variable is optional and its default value is '1'. (If indicated, its value must be positive.) Both iteration variables must be entered on the first line of the sub-function, separated by a space.
			The parameters must be entered in the following order: starting bound (positive), ending bound and increment interval.
			DO calls for 3 parameters, of which the first 2 are required. All 3 must appear on the first line.
			The System automatically generates an index: JfusfR, 'fu' standing for the function code, and 'sf' standing for the sub-function code.
			To generate another index, you must enter, at the beginning of the CONDITION FOR EXECUTION field:
			I=index bound1 bound2 step
			'index' being a numeric work area
			'bound1' being the starting bound
			'bound2' being the ending bound
			'step' being the incrementing interval
		'OR'	Continuation of the condition associated with the preceding lines by a logical 'OR'.
		'AN'	Continuation of the condition associated with the preceding lines by a logical 'AND'.
			NOTE: The parentheses which group the terms of a condition must be indicated in the text of the condition.
			In a Macro-Structure, the type of structure or condition cannot be parameterized.
		'WH'	You can use COBOL commands after an EVA or SEA command. Each of these commands indicates one processing to be performed according to the fulfilled condition. In the example below, processing1 will be performed when condition1 is fulfilled.
			EVA condition

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			---processing1--- 99WH ---condition1---
			---processing2--- 99WH ---condition2---
			---processing3--- 99WH ---condition3---
		'DC'	The execution of the processing depends on the comparison between two dates.
			In the CONDITION FOR EXECUTION field, you must enter:
			xy ffnn-date1 oper ssnn-date2
			'x' being the format assigned to 'date1' (optional). If you do not specify it, it will be extracted from the repository, provided the Data Element has a date format.
			'y' being the format assigned to 'date2' (optional). If you do not specify it, it will be extracted from the repository, provided the Data Element has a date format.
			'oper' being the type of comparison. It can be >, <, >=, <=, NOT>, NOT<, NOT=.
			If this condition is entered on an 'ffss' sub-function level, the comparison test will be followed by the following generated line:
			NEXT SENTENCE ELSE GO TO Fffss-FN.
		'DV'	The processing is executed if the date is valid.
			In the CONDITION FOR EXECUTION field, you must enter:
			x ffnn-date1
			'x' being the date format (optional). If you do not specify it, it will be extracted from the repository, provided the Data Element has a date format.
			'ffnn-date1' being the date Data Element to be tested
			If this condition is entered on a 99 level, the following line is generated so that the processing is executed: IF DEL-ER='1'
			If this condition is entered on an 'ffss' sub-function level, the following lines are generated so that the processing is excuted:
			IF DEL-ER='1'
			NEXT SENTENCE ELSE GO TO Fffss-FN.
		'DI'	The processing is executed if the date is invalid.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			In the CONDITION FOR EXECUTION field, you must enter:
			x ffnn-date1
			'x' being the date format (optional). If you do not specify it, it will be extracted from the repository, provided the Data Element has a date format.
			'ffnn-date1' being the date Data element to be tested
			If this condition is entered on a 99 level, the following line is generated so that the processing is executed: IF DEL-ER>'1
			If this condition is entered on an 'ffss' sub-function level, the following lines are generated so that the processing is executed:
			IF DEL-ER>'1'
			NEXT SENTENCE ELSE GO TO Fffss-FN.
			ON-LINE SYSTEMS DEVELOPMENT
			The following values, in the CONDITION TYPE OR S.F. STRUCTURE field, are used for relative positioning with the On-Line Systems Development function and Pacbench C/S.
		'*A'	Insertion of a sub-function before an automatic sub-function identified by the data element or the Segment it processes.
		'*P'	Insertion of a sub-function after an automatic sub-function identified by the Data Element or the Segment it processes.
			(The CONDITION FOR EXECUTION of the automatic sub-function applies to the inserted sub-function if the LEVEL NUMBER of the inserted sub-function is greater than that of the automatic sub-function.)
		'*R'	Replacement of an automatic sub-function identified by the Data Element or the Segment it processes. (The CONDITION FOR EXECUTION of the automatic sub-function does not apply to the replaced sub-function.)
			For more information, see chapter 'Use of Structured code', subchapter 'Specific Procedures' in the 'On-Line Systems' manual.
			The two following values are used for the relative positioning with Pacbench C/S (server components only):
		'*C'	Insertion or replacement of the processing on the server or the Logical View.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'*B'	Insertion in the elementary processing called by PERFORM.
			For more information, see Manual 'Business Logic and TUI clients', chapter 'Business Component', subchapter 'Writing Procedural Code'.
13	28		CONDITION FOR EXECUTION
			The CONDITION FOR EXECUTION statement is coded without using 'IF', 'AND', 'OR', 'GO TO', or a period (.).
			The first line must be associated with an explicit or implicit LEVEL NUMBER ('05' for a function; '06' to '98' for a sub-function; '99' for an elementary procedure within a function or sub-function). The LEVEL NUMBER need not be specified on continuation lines.
			For a 'CASE OF' type structure, the name of the variable (which may have alternative values) must be indicated on the first line.
			For structures dependent on a 'CASE OF', the CONDITION FOR EXECUTION indicates the possible value of the 'CO' variable.
		'$n'	In a Macro-Structure, the CONDITION FOR EXECUTION can be parameterized.

Programmer Flags and Variables

The purpose of this subchapter is to present the programmer with a list of flags, variables, counters and indices generated by the System and which are commonly used in Procedural Code.

The subchapter is divided into two parts, the first pertaining to batch Programs, the second to on-line Programs.

The chart below lists only those variables recommended for use by programmers.

BATCH PROGRAM VARIABLES

These symbols are used:

```
cc       = (report) CATEGORY CODE
dd       = DATA STRUCTURE CODE
eeeeee   = DATA ELEMENT CODE
r        = LAST CHARACTER OF REPORT CODE
```

```
ss       = SEGMENT CODE
st       = (report) STRUCTURE NUMBER
```

DATA NAME	FORMAT	MOST COMMON USAGE
ddss-eeeeee		
IK	X	file access error ind.:'1'= error
DATCE	X(8)	(group) dates with century
CENTUR	XX	century
DATOR	X(6)	(group)
DATOA	XX	year
DATOM	XX	month
DATOJ	XX	day
DAT6	X(6)	(group)
DAT61	XX	
DAT62	XX	
DAT63	XX	
DAT8	X(8)	(group) dates with slashes
DAT81	XX	(no century)
DAT8S1	X	slash
DAT82	XX	
DAT8S2	X	slash
DAT83	XX	
DAT8E		(redefines DAT8)
DAT81E	X(4)	century/year
DAT82E	XX	
DAT83E	XX	
DAT6C	X(8)	(group)
DAT61C	XX	
DAT62C	XX	
DAT63C	X(4)	century/year
DAT8C	X(10)	(group) dates with slashes and
DAT81C	XX	century
DAT82C	XX	(slashes via filler)
DAT83C	X(4)	century/year
FTBn	X	final total break '1'= yes

DATA NAME	FORMAT	MOST COMMON USAGE
FBL	9	current final brk lev: rarely used
IBL	9	current init. brk lev: rarely used
ITBn	X	initial total break '1'= yes
dd-FBn	X	final break indicator: '1'= yes
dd-IBn	X	initial break indicator: '1'= yes
dd-CFn	X	record to be processed: '1'= yes
dd-OCn	X	write file: '1' or delete it: '0'
FT		= ALL '1'
dd-FT	X	EOF indicator: '1'= yes
dd-FI	X	ctl brk: last rec. indic. '1'= yes
I01	S9(4)	which rec. type is being processed
I02	S9(4)	which action is being processed
I03	S9(4)	stores a pointer (rank) to the first element of the specific part Segment being processed
I04	S9(4)	stores a pointer to the last Data Element of the specific part Segment being processed only in
I06	S9(4)	in functions not autom. generated
I50	S9(4)	stores the rank of the first Data Element of the common part
I51	S9(4)	stores the number of record types
IddssL	S9(4)	subscript for loading tables
IddssR	S9(4)	subscript for searching tables
IddssM	S9(4)	maximum positions in a table
J00	S9(4)	looks-up for the category table
J01	S9(4)	looks-up for the 3-D table
J05, J06, J07	S9(4)	accumulators
Jddrcc	S9(4)	report repetitive category
JddrccM	S9(4)	report repetitive category
IND	S9(4)	stores the major-most key level of input data structures to be matched
ddIND	S9(4)	stores the current value off the key of the record on data structure dd
5-dd00-RECCNT	S(9)9	record counter
ID-ER	X	error rec. type or action; '0'= ok
ER-ss-eeeeee	X	0-5: (in)valid pres/abs/cont/class

DATA NAME	FORMAT	MOST COMMON USAGE
DEL-ER	X	performed validations
ER-PR(n)		
TR-ER	X	any errors on transaction: '1'= no
SEG-ER	X	
GR-ER	X	errors on group of transactions to
LE-FIENR	X(4)	hold the VA Pac code of the Segment currently being processed
UT-ERUT		errors detected by user valid.'s
UT-UPR(n)	X	user error messages
Trst-eeeeee		totaling
Grst-eeeeee		grand totals
5-dd00-rLCM	S999	report line count maximum
5-dd00-rLC	S999	report line count
5-dd00-rPC	S9(7)	page count
LSKP	99	line skips (user spooling)
NUPOL	X	laser printer
CATX	XX	category of report being printed
6-dd00		
6-dd00-r		rarely used
1-ddss		read area - control break files
1-ddss-eeeeee		
2-ddss		update area
2-ddss-eeeeee		
1-dd-TABLE		USAGE OF D.S. = 'T' (Table)
1-ddssT		(group)
1-ddss(n)		
1-ddss-eeeeee(n)		For instance 1-ddss-eeeeee(IddssR)

