CodeTran

Program Conversion Tool

Version: 1.1
Date: November, 2010

Harry M. Sneed

Harry.Sneed@anecon.com

Software Design und Beratung G.m.b.H.
Alserstraße 4 / Hof 1
A-1090 Wien

1. Purpose of the CodeTran Tools

CodeTran is as the name implies a set of tools for transforming programs written in one language into programs in another language. The three legacy languages transformed are IBM-Assembler, PLI and COBOL, all languages of the conventional mainframe world. Assembler programs are transformed to either ANSI-COBOL or PL/I. COBOL programs can be transformed to either OO-COBOL or Java. PL/I programs are only transformed to Java. The purpose of such a transformation is to free the user from his current environment. It is not to improve the quality of the code. For that, the tool SoftRedo is intended. SoftRedo reformats, refines, renames, restructures and refactors Assembler, PL/I, COBOL and C/C++ code. CodeTran is for transforming the code stepwise over into an object-oriented form in the language Java. To improve the quality of his code, the user should renovate it with SoftRedo before converting it with CodeTran.

The goal here is to migrate from one hardware platform to another and/or to migrate into a newer software development environment. One can also speak of software modernization. The application software is taken out of its original, procedural architecture and placed in a newer, object-oriented architecture. If the code happens to be in Assembler language, then the code has to be first translated into COBOL or PL/I. After that the COBOL or PL/I code is converted over into Java. The ultimate objective is to have the code in an open, non-proprietary language – Java - which is compatible with modern internet technologies and will execute on any machine. This gives users the necessary machine independence and software flexibility to be able to change vendors at will and to connect to other modern software systems bought or taken from the open source community.

Another important factor is the personnel problem. There are less and less programmers with Assembler, COBOL and PL/I knowledge and more and more programmers with Java knowledge. In order to attract young developers, a user organization must have a software development environment with which they are familiar. This and the flexibility issue are the most important motives for a change of language.

The motive for changing the architecture is similar. Whether object-oriented software is really superior to procedural software is open to question but that is not the point here. The point is that the young generation of software developers has been trained to think in that mode, whereas the older developers can only think in the procedural mode. Some may be able to change their way of thinking, but most won't. The older generation of developers is going out. The newer generation is coming in. Certainly there will be another paradigm change in the near future – see aspect orientation - and an even younger generation will follow it, but at the moment object technology is the current mode of constructing software systems. Users should be encouraged to adapt to it, even if it is not the panacea that they may believe.

In this respect there is also a need to make legacy software understandable to those who are taking over the migrated system. The code should not only be converted and put in an object-oriented form, but also documented in such a way as to make the converted version comprehensible to those who have to test and maintain it. The automated translation to Java may only be a preliminary step to reimplementing the software. The final goal of the user could be to reuse the old code in developing a new solution. By automatically documenting the code the user will gain an overview of his system and see which business functions are

being performed by which technical code units. Portions of the old code can be built into the new Java system and the documentation helps in discovering which portions can be reused.

Thus CodeTran actually serves a fourfold purpose:

- It translates old Assembler code into COBOL or PL/I.
- It translates COBOL and PL/I Code into Java Code
- It transforms the old procedural structure into an object-oriented one
- It creates a post documentation of the transformed code.

(*see Figure 1: The multipurpose of CodeTran*)

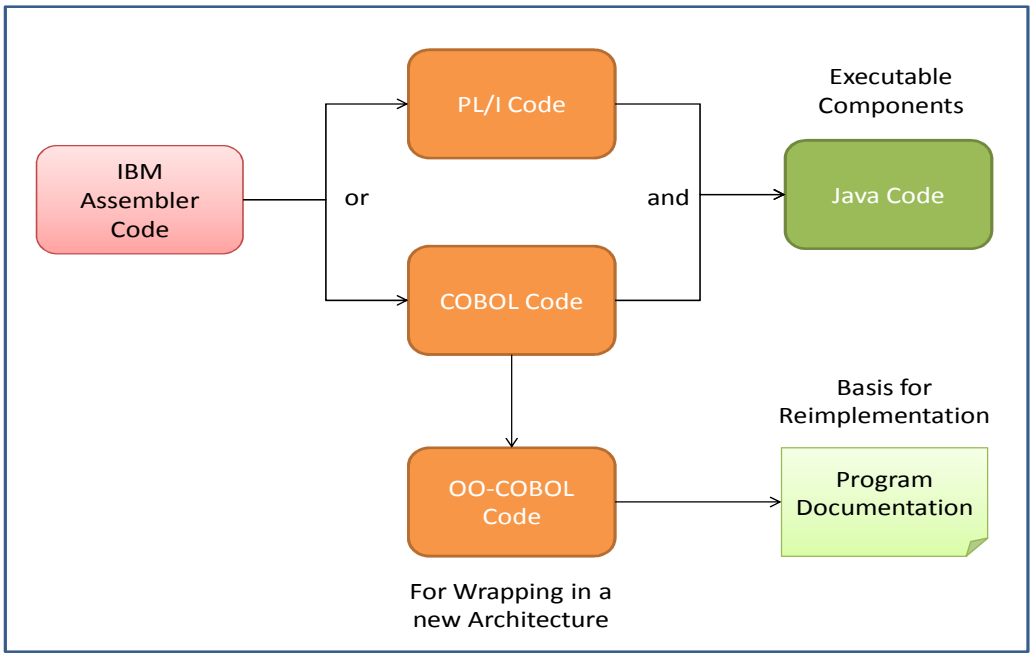


Figure 1: The Multipurpose of CodeTran

1.1. Translating Assembler into COBOL or PL/I

If the user still has Assembler code in operation he must see that this code is converted to the higher order language COBOL or PL/I, before it can be converted to Java. There is no direct translation from Assembler to Java. CodeTran supports the translation of Assembler to COBOL or to PL/I by automatically converting most statements and data type definitions. Of course, the resulting COBOL or PL/I code will have to be cleaned up and tested before being moved on to Java, but at least CodeTran offers this possibility to users with Assembler remnants.

1.2. Translating COBOL and PL/I into Java

Both COBOL and PL/I procedures are converted to Java on a statement by statement basis. For every COBOL or PL/I procedural statement there will be one or more equivalent Java statements. COBOL paragraphs and PL/I procedures become Java methods. Every COBOL or PL/I data structure becomes a Java class. With a few exceptions that remain to be manually translated, all statement and data types are automatically translated into a Java equivalent. The original code units – paragraphs, subroutines and procedures – are preserved

as methods in Java, including their comments. The COBOL and PL/I data structures are all converted to an array of ASCII characters with a Get and Set method for each data item. At runtime they are casted into the appropriate Java data type.

1.3. Transforming to an Object-Oriented Architecture

As opposed to other conversion tools on the market, which preserve the Java code in a procedural mode, CodeTran also creates an object-oriented architecture. The guiding principle for this is locality of reference. The procedural statements are placed together with the data they use. Both data and procedures are encapsulated into units of processing, i.e. classes, which can be invoked from outside. Every class has interfaces to the external environment. Each procedure, or method can have its own interface, but they all use the same data attributes. These processing units are then assembled into hierarchies referred to as class hierarchies. A component, or package, is a hierarchy of classes.

Classes are created from external data records, internal data groups, maps, reports and interface structures. These are taken over from the original program. So the starting point for the transformation of procedural to object-oriented code is the data objects, i.e. structures. To them are added the procedures and paragraphs which process them. The result is a set of classes each with its own methods. This avoids having one big God class containing all of the methods as is often done in code migration.

The goal of CodeTran is to create a class hierarchy for every COBOL or PL/I program. The assumption made, is that every program corresponds to a Use Case and that every Use Case has its own class hierarchy. The idea is to avoid dependencies between Use Cases by isolating the class hierarchies. Thus, classes within a component can interact with one another, but they cannot directly interact with classes in other components. This enables the components to be tested and used as independent web services. This approach has its disadvantages, the same data and the same procedures may appear in many components, but the goal of functional and data independence has precedence over that of minimizing the code amount. It should be possible to change or extend a component without worrying about which affect that may have on the other components. This is only possible if every component is self contained.

This partitioning of code into components with the duplicated classes may lead some to claim that this is more an object-based rather than a pure object-oriented solution. The important thing is that the data of a PLI or COBOL program is broken up into many classes – one for every storage class and for every data record within that storage class – and that the procedures and paragraphs are assigned to that class whose data they reference most. There will of course still be many references to data in other classes of that component. These will use the get and set methods of those classes. However, the dominating principle remains that of locality of reference. The fact that CodeTran generates genuine object-oriented code units, distinguishes it from other Java conversion tools.

1.4. Documenting the converted Code

The final goal of CodeTran is to provide the user with a comprehensive documentation of each and every component. This is done in three ways, one by placing comments within the source code, secondly by generating a JavaDoc directory and thirdly by creating an XML

table of contents. Every class and every method has a header. Every method also has a trailer. The method header indicates which data types are used in what way by the method, as input, output or predicate. The method trailer indicates which other methods may follow upon this method, i.e. all potential successors. The class header indicates which interfaces this class has to the outside world, i.e. its parameters.

The information for the JavaDoc directory is extracted from the class structures and the code comments. The goal is to have a directory of all components and classes. Within the classes there is a further directory of all methods and attributes. The classes are linked by their hierarchical and horizontal relationships. It should be possible for a user to navigate up and down and across the classes by tracing data and control flows. JavaDoc is a standard open source product which each user can set up for himself.

The third type of documentation produced is an XML file for each component with a table of contents of all classes, interfaces and methods contained therein. This XML document records the structure, the data flow between methods and the control flow from method to method. With it the user can trace the control flow through the component.

2. Functions of the CodeTran Tools

To fulfill the goals outlined above, CodeTran follows a well defined migration process with no less than 20 distinct steps of which 11 are carried out by CodeTran, 5 by SoftRedo and 4 by the user himself:

1) The user should comment the code and assign long meaningful names to the short meaningless data types.
2) SoftRedo reformats the code by splitting and indenting lines (Reformat).
3) SoftRedo refines the code and eliminates undesirable code constructs (Refine).
4) SoftRedo removes incompatible data types by converting all numeric data types into decimal character format (Remove).
5) SoftRedo replaces hard coded data by symbolic constants and text tables (Replace).
6) SoftRedo relocates the IO and database access operations to separate data access routines (Relocate).
7) The user should compile the reengineered programs to ensure that no syntax errors have occurred.
8) CodeTran reorganizes the data structures of each program putting individual variables which are not embedded within a user defined data structure into a global data structure (DataPrep)
9) CodeTran makes a dataflow analysis and creates a table of all data references, ordered by paragraph or procedure (DataRef)
10) CodeTran creates a data description table with the types and displacements of all data used by a program (DataTab).
11) CodeTran generates a Java class for each data structure with the data attributes and the put and set methods for each attribute (ClassGen)
12) CodeTran generates a Java method from each paragraph or procedure together with a comment header and a comment trailer (MethGen)
13) CodeTran merges the methods with the classes by assigning the methods to the class whose data they reference the most (JavaGen)
14) CodeTran creates an XML document for each Java Component
15) CodeTran generates a test driver based on the data description table to test the converted Java code against the original code.
16) CodeTran cuts up the generated classes into separate source members
17) CodeTran assigns the subprograms to a subprogram class library and marshals the parameters into parameter objects
18) CodeTran creates a JavaDoc Repository for each Java component.
19) The user should compile the converted Java classes to ensure that they are syntactically correct.
20) The user should test the converted Java components with the generated test driver to ensure that they are functionally equivalent to the original programs.

(*see Figure 2: CodeTran Code Transformation Process*)

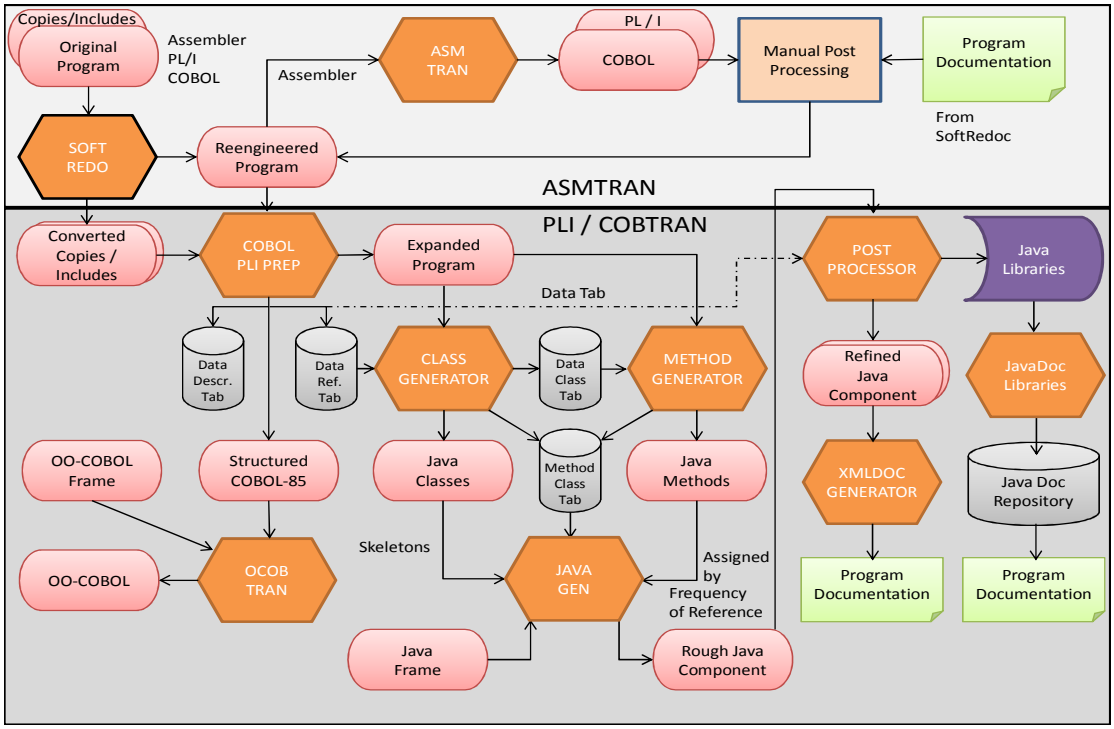


Figure 2: Code Transfprmation Process

2.1. Commenting and renaming the Data

Before launching a migration there is a task to be done which only the user programmers can do and that is to comment the code and rename the data. Of course the CodeTran tools will transform the legacy code and data even if they are not commented, but the result of the transformation is much better if the attribute names are meaningful and the methods are commented. In COBOL the user should insert a comment line after each data picture declaration describing what the data means. In PL/I the comment can be placed after the data declaration on the same line. In COBOL the user should insert one or more comment lines before each paragraph describing what function the paragraph is performing. If there happens to be some documentation, there should also be a reference to that documentation. In PL/I the comments should be placed before each internal procedure. In both languages there should be a comment header block at the beginning of each source module, describing the purpose of that module.

There is a separate step in SoftRedo for renaming the data items in a user program. It assumes that the user has created an excel table of short and long names. If so, the short names will be replaced by the long speaking names wherever they occur in the code. If the user has placed the long name in comment behind the data declaration in a systematic way, there is a pre-process which will extract the long names from the comments and create the excel table automatically. In any case, it should be the goal of the user to carry over speaking names and comments into the new code and not keep the original assembler like labels which will subvert the readability of the code.

2.2. Code Reengineering

A necessary prerequisite to the transformation of code by CodeTran is the reengineering of that code by SoftRedo. CodeTran assumes that the original code fulfills certain prerequisites. There are four prerequisites which must be fulfilled, in order for CodeTran to work properly.

- The procedural copies must be inserted into the code
- The code must be formatted, indented and the comments placed in separate lines.
- Obsolete and undesirable code features must be removed
- The machine specific data types like hexadecimal, binary, zoned decimal and floating-point must all be converted to ASCII character type fields with a picture clause. This applies to the data copies as well.

There are four other prerequisites which should be fulfilled if the user wants to attain a higher code quality:

- The short, non comprehensible names should be renamed into longer, more understandable ones
- The hard coded data – numeric and literal constants – should be taken out of the procedural code and placed in separate resource files or tables.
- The database access, screen operations and file input/output operations, i.e. all interactions with the environment, should be taken out of the mainline code, i.e. the business logic, and relocated to an access section at the end of the program.
- The deeply nested procedures or paragraphs should be refactored and the inner statements extracted to a separate subroutine which is invoked from the original location.

To fulfill the must requirements the following Redo steps must be executed:

- AddCopy to pull in the procedural copies and includes
- Reformat to reformat the code, split the lines and place comments in separate lines
- Refine to filter out obsolete and undesirable code features
- Remove to remove the machine specific data types.

To meet the additional requirements the following Redo steps can be executed:

- Rename to replace the short, mnemonic names with long, meaningful ones
- Replace to replace the hard coded data – literal texts and numeric constants – with symbolic constants
- Relocate to relocate the database access and input/output operations
- Refactor to factor out the deeply nested procedures to separate subroutines.

What should not be executed is the restructuring function – Restruct – to remove the GOTO branches, since CodeTran has its own method for handling GOTO branches in Java. The input to CodeTran is the output of SoftRedo – a source library of COBOL or PL/I modules and data copies produced by SoftRedo.

2.3. Compiling the reengineered programs

Before moving on to CodeTran the reengineered Assembler, COBOL or PL/I programs should in any case be compiled if not retested. There is another tool SoftVer which will verify the reengineered programs against the original versions. This is to ensure that neither syntax nor semantic errors have been introduced by the reengineering process. If so, then they have to be corrected before commencing with the code transformation process. One time syntax

errors can be corrected manually. If there is a systematic error which occurs in many places, the SoftRedo tool has to be corrected and the reengineering process repeated.

2.4. Reorganizing the Program Data Structures

The first step in the actual code transformation process is to reorganize the data structures. For the sake of creating objects, all data fields must be contained within some data structure. This means that all elementary data fields have to be collected together in one or more global data structures for each program. In COBOL that means all 77 level and also 01 level data items with no substructure. In PL/I all standalone data declarations with the exception of the procedure parameters must be put into a container structure. When this step is completed the source will have no more elementary standalone data elements. Except for the parameters of a procedure, all the data elements will be contained within a 01 record structure. These will be the future objects. This step is included in the CodeTran preprocessor job COBPrep or PLIPrep.

2.5. Program Data Flow Analysis

The second step in the actual code transformation process is to analyze the data usage and to create a data reference table. Here the procedural code is parsed to recognize each and every variable used. A variable can be used as either an argument = input, a result = output, a predicate = conditional operand, or a parameter to a module or procedure. In the data reference table these references are ordered in COBOL by paragraph and in PL/I by internal procedure. They are stored with their name and usage type. This step too is part of the CodeTran preprocessor job COBPrep or PLIPrep.

The data reference table is necessary to recognize later which data attributes are used by which methods and to eliminate those data items not used. This is very important for large mainframe programs containing many data which are never used. There is no reason whatsoever to carry them over into the new Java classes where they would only bloat the code. Since each data item will have its own address and its own set and get operations there is no need to keep the unused data just to maintain the displacements of the data.

2.6. Program Data Table Generation

The third step in the code transformation process is to generate a data description table for each data structure. This data table orders the data by program, storage class, structure and position. For each data item, the name, the type, the picture, the length, the decimal position, and the value are given. In addition, it is marked which data redefines which others. This table is used later for checking the displacements and lengths of the generated Java object structures. It is also intended to be used as a basis for testing the converted programs against the original ones. The test driver uses it as a reference to generate test data and to validate the test results, thus assuring functional equivalence of the Java component with the COBOL or PL/I program. The basis of comparison is the common data table.

This is the last step in the CodeTran preprocessor. After it, follows the actual transformation of the code into object-oriented Java.

2.7. Java Class Generation

The fourth step in the code transformation process is the generation of the Java classes. Here the data declarative part of the COBOL or PL/I program is analyzed and for each data record a class created. The class will have a class header, a class definition which extends the COBOL or PL/I master class, an object defined as a character array, a set and get operation for each data field within that array, a constructor and an initiator operation. In creating the class, the class generator filters out all data items not used. For this it has to access the data reference table. Itself, it creates two additional work tables, one for linking data attributes to classes and one for linking data elements to the paragraphs and procedures in which they are used. These two tables are needed later for the method generation and documentation.

The set and get operations inherit the data conversion functions from the standard COBOL or PL/I master class. Each time a data is accessed, it is converted from a character string to that data type required by Java, i.e. a String object, an integer or a real number. Each time a data is returned, it is converted back from the Java data type to the character array. The constructor allocates the character array and the initiator initializes the individual fields to their original values. In case there was no initial value the numbers are set to zero and the character fields to spaces. A place holder marks where the processing methods are to be inserted.

Besides creating a class for each data structure, the class generator also generates storage classes which are containers of the structures belonging to each storage class. The storage classes are language dependent. In COBOL the storage classes are the file storage, the working storage, the linkage storage, the database cache storage and the communication storage for the terminal input/output. In PL/I the storage classes are the file storage, the static storage, the automatic storage, the based storage and the procedure parameters. This intermediate class level is intended to help in classifying the elementary classes. The concept of a storage class is intended to group the elementary classes into a higher order abstraction, thereby reducing the number of classes at the same hierarchical level.

2.8. Java Method Generation

The fifth step in the code transformation process is the generation of the Java methods. Here the procedural part of the COBOL or PL/I program is analyzed and for each paragraph or procedure a method generated. The method code itself is converted on a statement by statement basis. The GOTO operations are converted to returns with the target label as the next method to be invoked. For the calls the output parameters are marshaled into a single result object. The database access, screen and I/O operations are converted to interfaces to the standard JDBC database access operations, the standard Java GUI operations and the standard Java file operations. For all of the other statement types – conditions, assignments, artithmetic operations, etc. - equivalent java statements are generated. Optionally, the original COBOL or PL/I statements can be left as comments in the code.

Before each method a comment header is created indicating which data attributes are used by that method and in what way, as input, as output or as both. The data attributes are associated with their classes and their types. This header documents the data flow. After each method a comment trailer is inserted indicating which methods are invoked and which methods can follow upon this method. The intention here is to document the control flow. In case of legacy code with many GOTO branches, this can be very important.

Another important function of the method generator is the assignment of the methods to classes. In processing the data references of a method the data2class table is accessed to identify to which class the referenced data belongs. That class to which the most referenced data belong is the class to which this method is assigned. If two classes have the same number of data referenced, then the class with the least number of methods is selected. The method2class assignments are recorded in a binary relation table - the method2class table. In addition, a list of all methods and method invocations – Performs in COBOL and internal Calls in PL/I – is generated. The purpose of all this is to ensure locality of reference. Methods should be kept together with the data they use. This is perhaps the most important aspect of object-oriented programming as opposed to procedural programming where the data is all kept together in one global storage area.

.

2.9. Merging the Methods with the Classes

The sixth step in the code generation process is the merging of the methods with the classes. When generating the classes, it is marked where the methods should be inserted. When reading the classes it is known from the method2class table which methods belong to what class. There may be classes with no methods. There may be other classes with many methods. It all depends on the density of reference. If a particular object, i.e. data structure, is referenced a lot, it will have many methods. If, on the other hand, it is hardly referenced at all, it will have no or few methods.

In merging the classes and methods some other features are also added. The storage classes which exist between the elementary classes and the super class, PL/I or COBOL, are completed here. If in COBOL sections are performed, then container classes are created here to include calls to all the methods contained within the performed sections. Finally, all references to methods are enhanced by the storage class and the structure class labels. Every method is referred to via the program name, the storage class and the structure class to which it belongs.

After the merging step there will be one big source file for each component = COBOL program or Java external procedure. This source files contains all the structure classes with all of their data attributes and methods plus the storage classes, the super class and, in COBOL, some control classes for invoking sections. The super class will contain all of the standard data conversion functions plus a recursive function for processing the GOTO references – the control flow driver.

2.10. Creating an XML Documentation of the Component Structure

The seventh step in the code generation process is the creation of an XML document from the Java component file. The XML file displays the hierarchy of the component listing out all classes belonging to that component, the data attributes of each class and the methods of each class with their inputs and outputs as well as their successor methods. In this way, in addition to the component structure, both data flow and control flow are documented in XML format.

2.11. Cutting up the Classes into separate Source Files

The remaining steps in the Java code generation process are carried out by the post processor. The eighth step is to cut up the single source components into a source file for each individual class and to collect them into a Java package library. At the same time some refinements are made to the Java code in order to make it compilable.

2.12. Linking the Subordinate Classes to the Superordinate Classes

The ninth step in the Java code generation process is the creation of a subprogram library and the linking of the sub programs to the main components. COBOL and PL/I programs can refer to other external programs and exchange data with them via parameter lists. These external references are resolved at linkage time by a linkage editor which connects the programs to one another via physical addresses. This is not possible in Java. In Java all references have to be resolved at compile time, in order to keep the compile units portable to any machine. For this reason CodeTran must fulfill the role of the linkage editor and satisfy the external module references with application program interfaces (APIs) in which the parameters are passed over to the called classes and the results returned.

The APIs and their underlying sub components are stored in a separate reusable library made available to all the other components. Thus, they can be invoked at run time from any component. This brings up the problem of concurrency, when two different components use the same sub component at the same time. This is resolved at the moment by locking the sub component whenever it is called. It is for the time being the task of the user to resolve this concurrency issue. CodeTran does not do it for him.

2.13. Creating a JavaDoc Repository

The final step in the Java code generation process is the creation of a JavaDoc repository. One of the purposes of the CodeTran transformation is to document the code. This is done by using JavaDoc. For each migration package a separate JavaDoc repository is generated which includes all the components and classes of that component, all of the data attributes and all of their methods with cross references between them. The user can then use the standard JavaDoc functions to view them at the component level, the class level and at the method level.

2.14. Compiling the converted Java Classes

What follows are steps not supported by CodeTran. The user has to compile the classes generated by CodeTran and to build the Java components. This depends on the environment he is working in. If he is working with Eclipse, he will use Eclipse to compile and administer the new Java code. If errors occur in the compilation he has the choice of correcting them manually or having the transformation tools corrected and repeating the whole process for the effected component. If the compile errors are local it might be better to correct them locally, where they occur. If they are global it is better to correct the tools and repeat the process. Of course this option assumes that the user is in a position to correct the tools. It may be better to rely on the tool developers to do this for him.

2.15. Generating a Test Driver to test the converted Component

One can never assume that such a complex code transformation will be without error. There will always be some errors that go undetected in the transformation process. To filter them out, the generated Java components have to be tested against the original COBOL or PL/I programs. To do this a test driver is required which will generate test data objects to feed to the program under test based on the data structures of that program. It must also be able to capture the test results based on the data structure descriptions and to make those results visible to the tester. The data for the regression test driver can be automatically generated out of the data description table created by the CodeTran preprocessor. Another tool SoftVer is designed to compare the logical paths thru a new component with the same paths thru the original COBOL or PL/I procedures. Should this approach not be used, then the user has to create a test driver to invoke the Java component, feed it data by means of stubs for the files and databases and to capture its outputs for comparison.

2.16. Testing the converted Java Classes

The final step in the overall migration process is to run the regression tests to validate the functional equivalence of the generated Java components with the original COBOL and PL/I programs. This is accomplished by feeding them the same input data and then comparing their outputs. These should be identical down to the single byte. To make the comparison the DataTest tool offered by ANECON can be used. It downloads the databases to CSV files and compares them record by record and field by field. Any fields which do not match are recorded in an exception report. The GUI outputs and output messages are converted to XML files and these are compared on an element by element basis. The non identical fields are recorded in the same way as the CSV fields. The user is given an exact report on the status of the component under test.

The component regression test needs to be repeated until all the key results are identical and all major errors have been removed. Then a component can be considered to be correct and can be integrated into the new Java architecture.

3. Inputs to the CodeTran Tools

The inputs to CodeTran are actually the outputs of SoftRedo without the C++.

- Assembler Programs & Macros
- PLI Programs & Includes
- COBOL Programs & Copies

(*see figure 3: CodeTran Inputs*)

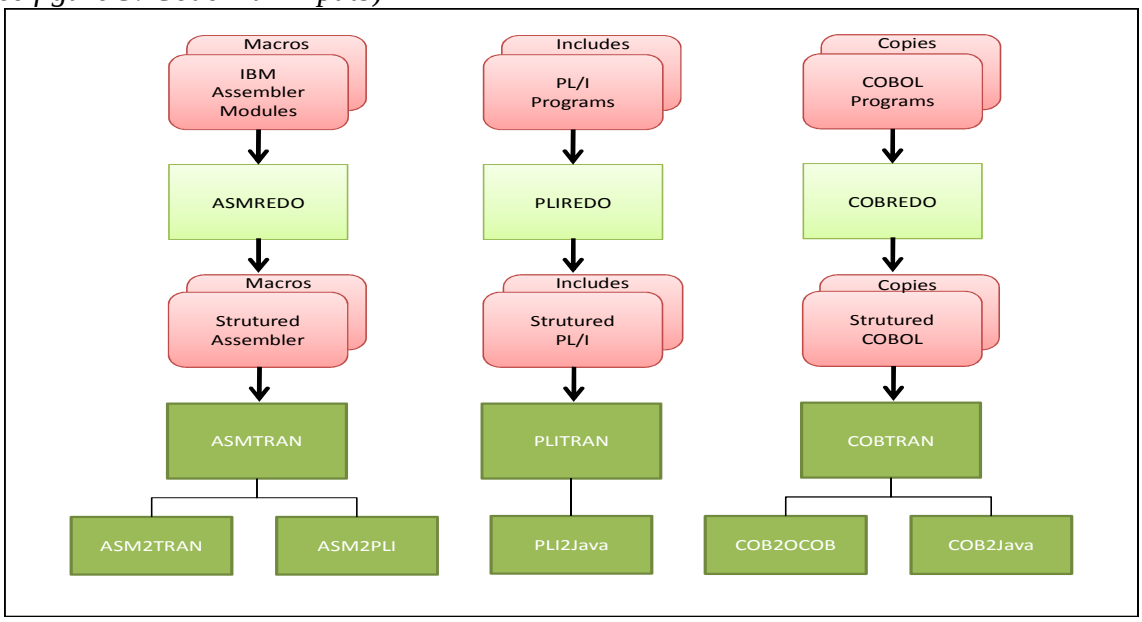


Figure 3: CodeTran Inputs

3.1 Assembler CSects & Macros

The original Assembler sources will contain standard IBM Assembler commands, enhanced by the vendor specific and user specific macros. These macros must be included in the macro table. The source should be in a standard MicroSoft text format and have one statement per line. The line length is 72 characters. The statements in Assembler have the standard layout: label, operation, operands. Comment lines are marked with a * in the first column. Continuation lines are indicated by a – in the last column.

The renovated Assembler sources should have the following characteristics:

- The source text has been reformatted with three columns. The first column contains the labels. The second column contains the operations. The third column contains the comments. All columns are aligned to start on the same position.
- Entry and Exit operations such as BR 15 will have been replaced with standard macros and physical addresses will have been replaced by logical ones using symbolic names in place of physical locations with displacements
- Text literals longer than 4 characters in Assembler operations have been replaced with a symbolic constant LIT999 and the symbolic constant inserted as a defined constant at the end of the source.
- Should there have been any file IO operations or database accesses (DLI, DB2) or teleprocessing operations (CICS, IMSDC), these will have been relocated to subroutines in an IO section at the end of the source.

- Forward branches have been converted to IF..ELSE macros and backward branches to WHILE loops. All branches are marked with comments. In this respect the code has been both restructured and annotated.

The renovated Assembler programs should first be compiled and the syntax errors corrected before starting to convert them to either COBOL or PL/I. Otherwise, the syntax errors will be carried over into the COBOL/PLI code. *(see Sample 6.1: Reengineered Assembler Source)*

3.2 PL/I Procedures & Includes

The original PL/I sources will contain standard IBM PL/I statements, enhanced by vendor specific macros and user defined functions. Typical vendor specific macros are EXEC-CICS, EXEC-DLI and EXEC-SQL macros which are embedded in the PL/I code. These macros and user defined functions must be included in the macro table. The source should be in a standard MicroSoft text format and have no more than one statement per line. The maximum line length is 80 characters.

The renovated PLI procedures will have been altered as follows:

- Should the PL/I procedures contain references to include members with procedural code, these procedural include texts will have been inserted into the main line procedural code. For this reason the user must assign the procedural copy library.
- The source has been realigned so that nested code is indented by two columns and lines with two or more statements are split. The comments in the code statements will be in separate lines before the original line where they were. This makes the code easier for the maintainer to read and easier for CodeTran to process.
- The external data declarations will have become parameters to the procedure, the Begin blocks will have been converted to Do blocks, the label variables removed and the Other clause will have been inserted into the Select statements. Besides the code blocks will be separated by comment lines and the Leave statement marked.
- If the user has selected the renaming function, the short names will have been replaced by long names.
- All of the binary, floating and packed data types will have been converted to decimal numeric and the bit types to bytes.
- If the user has selected the replacement function, the text literals and numeric constants will have been replaced by symbolic constant names and an include member with these constant data values will have been created. The Include statement for it is inserted right after the procedure declaration.
- If the user has selected the relocate function all IO operations, all database accesses (IMS-DB, DB2 and ADABAS) and all the teleprocessing operations (IMS-DC and CICS) will have been relocated to internal procedures after the main procedure. They are invoked by a call from their former location.
- If the user has selected the refactoring function, the nested code beyond the maximum nesting level specified will have been extracted and moved to a sub procedure at the end of the source. It is called from its original location.

The renovated PL/I programs should first be compiled and the syntax errors corrected before starting to convert them to Java. Otherwise, the syntax errors will be carried over into the Java code. *(see Sample 6.2: Reengineered PLI Source)*

3.3 COBOL Programs & Copies

The original COBOL sources will contain standard COBOL-85 or COB-2 statements, enhanced by vendor specific macros. Typical vendor specific macros are EXEC-ADABAS, EXEC-CICS, EXEC-DLI and EXEC-SQL macros which are embedded in the COBOL code. These macros must be included in the macro table. The source should be in a standard MicroSoft text format and have no more than one statement per line. The maximum line length is 72 characters.

The renovated COBOL programs will reflect the following changes:

- Should the COBOL Procedure Division contain references to Copy members, these procedural copy texts will have been inserted into the main line COBOL code. For this reason the user must assign the procedural copy library.
- The source text has been realigned so that nested code is indented by two columns and lines with two or more statements are split. This makes the code easier for the maintainer to read. In addition, comments between paragraphs are placed inside the paragraphs to enable the paragraphs to be relocated without losing the comments.
- The obsolete COBOL-74 statements such as Exhibit, Examine, Alter, Note, etc. will have been replaced, the Next Sentence commands will have become IF statements, the GOTO depending on constructs will be Evaluate statements, the Perform Thrus will be expressed as a sequence of Performs and all If statements will be closed out by End_Ifs. In addition, the code blocks, i.e. paragraphs, will be separated by comment lines.
- If the user has selected the renaming function, the short names will have been replaced by long names.
- If the user has selected the remove function, the packed data types and the binary data types will have been converted to decimal numeric.
- If the user has selected the replacement function, the text literals and numeric constants will have been replaced by symbolic constant names and a copy member with these constant data values will have been created. The copy statement is inserted at the end of the Working-Storage Section.
- If the user has selected the relocate function all IO operations, all database accesses (IMS-DB, DB2 and ADABAS) and all the teleprocessing operations (IMS-DC and CICS) will have been relocated to an IO Section at the end of the program. They are invoked by a Perform from their former location.
- If the user has selected the refactoring function, the nested code of a paragraph beyond the maximum nesting level specified will have been extracted and moved to a separate Section at the end of the source. It is performed from its original location.

The renovated COBOL programs should first be compiled and the syntax errors corrected before starting to convert them to Java. Otherwise, the syntax errors will be carried over into the Java code. *(see Sample 6.3: Reengineered COBOL Source)*

3.4 Macro/Function Table

The macro/function table contains a row for each macro or function with 3 columns. For the procedural languages Assembler, PL/I and COBOL the rows are macro operations. For the languages C and C++ the rows are standard functions for I/O operations.

The first column is a four letter code for the type of macro or functions. The second column is an 8 or 40 character string giving the name of the macro or function. The third column is used only by the tool ***SoftAudit***. It is the number of function–points associated with each I/O operation or file type. It can range from 3 to 6 for Input operations, from 4 to 7 for Output operations, from 5 to 10 for Files and from 7 to 15 for Databases.

In the case of macros the name of the macro is only 8 characters, followed by the number of parameters for the macro and the position of the parameter with the name of the object being processed by the macro, i.e. the file, record datagroup or map.

READ GETNEXT, PARAMS = 3, OBJECT = 1

The 25 valid macro or function types are:

- CALL = A procedure or method is called
- CLOS = A file or database ir opened
- DB = A database is declared
- DECL = A datatype is declared
- DELT = A file or database record is deleted
- ENTR = An entry to the module
- EXIT = An exit from the module
- FILE = A file is declared
- FUNC = A function to compute a value
- GETS = A data storage is allocated
- INCL = Another source is copied in
- INPT = An input operation
- ISRT = A database, record is inserted
- LIST = A report is printed
- MASK = A user interface is declared
- MESS = A message is declared
- OPEN = A file or database is opened
- OUTP = An output operation
- PROC = A procedure is declared
- RECV = A user interface or message is received
- READ = A file record is read
- SELT = A database record is queried
- UPDT = A database record is updated
- WRIT = A file record is written
- SEND = A user interface or message is sent.

A sample macro table for Assembler and a sample macro table for CICS are given in *Sample 6.4 – Assembler Macro-Table.* It is not possible to identify the input/output, DB and TP operations without the help of such a table. CodeTran needs it to identify and replace the IO, DB and TP operations. In Assembler these operations are user defined. Therefore it is up to the user to define them also for the reengineering tool. PL/I and COBOL have standard IO operations, but may be using a special macro language such as ADABAS, IMS or CICS. A macro table is required for processing those languages.

3.5 Data Name Tables

If the user wants to have the long names of the data placed in the comments, he should make a name table available. This can be the same name table as was used by SoftRedo to rename the data variables. This table can be made with Excel. It contains two comma separated columns. In the first column is the original short name of a variable. It may be no longer than 20 characters long. In the second column is the new long name. It can be up to 30 characters in COBOL and 32 in PLI. The format of the table is as follows:

<shortName>;<longName>
<shortName>;<longName> (*Sample 6.5 – Data Name-Table)*

There can be several tables for each set of programs to be converted. In any one table there can be any number of rows. Creating the tables is a responsibility of the user, but he may use the comment lines attached to the data declarations as a basis for creating the new long names.

3.6 GUI Parameters

The ***CodeTran*** user interface gives the user the opportunity to set the control parameters and to select which reengineering functions should be performed.

3.6.1Control Parameters

The Control parameters to be submitted by the user include the following names and directories:

- Name of the product to be converted
- Name of the system to be converted
- Output directory for storing the converted sources
- List of source files to be processed
- List of name tables to be used when renaming the data
- Language of input (Assembler, PL/I, COBOL)
- Language of output (PL/I, COBOL, OO-COBOL, Java)
- List of Options for the transformation.

3.6.2Transformation Options

The transformation options are:

- Keep original statements as comments
- Generate components with a built-in test frame
- Generate an XML documentation
- Generate a JavaDoc documentation
- Generate SQL Database Access Functions to simulate
  - ADABAS DB accesses
  - IDS/IDMS-DB accesses
  - IMS-DB accesses
- Generate File Access Functions to simulate

    - SAM/ISAM file accesses
    - VSAM file accesses
- Generate GUI User Interface Functions to simulate
    - COBOL screen processing operations
    - Bull Input/Output Messages
    - IMS-DC Input/Output operations
    - CICS Input/Output operations

The input parameters are collected together in an XML interface message which acts as a link to the batch processes. Should the user want to run CodeTran in another environment other than Windows, he can create his own driver and still use the underlying batch processes by setting up this XML interface and invoking the individual CodeTran processes. In this way it is even possible to run CodeTran on the mainframe. (*Sample 6.6 – Batch Process Interface)*

4 Outputs from the CodeTran Tools

The outputs from CodeTran are converted source files in the languages:

- COBOL programs from Assembler modules
- PL/I procedures from Assembler modules
- COBOL copies from Assembler copies
- PL/I includes from Assembler copies
- Java components from PL/I or COBOL programs
- OO-COBOL components from COBOL programs.

In addition to these source outputs CodeTran also produces four types of documents:

- Data Description Table
- Data Reference Table
- XML Component Structure Design
- JavaDoc Repository.

(*see Figure 4: CodeTran Outputs*)

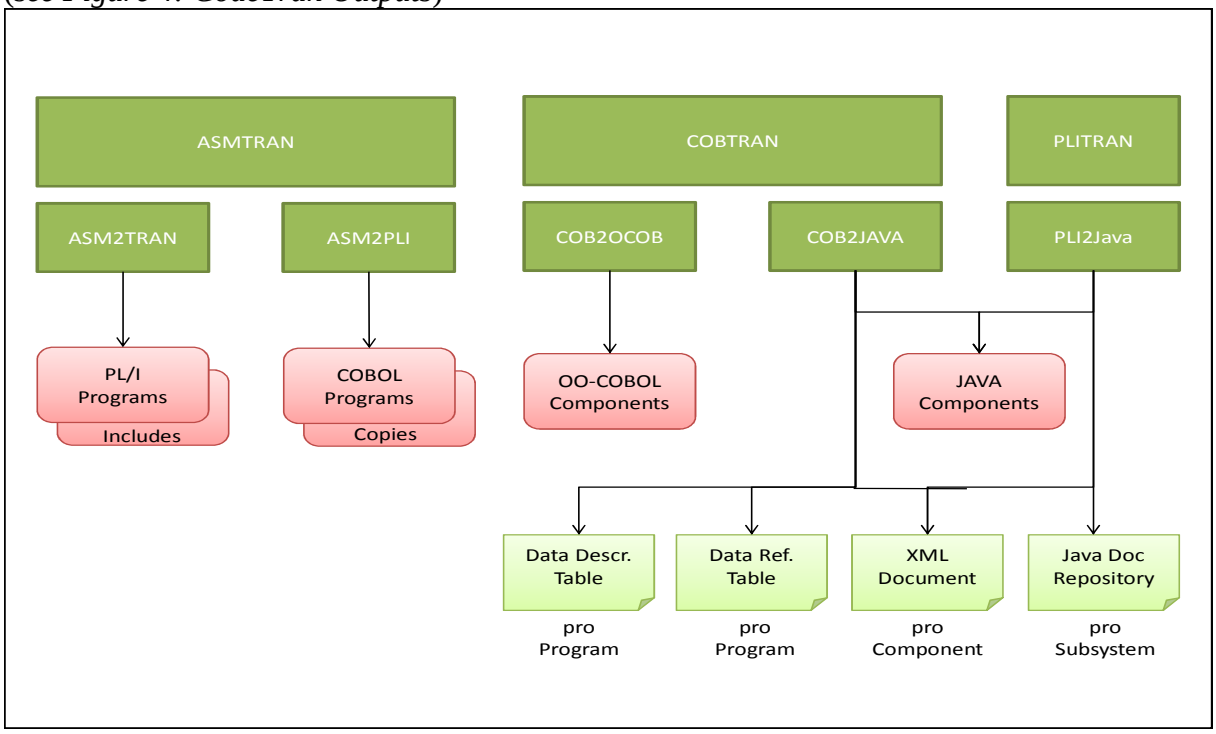


Figure 4: CodeTran Outputs

4.1 COBOL Programs from Assembler Modules

The COBOL programs produced from Assembler modules have a COBOL-74 syntax. They have an Identification Division with the program name and a skeleton Environment Division containing a Select statement for each file assigned in the Assembler program. The Data Division contains a picture declaration for every Assembler data definition. The Assembler DS = Define Storage variables are converted to variables with a data type corresponding to the Assembler type. The Assembler DC = Define Constant fields are converted to constants with a value taken from the Assembler initial value. Record structures

are extracted from the DSECTS = Data Sections and the ORG definitions. Overlay structures are declared as Redefines. The Working Storage Section is divided into two parts – the constant data and the variable data. The Linkage Section contains the parameters of the Assembler program. The user has an option whether to keep the original assembler commands as comments or to remove them. Thus, there are two output versions of the converted COBOL program, one with the original assembler statements and one without them. The former has the extension .COB, the latter has the extension .CBL. Both versions contain the following divisions and sections:

- Identification Division with the program name and version
- Environment Division with an Input-Output Section and a File Control Section with a select for each file declared.
- Data Division with a File Section containing the File Descriptions of all files, a Working-Storage Section with the constant data declarations followed by the variable data declarations, and a Linkage Section with the parameters received by the program.
- Procedure Division with a paragraph for each former Assembler label containing the commands that followed that label.

The COBOL program produced by the conversion should be at least 80% complete. The remaining 20% has to be manually inserted by the COBOL programmer. *(see Sample 6.7: Converted COBOL Program)*

4.2 PL/I Procedures from Assembler Modules

The PL/I procedures generated from Assembler modules have an IBM-PL/I syntax. They begin with a procedure statement in which the parameters to the original Assembler program are declared. This is followed by the Entry declarations to foreign modules called by this module, after which pointers for the Assembler registers are declared. Following the pointer declarations come the constant data and then the variables. Thus, all of the data is declared before the first procedural statement. The procedural statements are grouped by the labels taken from the original Assembler module. As with COBOL the Assembler DC = Define Constant fields are converted to init constants with a value taken from the Assembler initial value. The Assembler DS = Define Storage variables are converted to variables with a data type corresponding to the Assembler type. Record structures are taken from the DSECTS = Data Sections and the ORG definitions. Overlay structures are declared as Defined variables. The data are collected together in a single static data area. This is subdivided into two parts – the constant data and the variable data. The parameters of the Assembler module remain the parameters of the PL/I procedures. The user has an option whether to keep the original assembler commands as comments or to remove them. Thus, there are two output versions of the converted PL/I program, one with the original assembler statements and one without them. The former has the extension .PL1, the latter has the extension .PLI. Both versions have the following content:

- Procedure Declaration
- Entry Declarations
- Pointer Declarations
- Constant Data Declarations
- Variable Data Declarations
- Procedural statements grouped by label containing the commands that followed that label.

The generated PL/I program may not be directly compiled and executed. It will at most be 75% finished so it must first be manually completed. Only then can it be converted over into Java. *(see Sample 6.8: Converted PL/I Program)*

4.3 COBOL Copies from Assembler Copies

The COBOL Copies converted from Assembler copies have a header followed by a 01 Group declaration and a COBOL Picture declaration for every data field in the original Assembler copy. The data definitions are converted on a 1:1 basis. Only the data types are changed to COBOL types. The names remain as they were. *(see Sample 6.9: Converted COBOL Copy Member)*

4.4 PL/I Includes from Assembler Copies

The PL/I Includes converted from Assembler copies have a header followed by a 01 Group declaration and a PL/I data declaration for every data field in the original Assembler copy. The data definitions are converted on a 1:1 basis. Only the data types are changed to corresponding PL/I types. The names remain as they were. *(see Sample 6.10: Converted PL/I Include Member)*

4.5 Java Components from COBOL or PL/I programs

The Java components generated from the COBOL or PL/I programs have the same framework and similar contents. The result of the conversion with CodeTran is a Java library for each converted subsystem with two common sub libraries and a sub library for each Java component = converted PL/I or COBOL program, in that subsystem. The two common sub libraries are:

- Frame and
- Programs.

4.5.1 Java Framework

The Java Framework – Frame – contains seven standard classes:

- Controller
- ProcessingInfo
- Component Storage
- Component Interfaces
- DatabaseAccess
- COBOL/PLIObject and
- TestDriver

4.5.1.1 Controller

The controller class is an abstract class for all programs. It defines all of the storage areas and contains the method RunStateMachine, a recursive procedure for driving the control flow of the component. It determines which method, in which class will be executed next. This is necessary to resolve the GOTO branches in COBOL and PL/I programs. *(see Sample 6.11: Java Controller)*

4.5.1.2 ProcessingInfo

The processing information class is a class to supply the controller class with the names of the next methods to be executed. The objects of the class are the method names. It also provides the qualifiers of the methods, i.e. their data class and storage class names. As such this class can be viewed as the concrete implementation of the abstract controller class. *(see Sample 6.12: Java Processing Information)*

4.5.1.3 Component Storage

The component storage class is an abstract class for allocating instances of the storage classes. There will be one instance of each storage class type, such working storage, automatic storage, cache storage, etc. *(see Sample 6.13: Java Storage Classes)*

4.5.1.4 COBOL/PLIObject

The COBOL or PLI Object class is a super class containing all of the standard methods inherited by the data classes. These are in the majority data conversion methods, e.g. methods for converting character strings into numeric values and for converting numeric values back into character strings. There are also methods for computing the addresses of indexed array elements and for setting pointers to data addresses. *(see Sample 6.14: Java COBOLObject)*

4.5.1.5 Component Interfaces

The component interface class contains an interface definition, i.e. the signature, to each foreign component called within this subsystem. The call is declared together with the parameters - arguments passed and results received. Not only is the interconnection between components within a subsystem but also the interconnection of components across system boundaries enabled by this container class. *(see Sample 6.15: Java Component Interfaces)*

4.5.1.6 Database Access

The database access class contains standard operations for accessing an SQL database – open, create, read, update, delete, - as well as exception handling operations for handling the database return codes. All data classes which access databases inherit these same standard accessing and error handling methods. The methods may vary depending on the type of database system converted. *(see Sample 6.16: Java Database Access Class)*

4.5.1.7 Test Driver

The Test Driver class contains generic test data objects and test methods for reading, writing, sending and receiving the test data flowing to and from the component under test. The methods are inherited by the individual test drivers generated for each component. *(see Sample 6.17: Java Test Driver)*

4.5.2 Java Programs

The Java Programs library contains a super class for each Java component. This super class allocates an instance of each storage class and handles the exception conditions invoked by unsuccessful storage allocation. It also contains stub methods for those methods not resolved within the data classes by the Java converter. Finally, it controls the sequence in which multiple methods are performed in COBOL, i.e. PERFORM THRUs. This is a very extensive process, which is not required for PL/I so that these classes are much smaller for PL/I.

4.5.3 Java Components

The sub directory for each converted Java component contains a storage sub sub directory and a set of all data classes belonging to this component. The storage sub directions contains a storage class for each storage type. The storage class allocates a static singleton object for each storage class. The six storage classes for COBOL are:

- CONTROL
- CACHE
- WORK
- LINKAGE
- DATABASE
- DATACOM.

The six storage classes for PL/I are:

- PROCS
- LOCAL
- STATIC
- BASED
- DATABASE
- DATACOM.

(see Sample 6.18: Java PLI Storage Classes)

4.5.4 Java Data Classes

The data classes all have a similar structure. They begin with a package assignment, imported classes and a class header, followed by the object definition and allocation. Then follows the attribute definitions and after that the converted methods. The exact contents of a data class are as follows:

- Package assignment
- Imported classes
- Class Header
- Class Declaration
- Static class object allocation (Constructor)
- Global methods inherited

- Class attributes with a get and set method for each attribute
- Initialization of the attribute values (Initiator)
- Class methods

Each attribute definition includes a description of the original COBOL or PL/I data type as well as the current position and length of this field within the character array allocated. The get and set methods refer to the data conversion routines in the base class. The initialization method sets the data fields to those values assigned in the original program or to default values like space and zeroes. *(see Sample 6.19: Converted COBOL class and Sample 6.20 Converted PLI Class)*

Each method has a method header in which the inputs and outputs of the method are identified by attribute name and class, a method body containing the converted Java statements and a trailer in which the successor methods of this method are identified and how they are reached, e.g. by call, goto, fall thru, etc. Within the body of the method the Java statements can be proceeded by the former COBOL or PL/I statements if desired. Should there be a problem in converting a statement, then that statement will be followed by a comment requesting further examination. This is to ensure that no conversion defects go unnoticed. *(see Sample 6.21: Converted COBOL method and Sample 6.22: Converted PL/I method)*

4.6 OO-COBOL Components from COBOL programs

The OO-COBOL programs produced from COBOL-85 retain their original basic COBOL syntax but have an object-oriented architecture with the OO-COBOL language extensions. As opposed to the COBOL to Java transformation where many small sources are spawned from the original program source file, here there is a 1:1 transformation from source to source. However, internally the source is structured very differently. It is cut up into classes and methods. There are several additional command types introduced by COBOL-96, e g. the CLASS-ID with an inherits clause and END-CLASS, the METHOD-ID with END-METHOD and EXIT METHOD. There is also an OBJECT Section for allocating one or more objects. This can be referred to as the object factory for producing objects any time the class is referred to by a “new” invocation.

The Program-ID becomes the Class-ID and references to classes inherited are inserted. The former records or first level structures become the objects of the program. For each such object an object reference is created. The level 77 elementary data fields are collected together in an artificial structure to become an object. Data structures with an Occurs clause can produce multiple instances. The other data structures remain as static singleton objects which are allocated at the start and freed at the end. There is a single source for each program or component with nested classes, one for each data object. In the converted program the classes are only used as data containers. The procedural part of the program is assigned to a single processing class containing all of the methods. The methods of this super class have access to all of the data classes. This may violate the principle of “Locality of Reference” followed so closely by the Java transformation, but it simplifies the transformation and reduces the code size which grows anyway because of the additional OO statement types.

The procedure division belongs to the master class and contains multiple nested procedures, each with its own header, Data division and Procedure division with entry declaration. The parameters to a method are defined in the Linkage Section of that method. If

the COBOL program had contained GOTO branches before then it will have a label variable X-NEXT-LABEL to indicate which method is to be invoked next. At the beginning of the program there will be a constructor method which invokes the allocation of all objects. This is followed by a loop in which the other methods are invoked depending upon the value of the X-NEXT-LABEL variable until the termination condition is met. After the loop come the processing methods in the order in which the former paragraphs were. Each method corresponds to a paragraph of the former COBOL program. The method names are the paragraph names. A destructor method follows the last processing method, which invokes the deallocation of the objects.

Each method has its own data and procedure divisions. In the data division the local data and parameters of the method are declared. In the procedure division there are the same commands as had been in the previous paragraphs, only the GOTOs have been replaced by the setting of the next label variable and the Perform statements have been replaced by INVOKE statements. Except for the INVOKE command, the statements correspond to the COBOL-85 or COBOL-2 standard.

In summary, the converted OO-COBOL program will have the following contents:

- Identification Division
- Environment Division
- Data Division
- Nested Classes
- Procedure Division
- Nested Methods

(see Sample 6.23: Converted OO-COBOL Class)

4.6.1 Identification Division

The Identification Division contains the CLASS-ID and the original module header as well as a new header with the conversion version and date.

4.6.2 Environment Division

The Environment Division contains an INPUT-OUTPUT Section with the original file selection statements and a CONFIGURATION Section with a REPOSITORY or FACTORY declaration denoting the role of this class and one or more INHERITS clauses for multiple inheritance.

4.6.3 Data Division

The Data Division contains a FILE Section for those objects read from or written to a file or printer. Each file is defined as a class with an instance for each record type. This section is followed by a WORKING-STORAGE Section. Here the Object references are declared at the beginning after which the internal data structures are declared beginning with a CLASS-ID and ending with an END-CLASS statement. Should a structure contain a repeating group it will be factored out as a separate class. Elementary variables with multiple occurrences and redefined fields remain as they are. For the repeating data groups multiple instances will be allocated. Since every method has its own parameters there is no Linkage Section. This global Data Division is accessible to all methods.

4.6.4 Nested Classes

As pointed out above each level 01 data structure will become a nested class accessible to all methods. It will have its own CLASS-ID, Environment and Data Divisions but no Procedure Division. In the Environment Division there will be a Repository and an Object statement for allocating the object data. The Data Division contains the data declarations carried over from the original program. The class is ended by an END-CLASS statement.

4.6.5 Procedure Division

The Procedure Division contains a series of nested methods. The first method is a constructor method which invokes the allocation of all objects. The second method is a control method which invokes the other methods depending on the value of the X-NEXT-LABEL variable. It runs in a loop until the termination condition is fulfilled. The third method is a destructor method which deletes all existing objects. If an object state is to be saved, it must be written out to a file the before the termination is reached. After it follow the processing methods taken from the former paragraph that are invoked by the control method. Since they are invoked from above depending on the current state of the X-NEXT-LABEL variable, there sequence does not really matter.

4.6.6 Nested Methods

Each nested method begins with a METHOD-ID and ends with an END-METHOD. In between there is a Data Division and a Procedure Division. The Data Division has a Working-Storage Section and a Linkage Section. The Working-Storage Section contains only an X-COUNT variable for counting the number of times this method is invoked. The Linkage Section contains the parameters to the method – the X-NEXT-LABEL and the X-RETURN-CODE. The procedure division begins with an Entry Declaration giving the parameters to the method. After it follow the procedural statements of the original COBOL paragraph without the GOTOs and with INVOKE in place of PERFORM. Since the methods access the data defined in the classes of the global data division, the data names have to be qualified by the class names, just as in the converted Java programs. This is a good practice anyway as it makes the code more secure and better readable. *(see Sample 6.24: Converted OO-COBOL Method)*

4.7 Data Description Table

The data description table contains all of the data fields defined in the original COBOL or PL/I Program. There is a row for each data item and a column for the following attributes:

- Usage Type = C,I,O,P (C is as a Condition, I is as an Input, O is as an output, P is as a parameter)
- Storage Section (e.g. WORK, FILE, LINKAGE)
- Displacement of the data item within the structure
- Hierarchical level of the data item
- Data item name
- Data item picture

- Data type
- Data length
- Data fraction length
- Data occurrence factor
- Redefined data field.

This table has a dual purpose. One is to check the data positions and lengths within the class definitions. The other is to generate test data for testing a converted component. *(see Sample 6.25:Data Description Table)*

4.8 Data Reference Table

The data reference table indicates which data items are used in what way by the original program. It is ordered by code block as the code blocks appeared in the original program. Code blocks can be Sections or Paragraphs in COBOL and Procedures or Labels in PL/I. To the right of the code block name are the usage types and the data names used by that code block. There are altogether only four columns:

- Code block Type = P,L,S,M (P is Procedure, L is Label, S is Section, M = Paragraph)
- Code block Name
- Data Usage Type = C,I,O,P (C is as a Condition, I is as an Input, O is as an output, P is as a parameter)
- Data Name (can be qualified by structure name if not unique)

This table too has a dual purpose. One is to check the data usage definitions within the generated methods. The other is to trace the data flow when testing a converted component. *(see Sample 6.26:Data Reference Table)*

4.9 XML Component Structure

For each Java and OO-COBOL component converted an XML document is generated describing the structure of that component. The structure is depicted in the form of a hierarchy. At the top of the hierarchy is the component itself. Under it come the classes with a start and end tag. Within the classes the attributes are listed out as elementary definitions. After the classes come the methods. The methods have a start and an end tag. Between these tags are the input/output data used by the method and the successor methods which are invoked during and after the method is running. With this document the user can follow both the data flow and the control flow and see where and how the data are defined. It replaces the old program documentation should there be one. *(see Sample 6.27: XML Component Documentation)*

4.10 JavaDoc Repository

For each Java component converted a JavaDoc Repository is created. The Javadoc Repository can be found in the directory doc. This directory contains a subdirectory at for the individual Java components as well as for a number of super ordinate HTML Files. These are as follows:

- AllClasses-Frame.html with a list of all classes in this subsystem plus the storage classes of the target language.
- AllClasses-NoFrame.html with the same listing

- Constant-Values.html with a list of all the global constants and literals contained within the subsystem.
- Deprecated-List.html with a list of all the API Definitions.
- Help-Doc.html with the user instructions for the JavaDoc Tool
- Index.html with a list of all Packages, i.e. components of the subsystem
- Overview-Frame.html with a list of all Packages or Components with links to the classes they contain.
- Overview-Summary.html with a complete table of contents for each component including its classes, attributes and methods.
- Overview-Tree with a hierarchical structure of the Components, Super classes, storage classes and Data classes.
- Serialized-Form.html with a linked list of all elements of the subsystem – components, classes, attributes, methods and interfaces.

The sub directory „at“ has another sub named „<system>\Cobol2java“ in which all of the components or Java packages of this sub system are listed out. Under each component can be found a HTML file with a specification of each class within that component. The class specification consists of a class header, a list of all class attributes and a description of each method of that class with its input and output variables and its potential successor methods. Via the HTML Links it is possible to trace the data flow from class to class and the control flow from method to method within the component. *(see Sample 6.28: JavaDoc View of a Class)*

5 CodeTran Usage

The tool ***CodeTran*** offers the user two different screens for the transformation operations. The one is the Transformation screen for selecting the files to be converted, setting the transformation parameters and starting the automated transformation tasks. The other is for editing the Assembler macro tables. The macro tables are only required for the Assembler conversions. For converting COBOL and PL/I only the Transformation screen is needed. *(See Figure 5: CodeTran Usage Process)*

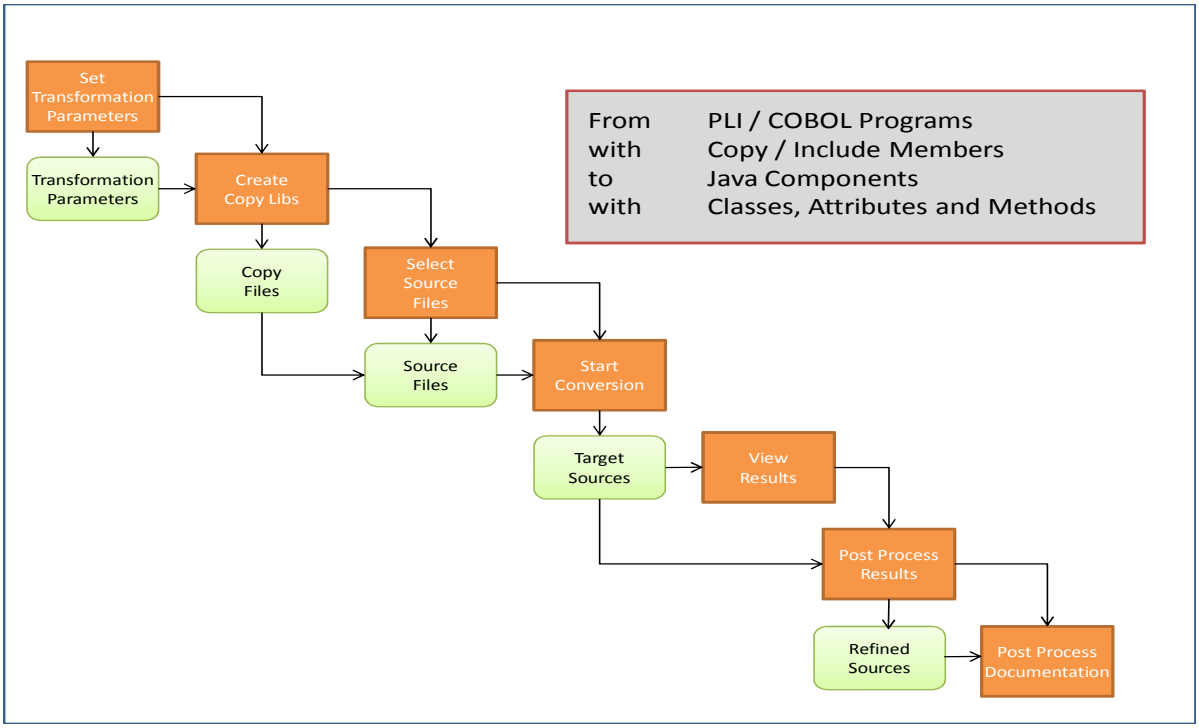


Figure 5: CodeTran Usage Process

5.1 Transformation screen

The transformation screen offers the user a graphical user interface from which he can select which language he wants to convert from and which language he wants to convert to. The first is the source language. The three source languages are:

- Assembler
- COBOL
- PL/I

The four target languages are:

- COBOL
- PL/I
- Java
- OO-COBOL.

In addition to the language selection, the user must give, as with the other tools, a product and a system name. The product name is the name of the entire system being converted. The name of the system is the name of this particular sub system or migration package. Both

should be 8-character short names. Under the language selection the user must select an output directory where the results of the conversions are stored. There should be a separate output directory for each sub system. Finally there are two check boxes – one to retain the original statements as comments in the converted code and the other to generate a test frame. They can be clicked on and off. This is all on the left hand side of the screen.

On the right side of the screen are many options. Most of these options are concerned with the post processing of the converted program. The name of the converted program and the name of a particular class are only required if one is correcting a particular class. The date also refers to the date of the correction. One should know that it is terribly difficult to convert programs from one language to another. The original transformation should convert at least 90% of the code, but there will still remain up to 10% of the code to be adjusted. There are here special post processing tools for making this adjustment. With their help the conversion should come close to 100%. If not then the user will have to finish the job manually.

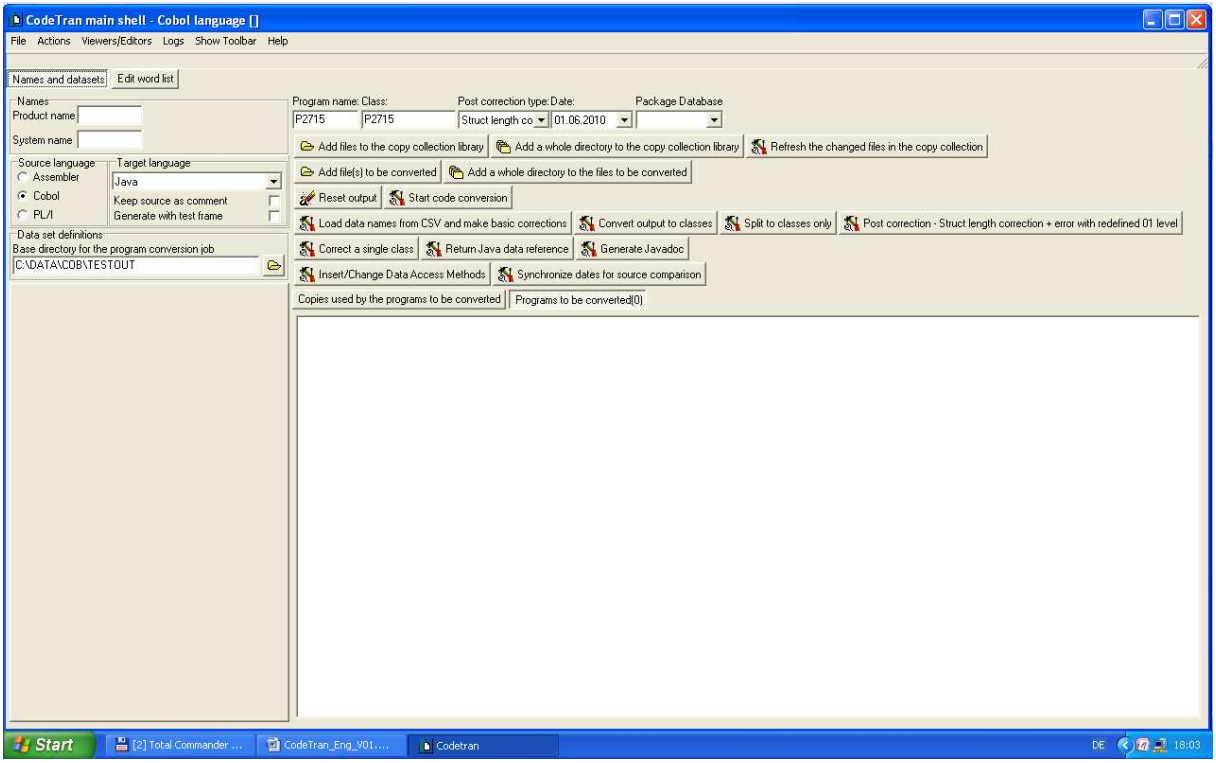
The only options required by the user in making the initial transformation are the lines 2 thru 5. In these lines he has the following 7 selections:

- Add Copy or Include Members to the COPY Library
- Add a whole Copy or Include Directory to the COPY Library
- Refresh the members of the COPY Library (this is in case it has changed since the last job was run)
- Select individual source files to be converted
- Select a whole directory of source files to be converted
- Reset the output directory (this will delete all of the contents of the output directory)
- Start the code conversion (this can be done when one or more source files have been selected).