ON-LINE PROGRAM VARIABLES

These symbols are used:

```
dd              = DATA STRUCTURE CODE
ss              = SEGMENT CODE
eeeeee          = DATA ELEMENT CODE
scrn            = SCREEN CODE
```

DATA NAME	FORMAT	MOST COMMON USAGE
IK	X	File access error indicator '0' = successful access;'1'= error (set after every file access)
OPER	X	Internal Operation Code Useful to specify conditions for Update, Scrolling, etc. (e.g. F06)
OPERD	X	Deferred Operation Code; used by OSD operator
CATX	X	Identifies category being executed Useful to condition logic in loops (e.g. F21, F31, F66, etc.)
CATM	X	Internal Transaction code governs file access for update Useful when update rules are complex (e.g. F16)
ICATR	99	Number of line item currently being processed
SCR-ER	X	Error on screen: '1' - no error; '4' - error
FT	X	'End-of-file' on read for display in repetitive category (either control break or true eof) Useful to condition structured code for Repetitive display (e.g., CATX = 'R' and FT = '0' and ICATR not > IRR is the best way to condition line item display logic in F66)
ICF	X	Governs execution of Reception Functions (F05 through F40); '0' in ICF generally implies first time processing for screen
OCF	X	Governs execution of Display Functions (F50 through F8Z)
SESSI	X(5)	Session number of generated prog.
LIBRA	X(3)	Library code
USERCO	X(8)	User code
DATGN	X(8)	Date of generated program (MM/DD/YY if if user language = E, or DD/MM/YY otherwise)
PROGR	X(8)	Program code in Library
TIMGN	X(8)	Time of generated program
COBASE	X(4)	Database code
DATGNC	X(10)	Date of generated program with century (MM/DD/CCYY if user language = E, or DD/MM/CCYY otherwise)
CAT-ER	X	Error in current category: ' ' - no error; 'E' - error Useful for conditioning Reception Functions after edits (e.g., F21, F31, F36, etc.)
CURPOS		Cursor pos. on screen in reception
CPOSL	S9(4)	Line number of cursor Useful for setting transfers on cursor positioning (e.g., F06)

DATA NAME	FORMAT	MOST COMMON USAGE
CPOSC	S9(4)	Column number of cursor
CPOSN	S9(4)	Absolute position of cursor in msg
IRR	99	No. of reps in the rep category
INT	999	No. of input fields in screen
IER	99	No. of screen-related error msg
DEL-ER	X	Error in current data element '1' - no error; < '1' - error
DATCE	X(8)	(Group) Dates with century
CENTUR	XX	(century)
DATOR	X(6)	(group)
DATOA	XX	year
DATOM	XX	month
DATOJ	XX	day
DATSEP	X	Separator used in date
DAT6	X(6)	(group) - fields for formatting
DAT61		the date - generated if the
DAT619	99	OPERATOR 'AD_' is used, or if
DAT62		a variable data element has a
DAT629	99	date format.
DAT63	XX	
DAT7		
DAT71	XX	
DAT72	XX	
DAT73	XX	
DAT8		
DAT81	XX	
DAT8S1	X	slash
DAT82	X	
DAT8S2	X	slash
DAT83	XX	
DATCTY	XX	field for loading the century
DAT6C	X(8)	fields for loading the non-for-
DAT61C	XX	matted date with the century.
DAT62C	XX	

DATA NAME	FORMAT	MOST COMMON USAGE
DAT63C	XX	
DAT64C	XX	
DAT7C		
DAT71C	XX	
DAT72C	XX	
DAT73C	XX	
DAT74C	XX	
DAT8C		
DAT81C	XX	
DAT8S1C	X	slash
DAT82C	XX	(slashes via filler)
DAT8S2C	X	slash
DAT83C	XX	
DAT84C	XX	
DAT8G	X(8)	Gregorian format date: CCYY/MM/DD
TIMCO		TIME; field for loading the time
TIMCOG		
TIMCOH	XX	
TIMCOM	XX	
TIMCOS	XX	
TIMCOC	XX	
TIMDAY		field for loading formatted time
TIMHOU	XX	
TIMS1	X	
TIMMIN	XX	
TIMS2	X	
TIMSEC	XX	
TIMCIC	9(7)	

DATA NAME	FORMAT	MOST COMMON USAGE
TIMCI1 TIMCIG TIMCIH TIMCIM TIMCIS DATCIC DATQTM DATQUD DATQUY	XX XX XX 9(7) 999 99	REDEFINES TIMCIC REDEFINES DATCIC

DATA NAME	FORMAT	MOST COMMON USAGE
ddss-CF	X	Segment configuration flag 0 - record not found; 1 - successful read. To check the status of a read, ddss-CF is preferred to IK (e.g., F25, F26, F60, F61, F66)
G-ddss		Pactables table description
G-ddss-eeeeee		Pactables fields
K-scrn		Fields common to dialogue
K-Ascrn-eeeeee		-screen top fields
K-Rscrn-eeeeee		-repetitive group fields
K-Rscrn-LINE(1)		Key of the first line item kept in common area for scrolling
K-Rscrn-LINE(2		Key of the record following the last line item on the screen kept in common area for scrolling
K-Zscrn-eeeeee		-screen bottom fields
T-scrn-eeeeee		("OFF" option only) With modify data tags off, a copy of each input field is stored in the common area. F8135 ensures consistency with the input map.
I-scrn		Group field for input fields
I-scrn-eeeeee		Input field label useful for Validation Transfer (e.g., F21-F24, F31)
E-scrn-eeeeee	alpha	Input field label for numeric fields after formatting
J-scrn-LINE		Group input field containing entire Repetitive group An occurrence of this field can be moved into I-scrn-LINE for manual processing of a specific line item.
Z-scrn-eeeeee		reception - attributes
O-scrn		group field for output fields
X-scrn-eeeeee		Cursor positioning field. This field is set in F70; any override must be made later (e.g., F71)

DATA NAME	FORMAT	MOST COMMON USAGE
Y-scrn-eeeeee		Attribute byte. This field is set in F70; any override must be made later (e.g., F71). Manually setting attributes may also be accomplished by setting the field A-scrn-eeeeee in Display Pre- paration (e.g., F68 or F69) and letting F70 fill Y-scrn-eeeeee (which is the actual attr. byte)
O-scrn-eeeeee		Output field label Useful for Transfer for Display (e.g., F66)
F-scrn-eeeeee		to test for class (numeric)
9-scrn-eeeeee		numeric fields
DE-ERR		validation table fields
DE-ER (n)	X	
ER-scrn-eeeeee		Data element error code variable used by the ERR operator; for details, see OLSD ref. manual, Documentary and Error Messages, sub-chapter Manual Explicit Error
ddss-FST	X	non-chained segs:'1'= first access '0'= next read
5-ddss-LTH	S9(4)	Segment length
5-dd00-LTH	S9(4)	Length of longest segment in file
I-PFKEY	XX	Attention Identifier variable
A-scrn-eeeeee(1)		For attribute byte processing before F70 (e.g., F68 or F69), move an N, B, or D to this field for normal, bright, or dark
A-scrn-eeeeee(4)		For cursor position processing before F70 (e.g., F68 or F69) move a Y to this field to set the cursor to the field manually
PROGR	X(8)	Program code
K-Sscrn-PROGR	X(8)	External name of the program we just came from. Can be used to check first time processing (e.g., K-Sscrn-PROGR = PROGR is used before F01; but after F01, ICF = 0 is the simpler check)
5-Sscrn-PROGR	X(8)	This field contains the name of the program to branch to
PROGE	X(8)	External name of the program
PRDOC	X(8)	Help program external name

Titles and Conditions Screen (-TC)

Programmers who begin using the System may have some difficulty mastering its generation possibilities.

It is not always easy to know beforehand which procedures of a batch or on-line Program will be generated.

Here, programming is done in two steps:

1. Call of automatically generated procedures,
2. Customizing the generated Program with structured code.

The first step is performed through data access in batch or on-line Programs:

1. Program Call of Data Structures screen (CH: P.....CD),
2. On-Line Call of Segments screen (CH: O.....CS),
3. On-Line Call of Data Elements screen (CH: O.....CE).

Depending on the kind of data that is entered on the call lines of these screens, the system either will or will not generate certain automatic (sub-)functions.

EXAMPLES:

1. In a batch program, 'M' entered in the USAGE OF DATA STRUCTURES field on a Call of Data Structures (-CD) screen causes the generation of validation functions.

2. In an on-line program, 'E' entered in the USE IN RECEPTION field on an On-Line Call of Segments (-CS) screen will generate the Segment access for validation with the setting of an error code.

The second step in programming is characterized by the use of the Structured Code function which allows you to complete automatically generated procedures with additional lines unique to the Program:

1. Call of P.M.S.'s screens for user-standard procedures.
   CH: P.....CP, O.....CP.
2. Direct input of Procedural Code
   CH: P.....P, O.....P.

A Program is then made up of automatically generated procedures and structured code.

GENERAL INFORMATION

Titles and Conditions (-TC) screen lines display the titles and conditions of all procedures of a batch or on-line Program, whether automatically generated or specified with procedural code.

Using this screen, the user can examine the general structure of a generated program and detect possible errors related to the uses of the Program's different parts.

ACCESSING THE TITLES AND CONDITIONS SCREEN

For batch Programs, enter the following in the CHOICE field:

CH: PppppppTCfusf<nn or Ppppppp<nnTCfusf

For on-line Programs, enter the following in the CHOICE field:

CH: OooooooTCfusf<nn or Ooooooo<nnTCfusf

```
where:
              pppppp = PROGRAM CODE
              oooooo = SCREEN CODE
              fu     = FUNCTION CODE (default:'  ')
              sf     = SUB-FUNCTION CODE (default:'  ')
              nn     = LEVEL NUMBER (default: 05)
```

EXAMPLE:

In order to obtain the complete list of titles and conditions for Program PGM001, the following should be entered in the CHOICE field:

CH: P PGM001 TC

In order to focus on a specific part of the Program, for instance starting with function F29, sub-function BB, and view the titles and conditions listed to the 15 level inclusive, the following should be entered in the CHOICE field:

CH: P PGM001 TC29BB<15 or CH: P PGM001 <15TC29BB

ORIGIN OF GENERATED LINES

The lines displayed on this screen originate from the following three sources:

1. VisualAge Pacbase automatic generation:

   Displayed lines come from the Program Call of Data Structures (-CD) screen for batch Programs, or On-Line Call of Segments (-CS) screen and On-Line Call of Data Elements (-CE) screen for on-line Programs, both having been previously entered by the user.