The lines 6 thru 9 are reserved for the post processing. The following post processing tasks are supported:

- Load data table to correct the displacements of a particular class
- Correct the data structure lengths
- Split the Classes into separate source files
- Correct a single class
- Trace a java data reference
- Insert data access methods
- Synchronize dates for data comparison
- Generate Javadoc Repository

To perform the post processing tasks the user must be a specialist in both COBOL or PL/I and Java. All of these tasks require a deep insight into the relationship between the source and the target sources as well as in the limits of automatic conversion. *(See Screenshot 1: The Transformation functions and parameters)*

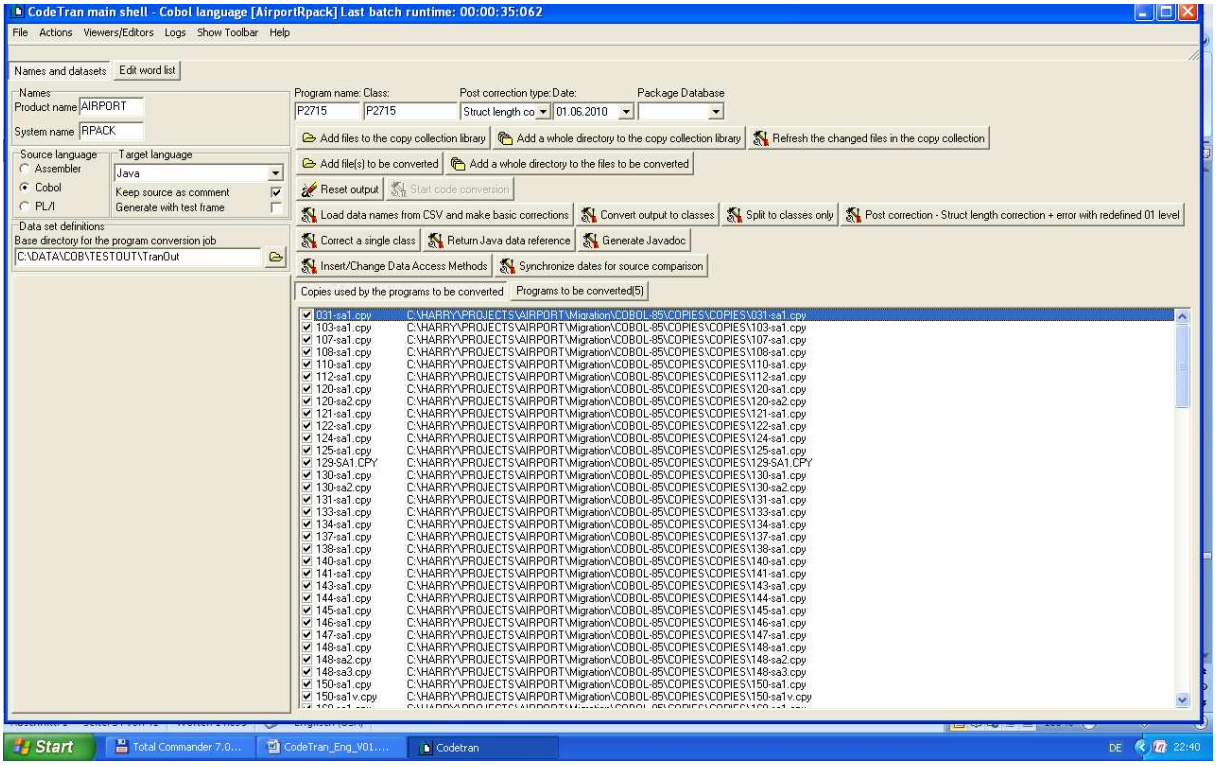


There are two categories of input sources to be transformed.

- Independent data descriptions in the form of copies and includes
- Program source files

The independent data descriptions have to be incorporated into the program source files before the transformation can start. This is made in the preprocessor. This is why the user must collect the copy/include members into a common library where they can be accesses by the preprocessor. The user can add individual copy/include members or entire directories. At the beginning he will want to select an entire directory. Later he may have to add individual members. If this is the case, he must also refresh the copy library.

The source files are also selected at the beginning as an entire source directory, namely the output directory of SoftRedo. It is important to note that all COBOL and PL/I programs to be converted must first be reengineered with SoftRedo. If later individual sources have to be reconverted they can be added one at a time to the list of programs. This will be the case when programs have to again pass thru the SoftRedo reengineering process. Therefore it is possible to select both directories and individual files. *(See Screenshot 2: File Selection)*



5.2 Assembler Macro Screen

The macro editing panel displays the current macro or function table for the target language. For Assember the table will contain all of the non-standard assembler statements which are possible macros. It is up to the user to identify and classify them.

The option „Edit Word List" will cause the current macro or function table for the target language to be displayed in two columns

- internal type and
- macro / function name.

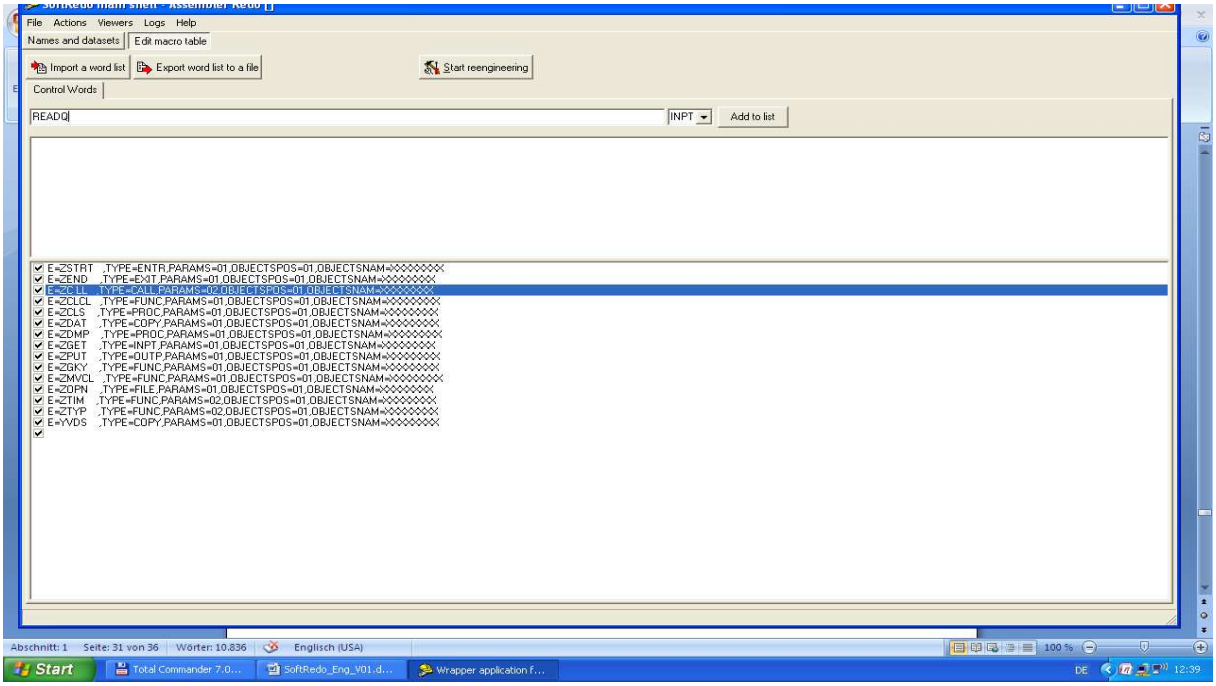
After a prescanning run, the macros and functions are displayed but their internal type will be unknown. The names of all macros will be listed. It will be up to the user to assign them one of the 30 predefined internal types.

Otherwise the contents of the existing standard table of macros will be displayed. The user can overwrite the macros. He may also add additional macros and assign them one of the predefined internal types. If the user wants to use an existing table from a previous analysis he may import it using the import function „Import Word List".

When the table is ready the user can export it by selecting the button „Export Word List". He will then be given a view of the windows directory and a box for assigning the name. The default name is FuncTab. It is recommended to store the FuncTab in the input directory together with the sources to be audited.

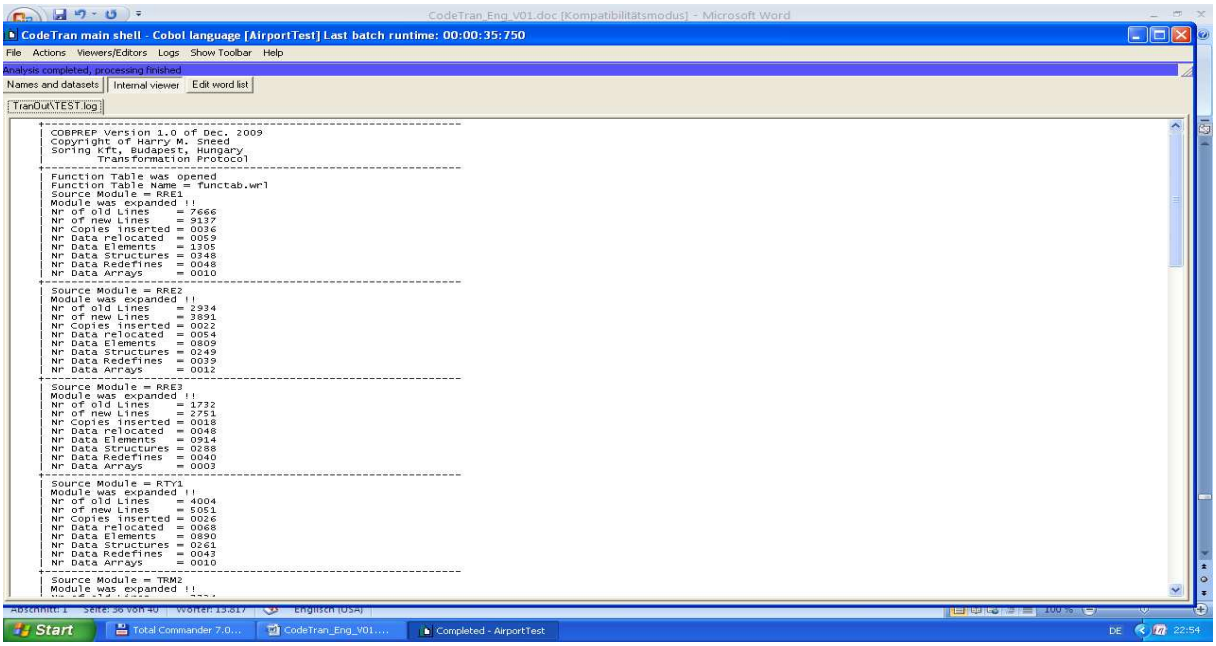
Later this table can be read in again by clicking the button „ Import a word list". Otherwise the existing standard table of macros will be loaded.

The check box to the left of the table entries allows the user to remove macros and functions by unchecking them and clicking the right mouse button to remove them from the list. Care should be taken in keeping the tables of macros and functions consistent with the sources they refer to. *(See Screenshot 3: Setting up the Macro table)*



5.3 Running a Transformation Job

To run a transformation job the user must select the language, give in the product and system names, select an output directory, select the copies/includes and select the source files to be transformed. He then needs only to click the start conversion button. The job will run in the background. After each and every transformation step the user will be requested to confirm the action has been completed. He does this by pressing the enter key. After the last conversion step has been completed a protocol of the job will be displayed showing what transformations have been performed upon which programs. *(see Screenshot 4: SoftTran Job Protocol)*



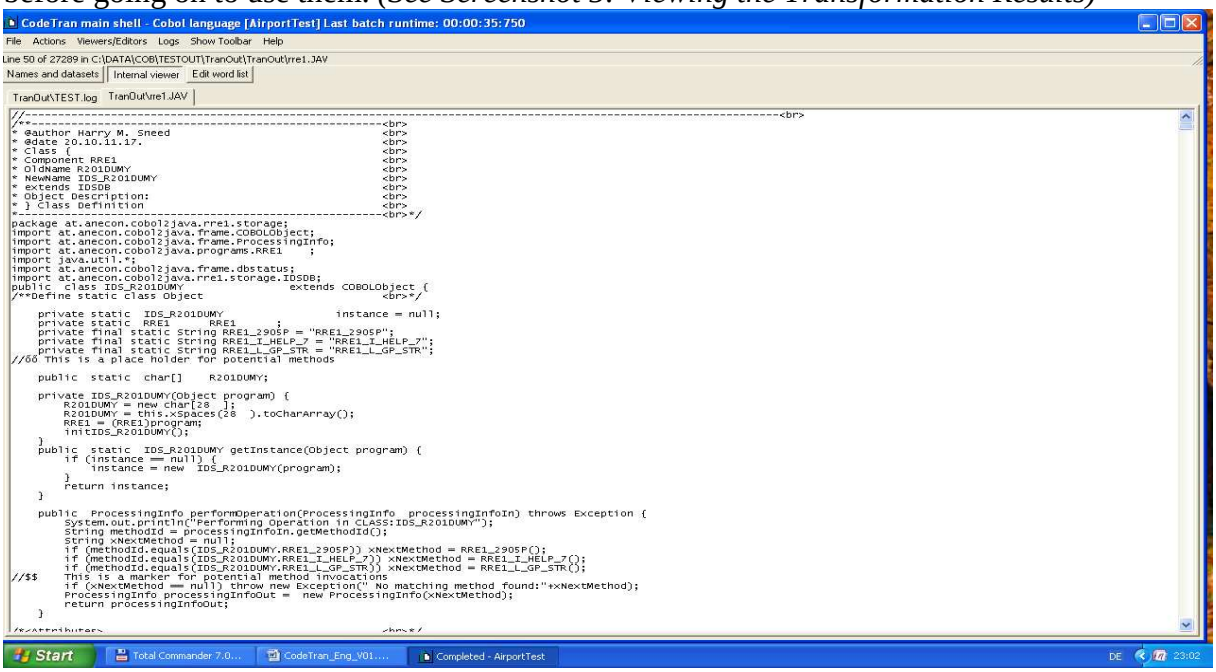
5.4 Viewing the Transformation Results

After running the transformation job, the user can view the resulting Java or OO-COBOL sources with the viewer function. All of them are stored in the output directory assigned by the user. The user can view them by using his own editor or by using the standard CodeTran viewer.

To view the results via the standard CodeTran viewer, the user should click the option „Viewers" in the upper menu bar. A pulldown menu will appear which will offer two options

- Open a file with the internal viewer
- Call an external editor.

The internal viewer will display all of the source files in the output directory. The user needs only to click on a file to view its contents. It is recommended to take a close look at the results before going on to use them. *(See Screenshot 5: Viewing the Transformation Results)*



6 CodeTran Transformation Samples

6.1 Reengineered Assembler Source

```
XM059    CSECT
         REGISTER
*        XM059   EWIGER KALENDER 1901 - 1999
*        PARAMETER
*        ---------
*
*            P1: DATUM TT,MM,JJ          8 BYTES  AN
*
*            P2: SPRACHCODE 1,2,3
*              WENN UNGUELTIG = 1        1 BYTE   AN
*
*            P3: ALINIERUNG L (LINKS)
*                       R (RECHTS)
*                 WENN UGUELTIG = L      1 BYTE   AN
*            P4: WOCHENTAG OUTPUT     10 BYTES  AN
*
*
*            WENN P1 NICH PLAUSIBEL, WIRD P4 MIT '?' GEFUELLT
*
*            DIESER KALENDER BERUHT AUF DER TATSACHE, DASS AM
*            1. JANUAR 1901   D I E N S T A G   WAR.
*
         STM   R14,R12,CON001(R13)
****     call subprogram
         BALR  R12,0
         USING *,R12
         ST    R13,SAVE+4
         LA    R11,SAVE
         ST    R11,8(R13)
         LA    R13,SAVE
         LM    R2,R5,0(R1)
*************************************************************************
*            CHECK INPUT FOR VALIDITY
         MVC   DD,0(R2)
         MVC   MM,3(R2)
         MVC   YY,6(R2)
         MVC   SPC,0(R3)
         MVC   LRS,0(R4)
*
         MVI   P16,LIT001
*************************************************************************
*REMOVE IO Operation removed to end of Program
         BAS   R12,CALL001
*************************************************************************
*
         IF    (P16,EQ,LIT001)
*        B     A1
         MVC   NEXTLAB,'A1'
         RETURN
*=====>* GOTO branch forward to A1
         ELSE
         MVC   0(10,R5),=10C'?'
*        B     RET
         MVC   NEXTLAB,'RET'
```

```
      RETURN
*=====>* GOTO branch forward to RET
      ENDIF
******** Label=A1      *****************
A1      EQU   *
*
      IF    (SPC,EQ,LIT002)
*      B     A3
      MVC   NEXTLAB,'A3'
      RETURN
*=====>* GOTO branch forward to A3
      ELSE
      ENDIF
*
      IF    (SPC,EQ,LIT003)
*      B     A3
      MVC   NEXTLAB,'A3'
      RETURN
*=====>* GOTO branch forward to A3
      ELSE
      ENDIF
*
      IF    (SPC,EQ,LIT004)
*      B     A3
      MVC   NEXTLAB,'A3'
      RETURN
*=====>* GOTO branch forward to A3
      ELSE
      MVI   SPC,LIT002         DEFAULT WERT
      ENDIF
******** Label=A3      *****************
A3      EQU   *
*
      IF    (LRS,EQ,LIT005)
*      B     A5
      MVC   NEXTLAB,'A5'
      RETURN
*=====>* GOTO branch forward to A5
      ELSE
      ENDIF
*
      IF    (LRS,EQ,LIT006)
*      B     A5
      MVC   NEXTLAB,'A5'
      RETURN
*=====>* GOTO branch forward to A5
      ELSE
      MVI   LRS,LIT005         DEFAULT WERT
      ENDIF
******** Label=A5      *****************
A5      EQU   *
      PACK  DDP,DD
      PACK  MMP,MM
      PACK  YYP,YY
      LA    R7,3
      LA    R8,DDP-2
      LA    R9,DDB-2
******** Label=B1      *****************
B1      EQU   *
      LA    R8,2(R8)
      LA    R9,2(R9)
      XC    DOPP,DOPP
```

```
      MVC   DOPP+6(2),0(R8)
      CVB   R1,DOPP
      STH   R1,0(R9)
*
      DOWHILE(LRS,EQ,C'R')
*      B     B1
      MVC   NEXTLAB,'B1'
      RETURN
      ENDDO
*<=====* GOTO branch backward to B1
*
*
*********************************************************************
*
      LH    R1,YYB            YEAR
*
      DOWHILE(LRS,EQ,C'R')
*      B     R1
      MVC   NEXTLAB,'R1'
      RETURN
      ENDDO
*<====>* GOTO branch to register R1
*
      MH    R1,LIT007         GET DAYS
      ST    R1,YTT            SAVE IT
*********************************************************************
*
*           KORREKTUR SCHALTJAHRE
*
      XR    R8,R8             CLEAR
      LH    R9,YYB              YEAR
      LA    R1,4              DIVISOR
      DR    R8,R1             YEAR / 4
*
      BNZ   D1                / NOT SCHALTJAHR, JUMP
      ENDIF
*
      IF    (MMB,GT,LIT008)
*      B     D1                / MONTH > FEBR. OK
      MVC   NEXTLAB,'D1'
      RETURN
*=====>* GOTO branch forward to D1
      ELSE
*
      DOWHILE(MMB,GT,=H'2')
*      B     R9
      MVC   NEXTLAB,'R9'
      RETURN
      ENDDO
*<====>* GOTO branch to register R9
*
      ENDIF
******** Label=D1      *****************
D1      EQU   *
      ST    R9,STT            SAVE CORRECTION
*********************************************************************
*
*
      LA    R8,TATAB
      LH    R9,MMB
*
      DOWHILE(MMB,GT,=H'2')
```

```
*       B     R9
        MVC   NEXTLAB,'R9'
        RETURN
        ENDDO
*<====>* GOTO branch to register R9
*
        SLL   R9,1              MULTIPLY BY 2
        AR    R8,R9              GET ADDR FOR DAYS
        LH    R9,0(R8)
        ST    R9,MTT             SAVE DAYS FROM PREVIOS MONTHS
*
*
        L     R9,YTT
        A     R9,STT
        A     R9,MTT
        AH    R9,DDB
*
        DOWHILE(MMB,GT,=H'2')
*       B     R9
        MVC   NEXTLAB,'R9'
        RETURN
        ENDDO
*<====>* GOTO branch to register R9
*
        XR    R8,R8
        LA    R1,7
        DR    R8,R1
*
*
        MH    R8,LIT009
*
        IF    (SPC,EQ,LIT002)
*       B     E3
        MVC   NEXTLAB,'E3'
        RETURN
*=====>* GOTO branch forward to E3
        ELSE
        LA    R8,CON002(R8)
        ENDIF
*
        IF    (SPC,EQ,LIT003)
*       B     E3
        MVC   NEXTLAB,'E3'
        RETURN
*=====>* GOTO branch forward to E3
        ELSE
        LA    R8,CON002(R8)
        ENDIF
******** Label=E3      ******************
E3      EQU   *
        LA    R9,WOTAB
        AR    R8,R9
        MVC   0(10,R5),0(R8)      MOVE WEEKDAY TO USER
*
        IF    (LRS,EQ,LIT005)
*       B     E9                FOR LEFT SCHIFT JUMP
        MVC   NEXTLAB,'E9'
        RETURN
*=====>* GOTO branch forward to E9
        ELSE
        LA    R1,9(R5)
        LA    R2,7
```

```
      ENDIF
******** Label=E4     *****************
E4      EQU   *
*
      IF    (0(R1),NE,LIT010)
*       B     E9
      MVC   NEXTLAB,'E9'
      RETURN
*=====>* GOTO branch forward to E9
      ELSE
      MVC   W1,0(R5)
      MVC   1(9,R5),W1
      MVI   0(R5),LIT011
*
      DOWHILE(0(R1),NE,LIT010)
*       B     E4
      MVC   NEXTLAB,'E4'
      RETURN
      ENDDO
*<=====* GOTO branch backward to E4
*
      ENDIF
******** Label=E9     *****************
E9      EQU   *
*
*
******** Label=RET    *****************
RET     EQU   *
      L     R13,SAVE+4
      LM    R14,R12,CON001(R13)
      SR    R15,R15
****    return to caller
*       B     R14
      MVC   NEXTLAB,'R14'
      RETURN
*
*********************************************************************
      EJECT
SAVE    DC    18F'-1'
DDMMYY  DS    0CL6
DD      DS    CL2
MM      DS    CL2
YY      DS    CL2
DDP     DS    CL2
MMP     DS    CL2
YYP     DS    CL2
DDB     DS    H
MMB     DS    H
YYB     DS    H
SPC     DS    C
LRS     DS    C
DOPP    DS    D
P16     DS    C
YTT     DC    F'-1'
STT     DC    F'-1'
MTT     DC    F'-1'
TTT     DC    F'-1'
W1      DS    CL9
*
*********************************************************************
*
WOTAB   DC    CL30'DIENSTAG  MARDI    MARTEDI  '
```

```
      DC   CL30'MITTWOCH  MERCREDI  MERCOLEDI '
      DC   CL30'DONNERSTAGJEUDI    GIOVEDI  '
      DC   CL30'FREITAG   VENDREDI  VENERDI  '
      DC   CL30'SAMSTAG   SAMEDI    SABATO   '
      DC   CL30'SONNTAG   DIMANCHE  DOMENICA  '
      DC   CL30'MONTAG    LUNDI     LUNEDI   '
      DC   X'FF'
*
*********************************************************************
*
TATAB   DC   H'0'
      DC   H'31'
      DC   H'59'
      DC   H'90'
      DC   H'120'
      DC   H'151'
      DC   H'181'
      DC   H'212'
      DC   H'243'
      DC   H'273'
      DC   H'304'
      DC   H'334'
      DC   X'FF'
*
*********************************************************************
*
      LTORG
*      END
*********************************************************************************
      TITLE ' ASSEMBLER IO ROUTINES '
XIORETRN DC    F              IO Return Address
*********************************************************************************
*********************************************************************
CALL001  EQU   *
      ST    R12,XIORETRN
      CALL  XM016,(DDMMYY,P16),VL
      L     R12,XIORETRN
*      B     R12
      MVC   NEXTLAB,'R12'
      RETURN
*********************************************************************************
*RECON   TABLE OF LITERAL CONSTANTS
NEXTLAB  DC    CL8
CON001   EQU   12
LIT001   EQU   C'0'
LIT002   EQU   C'1'
LIT003   EQU   C'2'
LIT004   EQU   C'3'
LIT005   EQU   C'L'
LIT006   EQU   C'R'
LIT007   EQU   H'365'
LIT008   EQU   H'2'
LIT009   EQU   H'30'
CON002   EQU   10
LIT010   EQU   C')
LIT011   EQU   C' '
*RECON   END OF LITERAL CONSTANTS
NEXTLAB  DC    CL8
*********************************************************************************
      END   XM059
```

6.2 Reengineered PL/I Source

```
DLITPLI: PROC(PCB_BE,PCB_TAA,
          PCB_TA,PCB_VA)
          OPTIONS(MAIN)  REORDER ;
/*********************************************************************/
/**** P2715    - COPYRIGHT IBM 1983 LICENSED MATERIAL -          ****/
/****                 PROGRAM PROPERTY OF IBM                   ****/
/*********************************************************************/
/*************************************************************/
/*                                                           */
/*      PARAMETER - DECLARATIONEN:                           */
/*      -------------------------                            */
/*                                                           */
/*************************************************************/
DCL COIBM CHAR(76)         INIT
('P2715    COPYRIGHT IBM CORP 1983 LICENSED MATERIAL- PROGRAM PROPERTY
OF IBM');
DCL  (PCB_BE,PCB_TAA,
     PCB_TA,PCB_VA)                    PTR ;
/* DECLARES   */
DCL SUCHKEY        CHAR(15)       INIT('') ;
DCL LINECOUNTER    PIC '(05)9'    INIT('') ;
DCL COUNTER        PIC '(15)9'    INIT('') ;
DCL TAWERT_Z       PIC 'Z.ZZZ.ZZZ.ZZ9V,99' ;
DCL TAWERT_NEU     CHAR(16) ;
  DCL BSTKNOM       PIC '(15)9' INIT(0) ;
  DCL BNOMSFRS      PIC '(15)9' INIT(0) ;
  DCL HFLOAT_1      PIC '(15)9' INIT(0) ;
  DCL HFLOAT_2      PIC '(15)9' INIT(0) ;
  DCL HFLOAT_3      PIC '(15)9' INIT(0) ;
DCL STEUER_VALUE   PIC '(1)9'  INIT('') ;
DCL LOOP_ERROR_COUNT     PIC '(05)9'     INIT(0) ;
DCL AKTUELLE_AKTSTUFE    CHAR(1) INIT('') ;
/* OPERATOR FUER DLI-ZUGRIFFE          */
DCL GLOBAL_OPERATOR CHAR(2) INIT(' =') ;
/* ZAEHLERFELDER FUER LISTEN_OUTPUT    */
DCL VALOREN_I      PIC '(05)9'   INIT(0) ;
DCL VALOREN_O      PIC '(05)9'   INIT(0) ;
DCL BESTAND_I      PIC '(05)9'   INIT(0) ;
DCL BESTAND_O      PIC '(05)9'   INIT(0) ;
DCL TABELLEN_I     PIC '(05)9'   INIT(0) ;
DCL TABELLEN_O     PIC '(05)9'   INIT(0) ;
DCL MARCHZINSEN            PIC '(11)9.(2)9' ;
DCL MARCHZ_MELDUNG         CHAR (1)        INIT('0') ;
DCL PARM_FAELL             CHAR    (10)    INIT('') ;
DCL PARM_SICHERST          PIC     '(8)9'  INIT('') ;
DCL PARM_LZPERB            CHAR    (10)    INIT('') ;
DCL PARM_LZPERV            CHAR    (10)    INIT('') ;
DCL PARM_EVERF             CHAR    (10)    INIT('') ;
DCL PARM_STKNOM            PIC '(13)9.(2)9'        ;
DCL PARM_ZSATZ             PIC '(07)9.(5)9'        ;
DCL PARM_WAEK              PIC     '999V9999'      ;
DCL PARM_WAEE              PIC     '999'           ;
DCL PARM_USANZ             CHAR    (1)     INIT('') ;
DCL INDEX_GEFUNDEN         CHAR (1)        INIT('0') ;
DCL I_COUNTER              PIC '(05)9'     INIT(0) ;
DCL AUSWAHL_PTR PTR ;
DCL CONVERSIONS_FELD_1     CHAR(11) INIT('') ;
DCL CONVERSIONS_FELD_2     CHAR(11) INIT('') ;
DCL CONVERSIONS_FELD_3     CHAR(12) INIT('') ;
DCL KEY_FELD               CHAR(33) INIT('') ;
```

```
 DCL (ONE,TWO,THREE,FOUR,FIVE,SIX) PIC '(08)9'     ;
 %INCLUDE P2715; /* SYMBOLIC CONSTANTS & LITERALS  */
    ONE   = 1 ;
    TWO   = 2 ;
    THREE = XCON3 ;
    FOUR  = XCON4 ;
    FIVE  = XCON5 ;
    SIX   = XCON6 ;
 DCL  BESTAND_DB       PIC '(05)9'      INIT (1) ;
 DCL  TABELLEN_DBA     PIC '(05)9'      INIT (2) ;
 DCL  TABELLEN_DB      PIC '(05)9'      INIT (3) ;
 DCL  VALOREN_DB       PIC '(05)9'      INIT (4) ;

DCL    PCB_PTR    PTR ;
       DCL  1   PCB      BASED (PCB_PTR) ,  /*  STOP             */
            3 DBNAME   CHAR  (   8) ,    /*  DB NAME          */
            3 SEGL     CHAR  (   2) ,    /*  SEGMENT-LEVEL     */
            3 STATUS   CHAR  (   2) ,    /*  STATUS-CODE       */
            3 F1       CHAR  (   4) ,    /*  IRRELEVANT IMS-INT */
            3 F2       PIC '9(08)',      /*  IMS-INTERN        */
            3 SNAME    CHAR  (   8) ,    /*  SEGMENT-NAME      */
            3 DKEYL    PIC '9(08)',      /*  KEY-LAENGE        */
            3 F3       PIC '9(08)',      /*  IMS-INTERN        */
            3 KEY      CHAR  (  32) ;    /*  KEY-FEEDBACKAREA   */
       DCL  1   PCBX     DEFINED PCB,
            3 XNAME    CHAR  (   8) ,    /*  DB NAME          */
            3 XREST    CHAR  (  60) ;    /*  DB DATA          */
                                 /*   UNTERTEILBAR     */
       /*************************************************************/
       /*                                   */
       /*     DL/I - ANGABEN:                        */
       /*                                   */
       /*************************************************************/
                              /************************/
                              /*  DL/I-FUNKTIONEN    */
                              /************************/
          3  FILLER             CHAR(12);
0/* Redefining Group Length does not match Redefined!*/
0/* Redefined  Group Length = 0080 */
0/* Redefining Group Length = 0068 */
0/* Difference is          = 12   */
0/* Generated Filler must pad the shorter Group!     */
       DCL  1   FUNC ,
            3 GU       CHAR  (   4)    INIT('GU  ') ,
            3 GHU      CHAR  (   4)    INIT('GHU ') ,
            3 GN       CHAR  (   4)    INIT('GN  ') ,
            3 GHN      CHAR  (   4)    INIT('GHN ') ,
            3 GNP      CHAR  (   4)    INIT('GNP ') ,
            3 GHNP     CHAR  (   4)    INIT('GHNP') ,
            3 DLET     CHAR  (   4)    INIT('DLET') ,
            3 REPL     CHAR  (   4)    INIT('REPL') ,
            3 ISRT     CHAR  (   4)    INIT('ISRT') ;
                              /************************/
                              /* DL/I-PARAMETER COUNT */
                              /************************/
       DCL  1   PARC ,
            3 C3       PIC '9(08)'    INIT(3) ,      /*CONVERTED*/
            3 C4       PIC '9(08)'    INIT(4) ,      /*CONVERTED*/
            3 C5       PIC '9(08)'    INIT(5) ,      /*CONVERTED*/
            3 C6       PIC '9(08)'    INIT(6) ;      /*CONVERTED*/
       /*************************************************************/
       /*                                   */
```

```
      /*       O N - E R R O R                          */
      /*                                                 */
      /*******************************************************************/
DCL PLIRETCODE    DECFIXED (08) INIT (0) ;              /*CONVERTED*/
ON  ERROR
CALL TESTME_002;
/*-------------------------------------------------*/
TESTME_002: PROC;
  ON ERROR SYSTEM ;
/* ON ERROR CALL ABEND(999) */     /**** ERROR IN ON-ERROR **/
  LOOP_ERROR_COUNT = LOOP_ERROR_COUNT + 1 ;
  IF LOOP_ERROR_COUNT > 5
  THEN
  DO ;
    PUT FILE(LISTE) SKIP EDIT
    ('LOOP INNERHALB ON ERROR BLOCK') (A) ;
    STOP ;
  END ;
  CLOSE  FILE(LISTE)      ;
  CALL PLIRETC(PLIRETCODE) ;
  IF PLIRETCODE > 4
  THEN
  DO ;
    IFE = 0 ;
    CALL FEHLER(IFE,PCB_PTR) ;
  END ;
  GO TO PROGRAMM_ENDE ;
  CALL   PLIDUMP('TFHCBA','P2715') ;
/* CALL   ABEND(2714)     */
END ;
ON CONVERSION
CALL TESTME_003;
/*-------------------------------------------------*/
TESTME_003: PROC;
  ON ERROR SYSTEM ;
  IF ONCODE ^= 612
  THEN
  DO ;
    PUT SKIP EDIT ('CONVERSION-ERROR, ONCODE: ',ONCODE) (A,P'ZZZ9');
    PLIRETCODE = 12 ;
    SIGNAL ERROR ;
  END ;
  ELSE
  DO ;
    PUT SKIP EDIT ('CONVERSION-ERROR 612   ',
    HERKUNFT,':>',ONSOURCE,'< ',KEY_FELD)
    (A,A,A,A,A,A) ;
    ONSOURCE = SUBSTR('00000000000000000000000000',1,
    LENGTH(ONSOURCE)) ;
  END ;
END ;
DCL FIXED_ERROR_COUNT    DECFIXED (04) INIT(0) ;         /*CONVERTED*/
ON FIXEDOVERFLOW
CALL TESTME_004;
/*-------------------------------------------------*/
TESTME_004: PROC;
  ON ERROR SYSTEM ;
  IF FIXED_ERROR_COUNT > 200
  THEN
  DO ;
    PUT SKIP EDIT('FIXED_ERROR_COUNT > 200 ',
    'TEST - TERMINATON       ') (A,A) ;
```

```
   PLIRETCODE = 12 ;
   SIGNAL ERROR ;
  END ;
  FIXED_ERROR_COUNT = FIXED_ERROR_COUNT + 1 ;
  ONSOURCE = 0 ;
END ;
/*************************************************************/
/*                                                           */
/*      UEBERSCHRIFT - LISTE                                 */
/*                                                           */
/*************************************************************/
ON  ENDPAGE(LISTE)
CALL TESTME_005;
/*-------------------------------------------------*/
TESTME_005: PROC;
  CT(19)  =  CT(19)  +  1 ;
  CT(20)  =  CT(20)  +  3 ;
  PUT FILE(LISTE) EDIT
  ('SBG-WAB','MANUELL GEAENDERTE TAGESWERTE PER',DATUM,'LISTE',
  '4',AKTUELLE_AKTSTUFE,'SEITE :',CT(19))
  (COL(01),A,COL(16),A,COL(50),A,COL(70),A,COL(76),A,
  COL(77),A,COL(100),A,COL(108),P'ZZZZ9') PAGE;      /* BVO001 */
  PUT FILE(LISTE) SKIP(2) EDIT
  ('FUER FOLGENDE POSITIONEN WURDEN B_RENDTA, B_TAGEFA,',
  ' B_VPFWET NEU GERECHNET:')
  (COL(1),A,COL(52),A) ;
  PUT FILE(LISTE) SKIP(3) EDIT
  ('VALNR','VALZUS','ZUGEH','EIGENT','NL','TITKUT',
  'TAGESWERT NEU')
  (COL(01),A,COL(16),A,COL(26),A,COL(35),A,COL(46),A,COL(53),A,
  COL(82),A) ;
  PUT FILE(LISTE) SKIP(2) EDIT ('   ') (COL(1),A) ;
  LINECOUNTER = 8 ;
END ;
/*************************************************************/
/*                                                           */
/*              INITIALISIERUNG                              */
/*                                                           */
/*************************************************************/
PGM_START:
      CT     =  0 ;
      DATUM  =  '99.99.99' ;
      ZEIT   =  TIME ;
      ZEIT   =  TRANSLATE('12:34:56,789',ZEIT,'123456789') ;
      PLIRETCODE = 0 ;
      OPEN  FILE(LISTE)            PAGESIZE(42) ;
/*************************************************************/
/*                                                           */
/*           V E R A R B E I T U  N G                        */
/*                                                           */
/*************************************************************/
   /*   FESTSTELLEN OB JOB UEBERHAUPT LAUFEN SOLL            */
KEY_FELD = LOW(32) ;
CALL DB_AKTION(TABELLEN_DB,'GU',KEY_FELD,'') ;
 IF PCB.STATUS ^= ' '
 THEN
DO ;
  PUT FILE(LISTE) SKIP EDIT
  ('DATUMRECORD IN TABELLEN_DB NICHT VORHANDEN') (A) ;
  PLIRETCODE = 12 ;
  SIGNAL ERROR ;
END ;
```

```
 ELSE
 DO ;
   PDAT = ADDR(IO_TA) ;
   AKTUELLE_AKTSTUFE = DATUMREC.AKTSTUFE ;
END ;
KEY_FELD = '9900' ;
GLOBAL_OP = '>=' ;
CALL DB_AKTION(TABELLEN_DBA,'GU',KEY_FELD,'') ;
 IF PCB.STATUS ^= ' ' ! SUBSTR(IO_TA,1,4) ^= KEY_FELD
 THEN
DO ;
  PUT FILE(LISTE) SKIP EDIT
  ('DATUMRECORD IN TABELLEN_DB NICHT VORHANDEN') (A) ;
  PLIRETCODE = 12 ;
  SIGNAL ERROR ;
END ;
 ELSE
 DO ;
   PDAT = ADDR(IO_TA) ;
   DATUM = TRANSLATE('AB.CD.EF',ABR9900.MASCHOV,'EFCDAB') ;
END ;
SIGNAL ENDPAGE(LISTE) ;
 IF AKTUELLE_AKTSTUFE = 'F'
 THEN
DO ;
  KEY_FELD = '9910' !! LOW(15) ;
  SUCHKEY  = '9910' ;
END ;
 ELSE
 DO ;
   KEY_FELD = '9909' !! LOW(15) ;
   SUCHKEY  = '9909' ;
END ;
IO_TA     = SUCHKEY ;
GLOBAL_OP = '>=' ;
CALL DB_AKTION(TABELLEN_DBA,'GHU',KEY_FELD,'') ;
 IF PCB.STATUS ^= ' ' ! SUBSTR(IO_TA,1,4) ^= SUCHKEY
 THEN
DO ;
  PUT FILE(LISTE) SKIP EDIT
  ('KEIN ABRUF') (A) ;
  PLIRETCODE = 4 ;
  SIGNAL ERROR ;
END ;
/*  DATUMSRECORD DER BESTANDS_DB HOLEN ZUR BESETZUNG PARM_SICHERST*/
KEY_FELD = LOW(32) ;
GLOBAL_OP = '>=' ;
CALL DB_AKTION(BESTAND_DB,'GU',KEY_FELD,'') ;
 IF PCB.STATUS ^= ' ' ! SUBSTR(IO_BE,1,15) ^= LOW(15)
 THEN
DO ;
  CALL FEHLERMELDUNG(1) ;
  PLIRETCODE = 8 ;
  SIGNAL ERROR ;
END ;
 ELSE
 DO ;
   PDAT = ADDR(IO_BE) ;
   PARM_SICHERST = TRANSLATE('ABCDEFGH',DATUMREC.SICHERST,
   'AB.CD.EFGH') ;
END ;
/*  BESTAND      LESEN, SOLANGE DL/I-STATUS 'OK'               */
```

```
LOOP:
DO WHILE(PCB.STATUS = ' ' ) ;
  CALL DB_AKTION(BESTAND_DB,'GHN','','U') ;
  IF PCB.STATUS ^= ' '
  THEN
  DO ;
    LEAVE LOOP ;
  END ;
  ELSE
  DO ;
    /* VERARBEITUNG NUR WENN FLAG GESETZT         */
    IF B_ABRTAGW = '1'
    THEN
    DO ;
      KEY_FELD = B_VALNR !!
      B_VALZUS !!
      B_ZUGEH  !!
      B_EIGENT !!
      B_NL     !!
      B_RESKEYBE ;
      CALL DB_AKTION(VALOREN_DB,'GU',KEY_FELD,'') ;
      IF PCB.STATUS ^= ' '
      THEN
      DO ;
        CALL FEHLERMELDUNG(2) ;
        PLIRETCODE = 8 ;
        SIGNAL ERROR ;
      END ;
      HERKUNFT = 'B_VPFWET' ;
      IF S_SNCD = 'S'
      THEN
      DO ;
        B_VPFWET = 0 ;
      END ;
      ELSE
      DO ;
        H1 = B_TAWERT ;
        H2 = S_VPFANS / 100 ;
        B_VPFWET = ROUND1(H1,H2) ;
      END ;
      /* OBLIGATION ODER AKTIE ? */
      IF VERIFY(SUBSTR(S_VALART,2,1),'6789') = 0
      THEN
      DO ;
        /* OBLIGATIONEN GEFUNDEN    */
        CALL RENDBER ;
      END ;
      ELSE
      DO ;
        /*  AKTIEN GEFUNDEN    */
        BSTKNOM = B_STKNOM ;
        SELECT (S_SNCD) ;
          WHEN ('N')
          DO ;
            KEY_FELD = '01' !! VALOR.C_OWAEC ;
            CALL DB_AKTION(TABELLEN_DB,'GU',KEY_FELD,'') ;
            IF PCB.STATUS ^= ' '
            THEN
            DO ;
              CALL FEHLERMELDUNG(3) ;
              PLIRETCODE = 8 ;
              SIGNAL ERROR ;
```

```
      END ;
      IF D_WAEE ^= 0 & B_TAWERT ^= 0
      THEN
      DO ;
        H1 =        (D_WAEK / D_WAEE) ;
        H2 = BSTKNOM * S_DIV / B_TAWERT ;
        B_RENDTA = ROUND2(H1,H2) ;
      END ;
      ELSE
      DO ;
        B_TAWERT = 0 ;
      END ;
    END ;
    WHEN ('S')
    DO ;
      KEY_FELD = '01' !! VALOR.C_HWAEC ;
      CALL DB_AKTION(TABELLEN_DB,'GU',KEY_FELD,'') ;
      IF PCB.STATUS ^= ' '
      THEN
      DO ;
        KEY_FELD = '01' !! VALOR.C_OWAEC ;
        CALL DB_AKTION(TABELLEN_DB,'GU',KEY_FELD,'') ;
        IF PCB.STATUS ^= ' '
        THEN
        DO ;
          CALL FEHLERMELDUNG(3) ;
          PLIRETCODE = 8 ;
          SIGNAL ERROR ;
        END ;
      END ;
      IF D_WAEE ^= 0 & B_TAWERT ^= 0
      THEN
      DO ;
        H1 =        (D_WAEK / D_WAEE) ;
        H2 = BSTKNOM * S_DIV * 100 / B_TAWERT ;
        B_RENDTA = ROUND2(H1,H2) ;
      END ;
      ELSE
      DO ;
        B_TAWERT = 0 ;
      END ;
    END ;  /*  'WHEN ('S')    */
    OTHERWISE
    DO ;
      B_RENDTA = 0 ;
    END ;
  END ;     /* SELECT S_SNCD */
END ;       /* OBLI / AKTIE  */
/*  B_TAGEFA BERECHNEN              */
HERKUNFT = 'B_TAGEFA' ;
BNOMSFR  = B_NOMSFR ;
H1       = B_RENDTA ;
B_TAGEFA = ROUND1(BNOMSFR,H1) ;
/*  FLAG RUECKSETZEN                */
B_ABRTAGW = '' ;
/*  REWRITE BESTANDES_RECORD        */
CALL DB_AKTION(BESTAND_DB,'REPL','','R') ;
IF PCB.STATUS ^= ' '
THEN
DO ;
  CALL FEHLERMELDUNG(4) ;
  PLIRETCODE = 12 ;
```

```
      SIGNAL ERROR ;
     END ;
     /*  MELDUNG AUF LISTE AUSGEBEN       */
     TAWERT_Z  = B_TAWERT ;
     /*  CHANGE PUNKTE IN HOCHKOMMA   */
     TAWERT_NEU = TRANSLATE(TAWERT_Z,"",'.') ;
     LINECOUNTER = LINECOUNTER + 2 ;
     IF LINECOUNTER > 60
     THEN
     DO ;
      SIGNAL ENDPAGE(LISTE) ;
     END ;
     PUT FILE(LISTE) SKIP  EDIT
     (S_VALNR,S_VALZUS,S_ZUGEH,S_EIGENT,S_NL,S_TIKUT,
     TAWERT_NEU)
     (COL(1),A,COL(18),A,COL(26),P'99999',COL(37),P'9',
     COL(45),P'9999',COL(53),A,COL(82),A) ;
     PUT FILE(LISTE) SKIP EDIT ('      ') (COL(1),A) ;
     COUNTER = COUNTER + 1 ;
    END ;          /* IF FLAG GESETZT                    */
  END ;             /* IF BESTANDS_RECORD FOUND           */
END ;               /* DO WHILE BESTANDS_RECORDS VORHANDEN    */
/* LOESCHEN DES ABRUFRECORDS NACH VERARBEITUNG                */
 IF AKTUELLE_AKTSTUFE = 'F'
 THEN
DO ;
  KEY_FELD = '9910' !! LOW(15) ;
  SUCHKEY  = '9910' ;
END ;
 ELSE
 DO ;
   KEY_FELD = '9909' !! LOW(15) ;
   SUCHKEY  = '9909' ;
END ;
IO_TA     = SUCHKEY ;
GLOBAL_OP = '>=' ;
CALL DB_AKTION(TABELLEN_DBA,'GHU',KEY_FELD,'') ;
 IF PCB.STATUS ^= ' ' ! SUBSTR(IO_TA,1,4) ^= SUCHKEY
 THEN
DO ;
  PUT FILE(LISTE) SKIP EDIT
  ('KEIN ABRUF') (A) ;
  PLIRETCODE = 4 ;
  SIGNAL ERROR ;
END ;
CALL DB_AKTION(TABELLEN_DBA,'DLET','','R') ;
 IF PCB.STATUS ^= ' '
 THEN
DO ;
  PLIRETCODE = 12 ;
  PUT FILE(LISTE) SKIP EDIT (
  'DELETE ERROR, TABELLEN (ABRUF_RECORD) ') (A) ;
  PLIRETCODE = 12 ;
  SIGNAL ERROR ;
END ;
/*  ENDSUMME AUSGEBEN             */
PUT FILE(LISTE) SKIP EDIT ('         ') (COL(2),A) ;
PUT FILE(LISTE) SKIP EDIT
   ('********** TOTAL ',COUNTER,'POSITIONEN NEU GERECHNET **********')
   (COL(30),A,COL(47),P'ZZZZ9',COL(53),A) ;
 IF UPDDAT(PCB_BE) ^= 0
 THEN
```

```
DO;
  CALL FEHLERMELDUNG(5);
  PCB_PTR = PCB_BE;
  PLIRETCODE = 12;
  SIGNAL ERROR;
END;
/* IF UPDDAT(PCB_TAA) ^= 0 THEN
DO;
  CALL FEHLERMELDUNG(5);
  PCB_PTR = PCB_TAA;
  PLIRETCODE = 12;
  SIGNAL ERROR;
END; */
PROGRAMM_ENDE:
CLOSE FILE(LISTE) ;
  /*   FEHLER-BEHANDLUNG      */
FEHLER:    PROC(JFE,PCB0) ;                /*  FEHLER BEHANDLUNG  */
          DCL  PCB0      PTR       ;
          DCL  JFE       DEC15,0) ;              /*CONVERTED*/
          PUT  FILE(LISTE)   EDIT
             ('DBNAME    = >',PCB_PTR->PCB.DBNAME ,'<',
              '  STATUS  = >',PCB_PTR->PCB.STATUS ,'<',
              '  SEGMENT = >',PCB_PTR->PCB.SNAME  ,'<',
              '  S.LEVEL =  ',PCB_PTR->PCB.SEGL   ,
              '  KEY     = >',PCB_PTR->PCB.KEY    ,'<',
              '  PLIRETC =  ',PLIRETCODE          )
             (SKIP(1),A,A,A,
              SKIP(1),A,A,A,
              SKIP(1),A,A,A,
              SKIP(1),A,A,
              SKIP(1),A,A,A,
              SKIP(1),A,A) ;
END  FEHLER ;
FEHLERMELDUNG:  PROC(FEHLER_INDEX) ;
DCL FEHLER_INDEX     DECFIXED (04)  ;                /*CONVERTED*/
DCL MAX_INDEX        DECFIXED (04) INIT (30) ;       /*CONVERTED*/
DCL FEHLERTEXT (30) CHAR(60) INIT (
    'FEHLERHAFTER START AUF BESTANDS_DB',
    'VALOREN-RECORD NICHT VORHANDEN (MUSS)',
    'TABELLEN_RECORD (ORIGIN.-WAEH.) NICHT GEFUNDEN' ,
    'REPLACE-ERROR IN BESTANDS_DB',
    'FEHLER  5',
    'FEHLER  6',
    'FEHLER  7',
    'FEHLER  8',
    'FEHLER  9',
    'PGM_FEHLER: FALSCHER SELCT DB_NR');

 IF (FEHLER_INDEX > MAX_INDEX) !
   (FEHLER_INDEX < 1)
   THEN
DO ;
  PUT FILE(LISTE) SKIP(3) EDIT
  ('FEHLER_INDEX OUT OF RANGE, VALUE: ',FEHLER_INDEX)
  (COL(40),A,P'ZZZZZZZ9',SKIP(3)) ;
  GO TO FEHLERM_END ;
END ;
PUT FILE(LISTE) SKIP EDIT (FEHLERTEXT(FEHLER_INDEX))
   (COL(01),A);
PUT FILE(LISTE) SKIP EDIT
    ('LAST FILESTATUS: ',PCB.STATUS) (A,A) ;
FEHLERM_END:
```

```
END FEHLERMELDUNG ;
DB_AKTION: PROC (DB_NR,DB_FUNKTION,DB_SCHLUESSEL,AKTION) ;
  DCL DB_NR              DECFIXED (04) ;            /*CONVERTED*/
  DCL SSA_PTR            PTR           ;
  DCL DB_FUNKTION        CHAR (4)      ;
  DCL DB_SCHLUESSEL      CHAR (*)      ;
  DCL AKTION             CHAR (1)      ;
  DCL FOUR               DECFIXED (08) INIT (4) ;     /*CONVERTED*/
  SELECT (DB_NR) ;
   WHEN (BESTAND_DB)
   DO ;
    SSA_BE.OP  = GLOBAL_OP ;
    SSA_BE.KEY = DB_SCHLUESSEL ;
    SELECT (AKTION) ;
     WHEN ('U')
      CALL  PLITDLI(FOUR,DB_FUNKTION,PCB_BE,IO_BE,SSA_BEU) ;
     WHEN ('R')
      CALL  PLITDLI(THREE,DB_FUNKTION,PCB_BE,IO_BE) ;
     OTHERWISE
      CALL  PLITDLI(FOUR,DB_FUNKTION,PCB_BE,IO_BE,SSA_BE) ;
    END ;
    PCB_PTR = PCB_BE ;
   END ;
   WHEN (TABELLEN_DBA)
   DO ;
    SSA_TAA.OP  = GLOBAL_OP ;
    SSA_TAA.KEY = DB_SCHLUESSEL ;
    SELECT (AKTION) ;
     WHEN ('U')
      CALL  PLITDLI(FOUR,DB_FUNKTION,PCB_TAA,IO_TA,SSA_TAAU) ;
     WHEN ('R')
      CALL  PLITDLI(THREE,DB_FUNKTION,PCB_TAA,IO_TAA) ;
     OTHERWISE
      CALL  PLITDLI(FOUR,DB_FUNKTION,PCB_TAA,IO_TA,SSA_TAA) ;
    END ;
    PCB_PTR = PCB_TAA  ;
   END ;
   WHEN (TABELLEN_DB)
   DO ;
    SSA_TA.OP  = GLOBAL_OP ;
    SSA_TA.KEY = DB_SCHLUESSEL ;
    SELECT (AKTION) ;
     WHEN ('U')
      CALL  PLITDLI(FOUR,DB_FUNKTION,PCB_TA,IO_TA,SSA_TAU) ;
     WHEN ('R')
      CALL  PLITDLI(THREE,DB_FUNKTION,PCB_TA,IO_TA) ;
     OTHERWISE
      CALL  PLITDLI(FOUR,DB_FUNKTION,PCB_TA,IO_TA,SSA_TA) ;
    END ;
    PCB_PTR = PCB_TA ;
   END ;
   WHEN (VALOREN_DB)
   DO ;
    SSA_VA.OP  = GLOBAL_OP ;
    SSA_VA.KEY = DB_SCHLUESSEL ;
    SELECT (AKTION) ;
     WHEN ('U')
      CALL  PLITDLI(FOUR,DB_FUNKTION,PCB_VA,IO_VA,SSA_VAU) ;
     WHEN ('R')
      CALL  PLITDLI(THREE,DB_FUNKTION,PCB_VA,IO_VA) ;
     OTHERWISE
      CALL  PLITDLI(FOUR,DB_FUNKTION,PCB_VA,IO_VA,SSA_VA) ;
```

```
      END ;
      PCB_PTR = PCB_VA   ;
     END ;
     OTHERWISE
     DO ;
      CALL FEHLERMELDUNG(10) ;
      PLIRETCODE = 12 ;
      SIGNAL ERROR ;
     END ;
    END ;
    GLOBAL_OP = ' =' ;
END DB_AKTION ;
%INCLUDE RENDITE ;
%INCLUDE RENDBER ;
%INCLUDE DAYDIFF ;
%INCLUDE ROUND1 ;
%INCLUDE ROUND2 ;
%INCLUDE UPDDAT ;
PROGRAMM_ENDE_ENDE:
END  DLITPLI ;                  /*     PROGRAMM ENDE       */
```

6.3 Reengineered COBOL Source

```
      IDENTIFICATION DIVISION.
      PROGRAM-ID. XM059.
     ****************************************************************
     *  XM059   EWIGER KALENDER 1901 - 1999
     *  PARAMETER
     *  ---------
     *       P1: DATUM TT,MM,JJ          8 BYTES  AN
     *
     *       P2: SPRACHCODE 1,2,3
     *         WENN UNGUELTIG = 1        1 BYTE   AN
     *
     *       P3: ALINIERUNG L (LINKS)
     *                  R (RECHTS)
     *           WENN UGUELTIG = L       1 BYTE   AN
     *       P4: WOCHENTAG OUTPUT     10 BYTES  AN
     *
     *       WENN P1 NICH PLAUSIBEL, WIRD P4 MIT '?' GEFUELLT
     *
     *       DIESER KALENDER BERUHT AUF DER TATSACHE, DASS AM
     *       1. JANUAR 1901   D I E N S T A G   WAR.
     ****************************************************************
      ENVIRONMENT DIVISION.
      DATA DIVISION.
      WORKING-STORAGE SECTION.
REFINE 77  COMMREG     PIC X(8) VALUE LOW-VALUES.
REFINE 01  ACCEPT-DATE.
REFINE     05  CURRENT-YEAR         PIC 99.
REFINE     05  CURRENT-MONTH        PIC 99.
REFINE     05  CURRENT-DAY          PIC 99.
REFINE 01  ACCEPT-TIME.
REFINE     05  CURRENT-HOUR         PIC 99.
REFINE     05  CURRENT-MINUTE       PIC 99.
REFINE     05  CURRENT-SECOND       PIC 99.
REFINE 01  CURRENT-DATE.
REFINE     05  CURRENT-YEAR         PIC 99.
REFINE     05  FILLER               PIC X VALUE '/'.
```

```
REFINE     05  CURRENT-MONTH        PIC 99.
REFINE     05  FILLER               PIC X VALUE '/'.
REFINE     05  CURRENT-DAY          PIC 99.
REFINE 01  TIME-OF-DAY.
REFINE     05  CURRENT-HOUR         PIC 99.
REFINE     05  FILLER               PIC X VALUE '/'.
REFINE     05  CURRENT-MINUTE       PIC 99.
REFINE     05  FILLER               PIC X VALUE '/'.
REFINE     05  CURRENT-SECOND       PIC 99.
    ************************************************************
     01  DDMMYY.
       02 DD       PIC 99.
RENAME     02 MONTH    PIC 99.
RENAME     02 CENTURY   PIC 99.
REMOVE*  Redefining Field Length does not match Redefined!
RENAME     02 CE-X     REDEFINES CENTURY
RENAME                    PIC XXX.
RENAME     02 YEAR     PIC 99.
REMOVE*  Redefining Field Length does not match Redefined!
RENAME     02 YY-X     REDEFINES YEAR
RENAME                    PIC XXX.
          01  P16         PIC X.
          01  DATUM-NUMBERS.
REMOVE     02 DDP      PIC 999.
REMOVE     02 MMP      PIC 999.
REMOVE     02 YYP      PIC 999.
REMOVE     02 DDB      PIC S9999.
REMOVE     02 MMB      PIC S9999.
REMOVE     02 YYB      PIC S9999.
          01  WORK-DATA-1.
RENAME     02 SPRACHENCODE              PIC 9.
RENAME     02 LEFT-RIGHT-ALIGNMENT     PIC X.
REMOVE     02 DOPPELT   PIC S9(15)       VALUE 0.
REMOVE     02 YTT      PIC S9(8)      VALUE -1.
REMOVE     02 STT      PIC S9(8)      VALUE 0.
REMOVE     02 MTT      PIC S9(8)      VALUE 0.
REMOVE     02 TTT      PIC S9(8)      VALUE 0.
     01  WW.
       02 W1       OCCURS 10 TIMES PIC X.
    *
    ************************************************************
    *
     01  WOTAB.
       02 FILLER PIC X(30) VALUE 'DIENSTAG  MARDI     MARTEDI   '.
       02 FILLER PIC X(30) VALUE 'MITTWOCH  MERCREDI  MERCOLEDI '.
       02 FILLER PIC X(30) VALUE 'DONNERSTAGJEUDI     GIOVEDI   '.
       02 FILLER PIC X(30) VALUE 'FREITAG   VENDREDI  VENERDI   '.
       02 FILLER PIC X(30) VALUE 'SAMSTAG   SAMEDI    SABATO    '.
       02 FILLER PIC X(30) VALUE 'SONNTAG   DIMANCHE  DOMENICA  '.
       02 FILLER PIC X(30) VALUE 'MONTAG    LUNDI     LUNEDI    '.
     01  WOTAB-REDEF  REDEFINES WOTAB.
       02 DAYS-3    OCCURS 7 TIMES.
         03 LANG-1 PIC X(10).
         03 LANG-2 PIC X(10).
         03 LANG-3 PIC X(10).
     01  TATAB.
REMOVE     02 FILLER      PIC S9999      VALUE 0.
    *  DC    H'31'
REMOVE     02 FILLER    PIC S9999      VALUE 31.
    *  DC    H'59'
REMOVE     02 FILLER    PIC S9999      VALUE 59.
    *  DC    H'90'
```

```
REMOVE    02 FILLER    PIC S9999     VALUE 90.
    *  DC    H'120'
REMOVE    02 FILLER    PIC S9999     VALUE 120.
    *  DC    H'151'
REMOVE    02 FILLER    PIC S9999     VALUE 151.
    *  DC    H'181'
REMOVE    02 FILLER    PIC S9999     VALUE 181.
    *  DC    H'212'
REMOVE    02 FILLER    PIC S9999     VALUE 212.
    *  DC    H'243'
REMOVE    02 FILLER    PIC S9999     VALUE 243.
    *  DC    H'273'
REMOVE    02 FILLER    PIC S9999     VALUE 273.
    *  DC    H'304'
REMOVE    02 FILLER    PIC S9999     VALUE 304.
    *  DC    H'334'
REMOVE    02 FILLER    PIC S9999     VALUE 334.
    *  DC    X'FF'
REMOVE    02 FILLER    PIC S9999     VALUE 336.
    *  DC    X'FF'
     01  TATAB-RED    REDEFINES TATAB.
REMOVE    02 TATAB-1   OCCURS 12 TIMES PIC S9999.
     01  WORK-DATA-2.
REMOVE    02 R1        PIC S9(8).
REMOVE    02 R9        PIC S9(8).
REMOVE    02 R8        PIC S9(8).
REMOVE    02 TAB-I     PIC S9(4).
REMOVE    02 W-I       PIC S9(4).
    *
     LINKAGE SECTION.
RENAME 01  PARAM-1.
RENAME    02 PARAM-1-TAG  PIC 99.
              02 FILLER    PIC X.
RENAME    02 PARAM-1-MONAT  PIC 99.
              02 FILLER    PIC X.
RENAME    02 PARAM-1-JAHRHUNDERT  PIC 99.
              02 FILLER    PIC X.
RENAME    02 PARAM-1-JAHR  PIC 99.
RENAME 01  PARAM-2.
             02 LANG-CODE PIC 9.
RENAME 01  PARAM-3.
             02 DIRECTION PIC X.
RENAME 01  PARAM-4.
             02 DAY-NAME  PIC X(10).
RENAME PROCEDURE DIVISION USING PARAM-1 PARAM-2 PARAM-3 PARAM-4.
REFINE X-START-LABEL.
RENAME    MOVE 19 TO PARAM-1-JAHRHUNDERT
RENAME    MOVE PARAM-1-TAG TO DD
RENAME    MOVE PARAM-1-MONAT TO MONTH
RENAME    MOVE PARAM-1-JAHR TO YEAR
RENAME    MOVE LANG-CODE TO SPRACHENCODE
          *    CALL  XM016,(DDMMYY,P16),VL
               MOVE DIRECTION TO LEFT-RIGHT-ALIGNMENT
               MOVE '0' TO P16
              CALL 'XM016' USING DDMMYY, P16
             IF P16 NOT = '0'
               MOVE ALL '?' TO DAY-NAME
               GOBACK
             END-IF.
    *                   DEFAULT WERT
RENAME    IF SPRACHENCODE = 1
RENAME    OR SPRACHENCODE = 2
```

```
RENAME    OR SPRACHENCODE = 3
RENAME      CONTINUE
RENAME    ELSE
   *                  DEFAULT WERT
         MOVE 1 TO SPRACHENCODE
      END-IF
      IF LEFT-RIGHT-ALIGNMENT = 'L'
      OR LEFT-RIGHT-ALIGNMENT = 'R'
       CONTINUE
      ELSE
       MOVE 'L' TO LEFT-RIGHT-ALIGNMENT
      END-IF
      MOVE DD TO DDP
      MOVE MONTH TO MMP
      MOVE YEAR TO YYP
      MOVE DDP TO DDB
   ******************************************************************
   *                  YEAR
   *                  MINUS 1
   *                  GET DAYS
   *                  SAVE IT
      MOVE MMP TO MMB
   ******************************************************************
   *
   *      KORREKTUR SCHALTJAHRE
   *
      MOVE YYP TO YYB
   *                  YEAR / 4
      COMPUTE YTT = ( YYB - 1 ) * 365
      MOVE YYB TO R9 R8
   *                  /    SCHALTJAHR
      COMPUTE R9 = R9 / 4
   *                  / MONTH > FEBR. OK
   *                  SUBTRACT 1
      COMPUTE R8 = R8 - 4 * R9
      IF R8 = 0 AND MMB <= 2
   *                  SAVE CORRECTION
       SUBTRACT 1 FROM R9
   ******************************************************************
   *
   *                  GET ADDR FOR DAYS
      END-IF
   *                  SAVE DAYS FROM PREVIOS MONTHS
      MOVE R9 TO STT
      MOVE MMB TO TAB-I
      IF TAB-I < 1 OR TAB-I > 12
       DISPLAY 'ERROR IN XM059, TATAB INDEX: ' TAB-I '  ERDOES'
       MOVE ALL 'X' TO DAY-NAME
       GOBACK
      END-IF
      MOVE TATAB-1 (TAB-I) TO MTT
      COMPUTE R9 = YTT + STT + MTT + DDB
      COMPUTE R1 = R9 / 7
      COMPUTE R8 = 7 * R1
      COMPUTE TAB-I = R9 - R8
      IF TAB-I = 0
       MOVE 7 TO TAB-I
      END-IF
      IF TAB-I < 1 OR TAB-I > 7
       DISPLAY 'ERROR IN XM059, WOTAB INDEX: ' TAB-I '  ERDOES'
       MOVE ALL 'X' TO DAY-NAME
   *                  MOVE WEEKDAY TO USER
```

```
      GOBACK
     END-IF
     EVALUATE TRUE
      WHEN SPRACHENCODE = 1
       MOVE LANG-1 (TAB-I) TO DAY-NAME
      WHEN SPRACHENCODE = 2
       MOVE LANG-2 (TAB-I) TO DAY-NAME
      WHEN SPRACHENCODE = 3
*                    FOR LEFT SCHIFT
       MOVE LANG-3 (TAB-I) TO DAY-NAME
     END-EVALUATE
     IF LEFT-RIGHT-ALIGNMENT = 'L'
      CONTINUE
     ELSE
      MOVE DAY-NAME TO WW
      MOVE 10 TO TAB-I
      PERFORM WITH TEST BEFORE UNTIL W1 (TAB-I) NOT = SPACE
       SUBTRACT 1 FROM TAB-I
      END-PERFORM
      PERFORM WITH TEST BEFORE VARYING W-I FROM 10 BY -1
                        UNTIL TAB-I = 0 OR TAB-I = 10
       MOVE W1 (TAB-I) TO W1 (W-I)
       MOVE SPACE TO W1 (TAB-I)
       SUBTRACT 1 FROM TAB-I
      END-PERFORM
      MOVE WW TO DAY-NAME
     END-IF.
     GOBACK.
```