   They are identified by a period ('.') in the ACTION CODE field of each line.

2. Macro-structure calls:

Displayed lines come from the Call of P.M.S.'s (-CP) lines (of Programs and Screens).

They are identified by an asterisk ('*') in the ACTION CODE field of each line.

3. Procedural Code (-P) lines attached directly to the Program or to the Screen:

   These lines are identified by a 'blank' in the ACTION CODE field for each line.

UPDATE POSSIBILITIES

The Titles and Conditions (-TC) screen displays functions or sub-functions titles with their conditions for execution. Therefore, updating affects a whole function or sub-function.

- FOR GENERATED PROCEDURES AND MACRO-STRUCTURES :

  The only updating possibility is the suppression of a function or sub-function generation. This is done with the 'S' OPERATOR (counterpart to the 'SUP' OPERATOR in Procedural Code (-P) lines).

  These procedures are identified with a ('.') or an ('*') in the ACTION CODE field. This code must be deleted if the update is to be taken into account.

- SPECIFIC PROCEDURES :

  The user may create, modify or delete the title of a function or sub-function written in procedural code.

  The user may also modify the LEVEL NUMBER, CONDITION TYPE OR S.F. STRUCTURE and CONDITION FOR EXECUTION of a function or sub-function. The corresponding ACTION CODES are : 'C', 'M', 'D' or blank.

  Each update is automatically channeled down to the corresponding Procedural Code (-P) screen.

RETURN TO A DISPLAYED (SUB-)FUNCTION

The user may return to the Procedural Code (-P) screen corresponding to a displayed line. In order to do this, the user places the cursor on the desired line and presses the relevant PFkey (standard: PF10).

The user can also branch to the '-PG' screen from the line where the cursor is positioned by using the appropriate PFkey (standard: PF9).

Finally, the user can request that the same screen be displayed from the line where the cursor is positioned by using the appropriate PFkey (standard: PF8, except under IMS).

UPDATE OPERATORS ALLOWED IN THIS SCREEN

```
N      Note (title).
S      Suppression (equivalent to 'SUP').
Blank  Continuation of conditioning of a (sub-)function.
```

Any other operator will be refused.

PREREQUISITE

The Program or Screen must have been previously defined.

NOTE TO ON-LINE SYSTEMS DEVELOPMENT FUNCTION USERS

Differences may appear between the Titles and Conditions (-TC) screen lines display and the actual generated Program:

FUNCTION F80

On the Titles and Conditions (-TC) screen, the sequence order of the sub-functions depends on the SEGMENT CODE IN THE PROGRAM, whereas in the generated Program, sub-functions are ordered according to the SEGMENT CODE IN THE LIBRARY. For a Segment called in on a Call of Segments (-CS) screen, which does not have a 'U'-type ORGANIZATION and is not used in display or reception, display is simulated on the Titles and Conditions (-TC) screen.

FUNCTION F81

Numeric and date validations are always considered to be generated, therefore sub-functions F8110 and F8120 are always displayed on the Titles and Conditions (-TC) screens. However, Sub-function F8110 is generated only if unprotected numeric Data Elements are called. Sub-function F8120 is generated only if unprotected date-type Data Elements are called.

SPECIAL PFKEYS

PF8:

Reset screen display starting from the line where the cursor is positioned (not available with the IMS version).

PF9:

From the Titles and Conditions (-TC) screen, branch to the Procedures Generated (-PG) screen and vice-versa, starting from the line where the cursor is positioned.

THE 'TITLES ONLY (-TO)' SCREEN

The Preview Facility includes the Titles Only (-TO) screen. This screen displays the list of program function titles only and illustrates their hierarchical organization.

The screen is accessed in the same manner as the Titles and Conditions (-TC) screen (substitute 'TO' for 'TC'); see Paragraph "ACCESSING THE TITLES AND CONDITIONS SCREEN" above.

The advantage of this screen is that the level numbers of the program functions are indented, showing the user a different view from the Titles and Conditions (-TC) screen. However, this screen cannot be used for updates.

(The Titles Only (-TO) screen image can be found at the end of this sub-chapter).

```
                  PURCHASING MANAGEMENT SYSTEM        SG000008.LILI.CIV.1583

 TITLES AND CONDITIONS BBINIT GENERAL PROCESSING
                       1

 2 3 4  5   6 7                                  8 9  10
 A FUSF LIN O OPERANDS                           LVTY CONDITION
 . 05       N READ SEQ.FILES NO CONTROL BREAK    05BL
 . 20       N END OF RUN                         05IT FT = ALL '1'

 *** END ***
 O: C1 CH: -TC
```

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	6		PROGRAM CODE OR SCREEN CODE
			This field contains the six-character program or on- line screen code.
2	1		ACTION CODE
		'C'	Creation of the line
		'M'	Modification of the line
		'D' or 'A'	Deletion of the line
		'T'	Transfer of the line
		'B'	Beginning of multiple deletion
		'G'	Multiple transfer
		'?'	Request for HELP documentation
		'E' or '-'	Inhibit implicit update
		'X'	Implicit update without upper/lowercase processing
3	2		FUNCTION CODE

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'AA to 99'	This code determines the placement of the Procedural Code lines in the sequence of functions. This is particularly important when used with the On-Line and Batch Systems Development functions in which automatic functions have pre-determined codes.
		' $n'	In a Macro-Structure, the FUNCTION CODE can be parameterized.
4	2		SUB-FUNCTION CODE
			Made up of numeric or alphabetic characters.
			This code determines the placement of the Procedural Code within the function.
		' $n'	In a macro-structure, the SUB-FUNCTION CODE can be parameterized.
5	3		LINE NUMBER
			PARAMETERIZABLE NUMERIC FIELD
		'0-999'	As a recommendation, number the lines starting with 10 by intervals of 10, thus facilitating future insertion insertions.
		$n0 to $n9	In a Macro-Structure, only the first two characters of the LINE NUMBER can be parameterized.
6	1		OPERATOR
			The following OPERATORS are the only ones allowed on this screen:
		'N'	Title of the function or the sub-function.
		'S'	Equivalent of 'SUP' in the Procedural Code (-P) lines.
			Used to suppress a function or sub-function.
		'blank'	Continuation lines of the CONDITION FOR EXECUTION.
7	32		OPERANDS
			The title of the corresponding function or sub-function is displayed in this field.
8	2		LEVEL NUMBER
			PARAMETERIZABLE NUMERIC FIELD
			The LEVEL NUMBER is indicated only on the first line of a Structure.
			The CONDITION FOR EXECUTION of a Structure at a given level applies to all the logically lower level Structures which follow the initial Structure, until the next logically higher level Structure is encountered.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'05'	This level is always assigned to functions. It is also the default level of the first line of a function Structure.
		'10'	This is the default level of the first line of a sub-function Structure.
		06 to 98	Possible levels for a sub-function.
		'99'	Defines an elementary procedure in a function or sub-function. (Maximum number of '99' levels in a sub-function = 98).
		'$n'	In a Macro-Structure the LEVEL NUMBER can be parameterized.
9	2		CONDITION TYPE OR S.F. STRUCTURE
			On the first line of a function or sub-function, the CONDITION TYPE OR S.F. STRUCTURE value indicates the 'Structure type' processing to be executed.
			In a Macro-Structure the CONDITION TYPE OR S.F. STRUCTURE cannot be parameterized.
			A (sub-)function constitutes a block of processing or structure. It's also possible to define, within a (sub-)function, elementary Structures characterized by a '99' level.
		'BL'	'Block' type Structure.
			Default value for all non-conditioned Structures.
		'IT'	'IF THEN' type structure.
			Default value for all conditioned Structures. Executed if the condition is satisfied.
		'EL'	'ELSE' type Structure.
			Prohibited at the function level.
			Executed if the preceding Structure at the same level (which must be an 'IF THEN') was not executed. An 'ELSE' Structure cannot have its own condition.
		'CO'	'CASE OF' type Structure.
			Prohibited at the '05' function level and at the '99' level.
			A 'CO' Structure is used for procedures that are exclusive of each other, and that are executed depending on the possible values of a variable.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			In a 'CASE OF' Structure only the name of the variable is defined (in the condition field and on a single line). The possible values of this variable are specified in the Structures at the next lower, non-elementary, hierarchical level. In a Dialog screen, nesting of 'CASE OF' loops is not authorized within another 'CASE OF' loop.
		'DW'	'DO WHILE' Structure.
			Prohibited at the '05' function level and at the '99' level.
			This Structure is executed repeatedly, as long as its condition is satisfied.
		'DU'	'DO UNTIL' Structure.
			Prohibited at the '05' function level and at the '99' level.
			This Structure is executed repeatedly until its condition is satisfied. Thus, it is executed at least one time.
			For the 'DW' and 'DU' type Structures, the user must set up the condition status (incrementation of an index, for example).
		DO	'DO' Structure ('loop' Structure).
			Cannot be used at an '05' function level or the '99' level.
			The 'DO' Structure is executed in a repetitive way depending upon the conditioning of three variables: the first variable being the iteration number where the processing should start; the second one, the iteration number where the processing should stop, the third being the incrementing interval.
			All three variables may be either numbers or Data Elements. The increment variable is optional and its default value is '1'. (If indicated, its value must be positive.) Both iteration variables must be entered on the first line of the sub-function, separated by a space.
			The parameters must be entered in the following order: starting bound (positive), ending bound and increment interval.
			DO calls for 3 parameters, of which the first 2 are required. All 3 must appear on the first line.
			The System automatically generates an index: JfusfR, 'fu' standing for the function code, and 'sf' standing for the sub-function code.
			To generate another index, you must enter, at the beginning of the CONDITION FOR EXECUTION field:

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			I=index bound1 bound2 step
			'index' being a numeric work area
			'bound1' being the starting bound
			'bound2' being the ending bound
			'step' being the incrementing interval
		'OR'	Continuation of the condition associated with the preceding lines by a logical 'OR'.
		'AN'	Continuation of the condition associated with the preceding lines by a logical 'AND'.
			NOTE: The parentheses which group the terms of a condition must be indicated in the text of the condition.
			In a Macro-Structure, the type of structure or condition cannot be parameterized.
		'WH'	You can use COBOL commands after an EVA or SEA command. Each of these commands indicates one processing to be performed according to the fulfilled condition. In the example below, processing1 will be performed when condition1 is fulfilled.
			EVA condition
			---processing1--- 99WH ---condition1---
			---processing2--- 99WH ---condition2---
			---processing3--- 99WH ---condition3---
		'DC'	The execution of the processing depends on the comparison between two dates.
			In the CONDITION FOR EXECUTION field, you must enter:
			xy ffnn-date1 oper ssnn-date2
			'x' being the format assigned to 'date1' (optional). If you do not specify it, it will be extracted from the repository, provided the Data Element has a date format.
			'y' being the format assigned to 'date2' (optional). If you do not specify it, it will be extracted from the repository, provided the Data Element has a date format.
			'oper' being the type of comparison. It can be >, <, >=, <=, NOT>, NOT<, NOT=.
			If this condition is entered on an 'ffss' sub-function level, the comparison test will be followed by the following generated line:

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
			NEXT SENTENCE ELSE GO TO Fffss-FN.
		'DV'	The processing is executed if the date is valid.
			In the CONDITION FOR EXECUTION field, you must enter:
			x ffnn-date1
			'x' being the date format (optional). If you do not specify it, it will be extracted from the repository, provided the Data Element has a date format.
			'ffnn-date1' being the date Data Element to be tested
			If this condition is entered on a 99 level, the following line is generated so that the processing is executed: IF DEL-ER='1'
			If this condition is entered on an 'ffss' sub-function level, the following lines are generated so that the processing is excuted:
			IF DEL-ER='1'
			NEXT SENTENCE ELSE GO TO Fffss-FN.
		'DI'	The processing is executed if the date is invalid.
			In the CONDITION FOR EXECUTION field, you must enter:
			x ffnn-date1
			'x' being the date format (optional). If you do not specify it, it will be extracted from the repository, provided the Data Element has a date format.
			'ffnn-date1' being the date Data element to be tested
			If this condition is entered on a 99 level, the following line is generated so that the processing is executed: IF DEL-ER>'1
			If this condition is entered on an 'ffss' sub-function level, the following lines are generated so that the processing is executed:
			IF DEL-ER>'1'
			NEXT SENTENCE ELSE GO TO Fffss-FN.
			ON-LINE SYSTEMS DEVELOPMENT
			The following values, in the CONDITION TYPE OR S.F. STRUCTURE field, are used for relative positioning with the On-Line Systems Development function and Pacbench C/S.

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'*A'	Insertion of a sub-function before an automatic sub-function identified by the data element or the Segment it processes.
		'*P'	Insertion of a sub-function after an automatic sub-function identified by the Data Element or the Segment it processes.
			(The CONDITION FOR EXECUTION of the automatic sub-function applies to the inserted sub-function if the LEVEL NUMBER of the inserted sub-function is greater than that of the automatic sub-function.)
		'*R'	Replacement of an automatic sub-function identified by the Data Element or the Segment it processes. (The CONDITION FOR EXECUTION of the automatic sub-function does not apply to the replaced sub-function.)
			For more information, see chapter 'Use of Structured code', subchapter 'Specific Procedures' in the 'On-Line Systems' manual.
			The two following values are used for the relative positioning with Pacbench C/S (server components only):
		'*C'	Insertion or replacement of the processing on the server or the Logical View.
		'*B'	Insertion in the elementary processing called by PERFORM.
			For more information, see Manual 'Business Logic and TUI clients', chapter 'Business Component', subchapter 'Writing Procedural Code'.
10	28		CONDITION FOR EXECUTION
			The CONDITION FOR EXECUTION statement is coded without using 'IF', 'AND', 'OR', 'GO TO', or a period (.).
			The first line must be associated with an explicit or implicit LEVEL NUMBER ('05' for a function; '06' to '98' for a sub-function; '99' for an elementary procedure within a function or sub-function). The LEVEL NUMBER need not be specified on continuation lines.
			For a 'CASE OF' type structure, the name of the variable (which may have alternative values) must be indicated on the first line.
			For structures dependent on a 'CASE OF', the CONDITION FOR EXECUTION indicates the possible value of the 'CO' variable.
		'$n'	In a Macro-Structure, the CONDITION FOR EXECUTION can be parameterized.

```
                  PURCHASING MANAGEMENT SYSTEM         SG000008.LILI.CIV.1583

      TITLES ONLY      BBINIT GENERAL PROCESSING
. FUSF . TITLE                          . LEVEL           . TY .SOURCE.LIBR
.....................................................................................
. 05   . READ SEQ.FILES NO CONTROL BREAK . 05              . BL .      .
. 20   . END OF RUN                      . 05              . IT .      .
.      .                                 .                 .    .      .
.      .                                 .                 .    .      .
.      .                                 .                 .    .      .
.      .                                 .                 .    .      .
.      .                                 .                 .    .      .
.      .                                 .                 .    .      .
.      .                                 .                 .    .      .
.      .                                 .                 .    .      .
.      .                                 .                 .    .      .
.      .                                 .                 .    .      .
.      .                                 .                 .    .      .
.      .                                 .                 .    .      .
.      .                                 .                 .    .      .
.      .                                 .                 .    .      .
.      .                                 .                 .    .      .
*** END ***
0: C1 CH: -TO
```

Chapter 6. Access Commands

On-Line Access Commands

LIST OF PROGRAMS

CHOICE	SCREEN	UPD
LCPaaaaaa	List of Programs by code (starting with Program 'aaaaaa').	NO
LNPaaaaaa	List of Programs by name (starting with program 'aaaaaa'). (case sensitive).	NO
LTPnPaaaaaa	List of Programs of type 'n' (starting with program 'aaaaaa').	NO
LEPeeeeeeee	List of Programs by external name (starting with external name 'eeeeeeee').	NO

DESCRIPTION OF PROGRAM 'aaaaaa'

CHOICE	SCREEN	UPD
Paaaaaa	Definition of Program 'aaaaaa'.	YES
PaaaaaaGCbbb	Comments for Program 'aaaaaa' (starting with line 'bbb').	YES
PaaaaaaGObbb	Generation option of Program 'aaaaaa' (starting with line 'bbb').	YES
PaaaaaaXVbbbbbb	X-references of Program 'aaaaaa' to Documents (starting with Document 'bbbbbb').	NO
PaaaaaaATbbbbbb	Text assigned to Program 'aaaaaa' (starting with text 'bbbbbb').	NO
PaaaaaaX	X-references of Program 'aaaaaa'.	NO
PaaaaaaXPbbbbbb	X-references of Program 'aaaaaa' to programs (starting with Program 'bbbbbb')	NO
PaaaaaaXObbbbbb	X-references of Program 'aaaaaa' to screens (starting with Screen 'bbbbbb').	NO
PaaaaaaXQrrrrrr	List of occurrences linked to Program 'aaaaaa' through User Relationship 'rrrrrr'.	NO
PaaaaaaCR	Occurrences linked to Program 'aaaaaa' through User Relationship	YES
PaaaaaaCDbb	Call of Data Structures of Program 'aaaaaa' (starting with Data Structure 'bb').	YES

PaaaaaaCPbbbbbb	Call of Parameterized Macro-Structure of Program 'aaaaaa' (starting with P.M.S. 'bbbbbb').	YES
PaaaaaaBbbccddd	Beginning Insertions Modifications of Program 'aaaaaa' (starting with section 'bb', paragraph 'cc', line 'ddd').	YES
PaaaaaaWbbccc	Description of Work Areas of Program 'aaaaaa' (starting with Work Area 'bb' line 'ccc').	YES
PaaaaaaPfusfnnn	Description of Procedural Code of Program 'aaaaaa' (starting with function 'fu', sub-function 'sf', line number 'nnn').	YES
PaaaaaaPGfusfnnn	View of Procedures Generated of Program 'aaaaaa' (starting with function 'fu', sub-function 'sf', line number 'nnn'), with display of generated procedure titles.	YES
Paaaaaa9bbbbbb	Description of Pure COBOL Source Code of Program 'aaaaaa' (starting with -9 line 'bbbbbb').	YES
PaaaaaaTCfusf	View of Titles and Conditions of automatic and specific procedures of Program 'aaaaaa' (starting with function 'fu', sub-function 'sf').	YES
PaaaaaaTCfusf<nn or Paaaaaa<nnTCfusf	View of Titles and Conditions of automatic and specific procedures of Program 'aaaaaa' up to level 'nn' (starting with function 'fu', sub-function 'sf').	YES
PaaaaaaTOfusf	View of Titles Only of automatic and specific procedures of Program 'aaaaaa' (starting with function 'fu', sub-function 'sf').	NO
PaaaaaaTOfusf<nn or Paaaaaa<nnTOfusf	View of Titles Only of automatic and specific procedures of Program 'aaaaaa' up to level 'nn' (starting with function 'fu', sub-function 'sf').	NO
PaaaaaaSCfusfnnn	Description of Source Code of 'reversed' Program 'aaaaaa' (starting with function 'fu', sub-function 'sf', line number 'nnn').	YES
PaaaaaaSTRfusf	Program Structure of 'reversed' Program 'aaaaaa' (starting with function 'fu', sub-function 'sf').	YES

NOTE: After the first choice of type 'Paaaaaa', 'Paaaaaa' can be replaced with '-'.