6.4 Assembler Macro Table

```
Lnr Lng Code Name
0001 04 PROC NOPR
0002 07 LINK OBJNAME
0003 06 FUNC YABVAN
0004 07 COPY YABZBER
0005 07 PROC YAD1401
0006 08 FUNC YADRESS1
0007 08 FUNC YADRESSE
0008 08 LINK YAGRCALL
0009 08 LINK YAGRCALT
0010 05 DB   YAIBD
0011 04 FUNC YAKV
0012 05 CNTR YALIN
0013 07 FUNC YANREDE
0014 04 FUNC YAUS
0015 05 FUNC YBAUS
0016 05 FUNC YBEIN
0017 06 FUNC YBEIN1
0018 05 PROC YBMIX
0019 04 FUNC YBOP
0020 05 FUNC YBPPZ
0021 07 PROC YBTCALL
0022 05 CALL YCALL
0023 05 FUNC YCAUS
0024 05 FUNC YCEIN
0025 04 FUNC YCLC
0026 05 FUNC YCLEN
0027 05 FILE YCLOS
0028 05 FUNC YCMIX
0029 07 LINK YCOCALL
```

```
0030 06 COPY YCOMRG
0031 07 COPY YCOMRG2
0032 06 ENTR YCSANF
0033 06 EXIT YCSEND
0034 07 FUNC YCTIME0
0035 07 FUNC YCTIME1
0036 05 FUNC YDATE
0037 06 ENTR YDCANF
0038 04 FILE YDCB
0039 06 EXIT YDCEND
0040 07 FUNC YDFTRAC
0041 04 FUNC YDIV
0042 07 MASK YDLCALL
0043 06 DB   YDLPCB
```

6.5 Data Name Table

```
OldName;NewName;
CRT-EIN;CRT-EINGABE;
CRT-AUS;CRT-AUSGABE;
RPARA;R-PARAMETER;
RPARA-ZE;R-PARAMETER-ZENTRALEINHEIT
KO;KASSENORDNUNG;
AP;APPLIKATION;
AR;ABRECHNUNG;
HB;HAUPTBEREICH;
PS;POSTESTELLE;
KOAN;KOSTENANNAHME;
KOAB;KOSTENABRECHNUNG;
KOPR;KOSTENPROGRAMM;
KOFR;KOSTENVERANSCHLAGUNG;
APAN;APPLIKATIONSANNAHME;
APAB;APPLIKATIONSABNAHME;
APFR;APPLIKATIONSVERANSCHLAGUNG;
APPR;APPLIKATIONSPROGRAMM;
ARAB;ABRECHNUNGSABNAHME;
ARAN;ABRECHNUNGSANNAHME;
ARFR;ABRECHNUNGSVERANSCHLAGUNG;
ARPR;ABRECHNUNGSPROGRAMM;
HBAN;HAUPTBEREICHSANNAHME;
HBAB;HAUPTBEREICHSABNAHME;
HBPR;HAUPTBEREICHSPROGRAMM;
HBFR;HAUPTBEREICHSVERANSCHLAGUNG;
PSAN;POSTSTELLEANNAHME;
PSAB;POSTSTELLEABNAHME;
PSFR;POSTSTELLEVERANSCHLAGUNG;
PSPR;POSTSTELLEPROGRAMM;
C-ORD1;C-ORDNUNG-1;
C-ORD2;C-ORDNUNG-2;
C-ANZ;C-ANZAHL;
C-ANZB;C-ANZAHL-BEZAHLT;
C-GEB;C-GEBUEHR;
C-GEB6;C-GEBUEHR-6;
C-MKZ;C-MEDIANKZ;
C-IND;C-INDEX;
C-LFNR;C-LAUFENDE-NUMMER;
C-MLDG-OK;C-MELDUNG-OK;
C-MLDG-WEITER;C-MELDUNG-WEITER;
C-LNFNR9;C-LAUFNR-9;
ZKTBL;ZENTRALKOSTENTABELLE;
```

```
ZKTNRB;ZENTRALKOSTENNUMMER;
ZPNAM;ZENTRALPOSTNAME;
ZPADR;ZENTRALPOSTADDRESSE;
ZK-UST;ZENTRALKASSE-UMSATZSTEUER;
ZK-BRF;ZENTRALKASSE-BERECHNUNGSFORMEL;
ZFUNK2;ZENTRALFUNKTION-2;
ZFUNK19;ZENTRALFUNKTION-19;
ZK-FRI;ZENTRALKASSE-FREITAG;
ZK-SKO;ZENTRALKASSE-ORDNUNG;
ZI-FAM;ZI-FAMILIENNAME;
ZI-VOR;ZI-VORNAME;
ZI-SUCH;ZI-SUCHAKTION;
ZPNR-TEL;ZP-TELEFONNR;
ZPNR-UNT;ZP-UNTERNEHMERNUMMER;
ZP05-SI;ZP-05-SIGNATUR;
ZP05-SIA;ZP-05-SIGNALE;
ZTB-ANF;ZTB-ANFANG;
K-KVNR;KUNDEN-KRANKENVERSICHERUNGSNUMMER;
K-GEBDAT;KUNDEN-GEBURTSDATUM;
K-GEBT;KUNDEN-GEBURTSTAG;
K-GEBM;KUNDEN-GEBURTSMONAT;
K-GEBJ;KUNDEN-GEBURTSJAHR;
K-MEHRL;KUNDEN-MEHRLEISTUNG;
K-FAMNAM;KUNDEN-FAMILIENNAME;
K-VORNAM;KUNDEN-VORNAME;
K-TITELKZ;KUNDEN-TITEL;
K-PLZ;KUNDENPOSTLEITZAHL;
K-STR;KUNDENSTRASSE;
K-GESCHL;KUNDENGESCHLECT;
```

6.6 Batch Process Interface

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--DOCTYPE CodeTran SYSTEM "CodeTran.dtd"-->
<CODETRAN>
 <Names
  ProductName = "AIRPORT"
  SystemName  = "I4-PACK"
  OutputType  = "cob2java"
 />
 <Parameters
  Comments = "Y"
  TestMode = "Y"
 />
 <Directories>
  <Orig_Input_Dir>c:\DATA\COB\TRANOUT\SourceCollection</Orig_Input_Dir>
  <Input_Dir>Workinp</Input_Dir>
  <Orig_Output_Dir>c:\DATA\COB\TRANOUT\TranOut</Orig_Output_Dir>
  <Output_Dir>TranOut</Output_Dir>
  <Work_Dir>Work</Work_Dir>
  <Copy_Dir>CopyLib</Copy_Dir>
  <Func_Tab>functab.wrl</Func_Tab>
 </Directories>
 <Files>
  <Orig_File_0001>i100.cbl</Orig_File_0001>
  <File_0001>SRC0001.CBL</File_0001>
 <Orig_File_0002>i101.cbl</Orig_File_0002>
 <File_0002>SRC0002.CBL</File_0002>
<Orig_File_0003>I300.CBL</Orig_File_0003>
<File_0003>SRC0003.CBL</File_0003>
```

<Orig_File_0004>I500.CBL</Orig_File_0004>
<File_0004>SRC0004.CBL</File_0004>
<Orig_File_0005>i900.cbl</Orig_File_0005>
<File_0005>SRC0005.CBL</File_0005>
<Orig_File_0006>i90a.cbl</Orig_File_0006>
<File_0006>SRC0006.CBL</File_0006>
<Orig_File_0007>iab1.cbl</Orig_File_0007>
<File_0007>SRC0007.CBL</File_0007>
<Orig_File_0008>iab6.cbl</Orig_File_0008>
<File_0008>SRC0008.CBL</File_0008>
<Orig_File_0009>ICAL.CBL</Orig_File_0009>
<File_0009>SRC0009.CBL</File_0009>
<Orig_File_0010>ice1.cbl</Orig_File_0010>
<File_0010>SRC0010.CBL</File_0010>
<Orig_File_0011>IDB1.CBL</Orig_File_0011>
<File_0011>SRC0011.CBL</File_0011>
<Orig_File_0012>idi1.cbl</Orig_File_0012>
<File_0012>SRC0012.CBL</File_0012>
<Orig_File_0013>iga1.cbl</Orig_File_0013>
<File_0013>SRC0013.CBL</File_0013>
<Orig_File_0014>igat.cbl</Orig_File_0014>
<File_0014>SRC0014.CBL</File_0014>
<Orig_File_0015>IKI1.CBL</Orig_File_0015>
<File_0015>SRC0015.CBL</File_0015>
<Orig_File_0016>imf9.cbl</Orig_File_0016>
<File_0016>SRC0016.CBL</File_0016>
<Orig_File_0017>imo1.cbl</Orig_File_0017>
<File_0017>SRC0017.CBL</File_0017>
<Orig_File_0018>IMON.CBL</Orig_File_0018>
<File_0018>SRC0018.CBL</File_0018>
<Orig_File_0019>iprt.cbl</Orig_File_0019>
<File_0019>SRC0019.CBL</File_0019>
<Orig_File_0020>isa1.cbl</Orig_File_0020>
<File_0020>SRC0020.CBL</File_0020>
<Orig_File_0021>ITIM.CBL</Orig_File_0021>
<File_0021>SRC0021.CBL</File_0021>
<Orig_File_0022>ITP1.CBL</Orig_File_0022>
<File_0022>SRC0022.CBL</File_0022>
<Orig_File_0023>iy10.cbl</Orig_File_0023>
<File_0023>SRC0023.CBL</File_0023>
</Files>
</CODETRAN>

6.7 Converted COBOL Program from Assembler

```
COB  *_____________________________________________________________
 COB   LINKAGE SECTION.
 COB   01  REGISTERS.
 COB       02  R0                  PIC S9(8) USAGE COMP.
 COB       02  R0-PTR  REDEFINES R0        POINTER.
 COB       02  R1                  PIC S9(8) USAGE COMP.
 COB       02  R1-PTR  REDEFINES R1        POINTER.
 COB       02  R2                  PIC S9(8) USAGE COMP.
 COB       02  R2-PTR  REDEFINES R2        POINTER.
 COB       02  R3                  PIC S9(8) USAGE COMP.
 COB       02  R3-PTR  REDEFINES R3        POINTER.
 COB       02  R4                  PIC S9(8) USAGE COMP.
 COB       02  R4-PTR  REDEFINES R4        POINTER.
 COB       02  R5                  PIC S9(8) USAGE COMP.
 COB       02  R5-PTR  REDEFINES R5        POINTER.
 COB       02  R6                  PIC S9(8) USAGE COMP.
```

```
COB      02  R6-PTR  REDEFINES R6          POINTER.
COB      02  R7                    PIC S9(8) USAGE COMP.
COB      02  R7-PTR  REDEFINES R7          POINTER.
COB      02  R8                    PIC S9(8) USAGE COMP.
COB      02  R8-PTR  REDEFINES R8          POINTER.
COB      02  R9                    PIC S9(8) USAGE COMP.
COB      02  R9-PTR  REDEFINES R9          POINTER.
COB      02  R10                    PIC S9(8) USAGE COMP.
COB      02  R10-PTR  REDEFINES R10        POINTER.
COB      02  R11                    PIC S9(8) USAGE COMP.
COB      02  R11-PTR  REDEFINES R11        POINTER.
COB      02  R12                    PIC S9(8) USAGE COMP.
COB      02  R12-PTR  REDEFINES R12        POINTER.
COB      02  R13                    PIC S9(8) USAGE COMP.
COB      02  R13-PTR  REDEFINES R13        POINTER.
COB      02  R14                    PIC S9(8) USAGE COMP.
COB      02  R14-PTR  REDEFINES R14        POINTER.
COB      02  R15                    PIC S9(8) USAGE COMP.
COB      02  R15-PTR  REDEFINES R15        POINTER.
COB   01  BASED-DATEN.
COB      02  X                   PIC X(0001).
       02  FILLER                REDEFINES X       .
         03  FILLER                PIC X(0000).
COB        03  X-0-L'REFTA1            PIC X(0001).
       02  FILLER                REDEFINES X       .
         03  FILLER                PIC X(0000).
COB        03  X-0-L'ZUSCHL            PIC X(0001).
COM  *_____________________________________________________
COB   PROCEDURE DIVISION USING REGISTERS, BASED-DATEN.
COB  * ZSTRT BASIS=R11,REGS=2
COB      MOVE X-0-R1 TO R3 .
COB      ENTRY KV0226
COB          USING ARB R3 .
COB      MOVE SAVA TO R10 .
COB      MOVE WT025A TO R5 .
COB      ENTRY KV0226
COB          USING T025F01 R5 .
COB      MOVE WT010A TO R4 .
COB      ENTRY KV0226
COB          USING T010F01 R4 .
COB      IF T010F01 = SBEH
COB         GO TO Z2261 .
COM  *                          Forward  Jump ===>>  Z2261
COB      MOVE T010F01 TO SBEH .
COB      PERFORM Z226U1 .
COB      IF T010F471 NOT = '1'
COB         GO TO Z22692 .
COM  *                          Forward  Jump ===>>  Z22692
COM  *_____________________________________________________
COB   Z2261.
COB      IF T025F021 NOT = 'E'
COB         GO TO Z2269 .
COM  *                          Forward  Jump ===>>  Z1169
COB      IF T025F86 NOT = '3'
COB         GO TO Z2268 .
COM  *                          Forward  Jump ===>>   Z2268
COB      IF T025F36 NOT = 'D'
COB         GO TO Z2263 .
COM  *                          Forward  Jump ===>>   Z2263
COB      MOVE 'G' TO T025F36 .
COB      GO TO Z2268 .
COM  *                          Forward  Jump ===>>  Z2268
```

```
COM  *_____________________________________________
COB   Z2263.
COB      IF T025F36 NOT = 'A'
COB         GO TO Z2262 .
COM  *                              Forward  Jump ===>>  Z2262
COB      MOVE 'C' TO T025F36 .
COB      GO TO Z2268 .
COM  *                              Forward  Jump ===>>  Z2268
COM  *_____________________________________________
COB   Z2262.
COB      IF T025F36 = 'B'
COB         GO TO Z2268 .
COB      IF T025F36 = 'E'
COB         GO TO Z2268 .
COB      IF T025F36 = 'C'
COB         GO TO Z2268 .
COB      IF T025F36 = 'F'
COB         GO TO Z2268 .
COB      IF T025F36 = 'G'
COB         GO TO Z2268 .
COB      CALL 'SESXCOR ' USING GEBA1 GEBA1 .
COB      CALL 'SESXCOR ' USING GEBA2 GEBA2 .
COB      MOVE ' ' TO ZUSCHL1 .
COB      MOVE ' ' TO KZLIN0 .
COB      MOVE X-1=) TO ZUSCHL1(1:L-BZAEHL1-.
COB      MOVE KP0 TO KZZUS .
COB      MOVE KP0 TO ZUBEL .
COB      MOVE KP0 TO REFTA1 .
COB      MOVE KP0 TO REFTA2 .
COB      MOVE KP0 TO REFTA3 .
COB      MOVE KP0 TO REFTA4 .
COB      MOVE KP0 TO REFLI .
COB      MOVE KP0 TO KZTAA .
COB      MOVE KP0 TO KZTAN .
COB      MOVE KP0 TO SMENG .
COB      MOVE KP0 TO LMENG .
COB      CALL 'SESXCOR ' USING KZLK KZLK .
COB      CALL 'SESXCOR ' USING KZBM KZBM .
COB      MOVE KP0 TO RF1 .
COB      MOVE KP0 TO SPREI .
COB      CALL 'SESXCOR ' USING BKZE BKZE .
COB      PERFORM Z226U2 .
COB      PERFORM Z226U3 .
COB      MOVE WT025A TO R5 .
COB      MOVE SAVA TO R10 .
COB      GO TO Z22681 .
COM  *                              Forward  Jump ===>>  Z22681
COM  *_____________________________________________________
COB   Z2268.
COB      ADD WLT025 TO R5 .
COB      ADD WLT025 TO R10 .
COM  *_____________________________________________
COB   Z22681.
COB      IF R5 NOT < WT025
COB         GO TO Z2269 .
COM  *                              Forward  Jump ===>>    Z2269
COM  *_____________________________________________
COB   Z2267.
COB      GO TO Z2261 .
COM  *                              Backward Loop <<===    Z2261
COM  *_____________________________________________
```

6.8 Converted PL/I Program from Assembler

```
XM059   :PROCEDURE ( PARM1,  PARM2,  PARM3,  PARM4,  PARM5,  );
/*********************************************************************/
/* F I L E  D E C L A R A T I O N S                                   */
/* E N T R Y   D E C L A R A T I O N S                                */
DECLARE  R12              ENTRY;
DECLARE  XM016              ENTRY;
/*********************************************************************/
DECLARE  SESTM  ENTRY RETURNS (FIXED BIN(15));
DECLARE  SESEDIT  ENTRY RETURNS (CHAR(*));
DECLARE  SESPACK  ENTRY RETURNS (CHAR(*));
DECLARE  SESUNPAC ENTRY RETURNS (CHAR(*));
DECLARE  SESXCOR  ENTRY RETURNS (BIT(8) );
DECLARE  SESXIOR  ENTRY RETURNS (BIT(8) );
DECLARE  SESXOI   ENTRY RETURNS (BIT(8) );
DECLARE  SESXOC   ENTRY RETURNS (BIT(8) );
/*********************************************************************/
/*   C O N S T A N T   D A T A  D E C L A R A T I O N S        /*
DECLARE 1  POINTER_DATEN ,
        2  R0         FIXED BINARY (31),
        2  REG_0 (31)   BIT(1)  DEFINES R0,
        2  R_0          POINTER DEFINES R0,
        2  R1         FIXED BINARY (31),
        2  REG_1 (31)   BIT(1)  DEFINES R1,
        2  R_1          POINTER DEFINES R1,
        2  R2         FIXED BINARY (31),
        2  REG_2 (31)   BIT(1)  DEFINES R2,
        2  R_2          POINTER DEFINES R2,
        2  R3         FIXED BINARY (31),
        2  REG_3 (31)   BIT(1)  DEFINES R3,
        2  R_3          POINTER DEFINES R3,
        2  R4         FIXED BINARY (31),
        2  REG_4 (31)   BIT(1)  DEFINES R4,
        2  R_4          POINTER DEFINES R4,
        2  R5         FIXED BINARY (31),
        2  REG_5 (31)   BIT(1)  DEFINES R5,
        2  R_5          POINTER DEFINES R5,
        2  R6         FIXED BINARY (31),
        2  REG_6 (31)   BIT(1)  DEFINES R6,
        2  R_6          POINTER DEFINES R6,
        2  R7         FIXED BINARY (31),
        2  REG_7 (31)   BIT(1)  DEFINES R7,
        2  R_7          POINTER DEFINES R7,
        2  R8         FIXED BINARY (31),
        2  REG_8 (31)   BIT(1)  DEFINES R8,
        2  R_8          POINTER DEFINES R8,
        2  R9         FIXED BINARY (31),
        2  REG_9 (31)   BIT(1)  DEFINES R9,
        2  R_9          POINTER DEFINES R9,
        2  R10               POINTER,
        2  REG_10 (31)   BIT(1)  DEFINES R10,
        2  R_10         FIXED BINARY (31) DEFINES R10,
        2  R11               POINTER,
        2  REG_11 (31)   BIT(1)  DEFINES R11,
        2  R_11         FIXED BINARY (31) DEFINES R11,
        2  R12               POINTER,
        2  REG_12 (31)   BIT(1)  DEFINES R12,
        2  R_12         FIXED BINARY (31) DEFINES R12,
        2  R13               POINTER,
```

```
        2  REG_13 (31)   BIT(1)  DEFINES R13,
        2  R_13         FIXED BINARY (31) DEFINES R13,
        2  R14               POINTER,
        2  REG_14 (31)   BIT(1)  DEFINES R14,
        2  R_14         FIXED BINARY (31) DEFINES R14,
        2  R15               POINTER,
        2  REG_15 (31)   BIT(1)  DEFINES R15,
        2  R_15         FIXED BINARY (31) DEFINES R15,
;   /* END_DECLARE */
DECLARE 1  KONSTANTEN_DATEN ,
        2  SAVE         (0018)  FIXED BINARY (31)
          INIT (  -1 ),
;   /* END_DECLARE */
DECLARE 1  DDMMYY       (0001) ,
        2  YTT               FIXED BINARY (31)
          INIT (  -1 ),
        2  STT               FIXED BINARY (31)
          INIT (  -1 ),
        2  MTT               FIXED BINARY (31)
          INIT (  -1 ),
        2  TTT               FIXED BINARY (31)
          INIT (  -1 ),
        2  WOTAB             CHAR(0030)
          INIT ( 'DIENSTAG  MARDI     MARTEDI   '),
        2  CHAR_???          CHAR(0030)
          INIT ( 'MITTWOCH  MERCREDI  MERCOLEDI '),
        2  CHAR_???          CHAR(0030)
          INIT ( 'DONNERSTAGJEUDI     GIOVEDI   '),
        2  CHAR_???          CHAR(0030)
          INIT ( 'FREITAG   VENDREDI  VENERDI   '),
        2  CHAR_???          CHAR(0030)
          INIT ( 'SAMSTAG   SAMEDI    SABATO    '),
        2  CHAR_???          CHAR(0030)
          INIT ( 'SONNTAG   DIMANCHE  DOMENICA  '),
        2  CHAR_???          CHAR(0030)
          INIT ( 'MONTAG    LUNDI     LUNEDI    '),
        2  HEX_????          CHAR(0002)
          INIT (X'FF'),
        2  TATAB             FIXED BINARY (15)
          INIT (  0 ),
        2  BINARY_?          FIXED BINARY (15)
          INIT (  31 ),
        2  BINARY_?          FIXED BINARY (15)
          INIT (  59 ),
        2  BINARY_?          FIXED BINARY (15)
          INIT (  90 ),
        2  BINARY_?          FIXED BINARY (15)
          INIT (  120 ),
        2  BINARY_?          FIXED BINARY (15)
          INIT (  151 ),
        2  BINARY_?          FIXED BINARY (15)
          INIT (  181 ),
        2  BINARY_?          FIXED BINARY (15)
          INIT (  212 ),
        2  BINARY_?          FIXED BINARY (15)
          INIT (  243 ),
        2  BINARY_?          FIXED BINARY (15)
          INIT (  273 ),
        2  BINARY_?          FIXED BINARY (15)
          INIT (  304 ),
        2  BINARY_?          FIXED BINARY (15)
          INIT (  334 ),
```

```
        2  HEX_????            CHAR(0002)
           INIT (X'FF'),
;   /* END_DECLARE */
/*****************************************************************/
/*    C O N S T A N T    D A T A  D E C L A R A T I O N S        /*
DECLARE 1  VARIABLEN_DATEN ,
DECLARE 1  DDMMYY       (0001) ,
        2  DD                CHAR(0002) ,
        2  MM                CHAR(0002) ,
        2  YY                CHAR(0002) ,
        2  DDP               CHAR(0002) ,
        2  MMP               CHAR(0002) ,
        2  YYP               CHAR(0002) ,
        2  DDB               FIXED BINARY (15) ,
        2  MMB               FIXED BINARY (15) ,
        2  YYB               FIXED BINARY (15) ,
        2  SPC               CHAR(0001) ,
        2  LRS               CHAR(0001) ,
        2  DOPP         (0064)  BIT (1) ,
        2  P16               CHAR(0001) ,
        2  W1                CHAR(0009) ,
;   /* END_DECLARE */
/*****************************************************************/
DECLARE 1  MASK_DATEN ,
        2  MASK_CODE           FIXED BINARY (15) ;
;   /* END_DECLARE */
/*****************************************************************/
;   /* END_DECLARE */
/*****************************************************************/
DECLARE 1  ADDRESS_DATEN ,
        2  DOPP_6_2            CHAR(0002) ,
        2  X           CHAR (0010),
        2  FILLER_001  DEFINED X         ,
           3  X_0_10_R5     CHAR (0010),
        2  FILLER_002  DEFINED X         ,
           3  X_0_R1        CHAR (0001),
        2  FILLER_003  DEFINED X         ,
           3  X_0_R5        CHAR (0001),
        2  FILLER_004  DEFINED X         ,
           3  X_0_R8        CHAR (0001),
        2  FILLER_005  DEFINED X         ,
           3  PADING_005    CHAR (0001),
           3  X_1_9_R5      CHAR (0009),
/*****************************************************************/
;   /* END_ADDR    */
/*****************************************************************/
/*****************************************************************/
/*   P L / 1   P R O C E D U R E                                 */
/* ADDR & LNG  A S S I G N M E N T S                             */
/*****************************************************************/
     /*!!!! ASSEMBLER STMT WAS NOT CONVERTED !!!!!!*/
/*       XM059   EWIGER KALENDER 1901 - 1999                     */
/*       PARAMETER                                               */
/*        -------                                                */
/*                                                               */
/*           P1: DATUM TT,MM,JJ          8 BYTES  AN           */
/*                                                               */
/*           P2: SPRACHCODE 1,2,3                                */
/*             WENN UNGUELTIG = 1        1 BYTE   AN           */
/*                                                               */
/*           P3: ALINIERUNG L (LINKS)                            */
/*                     R (RECHTS)                                */
```

```
/*              WENN UGUELTIG = L       1 BYTE   AN          */
/*          P4: WOCHENTAG OUTPUT     10 BYTES  AN             */
/*                                                            */
/*                                                            */
/*          WENN P1 NICH PLAUSIBEL, WIRD P4 MIT '?' GEFUELLT     */
/*                                                            */
/*          DIESER KALENDER BERUHT AUF DER TATSACHE, DASS AM     */
/*          1. JANUAR 1901   D I E N S T A G   WAR.           */
/*                                                            */
     SAVE_4      =  R13       ;
     R11         =  ADDR(SAVE       );
     X_8_R13     =  R11       ;
     R13         =  ADDR(SAVE       );
/*************************************************************** */
/*          CHECK INPUT FOR VALIDITY                          */
     DD          =  X_0_R2     ;
     MM           =  X_3_R2     ;
     YY          =  X_6_R2     ;
     SPC         =  X_0_R3     ;
     LRS         =  X_0_R4     ;
/*                                                            */
     P16         =   '0'       ;
     CALL  XM016   (DDMMYY,P16 );
   IF  P16         =     '0'
     GO TO A1          ;                /*FORWARD  JUMP ===>> */
     X_0_10_R5    =   10C'?'    ;
     GO TO RET         ;                 /*FORWARD  JUMP ===>> */
 A1:
   IF  SPC         =     '1'
     GO TO A3          ;                /*FORWARD  JUMP ===>> */
   IF  SPC         =     '2'
     GO TO A3          ;                /*FORWARD  JUMP ===>> */
   IF  SPC         =     '3'
     GO TO A3          ;                /*FORWARD  JUMP ===>> */
     SPC         =   '1'       ;
 A3:
   IF  LRS         =     'L'
     GO TO A5          ;                /*FORWARD  JUMP ===>> */
   IF  LRS         =     'R'
     GO TO A5          ;                /*FORWARD  JUMP ===>> */
     LRS         =   'L'       ;
 A5:
     DDP          = SESPACK  (DD        , ZONED-LNG , PACKED-LNG);
     MMP          = SESPACK  (MM        , ZONED-LNG , PACKED-LNG);
     YYP          = SESPACK  (YY        , ZONED-LNG , PACKED-LNG);
     R7          =  3          ;
     R8          =  ADDR(DDP_2       );
     R9          =  ADDR(DDB_2       );
 B1:
     R8          =  ADDR(X_2_R8      );
     R9          =  ADDR(X_2_R9      );
     DOPP         = SESXCOR  (DOPP      , OPRN-LNG  );
     DOPP_6_2     =  X_0_R8     ;
     R1          =  DOPP       ;
     X_0_R9       =  R1         ;
     R7          =  R7          -  1          ;
   IF  R7          >     ZERO
     GO TO B1          ;                /*BACKWARD LOOP <<=== */
/*                                                            */
/************************************************************** */
/*                                                            */
     R1          =  YYB        ;
```

```
    R1       = R1       *    365    ;
    YTT      = R1       ;
/*********************************************************************** */
/*                                                                       */
/*          KORREKTUR SCHALTJAHRE                                        */
/*                                                                       */
    R9       = YYB      ;
    R1       = 4        ;
    R8       = R8       / R1       ;
  IF MASK_CODE   =   0
    GO TO D1       ;              /*FORWARD  JUMP ===>> */
  IF MMB       >     2
    GO TO D1       ;              /*FORWARD  JUMP ===>> */
D1:
    STT      = R9       ;
/*********************************************************************** */
    R8       = ADDR(TATAB      );
    R9       = MMB      ;
    R8       = R8       + R9       ;
    R9       = X_0_R8   ;
    MTT      = R9       ;
    R9       = YTT      ;
    R9       = R9       + STT      ;
    R9       = R9       + MTT      ;
    R9       = R9       + DDB      ;
    R1       = 7        ;
    R8       = R8       / R1       ;
    R8       = R8       *    30    ;
  IF SPC       =    '1'
    GO TO E3       ;              /*FORWARD  JUMP ===>> */
    R8       = ADDR(X_10_R8    );
  IF SPC       =    '2'
    GO TO E3       ;              /*FORWARD  JUMP ===>> */
    R8       = ADDR(X_10_R8    );
 E3:
    R9       = ADDR(WOTAB      );
    R8       = R8       + R9       ;
    X_0_10_R5   = X_0_R8    ;
  IF LRS       =    'L'
    GO TO E9       ;              /*FORWARD  JUMP ===>> */
    R1       = ADDR(X_9_R5     );
    R2       = 7        ;
 E4:
  IF X_0_R1     ^ =   ' '
    GO TO E9       ;              /*FORWARD  JUMP ===>> */
    W1       = X_0_R5   ;
    X_1_9_R5    = W1       ;
    X_0_R5     =   ' '     ;
    R2       = R2       - 1        ;
  IF R2        >    ZERO
    GO TO E4       ;              /*BACKWARD LOOP <<=== */
 E9:
/*                                                                       */
 RET:
    R13      = SAVE_4   ;
    R15      = R15      - R15      ;
    RETURN;
/*                                                                       */
/*********************************************************************** */
 SAVE:
 DDMMYY:
/*                                                                       */
```

```
/*************************************************************** */
     RETURN;
 /*************************************************************************/
 /*| ASSEMBLER LINES PROCESSED 0152 */
 /*| COMMENT   LINES PROCESSED 0053 */
 /*| PL/I      LINES PRODUCED  0222 */
 /*************************************************************************/
```

6.9 Converted COBOL COPY Member from Assembler