All notations between parentheses are optional.

On-Line Display Options

PxxxxxxCP

C1: Displays the call lines of Macro-Structures.

C2: Displays the number of the session in which the line was updated.

PxxxxxxB, -W, -P, -8.

C1: Displays the input format.

C2: Displays information concerning the origin of the lines.

The Macro-Structure code will appear in the source area for the Program lines obtained by a Macro-Structure call.

For lines that belong to the Library in which you are working, the LIB field indicates the number of the session in which the line was updated. For lines that belong to other Libraries, it indicates their code.

The 'C2' option cannot be used for updating.

C3: Available only on -W screens. Displays the selected format (I-E-S) of the Data Element.

The 'C3' option cannot be used for updating.

On-Line Action Codes

On the Program Definition screen:

'C'= Create.

'M'= Modify.

'D'= Delete (possible only if there's no description line).

'?'= Request for documentation ('HELP' function).

On Description screens:

'C'= Create the line.

'M'= Modify the line.

'D'= Delete the line.

'X'= Create or modify the line. When a line contains fields which are automatically transformed into uppercase, for example Pure COBOL Source Code (-9) lines, this transformation is inhibited.

'B'= Multiple deletion beginning with this line,

'R'= Repeat the line.

'I'= Insert lines.

'T'= Transfer the line (enter target line number in the LINE NUMBER field),

'G'= Group transfer of lines (enter target line number in the LINE NUMBER fields), beginning with this line,

'L'= With action codes 'B' or 'G' above: last line to be deleted or transferred,

'?'= Request for documentation (HELP function).

For more details on these Action Codes, refer to the 'Character Mode User Interface' guide.

Generation and/or Printing

Programs can be generated and printed by entering certain commands, either on-line, on the Generation and Print Commands (GP) screen (used for documentation and generation requests), or in batch mode (see the 'Developer's Procedures' manual).

These commands are listed below:

- LCP

List of all Programs by code.

C1: without keywords,

C2: with keywords.

- LNP

List of all Programs by name.

- LEP

List of all Programs by external name.

- LKP

List of Programs by keywords. The user may limit the keywords to explicit or implicit only. The keywords are specified on a continuation line (see the The 'Character Mode User Interface' guide).

- LTP

List of all Programs by type.

- DCP

Description information for the Program whose code is entered in the ENTITY CODE field; if no code has been entered, the Description information for all Programs will be provided.

C1: without assigned text,

C2: with the assigned text.

- DSP

Description information for the reversed Program whose code is entered in the ENTITY CODE field.

- GCP

Generation and description of a Program whose code must be indicated.

Upon generation, the Segments of the Data Structures called in the Program can contain up to 9999 Data Elements. An error message is displayed in the generation report if this number is exceeded.

- GSP

Generation and description of the reversed Program (with SC lines).

- FLP

Specify the flow of the programs. The user may specificy the environment (PEI), control card options, and parameters (as needed).

C1 option only.

- FSP

Specify the flow of the reversed Programs.

Chapter 7. Example of a Generated Program

Introduction

This chapter is designed to provide examples of how certain input will affect the automatically generated Program.

Only those portions of the Program that can be modified by Structured Code will be described in this chapter.

The displayed examples are not from the same Program. They are simply examples.

A batch Program structure was used; however the principle is the same for on-line programs.

Environment Division

The ENVIRONMENT DIVISION may be adapted as needed, via the Beginning Insertions (-B) screen entries.

The example below illustrates how the Beginning Insertions lines may be used to modify the INPUT-OUTPUT SECTION. The user entered the following lines in order to code the SELECT statements for ESDS and RRDS VSAM files.

```
------------------------------------------------------------
!  A SE PA LIN INSTRUCTION TO BE INSERTED                  !
!    01 DD 090  * ENTRY-SEQUENCED DATA SET EXAMPLE         !
!    01 DD 100      SELECT DD-FILE                         !
!    01 DD 120             ASSIGN TO ESMSTR                !
!    01 DD 140             ORGANIZATION IS SEQUENTIAL      !
!    01 DD 160             ACCESS MODE IS SEQUENTIAL       !
!    01 DD 180             FILE STATUS IS DD00-STATUS.     !
!    01 EE 090  * RELATIVE RECORD DATA SET EXAMPLE         !
!    01 EE 100      SELECT EE-FILE                         !
!    01 EE 120             ASSIGN TO RRMSTR                !
!    01 EE 140             ORGANIZATION IS RELATIVE        !
!    01 EE 160             ACCESS MODE IS DYNAMIC          !
!    01 EE 170             RELATIVE KEY IS WS00-RRN        !
!    01 EE 180             FILE STATUS IS EE00-STATUS.     !
!                                                          !
!O: C1 CH: -B                                             !
------------------------------------------------------------
```

The Call of Data Structures (-CD) lines are coded as follows:

```
------------------------------------------------------------
! DP   DL  EXTERN  OARFU        U    SELECTION            !
! DD   BL  ESMSTR  VSFID        C    *10                  !
!          STAT.FLD: DD00STATUS                          !
! EE   BL  RRMSTR  VSFID        C    *20                  !
!          STAT.FLD: EE00STATUS                          !
!                                                        !
!O: C1 CH: -CD                                          !
------------------------------------------------------------
```

NOTE: For more input entered for the RELATIVE KEY IS statement on the RRDS file, see subchapter "WORKING-STORAGE SECTION".

The excerpt of COBOL that is generated as a result of these lines follows.

```
   ENVIRONMENT DIVISION.                                    BLPROG
   CONFIGURATION SECTION.                                   BLPROG
   SOURCE-COMPUTER. IBM-370.                                BLPROG
   OBJECT-COMPUTER. IBM-370.                                BLPROG
   INPUT-OUTPUT SECTION.                                    BLPROG
   FILE-CONTROL.                                            BLPROG
   * ENTRY-SEQUENCED DATA SET EXAMPLE                       D01DD
       SELECT DD-FILE                                       D01DD
             ASSIGN TO ESMSTR                               D01DD
             ORGANIZATION IS SEQUENTIAL                     D01DD
             ACCESS MODE IS SEQUENTIAL                      D01DD
             FILE STATUS IS DD00-STATUS.                    D01DD
   * RELATIVE RECORD DATA SET EXAMPLE                       D01EE
       SELECT EE-FILE                                       D01EE
             ASSIGN TO RRMSTR                               D01EE
             ORGANIZATION IS RELATIVE                       D01EE
             ACCESS MODE IS DYNAMIC                         D01EE
             RELATIVE KEY IS WS00-RRN                       D01EE
             FILE STATUS IS EE00-STATUS.                    D01EE
   DATA DIVISION.                                           BLPROG
   FILE SECTION.                                            BLPROG
```

Working-Storage Section: Beginning

The WORKING-STORAGE SECTION and other sections belonging to the end of the DATA DIVISION may be supplemented via the Work Areas (-W) screen. The example shows the implementation of this feature using the Formatted Line for a Work Areas screen with an alphabetic CODE FOR COBOL PLACEMENT.

The CODE FOR COBOL PLACEMENT, when alphabetic, causes the data to be placed in the beginning of the WORKING-STORAGE SECTION, with this code and the LINE NUMBER producing a sequencing number.

The Work Areas (-W) screen appeared as follows:

```
------------------------------------------------------------
!CODE FOR PLACEMENT..:  BB                                  !
!A LIN T LEVEL OR SECTION WORK AREA DESCRIPTION             !
!* 020 F PC: XW  LC: XW SEL: 02____ PICT: I DESC: 2 LEV: 1 !
!  110 F DP: WK  DL: BB SEL: ______ PICT: I DESC: 2 LEV: 1 !
!  120 F DP: WA  DL: WA SEL: 00____ PICT: I DESC: 2 LEV: 1 !
------------------------------------------------------------
```

NOTE: The formatted line that appears as line 020 in this example comes from a Macro-Structure. (Notice the asterisk in the ACTION CODE field for this line.) The field labels that appear on the screen for this Macro ('PC:' and 'LC:') result from an older version of this line, and are exactly the same as those fields labeled 'DP:' and 'DL:'.

In the generated COBOL, the WORKING-STORAGE SECTION will begin with the Description of Segment XW02, followed by the Descriptions of all the Segments that belong to Data Struc- ture WK. Segment WA00 follows.

Another Work Areas (-W) screen was entered for this example, to illustrate setting up a search field for an RRDS file. In this example, the user did not use a formatted line. The CODE FOR COBOL PLACEMENT used was higher than the code used for the screen with the formatted lines above, thus the COBOL corresponding to this entry follows those lines.