```
COB   01  X-TABLE-02       OCCURS 0001 TIMES.
 ASM  *        DC    4C'/',CL8'B06A',CL11'17.07.2002',CL5'16.09',4C'/'
 COB      02  X-NONAME-003               PIC X(0001)
 COB        VALUE  '/'
 COB                              OCCURS 0004 TIMES.
 COB   01  X-TABLE-03      OCCURS 0001 TIMES.
 ASM  *R010003  DC    F'0'                 * SICHERUNGSBEREICH R1
 COB      02  R010003                  PIC 9(0008) USAGE COMP    VALUE   0.
 ASM  *R140003  DC    F'0'                 * SICHERUNGSBEREICH R14
 COB      02  R140003                  PIC 9(0008) USAGE COMP   VALUE   0.
 ASM  *R150003  DC    F'0'                 * SICHERUNGSBEREICH R15
 COB      02  R150003                  PIC 9(0008) USAGE COMP   VALUE   0.
 ASM  *R14MODE  DC    F'0'                  * SICHERUNGSBER. R14 UND
 COB      02  R14MODE                  PIC 9(0008) USAGE COMP  VALUE   0.
 ASM  *        LTORG
 ASM  *PCBADR   DC    F'0'            * ADRESS-SICHERUNGS-BEREICH FUE
 COB      02  PCBADR                  PIC 9(0008) USAGE COMP   VALUE   0.
 COB   01  X-TABLE-04      OCCURS 0001 TIMES.
 ASM  *KCSECT   DC    CL8'B06A'
 COB      02  KCSECT                  PIC X(0008)   VALUE  'B06A'.
 ASM  *JOBNAME  DC    CL8' '           * JOBNAME
 COB      02  JOBNAME                  PIC X(0008)     VALUE  ' '.
 ASM  *BSHPSZ01 DC    A(Z01)           * SICHERUNGSBEREICH VON Z01
 COB      02  BSHPSZ01                 POINTER.
 ASM  *SAVE_B06 DC    18F'0'           * SAVE-AREA
 COB      02  SAVE_B06                 PIC 9(0008) USAGE COMP  VALUE   0.
COB                               OCCURS 0018 TIMES.
 ASM  *BSHPSKZ1 DC    CL8'BSHPSV02'     * MACRO-KENNZEICHEN 01.12.83
COB      02  BSHPSKZ1                 PIC X(0008)     VALUE  'BSHPSV02'.
ASM  *BSHPSKZ2 DC    CL2'AH'           * HPRO / UPRO KENNZEICHEN
 COB      02  BSHPSKZ2                 PIC X(0002)   VALUE  'AH'.
ASM  *BSHPSKZ3 DC    B'00000000'       * INTERNE BITSCHALTER
 COB      02  BSHPSKZ3                 PIC X(0002)    VALUE  '00000000'.
ASM  *BSHPSKZ4 DC    B'00100000'       * INTERNE BITSCHALTER MIT IMS-K
 COB      02  BSHPSKZ4                 PIC X(0002)   VALUE  '00100000'.
ASM  *BSHPSBAS DC    F'0'              * PGM-START-ADRESSE (1.BASISREG
 COB      02  BSHPSBAS                 PIC 9(0008) USAGE COMP    VALUE   0.
ASM  *BSHPSKZ5 DC    H'0'              * STEUERBYTE FUER U300A
 COB      02  BSHPSKZ5                 PIC 9(0005) USAGE COMP   VALUE   0.
ASM  *BSHPSKZ6 DC    H'0'              * STEUERBYTE FUER U301A
 COB      02  BSHPSKZ6                 PIC 9(0005) USAGE COMP    VALUE   0.
ASM  *BSHPSKZ7 DC    F'0'              * ADR. DES AUSGABEBER. FUER U300
 COB      02  BSHPSKZ7                 PIC 9(0008) USAGE COMP  VALUE   0.
ASM  *BSHPSKZ8 DC    F'0'              * RUECKGABE-BEREICHS-ADR. AUS
COB      02  BSHPSKZ8                 PIC 9(0008) USAGE COMP  VALUE   0.
ASM  *BSHPSKZ9 DC    A(PGMENDE-B06A)    * PROGRAMM-LAENGE
 COB      02  BSHPSKZ9                 POINTER.
 ASM  *BSHPSKZA DC    F'0'              * MEA-UEBERGABE-BEREICHS-ADRESS
 COB      02  BSHPSKZA                 PIC 9(0008) USAGE COMP   VALUE   0.
ASM  *BSHPSKZB DC    F'0'              * MEA-RUECKSPRUNG-ADRESSE - 4
```

```
 COB      02  BSHPSKZB                PIC 9(0008) USAGE COMP     VALUE   0.
ASM  *BSHPSKZC DC    F'0'            * RESERVE
 COB      02  BSHPSKZC                PIC 9(0008) USAGE COMP    VALUE   0.
ASM  *BSHPSKZD DC    F'0'            * UPRO-ADRESS-TABELLE VON U301A
 COB      02  BSHPSKZD                PIC 9(0008) USAGE COMP  VALUE   0.
ASM  *BSHPSKZE DC    F'0'            * U301A EP-ADRESSE
 COB      02  BSHPSKZE                PIC 9(0008) USAGE COMP  VALUE   0.
ASM  *BSHPSKZF DC    V(U301C)          * U301C-CHECKPOINT-BEREICH
 COB      02  BSHPSKZF               POINTER.
 ASM  *BSHPSKZG DC    CL1'N',3X'00'     * INTERNE PGM-KENNZEICHNUNG
 COB      02  BSHPSKZG                PIC X(0001)  VALUE  'N'.
ASM  *BSHPSKZH DC    A(J5ZZAD)         * ADR. DES UPRO-NAMEN U. EP-ADR
COB      02  BSHPSKZH               POINTER.
 ASM  *BSHPSKZI DC    F'0'            * INT. ADR. F. MUPRO UR= VERARB
 COB      02  BSHPSKZI               PIC 9(0008) USAGE COMP   VALUE   0.
ASM  *BSHPSKZJ DC    F'0'            * INTERNER ADRESS-BEREICH
 COB      02  BSHPSKZJ               PIC 9(0008) USAGE COMP   VALUE   0.
ASM  *BSHPSKZK DC    F'0'             * ADR. DER WE_BASIS-TAB UEBER W
 COB      02  BSHPSKZK                PIC 9(0008) USAGE COMP  VALUE   0.
ASM  *BSHPSKZL DC    F'0'            * INTERNER ADRESS-BEREICH
 COB      02  BSHPSKZL                PIC 9(0008) USAGE COMP    VALUE   0.
ASM  *BSHPSKZM DC    F'0'             * INTERNER ADRESS-BEREICH
 COB      02  BSHPSKZM                PIC 9(0008) USAGE COMP  VALUE   0.
ASM  *       ORG   BSHPSKZL          * UEBERLAGERUNGS-POSITION
 COB   01  REDEF-FIELD     REDEFINES    BSHPSKZL.
 ASM  *        DC    H'0'              * RESERVE
 COB      02  X-NONAME-004               PIC 9(0005) USAGE COMP  VALUE   0.
ASM  *BSHPS_WE DC    CL2'HW'         * WAEHRUNGS-ART DES HPROS = KW,HW.
COB      02  BSHPS_WE                PIC X(0002)      VALUE  'HW'.
ASM  *BSHPS_WE DC    CL3'WEZ'        * WAEHRUNGSEINHEIT -ZIEL- = DM, EU
 COB      02  BSHPS_WE                PIC X(0003)     VALUE  'WEZ'.
ASM  *BSHPS_WE DC    CL1'0'          * W„HRUNGSUMST.-PHASEN-NR.= 1,2,3
 COB      02  BSHPS_WE                PIC X(0001)    VALUE  '0'.
ASM  *       ORG   * ZAEHLER AUF HOECHSTEN STAND SETZEN
ASM  *ADR0041  DC    2F'0'
 COB      02  ADR0041                  PIC 9(0008) USAGE COMP  VALUE   0
COB                              OCCURS 0002 TIMES.
```

```
    *    SATZDEFINITION ERSTATTUNGSANTRAEGE-DATEI
    *       FUER ZUGRIFFE -> ERANSATZ
 *#  TI ERAN/
÷     *=+
             02  EA-BER.
                 03  EA-KEY.
                     04  EA-ORD.
    *                        VORLAUFSATZ = LOW-VALUE
                         05  EA-LFNR.
RENAME                       06  EA-LAUFENDE-NUMMER-PRO-ANTRAG PIC 9(06).
    *                        LAUFENDE NUMMER JE ANTRAG
                    04  EA-ONRE.
RENAME                   05  EA-ORD-NUMMER-PRO-EINZELANTRAG  PIC 9(03).
    *                        ORD-NUMMER EINZELANTRAG
RENAME       03  EA-KZ-STORNIEREN  PIC X(01).
    *                        KZ STORNIEREN BLANK/1
    *                        ONRE=0 LFNR-STORNO (GESANTR)
    *                        ONRE>0 ONRE-STORNO (EINANTR)
                 03  EA-KZ.
                     04  EA-KZ-FIX.
```

```
                    05  EA-KZ-GESANTR   PIC X(01).
                    05  EA-KZ-VORMERK1  PIC X(01).
                    05  EA-KZ-VORMERK2  PIC X(01).
                    05  EA-ANZ-MANANWS  PIC 9(02).
RENAME          04  EA-STEUERFELDER.
                    05  EA-KZ-RVDATEN   PIC X(01).
                    05  EA-KZ-ANSTAUF   PIC X(01).
                    05  EA-KZ-OEKOBEL   PIC X(01).
                    05  EA-ANZ-TEILBET  PIC 9(03).
    *                   STEUERFELDER SATZTEILE 0/1
RENAME      03  EA-EINLAUF-OHNE-KZ         PIC X(110).
    *                   OHNE KZ (IMMER VORHANDEN)
RENAME      03  FILLER  REDEFINES  EA-EINLAUF-OHNE-KZ.
RENAME          04  EA-EINLAUFDATUM        PIC 9(07) .
    *                   JJMMTT EINLAUFDATUM
RENAME          04  EA-KVNR-VERSICHERTER    PIC 9(06).
    *                   KVNR VERSICHERTER
RENAME          04  EA-KVNR  REDEFINES  EA-KVNR-VERSICHERTER.
                    05  EA-ADRV-KZ  PIC X(01).
                    05  EA-ADRV-NR.
RENAME                  06 EA-ADRV-NUMMER     PIC 9(05).
    *                   ODER KZ 'A' + ADRV-NUMMER    TZ
                    04  EA-GEBA-X.
RENAME              05  EA-GEBDAT-ANG   PIC 9(07)        OCCURS 05.
    *                   TTMMJJ GEBDAT ANG
RENAME          04  EA-LEISTUNGSART      PIC X(04).
    *                   LEISTUNGSART KV/6
RENAME          04  EA-VORGANGSART       PIC X(40).
    *                   VORGANG=
    *                   AKTWEITERGABE, ABLEHNUNG, USW
RENAME          04  EA-VORGANGSDATUM    PIC 9(07) .
    *                   JJMMTT VORGANGSDATUM
RENAME          04  EA-RUECKVERGUETUNGSBETRAG   PIC S9(7)V99 .
    *                   RUECKVERGUETUNGSBETRAG LFNR
    *                   (NUR ANZEIGE, AUCH FREMDWAE.)
RENAME          04  EA-ANTRAGSSTATUS   PIC X(01).
    *                    =OFFEN
    *                   E=ERLEDIGUNG
    *                   A=ABLEHNUNG
RENAME          04  EA-PERSONALNUMMER-RATENAUFTEIL  PIC 9(05) .
    *                   PERSONALNUMMER RATENAUFTEILG.
REMOVE          04  EA-PNRE       PIC 9(05) .
    *                   PERSONALNUMMER ERLEDIGUNG
    *                   NUR VON ERAN002,ANW VERWENDET
RENAME          04  EA-KZ-ANTRAG  PIC X(01).
    *                   KZ ANTRAG
    *                   V= NUR VER, A= NUR ANG, X=V+A
RENAME          04  EA-DATUM-ANWEISUNGS-DFG  PIC X(08).
    *                   CURRENT-DATE ANWEISUNGS-DFG
    *                   ODER 'BAGATELL'
    *                   BLANK= FALL OFFEN/ABGELEHNT;
RENAME          04  EA-ANWEISUNGSDATUM    PIC 9(07) .
    *                   JJMMTT ANWEISUNGSDATUM
    *                   0= KEINE ODER WIEDERANWEISUNG
RENAME          04  EA-IKZ-FREMDWAEHRUNG   PIC X(03).
    *                   FREMDWAEHRUNG (IKZ)
RENAME          04  EA-KZ-AUTOMAT-FREIGABE PIC X(01).
REFORM*                         1= KEINE AUTOMAT.
    *                   FREIGABE                Z
RENAME          04  EA-KZ-UNFERTIG                 PIC X(01).
    *                   U= UNFERTIG
RENAME          04  EA-KZ-VERLASSENSCHAFT    PIC X(01).
```

```
    *                       KZ VERLASSENSCHAFT
RENAME          04  EA-KZ-WIEDERANWEISUNG   PIC X(01).
    *                       KZ WIEDERANWEISUNG
    *                       W= DURCHFUEHREN, 1= OFFEN
```

6.10 Converted PL/I INCLUDE Member from Assembler

```
DECLARE 1  SU1         (0001) ,
 /*A:SU111   DC    PL4'0'              SUMMEN GIRO-RENTNER      */
         2  SU111             FIXED DECIMAL (0007)
           INIT (  0 ),
 /*A:SU112   DC    PL6'0'                                       */
         2  SU112             FIXED DECIMAL (0011)
           INIT (  0 ),
 /*A:SU113   DC    PL4'0'                                       */
         2  SU113             FIXED DECIMAL (0007)
           INIT (  0 ),
 /*A:SU114   DC    PL6'0'                                       */
         2  SU114             FIXED DECIMAL (0011)
           INIT (  0 ),
 /*A:SU115   DC    PL7'0'                                       */
         2  SU115             FIXED DECIMAL (0013)
           INIT (  0 ),
 /*A:SU121   DC    PL4'0'              SUMMEN GIRO-KVDR         */
         2  SU121             FIXED DECIMAL (0007)
           INIT (  0 ),
 /*A:SU122   DC    PL6'0'                                       */
         2  SU122             FIXED DECIMAL (0011)
           INIT (  0 ),
 /*A:SU123   DC    PL4'0'                                       */
         2  SU123             FIXED DECIMAL (0007)
           INIT (  0 ),
 /*A:SU124   DC    PL6'0'                                       */
         2  SU124             FIXED DECIMAL (0011)
           INIT (  0 ),
 /*A:SU125   DC    PL7'0'                                       */
         2  SU125             FIXED DECIMAL (0013)
           INIT (  0 ),
 /*A:SU131   DC    PL4'0'              SUMMEN GIRO-PFLEGE       */
         2  SU131             FIXED DECIMAL (0007)
           INIT (  0 ),
 /*A:SU132   DC    PL6'0'                                       */
         2  SU132             FIXED DECIMAL (0011)
           INIT (  0 ),
 /*A:SU133   DC    PL4'0'                                       */
         2  SU133             FIXED DECIMAL (0007)
           INIT (  0 ),
 /*A:SU134   DC    PL6'0'                                       */
         2  SU134             FIXED DECIMAL (0011)
           INIT (  0 ),
 /*A:SU135   DC    PL7'0'                                       */
         2  SU135             FIXED DECIMAL (0013)
           INIT (  0 ),
 /*A:SU141   DC    PL4'0'              SUMMEN POST-RENTNER      */
         2  SU141             FIXED DECIMAL (0007)
           INIT (  0 ),
 /*A:SU142   DC    PL6'0'                                       */
         2  SU142             FIXED DECIMAL (0011)
           INIT (  0 ),
 /*A:SU143   DC    PL4'0'                                       */
```

```
        2  SU143             FIXED DECIMAL (0007)
             INIT (  0 ),
/*A:SU144   DC   PL6'0'                              */
        2  SU144             FIXED DECIMAL (0011)
             INIT (  0 ),
/*A:SU145   DC   PL7'0'                              */
        2  SU145             FIXED DECIMAL (0013)
             INIT (  0 ),
/*A:SU151   DC   PL4'0'              SUMMEN INSGESAMT        */
        2  SU151             FIXED DECIMAL (0007)
             INIT (  0 ),
/*A:SU152   DC   PL7'0'                              */
        2  SU152             FIXED DECIMAL (0013)
             INIT (  0 ),
/*A:SU153   DC   PL4'0'                              */
        2  SU153             FIXED DECIMAL (0007)
             INIT (  0 ),
/*A:SU154   DC   PL7'0'                              */
        2  SU154             FIXED DECIMAL (0013)
             INIT (  0 ),
/*A:SU155   DC   PL7'0'                              */
        2  SU155             FIXED DECIMAL (0013)
             INIT (  0 ),
/*A:SU156   DC   PL4'0'              SUMMEN DYNAMISIERUNG    */
        2  SU156             FIXED DECIMAL (0007)
             INIT (  0 ),
/*A:SU157   DC   PL7'0'                              */
        2  SU157             FIXED DECIMAL (0013)
             INIT (  0 ),
;   /* END_DECLARE */
DECLARE 1  SU2         (0001) ,
/*A:SU211   DC   PL4'0'              SUMMEN GIRO-RENTNER     */
        2  SU211             FIXED DECIMAL (0007)
             INIT (  0 ),
/*A:SU212   DC   PL6'0'                              */
        2  SU212             FIXED DECIMAL (0011)
             INIT (  0 ),
/*A:SU213   DC   PL4'0'                              */
        2  SU213             FIXED DECIMAL (0007)
             INIT (  0 ),
/*A:SU214   DC   PL6'0'                              */
        2  SU214             FIXED DECIMAL (0011)
             INIT (  0 ),
/*A:SU215   DC   PL7'0'                              */
        2  SU215             FIXED DECIMAL (0013)
             INIT (  0 ),
/*A:SU221   DC   PL4'0'              SUMMEN GIRO-KVDR        */
        2  SU221             FIXED DECIMAL (0007)
             INIT (  0 ),
/*A:SU222   DC   PL6'0'                              */
        2  SU222             FIXED DECIMAL (0011)
             INIT (  0 ),
/*A:SU223   DC   PL4'0'                              */
        2  SU223             FIXED DECIMAL (0007)
             INIT (  0 ),
/*A:SU224   DC   PL6'0'                              */
        2  SU224             FIXED DECIMAL (0011)
             INIT (  0 ),
/*A:SU225   DC   PL7'0'                              */
        2  SU225             FIXED DECIMAL (0013)
             INIT (  0 ),
/*A:SU231   DC   PL4'0'              SUMMEN GIRO-PFLEGE      */
```

```
          2  SU231              FIXED DECIMAL (0007)
             INIT (  0 ),
/*A:SU232    DC    PL6'0'                          */
          2  SU232              FIXED DECIMAL (0011)
             INIT (  0 ),
/*A:SU233    DC    PL4'0'                          */
          2  SU233              FIXED DECIMAL (0007)
             INIT (  0 ),
/*A:SU234    DC    PL6'0'                          */
          2  SU234              FIXED DECIMAL (0011)
             INIT (  0 ),
/*A:SU235    DC    PL7'0'                          */
          2  SU235              FIXED DECIMAL (0013)
             INIT (  0 ),
/*A:SU241    DC    PL4'0'              SUMMEN POST-RENTNER      */
          2  SU241              FIXED DECIMAL (0007)
             INIT (  0 ),
/*A:SU242    DC    PL6'0'                          */
          2  SU242              FIXED DECIMAL (0011)
             INIT (  0 ),
/*A:SU243    DC    PL4'0'                          */
          2  SU243              FIXED DECIMAL (0007)
             INIT (  0 ),
/*A:SU244    DC    PL6'0'                          */
          2  SU244              FIXED DECIMAL (0011)
             INIT (  0 ),
/*A:SU245    DC    PL7'0'                          */
          2  SU245              FIXED DECIMAL (0013)
             INIT (  0 ),
/*A:SU251    DC    PL4'0'              SUMMEN INSGESAMT         */
          2  SU251              FIXED DECIMAL (0007)
             INIT (  0 ),
/*A:SU252    DC    PL7'0'                          */
          2  SU252              FIXED DECIMAL (0013)
             INIT (  0 ),
/*A:SU253    DC    PL4'0'                          */
          2  SU253              FIXED DECIMAL (0007)
             INIT (  0 ),
/*A:SU254    DC    PL7'0'                          */
          2  SU254              FIXED DECIMAL (0013)
             INIT (  0 ),
/*A:SU255    DC    PL7'0'                          */
          2  SU255              FIXED DECIMAL (0013)
             INIT (  0 ),
/*A:SU256    DC    PL4'0'              SUMMEN DYNAMISIERUNG     */
          2  SU256              FIXED DECIMAL (0007)
             INIT (  0 ),
/*A:SU257    DC    PL7'0'                          */
          2  SU257              FIXED DECIMAL (0013)
             INIT (  0 ),
;   /* END_DECLARE */
```

6.11 Java Controller

```
package cobol2java.frame;
import java.util.Vector;
```

```
/**
 * This class is the abstract class for all programs. It contains all storage areas.
 * Its runStateMachine method represents the control flow implementation.
 * @author Stephan Sneed
 * @version 2.0
 */
public abstract class Controller {
  /** The string constant for setting xNextLabel to termination */
  private static final String STOP = "STOP";
  /** The string constant for the WORK storage */
  public static final String WORKStorage = "WORK";
  /** The string constant for the INPUT storage */
  public static final String INPUTStorage = "INPUT";
  /** The string constant for the OUTPUT storage */
  public static final String OUTPUTStorage = "OUTPUT";
  /** The string constant for the CACHE storage */
  public static final String CACHEStorage = "CACHE";
  /** The string constant for the IDSDB storage */
  public static final String IDSDBStorage = "IDSDB";
  /** The string constant for the MAIN module */
  public static final String  MAINmodule = "MAIN";
   /**
   * Default Constructor.
   */
  public Controller() {
  }
    /**
   * This recursive method is an implementation of a finite state machine
   * @param processingInfoIn The processing info with information where to jump to
   * @throws Exception if no matching class found for the classId of processingInfoIn
   */
  public void runStateMachine(ProcessingInfo processingInfoIn) throws Exception {
    String xNextMethod = processingInfoIn.getXNextMethod();
    String storage = processingInfoIn.getStorage();
    ProcessingInfo processingInfoOut = null;
    System.out.println("----------------");
    if (!xNextMethod.equals(STOP)) {
      COBOLObject cobolObject = this.getCOBOLObject(processingInfoIn);
      if (storage.equals(MAINmodule)) {
          processingInfoIn = getLastProcessingInfo();
      }
      if (cobolObject != null) {
        processingInfoOut = cobolObject.performOperation(processingInfoIn);
        runStateMachine(processingInfoOut);
      } else {
        throw new Exception("No matching class found for label: "+xNextMethod);
      }
    } else {
      System.out.println("STATE MACHINE TERMINATED PROPERLY");
    }
  }
  /**
   * Returns the COBOLObject specified by the processinInfoIn from the storage area
   * also specified by the processingInfoIn.
   * @param processingInfoIn
   * @return The demanded COBOLObject from the demanded storage area
   */
  public abstract COBOLObject getCOBOLObject(ProcessingInfo processingInfoIn);
  /**
   * Returns the the changed processinginfo after a "MAIN" method call
   * @return The processinginfo after a "MAIN" method call
   */
```

```
 public abstract ProcessingInfo getLastProcessingInfo();
}
```

6.12 Java Processing Information

```
package cobol2java.frame;
import java.util.Stack;
import at.fwag.cobol2java.frame.Controller;
/**
 * An object of this class is used to determine the next jump to another
 * object[type:dataObject] and to one of its methods. Its variable xNextLabel
 * contains this information.
 */
public class ProcessingInfo {
	/** TODO: Declare Structure!! */
	private String xNextMethod;
	/** The storage program of xNextLabel */
	private String program;
	/** The storage substring of xNextLabel */
	private String storage;
	/** The class id substring of xNextLabel */
	private String classId;
	/** The method id substring of xNextLabel */
	private String methodId;
	/**
	 * The default constructor, initialized with a xNextMethod string
	 * @param xNextMethod {SOTRAGE:ClassId:MethodId}
"PH74:WORK.WRK_PH74_WORKING.PH74_R1"
	 */
	public ProcessingInfo(String xNextMethod) {
		this.xNextMethod = xNextMethod;
		if (xNextMethod.contains(":")) {
			if(xNextMethod.contains(".")) {
			int dpointPos = xNextMethod.indexOf(":");
			int spointPos1 = xNextMethod.indexOf(".");
			int spointPos2 = xNextMethod.indexOf(".",spointPos1 + 1);
			this.program = xNextMethod.substring(0, dpointPos);
			this.storage = xNextMethod.substring(dpointPos+1, spointPos1);
			this.classId = xNextMethod.substring(spointPos1+1, spointPos2);
			this.methodId = xNextMethod.substring(spointPos2+1);
		}
			else {
				int dpointPos = xNextMethod.indexOf(":");
				this.program = xNextMethod.substring(0, dpointPos);
				this.storage = Controller.MAINmodule;
				this.classId = this.program;
				this.methodId = xNextMethod.substring(dpointPos+1);
			}
		}
		else
		if(!xNextMethod.contains(".")) {
			this.program = "?";
			this.storage = "?";
			this.classId = "?";
			this.methodId = xNextMethod;
		}

	}

	/**
	 * Returns the xNextLabel
```

```
	 * @return xNextLabel
	 */
	public String getXNextMethod() {
		return this.xNextMethod;
	}
	/**
	 * Returns the storage substring of xNextLabel
	 * @return classId
	 */
	public String getStorage() {
		return this.storage;
	}
	/**
	 * Returns the program substring of xNextLabel
	 * @return classId
	 */
	public String getProgram() {
		return this.program;
	}
	/**
	 * Returns the classId substring of xNextLabel
	 * @return classId
	 */
	public String getClassId() {
		return this.classId;
	}

	/**
	 * Returns the methodId substring of xNextLabel
	 * @return methodId
	 */
	public String getMethodId() {
		return this.methodId;
	}
}
```

6.13 Java Storage Classes

```
package cobol2java.frame;
import java.util.Vector;
/**
 * Abstract class storage. Represents a storage area in the original COBOL74 program.
 * @author Stephan Sneed
 * @version 2.0
 */
@SuppressWarnings({ "unchecked", "serial" })
public abstract class Storage extends Vector {
	/**
	 * This method returns the demanded cobol object that is specified by the
	 * class Id input substring of xNextLAbel
	 * @param classId A substring of xNextLabel
	 * @return The demanded COBOL object, null if it is not yet initialized
	 */
	public COBOLObject getSingletonByClassId(String classId) {
		COBOLObject cobolObject = null;
		for(int i=0;i<this.size();i++) {
			cobolObject = (COBOLObject)this.elementAt(i);
			String fullClassName = cobolObject.getClass().toString();
			String className = fullClassName.substring(fullClassName.lastIndexOf(".")+1);
			if (className.equals(classId))break;
		}
```

```
                return cobolObject;
        }
}
```

6.14 Java COBOL Object

```
package cobol2java.frame;
import static java.lang.Math.round;
import static java.lang.Math.floor;
import java.util.ArrayList;
import java.util.List;
/**
 * This abstract class is the parent class of all cobol object classes.
 * Each COBOL object defines some operations on a char[] array which represents
 * the original COBOL data structure. This class provides all the necessary
 * operations for the manipulation and data provisioning of the char[] array.
 * Thus, a COBOL data structure is represented by a char[] array.
 * A COBOL data field is represented by a position and a length, defining a sub
 * sequence of this char[] array.
 * @author Stephan Sneed
 */
public abstract class COBOLObject {
 /** Constant for stopping the state machine when assigning xNextLabel with */
 public final static String STOP = "STOP";

 /**
  * Default Constructor.
  */
 public COBOLObject() {
 }
 /**
  * Abstract definition of the method each data object as to implement. This
  * method decides, which of the provided methods of the COBOL object should
  * be executed, using the methodId of the incoming processing information object.
  * @param processingInfoIn The processing info deciding which method to call.
  * @return The processing info that defines the next class and method to jump to.
  * @throws Exception if no matching method found for the methodId of processingInfoIn.
  */
 public abstract ProcessingInfo performOperation(ProcessingInfo processingInfoIn) throws Exception;
 /**
  * This method returns a char[] of blanks with the specified length.
  * @param length The demanded length of the spaces string
  * @return spaces A char[] of spaces with the specified length
  */
 public String xSpaces(int length) {
   char[] xSpaces = new char[length];
   for (int i=0;i<length;i++) {
     xSpaces[i] = ' ';
   }
   return new String(xSpaces);
 }


 /**
  * This method returns a char[] of zeros with the specified length.
  * @param length The demanded length of the zeros string
  * @return A char[] of zeros with the specified length
  */
 public String xZeros(int length) {
   String xZeros = new String();
   for (int i=0;i<length;i++) {
```

```
    xZeros = xZeros.concat("0");
  }
  return xZeros;
}

/**
 * This method returns a the integer input with leading zeros suppressed<br>
 * This routine is simulates the COBOL "Z" and "-" PICTURE character behavior
 * @param intValue The integer to be converted
 * @param length The demanded length of the string
 * @return A char[] of zeros with the specified length
 */
public String xZeroSuppresss(int intValue, int length) {
        String work = (new Integer(intValue)).toString();
        char [] xZeroSuppresss ;
  for (int i=0;i<(length - work.length());i++) {
        if (intValue >= 0)
           work = "0".concat(work);
        else
                 work = " ".concat(work);
  }
        xZeroSuppresss = work.toCharArray();
  for (int i=0;i<work.length()-1;i++) {
        if (xZeroSuppresss[i] == '0') {
                xZeroSuppresss[i] =  ' ';
        }
        else {
                break;
        }
  }
  return new String(xZeroSuppresss);
}

/**
 * This method returns a the string input with leading "what" characters replaced by "with " characters
 * @param source The String to be converted
 * @param what The leading char to be replaced
 * @param with The replacement character
 * @return the modified string
 */

public String replaceLeading(String source,char what, char with ) {
    char[] temp = source.toCharArray();
    for (int i = 0;i < source.length(); i++) {
       if ( temp[i] == what )
            temp[i] = with ;
       else
            break;
    }
    return  new String(temp);
}

/**
 * This method returns the input string(what) repated repetition_number times.
 * Primarily intended for the implemantation of the MOVE ALL <string> .... COBOL statement
 * @param repetition_number teh number of repetitions
 * @param what the string to repeat
 * @return a string consisting of the repetition of the input
 */
public String xRepeat(int repetition_number , String what) {
  String xRepeat = new String();
  for (int i=0;i<repetition_number;i++) {
```

```
            xRepeat = xRepeat.concat(what);
    }
    return xRepeat;
  }

  /**
   * This method implements the COBOL statement <br>
   *  INSPECT <instring> TALLYING <start_number> FOR ALL <substr>.
   *
   * @param instring the string to inspect
   * @param substr the string of to count in instring
   * @param start_number this value is increased by the number of occurrences os substr in instring
   * @return a string consisting of the repetition of the input
   */

  public int xInspectAll(String instring, String substr, int start_number ) {
  {
     int i = 0;
     int j = 0;
     int k = instring.length();
     int l = substr.length();
     while (i < (k - l)) {
       if (instring.substring(i,i + l).equals(substr)) {
         i += l;
         j++;
       }
       else {
         i++;
       }

     }
     return j + start_number;
    }
  }

  /**
   * This method overwrites the character subsequence of the static char array
   * determined by the start position and the length with the characters of the
   * source string. If the string is shorter than the determined length of the
   * character sequence, it is filled up with spaces.
   * @param staticStorage The char[] array that represents this data structure
   * @param source The string that should be set
   * @param position The start position within the char[] array (data structure)
   * @param length The length of the string within the char[] array (data structure)
   */
  public void setAsChar(char[] staticStorage, String source, int position, int length) {
        char[] temp = source.toCharArray();
        int end = position + length;
        for (int i = position,j = 0;i < end; i++,j++) {
            if (j < source.length()) staticStorage [i] =temp[j];
          else staticStorage [i] = ' ';
          }
    }

 /**
  * This method overwrites the character subsequence of the static char array
  * determined by the position and the length of the data field. The number
  * will be shifted to the right and the remaining characters on the left side
  * will be filled up with zeros.
  * @param staticStorage The char[] array that represents this data structure
  * @param source The integer that should be set
  * @param position The position of the integer data field within the char[] array
```

```
* @param length The length of the integer data field
*/
 public void setAsChar(char[] staticStorage, int source, int position, int length) {
    setAsCharAlignRight(staticStorage,convertIntToString(source),position,length);
 }
 /**
* This method overwrites the character subsequence of the static char array
* determined by the position and the length of the data field. The number
* will be shifted to the right and the remaining characters on the left side
* will be filled up with zeros.
* @param staticStorage The char[] array that represents this data structure
* @param source The double that should be set
* @param position The position of the data field within the char[] array
* @param length The length of the  data field
* @param precision The number of digits after the decimal point to be kept when storing the double value
*/
 public void setAsChar(char[] staticStorage, double source, int position, int length ,int precision) {
    setAsCharAlignRight(staticStorage,convertDoubleToString(source,precision),position,length);
 }

 /**
  * This method does the same like the setAsChar() method but it is meant for
  * setting integer numbers. Therefore it aligns the string representation of the
  * number to the right side of the data field and fills up the remaining characters
  * on the left side with zeros.
  * @param staticStorage The char[] array that represents this data structure
  * @param source The integer to be set (as String representation)
  * @param position The position of the data field
  * @param length The length of the data field
  */
 public void setAsCharAlignRight(char[] staticStorage, String source, int position, int length) {
      char[] asource = source.toCharArray();
      for (int i = position;i < position + length;i++) {
           staticStorage [i] = '0';
      }
      int firstdigitposition = position;
      if (asource[0] == '-')
      {
        asource[0] = '0';
        staticStorage [position] = '-';
        firstdigitposition++;
      }
      for (int outputposition = position + length - 1,inputposition = source.length() - 1;
            outputposition >= firstdigitposition && inputposition >= 0;
            outputposition--,inputposition--) {
           staticStorage [outputposition] = asource[inputposition];
       }
  }

 /**
  * This method returns the data field (a sub sequence of the char[] array)
  * specified by the position and the length.
  * @param staticStorage The char[] array
  * @param position The position of this data field in the char[] array
  * @param length The length of the data field
  * @return The string representation of this data field (sub sequence of the char[] array)
  */
 public String getString(char[] staticStorage, int position, int length) {
    return new String(staticStorage).substring(position,position + length);
 }
   /**
  * This method returns the data field (a sub sequence of the char[] array)
```

```
 * specified by the position and the length.
 * @param staticStorage The char[] array
 * @param position The position of this data field in the char[] array
 * @param length The length of the data field
 * @return The integer representation of of this data field (sub sequence of the char[] array)
 */
public int getInteger(char[] staticStorage, int position, int length) {
   return convertToInteger(new String(staticStorage).substring(position,position + length).trim());
  }
 /**
 * This method returns the data field (a sub sequence of the char[] array)
 * specified by the position and the length.
 * @param staticStorage The char[] array
 * @param position The position of this data field in the char[] array
 * @param length The length of the data field
 * @param precision The number of digits after the decimal point to be kept when storing the double value
 * @return The double representation of of this data field (sub sequence of the char[] array)
 */
public double getDouble(char[] staticStorage, int position, int precision, int length) {
  return convertToDouble(new String(staticStorage).substring(position,position + length).trim()) / precision ;
}
 /**
 * This method converts a string value to an integer.
 * @param stringValue The string representation of an integer
 * @return The native integer representation of a number string
 */
  public int convertToInteger(String stringValue) {
          stringValue = stringValue.trim();
          if (stringValue.matches(NumericMatch)) {
                   return (new Integer(stringValue).intValue());
          }
          else {
                   return 0;
          }
  }
 /**
 * This method converts a string value to a double.
 * @param doubleValue The string representation of a double
 * @return The native double representation of a number string
 */
  public double convertToDouble (String doubleValue) {
         doubleValue = doubleValue.trim();
                   if (doubleValue.matches(NumericMatch)) {
                            return (new Double(doubleValue).doubleValue());
                   }
                   else {
                            return 0;
                   }
  }
 /**
 * This method converts an integer value into a string representation.
 * @param intValue The integer value
 * @return The string representation of the integer value
 */
public String convertIntToString (int intValue) {
   return (new Integer(intValue)).toString();
  }

/**
 * This method converts an integer value into a string representation with a given length.
 * @param intValue The integer value
 * @param length The length of the returned string
```

```
   * @return The string representation of the integer value
   */
  public String convertIntToString (int intValue,int length) {
     char[] staticStorage = new char[length];
     String source = convertIntToString (Math.abs(intValue));
     char[] asource = source.toCharArray();
     int position = 0;
     int k = 0;
     int end = position + length;
     for (int i = end-1,j = source.length()-1;i >= position; i--,j--) {
        if (j >= k) staticStorage [i] = asource[j];
          else
            staticStorage [i] = '0';
      }
     return new String(staticStorage);
  }

  /**
   * This method converts a double value into a string representation.
   * @param doubleValue The double value
   * @param precision the required precision
   * @return The string representation of the double value
   */
  public String convertDoubleToString (double doubleValue, int precision) {
     return new Integer(round((float)(doubleValue * precision))).toString() ;
    }
  /**
   * This method returns a truncated double value as integer.
   * @param doubleValue The double value
   * @return The truncated double value as integer
   */
  public int roundDouble (double doubleValue) {
     return new Integer((int)(floor(doubleValue))).intValue() ;
    }
  /**
   * This method returns a rounded double value as integer.
   * @param doubleValue The double value
   * @return The rounded double value as integer
   */
  public int truncateDouble (double doubleValue) {
     return new Integer((int)(floor(doubleValue))).intValue() ;
    }
}
```

6.15 Java Component Interfaces

```
package cobol2java.frame;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.List;
import at.fwag.cobol2java.externalcalls.*;
/**
 * @author Majnár Rudolf<br>
 *
 * This class contains the methods and stubs implementing the COBOL external subroutines
 *
 */
public class ExternalCalls  extends COBOLObject {
```

```
//    private static  ExternalCalls           instance = null;
    protected static  char[]    WORK_AREA;
    public ExternalCalls() {
            WORK_AREA = new char[200];
            WORK_AREA = this.xSpaces(200).toCharArray();
    }

    public  ProcessingInfo performOperation(ProcessingInfo  processingInfoIn) throws Exception {
       return new ProcessingInfo(STOP);
    }
/**
 * Test converted subroutines
 * @param args
 */
    public static void main(String[] args) {
              ExternalCalls y = new ExternalCalls();

            System.out.println("Call_FSLCD + Call_FSLDC");
              {
                        String[] TesztParameters        = {"y1234","x4321"};
                    System.out.println(TesztParameters[0]+ "/" +TesztParameters[1]);
                    TesztParameters = y.Call_FSLCD(TesztParameters[0],TesztParameters[1]);
                    System.out.println(TesztParameters[0]+ "/" +TesztParameters[1]);
                    TesztParameters = y.Call_FSLDC(TesztParameters[1],TesztParameters[0]);
                    System.out.println(TesztParameters[0]+ "/" +TesztParameters[1]);
              }
              System.out.println("Call_LCAS + Call_UCAS");
              {
                        String[] TesztParameters        = {"3456 yABCDEF1234","8"};
                    System.out.println(TesztParameters[0]+ "/" +TesztParameters[1]);
                    TesztParameters = y.Call_LCAS(TesztParameters[0],TesztParameters[1]);
                    System.out.println(TesztParameters[0]+ "/" +TesztParameters[1]);
                    TesztParameters = y.Call_UCAS(TesztParameters[0],TesztParameters[1]);
                    System.out.println(TesztParameters[0]+ "/" +TesztParameters[1]);
              }
              System.out.println("Call_SQUE");
              {
                        String[] TesztParameters        = {"3   45 6 y     ABCDEF1234","25"};
                    System.out.println(TesztParameters[0]+ "/" +TesztParameters[1]);
                    TesztParameters = y.Call_SQUE(TesztParameters[0],TesztParameters[1]);
                    System.out.println(TesztParameters[0]+ "/" +TesztParameters[1]);
              }
              System.out.println("Call_PH74");
              {
                        String[] TesztParameters        = {"SCHONGUT","12345678901234567890"};
                    System.out.println(TesztParameters[0]+ "/" +TesztParameters[1]);
                    TesztParameters = y.Call_PH74(TesztParameters[0],TesztParameters[1]);
                    System.out.println(TesztParameters[0]+ "/" +TesztParameters[1]);
              }
              {
                        String[] TesztParameters        = {"SCHONGUCHT","12345678901234567890"};
                    System.out.println(TesztParameters[0]+ "/" +TesztParameters[1]);
                    TesztParameters = y.Call_PH74(TesztParameters[0],TesztParameters[1]);
                    System.out.println(TesztParameters[0]+ "/" +TesztParameters[1]);
              }

              System.out.println("Call_ABCD + Call_BCDA");
              {
                        String[] TesztParameters        = {"111111111111","00000000"};
                    System.out.println(TesztParameters[0]+ "/" +TesztParameters[1]);
                    TesztParameters = y.Call_ABCD(TesztParameters[0],TesztParameters[1]);
```

```
                System.out.println(TesztParameters[0]+ "/" +TesztParameters[1]);
            }
            System.out.println("Call_ABCD + Call_BCDA");
            {
                    String[] TesztParameters    = {"222111333111","00000000"};
               System.out.println("      " + TesztParameters[0]+ "/" +TesztParameters[1]);
                TesztParameters = y.Call_ABCD(TesztParameters[0],TesztParameters[1]);
                System.out.println("Call_ABCD " + TesztParameters[0]+ "/" +TesztParameters[1]);
                TesztParameters = y.Call_BCDA(TesztParameters[1],"000000000000");
                System.out.println("Call_BCDA " + TesztParameters[0]+ "/" +TesztParameters[1]);
            }
            {
                    String[] TesztParameters    = {"aaabbbcccddd","00000000"};
               System.out.println("      " + TesztParameters[0]+ "/" +TesztParameters[1]);
                TesztParameters = y.Call_ABCD(TesztParameters[0],TesztParameters[1]);
                System.out.println("Call_ABCD " + TesztParameters[0]+ "/" +TesztParameters[1]);
                TesztParameters = y.Call_BCDA(TesztParameters[1],"000000000000");
                System.out.println("Call_BCDA " + TesztParameters[0]+ "/" +TesztParameters[1]);
            }
            {
                    String[] TesztParameters    = {"1aa2bb3cc4dd","00000000"};
               System.out.println("      " + TesztParameters[0]+ "/" +TesztParameters[1]);
                TesztParameters = y.Call_ABCD(TesztParameters[0],TesztParameters[1]);
                System.out.println("Call_ABCD " + TesztParameters[0]+ "/" +TesztParameters[1]);
                TesztParameters = y.Call_BCDA(TesztParameters[1],"000000000000");
                System.out.println("Call_BCDA " + TesztParameters[0]+ "/" +TesztParameters[1]);
            }
        }

/**     External Call: CALL ".ABORT" USING TP-STORAGE */
  public String[] Call_ABORT(String TP_STORAGE) {
            String[] inputParameters    = {TP_STORAGE};
            return inputParameters;
        }
/**     External Call: CALL ".LSTAT" USING REM-LID-DATA. */
  public String[] Call_LSTAT(String REM_LID_DATA) {
            String[] inputParameters    = {REM_LID_DATA};
            return inputParameters;
        }
/**     External Call: CALL ".NABRT" USING TP-STORAGE. */
  public String[] Call_NABRT(String TP_STORAGE) {
            String[] inputParameters    = {TP_STORAGE};
            return inputParameters;
        }
/**   External Call: CALL ".SLEEP" USING TP-STORAGE SLEEP-DATA. */
  public String[] Call_SLEEP(String TP_STORAGE,String SLEEP_DATA) {
            String[] inputParameters    = {TP_STORAGE,SLEEP_DATA};
            return inputParameters;
        }
/**     External Call: CALL ".SPWNB" USING SP-JCL-P SP-NRL SP-STG SP-STT */
  public String[] Call_SPWNB(String v1, String v2, String v3, String v4) {
            String[] inputParameters    = {v1,v2,v3,v4};
            return inputParameters;
        }
/**     External Call: CALL ".FREE" USING TP-STORAGE. */
  public String[] Call_FREE(String TP_STORAGE) {
            String[] inputParameters    = {TP_STORAGE};
            return inputParameters;
        }
/**     External Call: CALL ".UNLCK" . */
  public void Call_UNLCK() {
        }
```

```
/**       External Call:  CALL ".SWTM" USING TIM-DATA. */

  public String[] Call_SWTM(String TIM_DATA) {
            String[] inputParameters       = {TIM_DATA};
            return inputParameters;
          }
/**       External Call: CALL "WTYP1" USING CONSOL-ZEILE */
  public String[] Call_WTYP1(String CONSOL_ZEILE) {
            String[] inputParameters       = {CONSOL_ZEILE};
            return inputParameters;
          }
/**       External Call: CALL "WSNAM" USING REM-WS-DATA */
  public String[] Call_WSNAM(String REM_WS_DATA) {
            String[] inputParameters       = {REM_WS_DATA};
            return inputParameters;
          }
/**       External Call: CALL ".CAPRM" USING DYNAMIC-FILE-PARAMETERS. */
  public String[] Call_CAPRM(String DYNAMIC_FILE_PARAMETERS) {
            String[] inputParameters       = {DYNAMIC_FILE_PARAMETERS};
            return inputParameters;
          }
/**       External Call: CALL ".PGPRM" USING DYNAMIC-FILE-PARAMETERS. */
  public String[] Call_PGPRM(String DYNAMIC_FILE_PARAMETERS) {
            String[] inputParameters       = {DYNAMIC_FILE_PARAMETERS};
            return inputParameters;
          }
/**       External Call: CALL ".ALPRM" USING DYNAMIC-FILE-PARAMETERS. */
  public String[] Call_ALPRM(String DYNAMIC_FILE_PARAMETERS) {
            String[] inputParameters       = {DYNAMIC_FILE_PARAMETERS};
            return inputParameters;
          }
/**       External Call: CALL "PWC74" USING WX8 WPC. */
  public String[] Call_PWC74(String WX8,String WPC) {
            String[] inputParameters       = {"WX8","WPC"};
            inputParameters[0] = WX8;
            inputParameters[1] = WPC;
            return inputParameters;
          }
/**       External Call:   CALL ".CRCLG" USING DYNAMIC-FILE-PARAMETERS */
  public String[] Call_CRCLG(String DYNAMIC_FILE_PARAMETERS) {
            String[] inputParameters       = {DYNAMIC_FILE_PARAMETERS};
            return inputParameters;
          }
/**       External Call: CALL ".CRPRM" USING DYNAMIC-FILE-PARAMETERS */
  public String[] Call_CRPRM(String DYNAMIC_FILE_PARAMETERS) {
            String[] inputParameters       = {DYNAMIC_FILE_PARAMETERS};
            return inputParameters;
          }
/**       External Call: CALL ".UJRNL" USING ER-RET J-LGT J-REC */
  public String[] Call_UJRNL(String ER_RET,String J_LGT,String J_REC) {
            String[] inputParameters       = {ER_RET,J_LGT,J_REC};
            return inputParameters;
          }

/**
 * External Call: CALL "DZD8" USING VDZ BDZ MINUTEN<br>
 * Minuten-Differenz unter Beruecksichtigung des Datums
 * @param VDZ
 * @param BDZ
 * @param MINUTEN
 * @return  an array of strings with the modified  input parameters
 */
```

```
  public String[] Call_DZD8(String VDZ,String BDZ, String MINUTEN) {
              Do_DZD8 w = new Do_DZD8(this);
              return w.Do_DZD8_A(VDZ,BDZ,MINUTEN);
            }

/**
 * External Call: CALL "DELDA" USING FWBCON TX-STORAGE
 *AUTHOR. KURT LANGER.<br>
 *DELTA-TIME FUER TEST<br>
 *   STATT   ACCEPT TXDATE FROM DATE<br>
 *   UND     ACCEPT TXTIME FROM TIME<br>
 *   CALL "DELDA" USING TX-STORAGE <br>
 * @param FWBCON
 * @param TX_STORAGE
 * @return an array of strings with the modified  input parameter
 */
  public String[] Call_DELDA(String FWBCON,String TX_STORAGE) {
              Do_DELDA w = new Do_DELDA(this);
              return w.Do_DELDA_STRT_1(FWBCON,TX_STORAGE);
            }
/**
 * External Call: CALL "FMT74" USING TX-STORAGE WAUF01 TAUF01
 * FORMATIERUNG VON TEMINALMELDUNGEN
 * @param TX_STORAGE
 * @param WAUF01
 * @param TAUF01
 * @return an array of strings with the modified  input parameter
 */
  public String[] Call_FMT74(String TX_STORAGE,String WAUF01,String TAUF01) {
      Do_FMT74 w = new Do_FMT74(this);
      return w.Do_FMT74_A(TX_STORAGE,WAUF01,TAUF01);
            }
/**
 * External Call: CALL "FMX74" USING TX-STORAGE WAUF01 TAUF01
 * FORMATIERUNG VON TEMINALMELDUNGEN
 * @param TX_STORAGE
 * @param WAUF01
 * @param TAUF01
 * @return an array of strings with the modified  input parameter
 */
  public String[] Call_FMX74(String TX_STORAGE,String WAUF01,String TAUF01) {
      Do_FMX74 w = new Do_FMX74(this);
      return w.Do_FMX74_A(TX_STORAGE,WAUF01,TAUF01);
            }
/**
 * External Call: CALL "SYNWEG" USING OMSG OUT-MSG<br>
 * LOESCHE SYNS AUS MSG UND SETZE PARAMETER IN OU-MSG<br>
 * @param OMSG
 * @param OUT_MSG
 * @return an array of strings with the modified  input parameter
 */
  public String[] Call_SYNWEG(String OMSG,String OUT_MSG) {
              Do_SYNWEG w = new Do_SYNWEG(this);
              return w.Do_SYNWEG_A(OMSG,OUT_MSG);
            }

}
```

6.16 Java Database Access Methods

```
/*<AccessMethods>                                <br>*/
   import at.anecon.cobol2java.frame.dbStatus;
   public dbStatus DB_STATUS;
      int TestCaseNr = 0;
/*<Method name = "storeR223AUTY" type = "access" />     <br>*/
   public String  storeR223AUTY(String R223AUTY, String DB_Key){
   //  INSERT R223AUTY IN IDS_R223AUTY;
      String TestObject this.getR223AUTY();
      TESTME.TestDBStore("R223AUTY", TestCaseNr, TestObject);
      return  this.DB_STATUS = 0;
   }

/*<Method name = "findR223AUTY" type = "access" />      <br>*/
   public String  findR223AUTY(String R223AUTY, String DB_Key){
   //  SELECT R223AUTY FROM IDS_R223AUTY WHERE (DB_Key = "????");
      return  this.DB_STATUS = 0;
   }

/*<Method name = "getR223AUTY" type = "access" />       <br>*/
   public String  getnextR223AUTY(String R223AUTY, String DB_Key){
   //  FETCH  R223AUTY;
      TestCaseNr++;
      String TestObject TESTME.TestDBFetch("R223AUTY", TestCaseNr);
      this.setR223AUTY(TestObject);
      return  this.DB_STATUS = 0;
   }

/*<Method name = "modifyR223AUTY" type = "access" />    <br>*/
   public String  modifyR223AUTY(String R223AUTY, String DB_Key){
   //  UPDATE R223AUTY;
      return  this.DB_STATUS = 0;
   }

/*<Method name = "eraseR223AUTY" type = "access" />     <br>*/
   public String  eraseR223AUTY(String R223AUTY, String DB_Key){
   //  DELETE R223AUTY;
      return  this.DB_STATUS = 0;
   }

/*</AccessMethods>                               <br>*/
```

6.17 Java Test Driver

```
package at.fwag.cobol2java.frame;
/**
 * @author Rudi
 *
 * To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Generation - Code and Comments
 */
public class TestMe {
        /**
         *
         */
        private String ProgramName;
        public TestMe(String iProgramName) {
                // TODO Auto-generated constructor stub
                ProgramName = iProgramName;
```

```
	}

	/**
	 * @param args
	 */
	public static void main(String[] args) {
		// TODO Auto-generated method stub
	}
/**
 * This method returns a test data for the TP input buffer
 *
 * @param TestObjectName the name of the calling class
 * @param TestCaseNr The demanded test case
 * @return The test data string read from a database or a text file
 */
  public String  ReceiveTPInput  (String TestObjectName, int TestCaseNr){
   //  SELECT  TESTDATA FROM TPINPUT WHERE (PROGRAMNAME = :ProgramName AND
TESTKEY1 = :TestObjectName AND TESTKEY2 = :TestCaseNr);
      return  new String("TESTDATA");
   }
/**
 * This method receives a string intended for sending to the terminal and writes it
 * to the test database.
 * @param TestObjectName the name of the calling class
 * @param TestCaseNr The number of the test case
 * @param TestObject Text to be written to the database
 */
   public void  SendTPOutput    (String TestObjectName, int TestCaseNr, String TestObject){
   //  INSERT  (:ProgramName, :TestObjectName, :TestCaseNr, :TestObject) INTO TPOUTPUT;
   }

/**
 * This method returns a char[] of blanks with the specified length.
 * @param TestObjectName The name of the required the test case type (COBOL record type)
 * @param TestCaseNr The number of the test case
 * @return The test data string read from a database or a text file
 */
  public String  FetchDBInput    (String TestObjectName, int TestCaseNr){
   //  SELECT TESTDATA FROM DBINPUT WHERE (PROGRAMNAME = :ProgramName AND
TESTKEY1 = TestObjectName AND TESTKEY2 = TestCaseNr);
      return  new String("TESTDATA from DB");
   }

/**
 *
 * This method receives a string intended for writing into the database
 * and writes it  to the test database.
 * @param TestObjectName Testobject name
 * @param TestCaseNr The running tescase number
 * @param TestObject The produced data to be stored
 */
   public void  StoreDBOutput   (String TestObjectName, int TestCaseNr, String TestObject){
   //  INSERT  (:ProgramName, 'TP-OUTPUT', :TestCaseNr, :TestObject) INTO DBOUTPUT;
   }
}
```

6.18 Java PLI Storage Classes

```
package at.anecon.pli2java.programs;

import at.anecon.pli2java.frame.*;
import at.anecon.pli2java.p2715.*;
```

```
import at.anecon.pli2java.p2715.storage.PROCS;
import at.anecon.pli2java.p2715.storage.STATIC;
import at.anecon.pli2java.p2715.storage.LOCAL;
import at.anecon.pli2java.p2715.storage.BASED;
import at.anecon.pli2java.p2715.storage.DATABASE;
//import at.anecon.pli2java.p2715.storage.MESSAGE;
import at.anecon.pli2java.p2715.storage.DATACOM;

/**
 * This is a java representation of a PL/I program P2715.<br>
 * Object Description: Control Class for Method Sequences<br>
 * TODO: INSERT COMMENT HERE
 * @author jSTSN
 * @version 2.0
 */
public class P2715 extends Controller {
  /** singleton instance */
  private static P2715 instance = null;
  /** The BASED storage area of the program */
  public static BASED BASED;
  /** The STATIC storage area of the program */
  public static STATIC STATIC;
  /** The LOCAL storage area of the program */
  public static LOCAL LOCAL;
  /** The PROCS storage area of the program */
  public static PROCS PROCS;
  /** The DATABASE storage area of the program */
  public static DATABASE DATABASE;
//  /** The MESSAGE storage area of the program */
//  public static MESSAGE MESSAGE;
  /** The DATACOM storage area of the program */
  public static DATACOM DATACOM;
  /** The test case supporter object of the program */
  public static TestMe TESTME;
  /** The test case supporter object of the program */
  public static ExternalCalls EXTERNALCALLS;
  /**
   * Default singleton constructor.
   */
  private P2715() {
    super();
    System.out.println("Initalizing Memory....");
    System.out.println("*******************************************************");
    BASED = BASED.getInstance(this);
    STATIC = STATIC.getInstance(this);
    LOCAL = LOCAL.getInstance(this);
    PROCS = PROCS.getInstance(this);
    DATABASE = DATABASE.getInstance(this);
//    MESSAGE = MESSAGE.getInstance(this);
    DATACOM = DATACOM.getInstance(this);
    TESTME = new TestMe("P2715");
    EXTERNALCALLS = new ExternalCalls();
    System.out.println("*******************************************************");
    System.out.println("...done.");
  }

  /**
   * This method is called to get the single instance of this class.
   * If it does not exist, a single instance will be invoked.
   * @return The singleton instance of this class
   */
  public static P2715 getInstance() {
```

```
    if (instance == null) {
      instance = new P2715();
    }
    return instance;
  }
  /**
   * This is the main method of the program P2715.
   * TODO: INSERT COMMENT HERE
   * TODO: This method might be replaced by a normal static method and called by the user interface.
   * @param args
   */
  public static void main(String[] args) {
    String xNextLabel = "P2715.PROCS.PROC_DB_AKTION.P2715_DB_AKTION";
    ProcessingInfo processingInfoIn = new ProcessingInfo(xNextLabel);
    P2715 p2715 = P2715.getInstance();
    try {
      System.out.println("STATE MACHINE STARTED with: "+xNextLabel);
      /*<Program_starts_here>*/
      /*</Program_starts_here>*/
      p2715.runStateMachine(processingInfoIn);
    } catch(Exception e) {
      System.out.println(e.toString());
    }
  }
  /** Used to keep the last next method label from a call to a "Main" method */
    static String LastLabel;
  /**
   * Returns processinInfo resulting from a call to a "Main" method
   * @return processinInfo resulting from a call to a "Main" method
   */
    public ProcessingInfo getLastProcessingInfo() {
      return new ProcessingInfo(LastLabel);
    }

  /**
   * Returns the PLIObject specified by the processinInfoIn from the storage area
   * also specified by the processingInfoIn.
   * @param processingInfoIn
   * @return The demanded PLIObject from the demanded storage area
   */
  public PLIObject getPLIObject(ProcessingInfo processingInfoIn) {
    PLIObject pliObject = null;
    String storage = processingInfoIn.getStorage();
    String classId = processingInfoIn.getClassId();
    if(storage.equals(BASEDStorage)) {
      System.out.println("Entering Storage Area: "+BASEDStorage);
      pliObject = BASED.getSingletonByClassId(classId);
    }
    if(storage.equals(STATICStorage)) {
      System.out.println("Entering Storage Area: "+STATICStorage);
      pliObject = STATIC.getSingletonByClassId(classId);
    }
    if(storage.equals(LOCALStorage)) {
      pliObject = LOCAL.getSingletonByClassId(classId);
    }
    if(storage.equals(PROCSStorage)) {
      pliObject = PROCS.getSingletonByClassId(classId);
    }
    if(storage.equals(DATABASEStorage)) {
      pliObject = DATABASE.getSingletonByClassId(classId);
    }
//    if(storage.equals(MESSAGEStorage)) {
```

```
//      pliObject = MESSAGE.getSingletonByClassId(classId);
//    }
    if(storage.equals(DATACOMStorage)) {
      pliObject = DATACOM.getSingletonByClassId(classId);
    }
//    if(storage.equals(MAINmodule)) {
//        try {
//            ProcessingInfo MainMethod = performOperation (processingInfoIn);
//            LastLabel = MainMethod.getXNextMethod();
//            if (!LastLabel.equals(cobolObject.STOP)) {
//                    pliObject = this.getPLIObject(MainMethod);
//                }
//                else {
//                    pliObject = new ExternalCalls();
//                }
//             } catch(Exception e) {
//                System.out.println(e.toString());
//        }
//    }
    return pliObject;
  }

//
//   The following variables and methods serve as dummy stubs for common functions,
//   that cannot be solved by the generator
//

// --------- End of standard part

/*<ControlMethods>                              <br>*/
/*</ControlMethods>                             <br>*/
} //End of Class: P2715
```

6.19 Converted COBOL Class

```
/**
* This is a java representation of a COBOL74 data structure 223-SA1-R. <br>
* Component RTY1                                <br>
* OldName 223-SA1-R                               <br>
* NewName DBS_223_SA1_R                                 <br>
* extends CACHE                                <br>
* Object Description:                          <br>
* TODO: INSERT COMMENT HERE<br>
* @author Harry M. Sneed                          <br>
* @since 2010.05.01.  <br>
* @version 0.2.1  <br>
*/
public  class DBS_223_SA1_R       extends COBOLObject {
/**Define static class Object                    <br>*/

    private static  DBS_223_SA1_R        instance = null;
    private static  RTY1      RTY1      ;
    private final static String RTY1_AUT_9 = "RTY1_AUT_9";
    public int testCaseNumber = 0;
//õõ This is a place holder for potential methods
    public  static  char[]    DB_223_SA1_R;

    private DBS_223_SA1_R(Object program) {
       DB_223_SA1_R = new char[12  ];
```

```
      DB_223_SA1_R = this.xSpaces(12  ).toCharArray();
      RTY1 = (RTY1)program;
      initDBS_223_SA1_R();
   }
   public  static  DBS_223_SA1_R getInstance(Object program) {
      if (instance == null) {
         instance = new  DBS_223_SA1_R(program);
      }
      return instance;
   }
   public  ProcessingInfo performOperation(ProcessingInfo  processingInfoIn) throws Exception {
      System.out.println("Performing Operation in CLASS:DBS_223_SA1_R");
      String methodId = processingInfoIn.getMethodId();
      String xNextMethod = null;
      if (methodId.equals(DBS_223_SA1_R.RTY1_AUT_9)) xNextMethod = RTY1_AUT_9();
//$$   This is a marker for potential method invocations
      if (xNextMethod == null) throw new Exception(" No matching method found:" + methodId);
      ProcessingInfo processingInfoOut =  new ProcessingInfo(xNextMethod);
      return processingInfoOut;
   }
/*<Attributes>                                  <br>*/
/**Get method for Cobol field 223-SA1.<br>
 * Original type: Struct<br>
 * New name: 223-SA1<br>
 * Position: 0<br>
 * Length: 8<br>
 * Occurs: <br>
 * @return the value as fixed length string
 */
   public String getDB_223_SA1() {
      return  getString(DB_223_SA1_R,0,8);
   }
/**Set method for Cobol field 223-SA1.<br>
 * Original type: Struct<br>
 * New name: 223-SA1<br>
 * Position: 0<br>
 * Length: 8<br>
 * Occurs: <br>
 * @param inStr the fixed length string value to be stored
 */
   public void setDB_223_SA1(String inStr) {
      setAsChar(DB_223_SA1_R,inStr,0,8);
   }
/**Get method for Cobol field 223-PASW.<br>
 * Original type: X(6)<br>
 * New name: 223-PASW-STEUERUNG<br>
 * Position: 0<br>
 * Length: 6<br>
 * Occurs: <br>
 * @return the value as fixed length string
 */
  public String getDB_223_PASW() {
      return  getString(DB_223_SA1_R,0,6);
   }
/**Set method for Cobol field 223-PASW.<br>
 * Original type: X(6)<br>
 * New name: 223-PASW-STEUERUNG<br>
 * Position: 0<br>
 * Length: 6<br>
 * Occurs: <br>
 * @param inStr the fixed length string value to be stored
 */
```

```
    public void setDB_223_PASW(String inStr) {
        setAsChar(DB_223_SA1_R,inStr,0,6);
    }
/**Get method for Cobol field 223-GRP.<br>
 * Original type: X(3)<br>
 * New name: 223-GRP-GRUPPE<br>
 * Position: 0<br>
 * Length: 3<br>
 * Occurs: <br>
 * @return the value as fixed length string
 */

    public String getDB_223_GRP() {
        return  getString(DB_223_SA1_R,0,3);
    }
/**Set method for Cobol field 223-GRP.<br>
 * Original type: X(3)<br>
 * New name: 223-GRP-GRUPPE<br>
 * Position: 0<br>
 * Length: 3<br>
 * Occurs: <br>
 * @param inStr the fixed length string value to be stored
 */
    public void setDB_223_GRP(String inStr) {
        setAsChar(DB_223_SA1_R,inStr,0,3);
    }
/**Get method for Cobol field 223-PS1.<br>
 * Original type: X(1)<br>
 * New name: 223-PS1-DZT-FREI<br>
 * Position: 8<br>
 * Length: 1<br>
 * Occurs: <br>
 * @return the value as fixed length string
 */
    public String getDB_223_PS1() {
        return  getString(DB_223_SA1_R,8,1);
    }
/**Set method for Cobol field 223-PS1.<br>
 * Original type: X(1)<br>
 * New name: 223-PS1-DZT-FREI<br>
 * Position: 8<br>
 * Length: 1<br>
 * Occurs: <br>
 * @param inStr the fixed length string value to be stored
 */
    public void setDB_223_PS1(String inStr) {
        setAsChar(DB_223_SA1_R,inStr,8,1);
    }
/**Get method for Cobol field 223-PS2.<br>
 * Original type: X(1)<br>
 * New name: 223-PS2-X-UID-DARF-AUFG-BER-WEITERGEBE22<br>
 * Position: 9<br>
 * Length: 1<br>
 * Occurs: <br>
 * @return the value as fixed length string
 */
    public String getDB_223_PS2() {
        return  getString(DB_223_SA1_R,9,1);
    }
/**Set method for Cobol field 223-PS2.<br>
 * Original type: X(1)<br>
 * New name: 223-PS2-X-UID-DARF-AUFG-BER-WEITERGEBE22<br>
```

```
 * Position: 9<br>
 * Length: 1<br>
 * Occurs: <br>
 * @param inStr the fixed length string value to be stored
 */
   public void setDB_223_PS2(String inStr) {
      setAsChar(DB_223_SA1_R,inStr,9,1);
   }
/**Get method for Cobol field 223-PSW.<br>
 * Original type: X(1)<br>
 * New name: 223-PSW<br>
 * Position: 6<br>
 * Length: 1<br>
 * Occurs: <br>
 * @return the value as fixed length string
 */
   public String getDB_223_PSW() {
      return  getString(DB_223_SA1_R,6,1);
   }
/**Set method for Cobol field 223-PSW.<br>
 * Original type: X(1)<br>
 * New name: 223-PSW<br>
 * Position: 6<br>
 * Length: 1<br>
 * Occurs: <br>
 * @param inStr the fixed length string value to be stored
 */
   public void setDB_223_PSW(String inStr) {
      setAsChar(DB_223_SA1_R,inStr,6,1);
   }
/**Get method for Cobol field 223-ACK.<br>
 * Original type: X(1)<br>
 * New name: 223-ACK<br>
 * Position: 7<br>
 * Length: 1<br>
 * Occurs: <br>
 * @return the value as fixed length string
 */
   public String getDB_223_ACK() {
      return  getString(DB_223_SA1_R,7,1);
   }
/**Set method for Cobol field 223-ACK.<br>
 * Original type: X(1)<br>
 * New name: 223-ACK<br>
 * Position: 7<br>
 * Length: 1<br>
 * Occurs: <br>
 * @param inStr the fixed length string value to be stored
 */
   public void setDB_223_ACK(String inStr) {
      setAsChar(DB_223_SA1_R,inStr,7,1);
   }

/*</Attributes>                              <br>*/
/**Initialize Class Attributes                    <br>*/
   public  void  initDBS_223_SA1_R()       {
      this.setDB_223_SA1(Spaces);
      this.setDB_223_PASW(Spaces);
      this.setDB_223_PSW(Spaces);
      this.setDB_223_ACK(Spaces);
   }
/*<Processing Methods>                          <br>*/
```

/**

6.20 Converted COBOL Method