This Work Areas (-W) screen appeared as follows:

```
------------------------------------------------------------
!CODE FOR PLACEMENT..:  CC                                  !
!A LIN T LEVEL OR SECTION WORK AREA DESCRIPTION             !
!  020 * RELATIVE RECORD DATA SET SEARCH FIELD              !
!  040   01              WS00-RRN   PIC 99 VALUE ZEROS.     !
------------------------------------------------------------
```

The System-generated 'WSS-BEGIN' will be generated after these supplementary lines.

```
   WORKING-STORAGE SECTION.                                          FL10UP
  *PC: XW  LC: XW SEL: 02______ PICT: I DESC: 2 LEV: 1 ORG: _ SS: _ 7BB020
   01                XW02.                                          FL10UP
        10           XW02-XDATE.                                    FL10UP
        11           XW02-XDAT1.                                    FL10UP
        12           XW02-XDAT19 PICTURE  99                        FL10UP
                     VALUE                ZERO.                     FL10UP
        11           XW02-XDAT2.                                    FL10UP
        12           XW02-XDAT29 PICTURE  99                        FL10UP
                     VALUE                ZERO.                     FL10UP
        11           XW02-XDAT3.                                    FL10UP
        12           XW02-XDAT39 PICTURE  99                        FL10UP
                     VALUE                ZERO.                     FL10UP
        10           XW02-XLEAPY PICTURE  99                        FL10UP
                     VALUE                ZERO.                     FL10UP
  *DP: WK  DL: BB SEL: ________ PICT: I DESC: 2 LEV: 1 ORG: _ SS: _ 7BB110
```

```
01               WK00.                                          FL10UP
     10          WK00-FLTKEY.                                   FL10UP
     11          WK00-LIBKEY PICTURE  X                         FL10UP
                 VALUE                SPACE.                    FL10UP
     11          WK00-FLM    PICTURE  999                       FL10UP
                 VALUE                ZERO.                     FL10UP
     11          WK00-FLDDEP PICTURE  X(6)                      FL10UP
                 VALUE                SPACE.                    FL10UP
     11          WK00-PSNAME PICTURE  X(4)                      FL10UP
                 VALUE                SPACE.                    FL10UP
     11          WK00-PANFI  PICTURE  X                         FL10UP
                 VALUE                SPACE.                    FL10UP
01               WK10.                                          FL10UP
     10          WK10-FLDCAN PICTURE  X(6)                      FL10UP
                 VALUE                SPACE.                    FL10UP
     10          WK10-FLCADE PICTURE  XXX                       FL10UP
                 VALUE                SPACE.                    FL10UP
     10          WK10-FLCAAR PICTURE  XXX                       FL10UP
                 VALUE                SPACE.                    FL10UP
     10          WK10-FLTDEP PICTURE  9(4)                      FL10UP
                 VALUE                ZERO.                     FL10UP
     10          WK10-FLTARR PICTURE  9(4)                      FL10UP
                 VALUE                ZERO.                     FL10UP
     10          WK10-ACMSER PICTURE  9(6)                      FL10UP
                 VALUE                ZERO.                     FL10UP
     10          WK10-FLCFS  PICTURE  X                         FL10UP
                 VALUE                SPACE.                    FL10UP
     10          WK10-FLQSTP PICTURE  9                         FL10UP
                 VALUE                ZERO.                     FL10UP
     10          WK10-FLQRFC PICTURE  9(3)                      FL10UP
                 VALUE                ZERO.                     FL10UP
     10          WK10-FLQRCC PICTURE  9(3)                      FL10UP
                 VALUE                ZERO.                     FL10UP
     10          WK10-FILLER PICTURE  X(31)                     FL10UP
                 VALUE                SPACE.                    FL10UP
01               WK20.                                          FL10UP
     10          WK20-PANTTL PICTURE  XXX                       FL10UP
                 VALUE                SPACE.                    FL10UP
     10          WK20-PANL   PICTURE  X(12)                     FL10UP
                 VALUE                SPACE.                    FL10UP
     10          WK20-PAMPHH PICTURE  9(10)                     FL10UP
                 VALUE                ZERO.                     FL10UP
     10          WK20-PAMPHW PICTURE  9(10)                     FL10UP
                 VALUE                ZERO.                     FL10UP
     10          WK20-RECCLA PICTURE  X                         FL10UP
                 VALUE                SPACE.                    FL10UP
     10          WK20-RECSMK PICTURE  X                         FL10UP
                 VALUE                SPACE.                    FL10UP
     10          WK20-REDCAN PICTURE  X(6)                      FL10UP
                 VALUE                SPACE.                    FL10UP
     10          WK20-FILLER PICTURE  X(22)                     FL10UP
                 VALUE                SPACE.                    FL10UP
*DP: WA  DL: WA SEL: 00______ PICT: I DESC: 2 LEV: 1 ORG: _ SS: _ 7BB120
 01              WA00.                                          FL10UP
     10          WA00-1TSTD.                                    FL10UP
```

```
       11            WA00-1HSTD  PICTURE  99                      FL10UP
                     VALUE                ZERO.                   FL10UP
       11            WA00-1RRCOL PICTURE  X                       FL10UP
                     VALUE                SPACE.                  FL10UP
       11            WA00-1MSTD  PICTURE  99                      FL10UP
                     VALUE                ZERO.                   FL10UP
       11            WA00-1AMPM  PICTURE  X                       FL10UP
                     VALUE                SPACE.                  FL10UP
       10            WA00-1TMIL.                                  FL10UP
       11            WA00-1HMIL  PICTURE  99                      FL10UP
                     VALUE                ZERO.                   FL10UP
       11            WA00-1MMIL  PICTURE  99                      FL10UP
                     VALUE                ZERO.                   FL10UP
       10            WA00-1FCCTR PICTURE  999                     FL10UP
                     VALUE                ZERO.                   FL10UP
       10            WA00-1CCCTR PICTURE  999                     FL10UP
                     VALUE                ZERO.                   FL10UP
  *RELATIVE RECORD DATA SET SEARCH FIELD                          7CC020
   01              WS00-RRN       PIC 99 VALUE ZEROS.            7CC040
   01   WSS-BEGIN.                                                FL10UP
        05   FILLER PICTURE X(7) VALUE 'WORKING'.                  FL10UP
        05   BLANC  PICTURE X    VALUE SPACE.                      FL10UP
        05   IK     PICTURE X.                                     FL10UP
```

Working-Storage Section: End

When the CODE FOR COBOL PLACEMENT is numeric, the data description is placed after those with alphabetic codes, and after most Data Structures Descriptions which come from the Call of Data Structures (-CD) or On-line Screen Call of Elements (-CE) or Call of Segments (-CS) screens.

The lines entered on the screen used for this example are generated just before the PROCEDURE DIVISION statement.

The Work Areas (-W) screen was coded as follows:

```
------------------------------------------------------------
!CODE FOR PLACEMENT..:  90                                  !
! LIN T LEVEL OR SECTION WORK AREA DESCRIPTION               !
! 000 F DP: WB  DL: WG SEL: 01_____ PICT: I DESC: 4 LEV: 3!
------------------------------------------------------------

  *DP: WB  DL: WG SEL: 01_____ PICT: I DESC: 4 LEV: 3 ORG: _ SS: _ 790000
   01               WB00.                                       PJJPS1
        02          WB01T.                                      PJJPS1
        03          WB01.                                       PJJPS1
        10          WB01-FILLER PICTURE  X(18).                 PJJPS1
        10          WB01-FILLER PICTURE  X(4).                  PJJPS1
        10          WB01-TABCPT PICTURE  X(44).                 PJJPS1
   PROCEDURE DIVISION.
```

Procedure Division

The user may modify the PROCEDURE DIVISION in any number of ways. The lines that are generated may be overridden, supplemented, or suppressed. New functions may be created for a Program by calling in Macros or by attaching Procedural Code (-P) lines directly to the Program. Lines of the Macro may be overridden, supplemented or suppressed.

The Procedural Code lines below illustrate different types of modifications that the user may make to the PROCEDURE DIVISION.

MODIFYING AUTOMATICALLY GENERATED FUNCTIONS

The Procedural Code (-P) lines below illustrate the overriding of the generation of the OPEN of the TR-FILE that would normally have occurred in Function F01.

```
------------------------------------------------------------
!                                          FUNCTION:  01 !
!A SF LIN OPE OPERANDS                    LVTY CONDITION  !
!* TR     N   INITIALIZATION OF FILE TR    10BL            !
!* TR     M   '1'  TR-FT                                   !
------------------------------------------------------------
```

Although the source of the lines above is a Macro which was called into the Program, the lines themselves are generated exactly as those lines that are attached directly would be.

Without these lines, the OPEN of the TR file would look just like that of the EM file.

Specific lines in certain automatically generated functions may be suppressed. The lines below illustrate suppressing the CLOSE of the TR-FILE that would normally occur in F20.

```
------------------------------------------------------------
!                                          FUNCTION:  20 !
!A SF LIN OPE OPERANDS                    LVTY CONDITION  !
!* TR     SUP                                             !
------------------------------------------------------------
```

The next excerpt shows the supplementation of Function F76. Here, the lines identified with an asterisk in the ACTION CODE field come from a Macro. The user has supplemented the lines of the Macro using Procedural Code (-P) lines attached directly to the Program. They are interspersed with lines of the Macro. This is controlled by the key : FUNCTION CODE, SUB-FUNCTION CODE, and LINE NUMBER. The example with F76AL shows that the user can override lines of a Macro with Procedural code lines of the same key:

```
------------------------------------------------------------
!                                          FUNCTION:  76 !
!A SF LIN OPE OPERANDS            LVTY CONDITION           !
!* AL     N   SEARCH              15DW I06 NOT > 27        !
!  AL  10 M   6    EM00-ERTYP     99IT UT-PR (I06) NOT = 0 !
!* AL  10 M   6    EM00-ERTYP     99IT UT-UPR (I06) NOT = 0!
!* AL  20 M   I06   EM00-ERCOD9                           !
!* AL  30 P   F76AU                                       !
!* AL  40 A   1    I06                                     !
------------------------------------------------------------
```

In F76AT, lines have been added to those of the Macro.

```
------------------------------------------------------------
!                                          FUNCTION:  76 !
!A SF LIN OPE OPERANDS            LVTY CONDITION           !
!* AT     N   PRINT GOOD TRANS..  10IT XW01-XERRCT = ZERO  !
!  AT   2 P   F92                 99IT 1-MB00-STRUCT = 'P' !
!  AT   5 GT  10                  99IT XW01-XERRCT NOT = 0 !
!* AT  10 M   SPACE  EM00                                 !
!* AT  20 P   F8RBB  F8R-FN       99IT XW01-XIPRIN = '1'   !
!* AT  30 GT  10                  99BL                     !
------------------------------------------------------------
```

Note that the generated code contains lines that are generated automatically by the System as well as these lines.

CREATING NEW FUNCTIONS

New functions may be added to the generated skeleton simply by using a FUNCTION CODE that is not generated. The lines will be placed within the Program according to the value of the FUNCTION CODE. Our example uses Function F93.

```
   N01.      NOTE ************************************.           FL10UP
                  *                                   *           FL10UP
                  *        INITIALIZATIONS            *           FL10UP
                  *                                   *           FL10UP
                  ************************************.           FL10UP
   F01.      EXIT.                                                FL10UP
   N01BB.    NOTE *INITIALIZATION OF FILE  BB-FILE    *.          FL10UP
   F01BB.    OPEN I-O                      BB-FILE.               FL10UP
   F01BB-FN. EXIT.                                                FL10UP
   N01EM.    NOTE *INITIALIZATION OF FILE  EM-FILE    *.          FL10UP
   F01EM.    OPEN INPUT                    EM-FILE.               FL10UP
   F01EM-FN. EXIT.                                                FL10UP
   N01MB.    NOTE *INITIALIZATION OF FILE  MB-FILE    *.          FL10UP
   F01MB-10. RETURN   MB-FILE      AT END                         FL10UP
             MOVE  1 TO                  MB-FI.                   FL10UP
   F01MB-FN. EXIT.                                                FL10UP
   N01TR.    NOTE *INITIALIZATION OF FILE TR          *.          P000
   F01TR.                                                         P000
       MOVE        '1' TO TR-FT.                                  P010
   F01TR-FN. EXIT.                                                P010
```

```
N01XE.    NOTE *INITIALIZATION OF FILE  XE-FILE    *.           FL10UP
F01XE.    OPEN OUTPUT                   XE-FILE.               FL10UP
F01XE-FN. EXIT.                                                FL10UP
F01-FN.   EXIT.                                                FL10UP
                              .
                              .
                              .
                              .
N20.      NOTE ************************************.           FL10UP
               *                                  *            FL10UP
               *           END OF RUN             *            FL10UP
               *                                  *            FL10UP
               ************************************.           FL10UP
F20.      IF FT =            ALL '1'                           FL10UP
          NEXT SENTENCE ELSE GO TO     F20-FN.                 FL10UP
F20BB.    CLOSE    BB-FILE.                                    FL10UP
F20BB-FN. EXIT.                                                FL10UP
F20EM.    CLOSE    EM-FILE.                                    FL10UP
F20EM-FN. EXIT.                                                FL10UP
F20XE.    CLOSE    XE-FILE.                                    FL10UP
F20XE-FN. EXIT.                                                FL10UP
                              .
                              .
                              .
                              .
N76.      NOTE ************************************.           FL10UP
               *                                  *            FL10UP
               * STORE ERRORS, RETRIEVE INIT. STATE*           FL10UP
               *                                  *            FL10UP
               ************************************.           FL10UP
N76-A.    NOTE *      STORE ERRORS                *.           FL10UP
F76-A.    IF ID-ER NOT = '0' MOVE ID-ER TO TR-ER               FL10UP
          GO TO F76-C. MOVE SE-ER (I01) TO SEG-ER.             FL10UP
          IF SEG-ER < '0' OR SEG-ER > '1'                      FL10UP
          MOVE SEG-ER TO TR-ER GO TO F76-C.                    FL10UP
          MOVE 1 TO I06.                                       FL10UP
F76-B.    MOVE DE-ER (I06) TO DEL-ER.                          FL10UP
     IF DEL-ER = '1'  OR DEL-ER = '0' GO TO F76-B1.            FL10UP
          MOVE 4 TO TR-ER GO TO F76-C.                         FL10UP
F76-B1.   IF I06 = I50 MOVE I03 TO I06 GO TO F76-B.            FL10UP
          IF I06 < I04 ADD  1   TO I06 GO TO F76-B.            FL10UP
F76-C.    IF TR-ER NOT = '1' MOVE '1' TO GR-ER.                FL10UP
N76AB.    NOTE *INITIALIZATIONS                       *.       P000
F76AB.                                                         P000
    MOVE        SPACE TO EM00                                  P010
    MOVE        ZERO TO XW01-XERRCT                            P020
    MOVE        '0' TO XW01-XIPRIN                             P030
    MOVE        1-MB00 TO XW01-XTRAIM.                         P040
F76AB-FN. EXIT.                                                P040
N76AC.    NOTE *IDENTIFICATION ERROR                  *.       P000
F76AC.    IF    ID-ER NOT = ZERO                               P000
          NEXT SENTENCE ELSE GO TO     F76AC-FN.               P000
    MOVE        '1' TO EM00-ERTYP                              P010
    MOVE        ID-ER TO EM00-ERCOD                            P020
    PERFORM     F76AU THRU F76AU-FN.                           P030
```

```
F76AC-900. GO TO F76AD-FN.                                          P030
F76AC-FN. EXIT.                                                     P030
N76AD.    NOTE *ELEMENT/RECORD ERROR                 *.           P000
F76AD.         EXIT.                                                P000
N76AE.    NOTE *RECORD ERROR                         *.           P000
F76AE.                                                              P000
    MOVE        SE-ER (I01) TO EM00-ERCOD.                        P010
          IF    EM00-ERCOD NOT = '0  '                            P020
          AND   EM00-ERCOD NOT = '1  '                            P030
    MOVE        '0' TO EM00-ERTYP                                 P020
    PERFORM     F76AU THRU F76AU-FN.                              P030
F76AE-FN. EXIT.                                                     P030
N76AF.    NOTE *ERROR IN COMMON PART OF SEGMENT    *.           P000
F76AF.                                                              P000
    MOVE        1 TO I06.                                         P010
N76AG.    NOTE *SEARCH                               *.           P000
F76AG.    IF    I06 NOT > I50                                     P000
          NEXT SENTENCE ELSE GO TO     F76AG-FN.                    P000
    MOVE        DE-ER (I06) TO EM00-ERTYP.                        P010
          IF    EM00-ERTYP NOT = ZERO                             P020
          AND   EM00-ERTYP NOT = '1'                              P030
    MOVE        I06 TO EM00-ERCOD9                                P020
    PERFORM     F76AU THRU F76AU-FN.                              P030
    ADD         1 TO I06.                                         P040
F76AG-900. GO TO F76AG.                                             P040
F76AG-FN. EXIT.                                                     P040
F76AF-FN. EXIT.                                                     P040
N76AH.    NOTE *ERROR IN SPECIFIC PART OF SEGM.    *.           P000
F76AH.                                                              P000
    MOVE        I03 TO I06.                                       P010
N76AI.    NOTE *SEARCH                               *.           P000
F76AI.    IF    I06 NOT > I04                                     P000
          NEXT SENTENCE ELSE GO TO     F76AI-FN.                    P000
    MOVE        DE-ER (I06) TO EM00-ERTYP.                        P010
          IF    EM00-ERTYP NOT = ZERO                             P020
          AND   EM00-ERTYP NOT = '1'                              P030
    COMPUTE     EM00-ERCOD9 = I06 - I03 + 1                       P020
    PERFORM     F76AU THRU F76AU-FN.                              P030
    ADD         1 TO I06.                                         P040
F76AI-900. GO TO F76AI.                                             P040
F76AI-FN. EXIT.                                                     P040
F76AH-FN. EXIT.                                                     P040
F76AD-FN. EXIT.                                                     P040
N76AK.    NOTE *USER ERRORS                          *.           P000
F76AK.                                                              P000
    MOVE        1 TO I06.                                         P010
N76AL.    NOTE *SEARCH                               *.           P000
F76AL.    IF    I06 NOT > 27                                      P000
          NEXT SENTENCE ELSE GO TO     F76AL-FN.                    P000
          IF    UT-PR (I06) NOT = ZERO                            P010
    MOVE        6 TO EM00-ERTYP                                   P010
    MOVE        I06 TO EM00-ERCOD9                                P020
    PERFORM     F76AU THRU F76AU-FN.                              P030
    ADD         1 TO I06.                                         P040
F76AL-900. GO TO F76AL.                                             P040
```

```
F76AL-FN. EXIT.                                                    P040
F76AK-FN. EXIT.                                                    P040
N76AR.    NOTE *TRANSACTION ERROR (GROUP)          *.              P000
F76AR.    IF    FTB7 = 1                                           P000
          NEXT SENTENCE ELSE GO TO     F76AR-FN.                   P000
    MOVE        1 TO I06.                                          P010
N76AS.    NOTE *SEARCH                             *.              P000
F76AS.    IF    I06 NOT > I51                                      P000
          NEXT SENTENCE ELSE GO TO     F76AS-FN.                   P000
          IF    SE-ER (I06) = '2'                                  P010
    MOVE        SE-ER (I06) TO XW01-XERRN1                         P010
    MOVE        I06 TO XW01-XERRN2                                 P020
    MOVE        XW01-XERRNU TO EM00-ERCOD                          P030
    MOVE        '0' TO EM00-ERTYP                                  P040
    PERFORM     F76AU THRU F76AU-FN.                               P050
    ADD         1 TO I06.                                          P060
F76AS-900. GO TO F76AS.                                            P060
F76AS-FN. EXIT.                                                    P060
F76AR-FN. EXIT.                                                    P060
N76AT.    NOTE *PRINT GOOD TRANSACTIONS            *.              P000
F76AT.    IF    XW01-XERRCT = ZERO                                 P000
          NEXT SENTENCE ELSE GO TO     F76AT-FN.                   P000
          IF    1-MB00-STRUCT = 'P'                                P002
    PERFORM     F92 THRU F92-FN.                                   P002
          IF    XW01-XERRCT NOT = ZERO                             P005
        GO TO     F76AT-FN.                                        P005
    MOVE        SPACE TO EM00.                                     P010
          IF    XW01-XIPRIN = '1'                                  P020
    PERFORM     FBRBB THRU FBR-FN.                                 P020
        GO TO     F76AT-FN.                                        P030
N76AU.    NOTE *PRINT BAD TRANSACTIONS             *.              P000
F76AU.                                                             P000
    MOVE        'A' TO EM00-ENTYP                                  P010
    ADD         1 TO XW01-XERRCT.                                  P020
N76AW.    NOTE *READ ERROR MESSAGE FILE            *.              P000
F76AW.                                                             P000
    MOVE        LE-FIENR TO EM00-PROGR                             P005
    MOVE        LIBRA TO EM00-LIBRA                                P010
    MOVE        0 TO EM00-LINUM                                    P015
    MOVE 0 TO IK                                                   P020
    READ           EM-FILE                                         P020
    INVALID KEY MOVE 1 TO IK                                       P020
    MOVE        'UNKNOWN MESSAGE' TO EM00-ERMSG                    P030
    DISPLAY     '  UNKNOWN MESSAGE, KEY IS '                       P040
    EM00-EMKEY.                                                    P050
F76AW-FN. EXIT.                                                    P050
N76AX.    NOTE *STORE ERROR                        *.              P000
F76AX.                                                             P000
    MOVE        4 TO TR-ER.                                        P010
F76AX-FN. EXIT.                                                    P010
N76AZ.    NOTE *PERFORM THE PRINT ROUTINE          *.              P000
F76AZ.                                                             P000
    PERFORM     FBRBB THRU FBR-FN.                                 P010
F76AZ-FN. EXIT.                                                    P010
F76AU-FN. EXIT.                                                    P010
```

```
F76AT-FN. EXIT.                                                 P010
F76-FN.   EXIT.                                                 P010
                              .
                              .
                              .
                              .
N93DA.    NOTE *DATE VALIDATION                    *.           P000
F93DA.                                                          P000
    MOVE        1 TO DEL-ER.                                    P010
          IF    XW02-XDATE NOT NUMERIC                          P020
    MOVE        4 TO DEL-ER                                     P020
        GO TO     F93DA-FN.                                     P030
          IF    XW02-XDAT1 > '12'                               P040
          OR    XW02-XDAT1 = '00'                               P050
          OR    XW02-XDAT2 > '31'                               P060
          OR    XW02-XDAT2 = '00'                               P070
    MOVE        5 TO DEL-ER                                     P040
        GO TO     F93DA-FN.                                     P050
          IF    XW02-XDAT2 > '30'                               P080
          AND   (XW02-XDAT1 = '04'                              P090
          OR    XW02-XDAT1 = '06'                               P100
          OR    XW02-XDAT1 = '09'                               P110
          OR    XW02-XDAT1 = '11')                              P120
    MOVE        5 TO DEL-ER                                     P080
        GO TO     F93DA-FN.                                     P090

          IF    XW02-XDAT1 NOT = '02'                           P130
        GO TO     F93DA-FN.                                     P130
          IF    XW02-XDAT2 > '29'                               P140
    MOVE        5 TO DEL-ER                                     P140
        GO TO     F93DA-FN.                                     P150
          IF    XW02-XDAT3 = '00'                               P160
    MOVE        1 TO XW02-XLEAPY                                P160
          ELSE                                                  P170
    COMPUTE     XW02-XLEAPY = XW02-XDAT39 -                     P170
    (XW02-XDAT39 / 4) * 4.                                      P180
          IF    XW02-XLEAPY NOT = ZERO                          P190
          AND   XW02-XDAT2 > '28'                               P200
    MOVE        5 TO DEL-ER                                     P190
        GO TO     F93DA-FN.                                     P200
F93DA-FN. EXIT.                                                 P200
```

Chapter 8. Appendix: Pure Cobol Source Code (-9)

The Pure COBOL Source Code (-9) screen contains COBOL source code statements.

It is used for the following:

- To use data descriptions generated by the System in COBOL programs,
- To use library and documentation components to manage existing COBOL source code.

DATA GENERATION

It is possible to generate the first three Divisions of a COBOL program (using the System's functions) and to write the PROCEDURE DIVISION exclusively with Pure COBOL Source Code (-9) lines.

This kind of program is defined with a 'D' value in the TYPE AND STRUCTURE OF PROGRAM field and the appropriate variant in the TYPE OF COBOL TO GENERATE field.

The program will be made up of Call of Data Structures (-CD) lines, of Beginning Insertions (-B) lines, of Work Areas (-W) lines and of Pure COBOL Source Code (-9) lines for the PROCEDURE DIVISION.

It is also possible to generate the data description only, and to write the rest of the Program using Pure COBOL Source Code (-9) lines.

The Program would then have an 'F' in the TYPE AND STRUCTURE OF PROGRAM field and a corresponding variant in the TYPE OF COBOL TO GENERATE field.

This Program will be made up of Call of Data Structures(-CD) lines and Pure COBOL Source Code (-9) lines. The positioning of the generated descriptions in the Program is determined by a Pure COBOL Source Code (-9) line for each Call of Data Structures (-CD) line. This Pure COBOL Source Code (-9) line must have an 'F' value in COBOL column '7', followed by a blank, and the DATA STRUCTURE CODE IN THE PROGRAM. The generated descriptions begin on the first '01' level and do not include the COBOL 'FD' clause.

PURE COBOL SOURCE

It is possible to manage pure COBOL Source Code (-9) in the System Database. It can be extracted by generating the Program with the 'C' variant in the TYPE OF COBOL TO GENERATE field.

There can be two versions of a Program:

COBOL (Pure COBOL Source Code (-9) lines); and VA Pac (Call of Data Structures (-CD), Call of a P.M.S.'s (-CP), Beginning Insertions (-B), Work Areas (-W), and Procedural Code (-P) lines).

If the variant is 'C' in the TYPE OF COBOL TO GENERATE field, the Pure COBOL Source Code (-9) lines will be generated and the programming lines will be ignored.

If the value in the TYPE OF COBOL TO GENERATE field specifies a COBOL variant, the program will be generated with programming lines, and the Pure COBOL Source Code (-9) lines will be ignored.

If the user wishes to see the information found in the 'CONSTANTS' area of a generated Program in the Pure COBOL Source program, he/she must use the Pure COBOL Source Code (-9) lines to code this information, being sure to use line numbers in the WORKING-STORAGE SECTION to ensure that this information is properly placed.

The -9 line must be coded in the following way:

```
01   PAC-CONSTANTS PICTURE X(20) VALUE '20 characters..'
col  12                    35           48
```

For example:

01 PAC-CONSTANTS PICTURE X(20) VALUE 'XXXXXXXXXXXXXXXXXXX'

The result will have the following structure:

AAA9999V000 DDDDDDDD, where:

- (AAA) is the application code,
- (9999V) is the number of the session during which the program was extracted,
- (DDDDDDDD) the date of the extraction.

If the user wishes to see more information (e.g. Database code, user code...), he must code the -9 line in the following way:

```
col  12                     35
```

```
 01  PAC-CONSTANTS PICTURE  X(87) VALUE
-              'PACBASE-C20
       'DATGNC    '.
```

The result will be composed of the following concatenated elements:

- Session number (5 char.),
- Application code (3 char.),
- Generation date (8 char.),
- System Program code (6 char.),
- User code (8 char.),
- Time of Program generation (8 char.),
- COBOL Program-Id (8 char.),
- Database code (4 char.),
- Date of Program generation with century (10 char.),
- Version (7 char.),
- Date of the generator (10 char.),
- Date of the generation skeleton (10 char.).

OPERATION FIELD

C1: default value.

C2: source and complementary input field display.

```
                  PURCHASING MANAGEMENT SYSTEM        SG000008.LILI.CIV.1583

 PURE COBOL PROCEDURE    LTVAL1 FIRST TEST PROGRAM
                         1

 2 3      4 6      5
 A LINE   C COMPL. COBOL INSTRUCTION                          : SOURCE LIBR.
   006000 * PJJPS1 DUPLICATE RECORD VALIDATION                  0318
   006001   PJJPS1 F36.                                        0318
   006002   PJJPS1     IF CATX  = 'HA'                         0318
   006003   PJJPS1     MOVE '1' TO W-WW00-REC                  0318
   006004   PJJPS1     ELSE MOVE ZERO TO W-WW00-REC.           0318
   006005   PJJPS1 F36-FN.                                     0318

 *** END ***
 0: C2 CH: -9
```

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
1	6		PROGRAM CODE
			Code identifying the program in the library.
2	1		ACTION CODE
		'C'	Creation of the line
		'M'	Modification of the line
		'D' or 'A'	Deletion of the line
		'T'	Transfer of the line
		'B'	Beginning of multiple deletion
		'G'	Multiple transfer
		'?'	Request for HELP documentation
		'E' or '-'	Inhibit implicit update
		'X'	Implicit update without upper/lowercase processing
3	6	NUMER.	COBOL LINE NUMBER
			FALSE NUMERIC FIELD

NUM	LEN	CLASS VALUE	DESCRIPTION OF FIELDS AND FILLING MODE
		'0-999999'	The line number can be entered for each COBOL instruction and renumbered after an update. It is advisable to use a line increment of 100 in COBOL.
		'nn00 -' 'nn99'	In a macro-structure, the COBOL line number can be parameterized on the four leftmost digits by two-digit groups.
4	1		CONTINUATION (COBOL COLUMN 7)
		'-'	Continuation of a literal (normal COBOL use).
		'*'	Comment line (ANSI COBOL only).
5	65		COBOL INSTRUCTION
			First part of the COBOL line. The end of the line (column 66 to 72) is displayed with the C2 OPERATION (O: C2).
6	6		END OF COBOL LINE
			This field is displayed as a complementary field which is viewed only with the C2 OPERATION (O: C2).
			(1) This field is used to complete a COBOL line up to 72 positions.
			(2) It also corresponds to the Identification area on a COBOL coding form (columns 73 to 80) for a Program identification entry. This entry appears on the far right side of a VisualAge Pacbase generated program.

Part Number: DDSTR000352A - 7464

Printed in USA