```
/**----------------------------------------------------------------------------------------------------------------<br>
 * Method {                                  <br>
 * NewName:TRM2_0040P                              <br>
 * <Parameters>                                <br>
 * <Param use = "Output" name = "TXSWE" class = "INP_TX_STORAGE" type = "String" /> <br>
 * <Error text= "SDUBB was referenced but not declared" />
 * <Param use = "InOut" name = "SDUBB" class = "INP_TX_STORAGE" type = "String" /> <br>
 * <Param use = "Input" name = "DB_STATUS" class = "WRK_TRM2_WORKING" type = "int" /> <br>
 * <Param use = "Input" name = "R262_SA1" class = "IDS_R262PRQD" type = "String" /> <br>
 * <Param use = "Output" name = "DB_262_SA2" class = "DBS_262_SA2" type = "String" Comment = "262-
SA2-UNTERGRUPPENSATZ"/> <br>
 * <Param use = "Input" name = "R262_PRQD" class = "IDS_R262PRQD" type = "String" /> <br>
 * <Param use = "Input" name = "DB_262_TAB" class = "DBS_262_SA2" type = "String" /> <br>
 * <Param use = "Output" name = "DB_262_TRD" class = "DBS_262_TRD" type = "String" /> <br>
 * <Param use = "Input" name = "DB_262_TORD" class = "DBS_262_TORD" type = "String" /> <br>
 * <Param use = "Input" name = "X0" class = "WRK_TRM2_WORKING" type = "int" /> <br>
 * <Error text= "DB_311 was referenced but not declared" />
 * <Param use = "Input" name = "DB_311" class = "WRK_TRM2_WORKING" type = "int" /> <br>
 * <Param use = "Input" name = "DB_262_TRD" class = "DBS_262_TRD" type = "String" /> <br>
 * <Param use = "Output" name = "R262_PRQD" class = "IDS_R262PRQD" type = "String" /> <br>
 * <Param use = "InOut" name = "R262PRQD" class = "IDS_R262PRQD" type = "String" /> <br>
 * <Param use = "Input" name = "DB_262_SA2" class = "DBS_262_SA2" type = "String" Comment = " 262-
SA2-UNTERGRUPPENSATZ"/> <br>
 * <Param use = "Output" name = "R262_SA1" class = "IDS_R262PRQD" type = "String" /> <br>
 * <Param use = "InOut" name = "R262_SA1" class = "IDS_R262PRQD" type = "String" /> <br>
 * </Parameters>                               <br>
 * Test cases: [to be inserted]                    <br>
 * @return new xNextMethod }                        <br>
 *----------------------------------------------------------------------------------------------------------------<br>*/
 public  String  TRM2_0040P() {
  String xNextMethod = Spaces();
  //       MOVE "0040P" TO TXSWE.
         TRM2.INPUT.INP_TX_STORAGE.setTXSWE("0040P");
  //*
  //       FIND NEXT WITHIN SDUBB.
         DB_STATUS = IDSDB.IDS_R201_SA1.find(R201_SA1, "NEXT");
  //*       MOVE DB-STATUS TO OTX1
  //*       MOVE SPACE   TO OTX3
  //*       MOVE " 40P"  TO OTX2              PERFORM S-A09.
  //       IF DB-STATUS NOT = ZERO
         if (!(TRM2.WORK.WRK_TRM2_WORKING.getDB_STATUS().matches(Zeros))) {
  //         GO NXTM
           xNextMethod = "TRM2:IDSDB.IDS_R201DUMY.TRM2_NXTM";
           return xNextMethod;
  //       END-IF.
         }
  //       GET
         DB_STATUS = IDSDB.IDS_R201_SA1.getnext(R201_SA1, this);
  //         MOVE R262-SA1 TO 262-SA2.
           TRM2.CACHE.DBS_262_SA2.setDB_262_SA2(this.getR262_SA1());
  //       IF R262-PRQD = "TRM--"
         if (this.getR262_PRQD().compareTo("TRM--") == 0) {
  //         PERFORM I0
           xNextMethod = TRM2.WORK.WRK_TRM2_WORKING.TRM2_I0();
  //         GO 0040P
           xNextMethod = "TRM2:IDSDB.IDS_R262PRQD.TRM2_0040P";
           return xNextMethod;
  //       END-IF.
```

```
        }
//      IF 262-TAB = SPACE
        if (TRM2.CACHE.DBS_262_SA2.getDB_262_TAB().matches(Spaces)) {
//        PERFORM I0
          xNextMethod = TRM2.WORK.WRK_TRM2_WORKING.TRM2_I0();
//        GO 0040P
          xNextMethod = "TRM2:IDSDB.IDS_R262PRQD.TRM2_0040P";
          return xNextMethod;
//      END-IF.
        }
//      MOVE R262-PRQD TO 262-TRD.
        TRM2.CACHE.DBS_262_TRD.setDB_262_TRD(this.getR262_PRQD());
//*       MOVE 262-C1 TO 262-D1       MOVE 262-C2 TO 262-D2
//*       MOVE 262-TD TO OTX1
//*       MOVE 262-N1 TO 262-D1       MOVE 262-N2 TO 262-D2
//*       MOVE 262-TD TO OTX3
//*       MOVE " 41P"  TO OTX2              PERFORM S-A09.
//      IF R262-PRQD > 262-TORD
        if (this.getR262_PRQD().compareTo(TRM2.CACHE.DBS_262_TORD.getDB_262_TORD()) > 0) {
//        GO 0040P
          xNextMethod = "TRM2:IDSDB.IDS_R262PRQD.TRM2_0040P";
          return xNextMethod;
//      END-IF.
        }
//      MOVE "I0"      TO TXSWE.
        TRM2.INPUT.INP_TX_STORAGE.setTXSWE("I0");
//      PERFORM I1 VARYING X0 FROM 1 BY 1 UNTIL  X0 > 311.
        TRM2.WORK.WRK_TRM2_WORKING.setX0(1);
        while (!(TRM2.WORK.WRK_TRM2_WORKING.getX0() >    311)) {
           xNextMethod = TRM2:CACHE.DBS_262_SA2.TRM2_I1();
       TRM2.WORK.WRK_TRM2_WORKING.setX0(TRM2.WORK.WRK_TRM2_WORKING.getX0() + 1);
        }
//      MOVE "0040P" TO TXSWE.
        TRM2.INPUT.INP_TX_STORAGE.setTXSWE("0040P");
//      MOVE 262-TRD TO R262-PRQD.
        this.setR262_PRQD(TRM2.CACHE.DBS_262_TRD.getDB_262_TRD());
//      FIND ANY R262PRQD.
        DB_STATUS = IDSDB.IDS_R262PRQD.find(R262PRQD, "ANY");
//        PERFORM DBERROR.
          xNextMethod = TRM2.TRM2.TRM2.TRM2_DBERROR();
//      IF 262-TAB = SPACE
        if (TRM2.CACHE.DBS_262_SA2.getDB_262_TAB().matches(Spaces)) {
//        PERFORM I0
          xNextMethod = TRM2.WORK.WRK_TRM2_WORKING.TRM2_I0();
//      ELSE
        }
        else {
//            MOVE 262-SA2 TO R262-SA1
            this.setR262_SA1(TRM2.CACHE.DBS_262_SA2.getDB_262_SA2());
//            MODIFY R262-SA1
            DB_STATUS = IDSDB.IDS_R262_SA1.modify(R262_SA1, this);
//      END-IF.
        }
//      PERFORM DBERROR.
        xNextMethod = TRM2.TRM2.TRM2.TRM2_DBERROR();
//        GO 0040P.
          xNextMethod = "TRM2:IDSDB.IDS_R262PRQD.TRM2_0040P";
          return xNextMethod;
//*---------------------------------------------------------------
 xNextMethod = "TRM2:WORK.WRK_TM2_000.TRM2_0044P";
 return xNextMethod;
} //End of Method:TRM2_0040P
```

```
 /** List of potential successor Methods              <br>
 * <Successors>                                 <br>
 * <NextMethod name = "TRM2:IDSDB.IDS_R201DUMY.TRM2_NXTM" type = "GOTO"/> <br>
 * <NextMethod name = "TRM2:WORK.WRK_TRM2_WORKING.TRM2_I0" type = "PERF"/> <br>
 * <NextMethod name = "TRM2:IDSDB.IDS_R262PRQD.TRM2_0040P" type = "GOTO"/> <br>
 * <NextMethod name = "TRM2:CACHE.DBS_262_SA2.TRM2_I1" type = "PERF"/> <br>
 * <NextMethod name = "TRM2:TRM2.TRM2.TRM2_DBERROR" type = "PERF"/> <br>
 * <NextMethod name = "TRM2:WORK.WRK_TM2_000.TRM2_0044P" type = "NEXT"/> <br>
 * </Successors>                                <br>
 * EndMethod                                    <br>*/
/*</Processing Methods>                             <br>*/
```

6.21 Converted PLI Class

```
//-------------------------------------------------------------------------------------------------------------------------<br>
/**-------------------------------------------------------<br>
* @author Harry M. Sneed                                 <br>
* @date 20.10.11.11.                                    <br>
* Class {                                          <br>
* Component P2715                                       <br>
* OldName IO_TA_BASED                                       <br>
* NewName BASE_IO_TA_BASED                                      <br>
* extends BASED                                        <br>
* Object Description:                                  <br>
* } Class Definition                                   <br>
*-------------------------------------------------------<br>*/
package at.anecon.pli2java.p2715.storage;
import at.anecon.pli2java.frame.PLIObject;
import at.anecon.pli2java.frame.ProcessingInfo;
import at.anecon.pli2java.programs.  P2715   ;
import java.util.*;
import at.anecon.pli2java.frame.dbstatus;
public  class BASE_IO_TA_BASED          extends PLIObject {
/**Define static class Object                          <br>*/
   private static  BASE_IO_TA_BASED            instance = null;
   private static  P2715     P2715     ;
//õõ This is a place holder for potential methods
   public  static  char[]    IO_TA_BASED;
   private BASE_IO_TA_BASED(Object program) {
      IO_TA_BASED = new char[2000];
      IO_TA_BASED = this.xSpaces(2000).toCharArray();
      P2715 = (P2715)program;
      initBASE_IO_TA_BASED();
   }
   public  static  BASE_IO_TA_BASED getInstance(Object program) {
      if (instance == null) {
         instance = new  BASE_IO_TA_BASED(program);
      }
      return instance;
   }
   public  ProcessingInfo performOperation(ProcessingInfo  processingInfoIn) throws Exception {
      System.out.println("Performing Operation in CLASS:BASE_IO_TA_BASED");
      String methodId = processingInfoIn.getMethodId();
      String xNextMethod = null;
//$$   This is a marker for potential method invocations
      if (xNextMethod == null) throw new Exception(" No matching method found:"+xNextMethod);
      ProcessingInfo processingInfoOut =  new ProcessingInfo(xNextMethod);
      return processingInfoOut;
   }
/*<Attributes>                                 <br>*/
/*<Attr name = "IO_TA_BASED            " type = "Struct  " pos = "0000" lng = "2000" comment =
"undefined"/> <br> */
```

```
/**Get Attribute Method                          <br>*/
   public String getIO_TA_BASED() {
      return  getString(IO_TA_BASED,0,2000);
   }
/**Set Attribute Method                          <br>*/
   public void setIO_TA_BASED(String inStr) {
      setAsChar(IO_TA_BASED,inStr,0,2000);
   }
/*<Attr name = "IO_TAA                 " type = "X(2000) " pos = "0000" lng = "2000" comment = "undefined"/>
<br> */
/**Get Attribute Method                          <br>*/
   public String getIO_TAA() {
      return  getString(IO_TA_BASED,0,2000);
   }
/**Set Attribute Method                          <br>*/
   public void setIO_TAA(String inStr) {
      setAsChar(IO_TA_BASED,inStr,0,2000);
   }
/*</Attributes>                                 <br>*/
/**Initialize Class Attributes                   <br>*/

   public  void  initBASE_IO_TA_BASED()     {
      this.setIO_TAA(Spaces);
   }
/*<Processing Methods>                           <br>*/
/*</Processing Methods>                          <br>*/
} //End of Class: BASE_IO_TA_BASED
```

6.22 Converted PLI Method

```
  *-------------------------------------------------------------------------------------------------------------------<br>*/
 public  String  P2715_DB_AKTION (
                    String DB_NR,
                    String DB_FUNKTION,
                    String DB_SCHLUESSEL,
                    String AKTION ) {
   String XNextMethod = Spaces();
   // DB_AKTION: PROC (DB_NR,DB_FUNKTION,DB_SCHLUESSEL,AKTION) ;
   //    SELECT (DB_NR) ;
      int case_indicator_1 = 0;
   //      WHEN (BESTAND_DB)
        if (P2715.LOCAL.WORK_P2715_AUTO.getDB_NR() ==
P2715.LOCAL.WORK_P2715_AUTO.getBESTAND_DB())
           case_indicator_1 = 01;
   //      DO ;
        {
   //        SSA_BE.OP  = GLOBAL_OPERATOR ;
 P2715.LOCAL.WORK_SSA_BE.setOP(P2715.LOCAL.WORK_P2715_AUTO.getGLOBAL_OPERATOR());
   //        SSA_BE.KEY = DB_SCHLUESSEL ;
          P2715.LOCAL.WORK_SSA_BE.setKEY(this.getDB_SCHLUESSEL());
   //        SELECT (AKTION) ;
          int case_indicator_2 = 0;
   //          WHEN ('U')
           if (this.getAKTION() == "U")
              case_indicator_2 = 01;
   //           CALL  PLITDLI(FOUR,DB_FUNKTION,PCB_BE,IO_BE,SSA_BEU) ;
             P2715_PLITDLI_RESULT = PLITDLI(P2715.LOCAL.WORK_P2715_AUTO.getFOUR(),
                        this.getDB_FUNKTION(),
                        P2715.PROCS.PROC_P2715.getPCB_BE(),
                        P2715.LOCAL.WORK_P2715_AUTO.getIO_BE(),
                        P2715.LOCAL.WORK_SSA_BEU.getSSA_BEU());
```

```
//        WHEN ('R')
          if (this.getAKTION() == "R")
            case_indicator_2 = 02;
//          CALL  PLITDLI(THREE,DB_FUNKTION,PCB_BE,IO_BE) ;
            P2715_PLITDLI_RESULT = PLITDLI(P2715.LOCAL.WORK_P2715_AUTO.getTHREE(),
                        this.getDB_FUNKTION(),
                        P2715.PROCS.PROC_P2715.getPCB_BE(),
                        P2715.LOCAL.WORK_P2715_AUTO.getIO_BE());
//        OTHERWISE
          if (case_indicator_2 == 0)
//          CALL  PLITDLI(FOUR,DB_FUNKTION,PCB_BE,IO_BE,SSA_BE) ;
            P2715_PLITDLI_RESULT = PLITDLI(P2715.LOCAL.WORK_P2715_AUTO.getFOUR(),
                        this.getDB_FUNKTION(),
                        P2715.PROCS.PROC_P2715.getPCB_BE(),
                        P2715.LOCAL.WORK_P2715_AUTO.getIO_BE(),
                        P2715.LOCAL.WORK_SSA_BE.getSSA_BE());
//      END ;
        }  /* End */
//      PCB_PTR = PCB_BE ;
        P2715.PROCS.PROC_P2715.setPCB_PTR(P2715.PROCS.PROC_P2715.getPCB_BE());
//    END ;
      }  /* End */
//    WHEN (TABELLEN_DBA)
      if (this.getAKTION() == P2715.LOCAL.WORK_P2715_AUTO.getTABELLEN_DBA())
        case_indicator_1 = 02;
//    DO ;
      {
//      SSA_TAA.OP  = GLOBAL_OPERATOR ;
P2715.LOCAL.WORK_SSA_BE.setOP(P2715.LOCAL.WORK_P2715_AUTO.getGLOBAL_OPERATOR());
//      SSA_TAA.KEY = DB_SCHLUESSEL ;
        P2715.LOCAL.WORK_SSA_BE.setKEY(this.getDB_SCHLUESSEL());
//      SELECT (AKTION) ;
        int case_indicator_2 = 0;
//        WHEN ('U')
          if (this.getAKTION() == "U")
            case_indicator_2 = 01;
//          CALL  PLITDLI(FOUR,DB_FUNKTION,PCB_TAA,IO_TA,SSA_TAAU) ;
            P2715_PLITDLI_RESULT = PLITDLI(P2715.LOCAL.WORK_P2715_AUTO.getFOUR(),
                        this.getDB_FUNKTION(),
                        P2715.PROCS.PROC_P2715.getPCB_TAA(),
                        P2715.LOCAL.WORK_P2715_AUTO.getIO_TA(),
                        P2715.LOCAL.WORK_SSA_TAAU.getSSA_TAAU());
//        WHEN ('R')
          if (this.getAKTION() == "R")
            case_indicator_2 = 02;
//          CALL  PLITDLI(THREE,DB_FUNKTION,PCB_TAA,IO_TAA) ;
            P2715_PLITDLI_RESULT = PLITDLI(P2715.LOCAL.WORK_P2715_AUTO.getTHREE(),
                        this.getDB_FUNKTION(),
                        P2715.PROCS.PROC_P2715.getPCB_TAA(),
                        P2715.BASED.BASE_IO_TA_BASED.getIO_TAA());
//        OTHERWISE
          if (case_indicator_2 == 0)
//          CALL  PLITDLI(FOUR,DB_FUNKTION,PCB_TAA,IO_TA,SSA_TAA) ;
            P2715_PLITDLI_RESULT = PLITDLI(P2715.LOCAL.WORK_P2715_AUTO.getFOUR(),
                        this.getDB_FUNKTION(),
                        P2715.PROCS.PROC_P2715.getPCB_TAA(),
                        P2715.LOCAL.WORK_P2715_AUTO.getIO_TA(),
                        P2715.LOCAL.WORK_SSA_TAA.getSSA_TAA());
//      END ;
        }  /* End */
//      PCB_PTR = PCB_TAA  ;
        P2715.PROCS.PROC_P2715.setPCB_PTR(P2715.PROCS.PROC_P2715.getPCB_TAA());
```

```
//      END ;
        }  /* End */
//      WHEN (TABELLEN_DB)
        if (this.getAKTION() == P2715.LOCAL.WORK_P2715_AUTO.getTABELLEN_DB())
          case_indicator_1 = 03;
//      DO ;
        {
//        SSA_TA.OP  = GLOBAL_OPERATOR ;
P2715.LOCAL.WORK_SSA_BE.setOP(P2715.LOCAL.WORK_P2715_AUTO.getGLOBAL_OPERATOR());
//        SSA_TA.KEY = DB_SCHLUESSEL ;
          P2715.LOCAL.WORK_SSA_BE.setKEY(this.getDB_SCHLUESSEL());
//        SELECT (AKTION) ;
          int case_indicator_2 = 0;
//          WHEN ('U')
           if (this.getAKTION() == "U")
             case_indicator_2 = 01;
//           CALL  PLITDLI(FOUR,DB_FUNKTION,PCB_TA,IO_TA,SSA_TAU) ;
            P2715_PLITDLI_RESULT = PLITDLI(P2715.LOCAL.WORK_P2715_AUTO.getFOUR(),
                        this.getDB_FUNKTION(),
                        P2715.PROCS.PROC_P2715.getPCB_TA(),
                        P2715.LOCAL.WORK_P2715_AUTO.getIO_TA(),
                        P2715.LOCAL.WORK_SSA_TAU.getSSA_TAU());
//          WHEN ('R')
           if (this.getAKTION() == "R")
             case_indicator_2 = 02;
//           CALL  PLITDLI(THREE,DB_FUNKTION,PCB_TA,IO_TA) ;
            P2715_PLITDLI_RESULT = PLITDLI(P2715.LOCAL.WORK_P2715_AUTO.getTHREE(),
                         this.getDB_FUNKTION(),
                         P2715.PROCS.PROC_P2715.getPCB_TA(),
                         P2715.LOCAL.WORK_P2715_AUTO.getIO_TA());
//          OTHERWISE
           if (case_indicator_2 == 0)
//           CALL  PLITDLI(FOUR,DB_FUNKTION,PCB_TA,IO_TA,SSA_TA) ;
            P2715_PLITDLI_RESULT = PLITDLI(P2715.LOCAL.WORK_P2715_AUTO.getFOUR(),
                        this.getDB_FUNKTION(),
                        P2715.PROCS.PROC_P2715.getPCB_TA(),
                        P2715.LOCAL.WORK_P2715_AUTO.getIO_TA(),
                        P2715.LOCAL.WORK_SSA_TA.getSSA_TA());
//      END ;
        }  /* End */
//      PCB_PTR = PCB_TA ;
        P2715.PROCS.PROC_P2715.setPCB_PTR(P2715.PROCS.PROC_P2715.getPCB_TA());
//      END ;
        }  /* End */
//      WHEN (VALOREN_DB)
        if (this.getAKTION() == P2715.LOCAL.WORK_P2715_AUTO.getVALOREN_DB())
          case_indicator_1 = 04;
//      DO ;
        {
//        SSA_VA.OP  = GLOBAL_OPERATOR ;
P2715.LOCAL.WORK_SSA_BE.setOP(P2715.LOCAL.WORK_P2715_AUTO.getGLOBAL_OPERATOR());
//        SSA_VA.KEY = DB_SCHLUESSEL ;
          P2715.LOCAL.WORK_SSA_BE.setKEY(this.getDB_SCHLUESSEL());
//        SELECT (AKTION) ;
          int case_indicator_2 = 0;
//          WHEN ('U')
           if (this.getAKTION() == "U")
             case_indicator_2 = 01;
//           CALL  PLITDLI(FOUR,DB_FUNKTION,PCB_VA,IO_VA,SSA_VAU) ;
            P2715_PLITDLI_RESULT = PLITDLI(P2715.LOCAL.WORK_P2715_AUTO.getFOUR(),
                        this.getDB_FUNKTION(),
                        P2715.PROCS.PROC_P2715.getPCB_VA(),
```

```
                        P2715.LOCAL.WORK_P2715_AUTO.getIO_VA(),
                        P2715.LOCAL.WORK_SSA_VAU.getSSA_VAU());
   //         WHEN ('R')
             if (this.getAKTION() == "R")
                case_indicator_2 = 02;
   //          CALL  PLITDLI(THREE,DB_FUNKTION,PCB_VA,IO_VA) ;
              P2715_PLITDLI_RESULT = PLITDLI(P2715.LOCAL.WORK_P2715_AUTO.getTHREE(),
                        this.getDB_FUNKTION(),
                        P2715.PROCS.PROC_P2715.getPCB_VA(),
                        P2715.LOCAL.WORK_P2715_AUTO.getIO_VA());
   //         OTHERWISE
             if (case_indicator_2 == 0)
   //          CALL  PLITDLI(FOUR,DB_FUNKTION,PCB_VA,IO_VA,SSA_VA) ;
              P2715_PLITDLI_RESULT = PLITDLI(P2715.LOCAL.WORK_P2715_AUTO.getFOUR(),
                        this.getDB_FUNKTION(),
                        P2715.PROCS.PROC_P2715.getPCB_VA(),
                        P2715.LOCAL.WORK_P2715_AUTO.getIO_VA(),
                        P2715.LOCAL.WORK_SSA_VA.getSSA_VA());
   //       END ;
            }  /* End */
   //        PCB_PTR = PCB_VA   ;
            P2715.PROCS.PROC_P2715.setPCB_PTR(P2715.PROCS.PROC_P2715.getPCB_VA());
   //       END ;
           }  /* End */
   //       OTHERWISE
           if (case_indicator_1 == 0)
   //       DO ;
           {
   //        CALL FEHLERMELDUNG(10) ;
            XNextMethod = $$Class.P2715_FEHLERMELDUNG(10) ;
   //        PLIRETCODE = XCON12 ;

P2715.LOCAL.WORK_P2715_AUTO.setPLIRETCODE(P2715.LOCAL.WORK_XCONSTANT_TAB.getXC
ON12());
   //         SIGNAL ERROR ;
             // try XSTATE = ERROR();
   //       END ;
            }  /* End */
   //     END ;
         }  /* End */
   //    GLOBAL_OPERATOR = ' =' ;
        P2715.LOCAL.WORK_P2715_AUTO.setGLOBAL_OPERATOR(" =");
   // END DB_AKTION ;
   return XNextMethod;
 } //End of Method:P2715_DB_AKTION
```

6.23 Converted OO-COBOL Class

```
IDENTIFICATION DIVISION.
PROGID/
COB96  CLASS-ID. OLD01 inherits Base.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
COB96  REPOSITORY.
COB96      Class Base is "base".
       SPECIAL-NAMES.
      *
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
      *
          SELECT CUSTOMER-FILE
```

```
           ASSIGN TO "SYS015"
           RECORD KEY IS CUST-NO IN CUSTOMER-RECORD
           ORGANIZATION IS INDEXED
           ACCESS RANDOM.
       *
COB96  OBJECT.
        DATA DIVISION.
        FILE SECTION.
       *
        FD  CUSTOMER-FILE
           LABEL RECORD STANDARD
           RECORDING MODE F
           DATA RECORD IS CUSTOMER-RECORD.
        01  CUSTOMER-RECORD.
           02  REC-TYPE               PIC XX.
           02  CUST-NO                PIC 9(8) COMP.
           02  CUST-NAME              PIC X(20).
           02  CUST-ADDRESS.
              03  STR                 PIC X(30).
              03  ZIP                 PIC 9(4).
              03  CITY                PIC X(20).
              03  STATE               PIC X(4).
           02  CUST-CREDIT            PIC 9.
       *
            WORKING-STORAGE SECTION.
       *
       *
            PROCEDURE DIVISION.
       *
OPEN   ************************************************************************
OPEN   *XOPEN01A.
COB96  METHOD-ID. XOPEN01A.
COB96  DATA DIVISION.
COB96  WORKING-STORAGE SECTION.
COB96  77  X-COUNT          PIC 999.
        LINKAGE SECTION.
       *
LINK   /
        77  X-RETCODE              PIC XX.
        01  X-CUSTOMER-RECORD.
           02  REC-TYPE               PIC XX.
           02  CUST-NO                PIC 9(8) COMP.
           02  CUST-NAME              PIC X(20).
           02  CUST-ADDRESS.
              03  STR                 PIC X(30).
              03  ZIP                 PIC 9(4).
              03  CITY                PIC X(20).
              03  STATE               PIC X(4).
           02  CUST-CREDIT            PIC 9.
       *
       ****************************************************************
       *
COB96  PROCEDURE DIVISION   USING X-CUSTOMER-RECORD X-RETCODE.
           MOVE ZEROES TO X-RETCODE.
           OPEN INPUT  CUSTOMER-FILE.
COB96      EXIT METHOD.
COB96  END METHOD XOPEN01A.
             OPEN   ************************************************************************
             READ   ************************************************************************
             READ   *XREAD01B.
COB96  METHOD-ID. XREAD01B.
COB96  DATA DIVISION.
```

```
COB96  WORKING-STORAGE SECTION.
COB96  77  X-COUNT          PIC 999.
      LINKAGE SECTION.
     *
LINK  /
      77  X-RETCODE                  PIC XX.
      01  X-CUSTOMER-RECORD.
         02  REC-TYPE                PIC XX.
         02  CUST-NO                 PIC 9(8) COMP.
         02  CUST-NAME               PIC X(20).
         02  CUST-ADDRESS.
            03  STR                  PIC X(30).
            03  ZIP                  PIC 9(4).
            03  CITY                 PIC X(20).
            03  STATE                PIC X(4).
         02  CUST-CREDIT             PIC 9.
     *
     ******************************************************************
```

6.24 Converted OO-COBOL Method

```
******************************************************************
     *REPORT-ORDER.
     *OLD007.
COB96  METHOD-ID. OLD007.
COB96  DATA DIVISION.
COB96  WORKING-STORAGE SECTION.
COB96  77  X-COUNT          PIC 999.
COB96  LINKAGE SECTION.
COB96  77  X-RETCODE        PIC XX.
COB96  77  X-NEXT-LABEL     PIC X(30).
COB96  PROCEDURE DIVISION USING     X-RETCODE
COB96                  RETURNING X-NEXT-LABEL.
TEST       ADD 1 TO X-COUNT.
           MOVE SPACES TO X-NEXT-LABEL.
     *       GO TO REPORT-ERROR
           IF ERROR-TYPE > 0
            MOVE "REPORT-ERROR" TO X-NEXT-LABEL
           END-IF
           IF X-NEXT-LABEL = SPACES
            MOVE SPACES TO DATA-PRT-LINE
            MOVE CUST-NO IN CUSTOMER-RECORD TO CUST-NO IN
           DATA-PRT-LINE
            MOVE CUST-NAME IN CUSTOMER-RECORD TO CUST-NAME IN
            DATA-PRT-LINE
            MOVE ART-NO IN ARTICLE-RECORD TO ART-NO IN DATA-PRT-LINE
            MOVE ART-NAME IN ARTICLE-RECORD TO ART-NAME IN
            DATA-PRT-LINE
            MOVE ITEM-QUAN IN ORDER-RECORD (POS) TO
            ART-QUAN IN DATA-PRT-LINE
            MOVE ART-PRICE IN ARTICLE-RECORD TO
            ART-PRICE IN DATA-PRT-LINE
            MULTIPLY ART-PRICE IN ARTICLE-RECORD BY ITEM-QUAN
            IN ORDER-RECORD (POS) GIVING PRICE
            MOVE PRICE TO TOTAL-PRICE
            ADD PRICE TO TOTAL-CUST-PRICE
            MOVE DATA-PRT-LINE TO PRT-LINE
COB96         INVOKE THIS-OLD06 "XWRIT06C"
                        USING PRT-LINE
                              X-RETCODE
            ADD XCON2 TO LINE-COUNT
            IF LINE-COUNT > XCON50
```

```
           ADD 1 TO PAGE-COUNT
           MOVE PAGE-COUNT TO PAGE-COUNT-NO
           MOVE HEADER-PRT-LINE TO PRT-LINE
COB96          INVOKE THIS-OLD06 "XWRIT06D"
                       USING PRT-LINE
                           X-RETCODE
           MOVE UNDER-LINE TO PRT-LINE
COB96          INVOKE THIS-OLD06 "XWRIT06E"
                       USING PRT-LINE
                           X-RETCODE
           MOVE 0 TO LINE-COUNT
          END-IF
          IF X-NEXT-LABEL = SPACES
            MOVE "PROCESS-ORDER" TO X-NEXT-LABEL
          END-IF
        END-IF
        IF X-NEXT-LABEL = SPACES
          MOVE "REPORT-ERROR" TO X-NEXT-LABEL
        END-IF.
        EXIT.
RECONS*---------------------------------------------------------------
    * PRINT ERROR MESSAGE
    *---------------------------------------------------------------
TEST       DISPLAY "Method executed = OLD007"
TEST       DISPLAY "Method count    = " X-COUNT
COB96      EXIT METHOD.
COB96  END METHOD OLD007.
```

6.25 Data Description Table

```
Nr. ;Use;Sect ;Pos ;Lev;PLI Data Name ;PLI Data Pict   ;Typ;Lng ;Fr;Occr;Value ;Redefines
0001;   ;PROC ;0000;01 ;PROC_P2715            ;              ;Str;0000;00;0001;       ;
0002;P  ;PROCS;0000;77 ;COIBM                 ;X(76)         ;Chr;0076;00;0001;'P2715  ;
0003;C  ;PROCS;0076;77 ;PCB_BE                ;POINTER       ;Ptr;0008;00;0001;       ;
0004;P  ;PROCS;0084;77 ;PCB_TAA               ;POINTER       ;Ptr;0008;00;0001;       ;
0005;P  ;PROCS;0092;77 ;PCB_TA                ;POINTER       ;Ptr;0008;00;0001;       ;
0006;P  ;PROCS;0100;77 ;PCB_VA                ;POINTER       ;Ptr;0008;00;0001;       ;
0007;   ;PROCS;0108;77 ;AUSWAHL_PTR           ;POINTER       ;Ptr;0008;00;0001;       ;
0008;P  ;PROCS;0116;77 ;PLIXOPT               ;X(12)         ;Chr;0012;00;0001;'ISA(12K;
0009;P  ;PROCS;0128;77 ;PLIXHD                ;X(50)         ;Chr;0050;00;0001;'WABE   ;
0038;   ;LOCAL;0000;01 ;P2715_AUTO            ;              ;Str;7417;00;0001;       ;
0039;CIO;LOCAL;0000;02 ;SUCHKEY               ;X(15)         ;Chr;0015;00;0001;"      ;
0040;CIO;LOCAL;0015;02 ;LINECOUNTER           ;(05)9         ;Num;0005;00;0001;"
0041;IO ;LOCAL;0020;02 ;COUNTER               ;(15)9         ;Num;0015;00;0001;"      ;
0042;IO ;LOCAL;0035;02 ;TAWERT_Z              ;Z.ZZZ.ZZZ.ZZ9V,9;Num;0016;03;0001;  ;
0043;O  ;LOCAL;0051;02 ;TAWERT_NEU            ;X(16)         ;Chr;0016;00;0001;       ;
0044;IO ;LOCAL;0067;02 ;BSTKNOM               ;(15)9         ;Num;0015;00;0001;0      ;
0045;IO ;LOCAL;0082;02 ;BNOMSFRS              ;(15)9         ;Num;0015;00;0001;0      ;
0046;IO ;LOCAL;0097;02 ;HFLOAT_1              ;(15)9         ;Num;0015;00;0001;0      ;
0047;IO ;LOCAL;0112;02 ;HFLOAT_2              ;(15)9         ;Num;0015;00;0001;0      ;
0048;   ;LOCAL;0127;02 ;HFLOAT_3              ;(15)9         ;Num;0015;00;0001;0      ;
0049;   ;LOCAL;0142;02 ;STEUER_VALUE          ;(1)9          ;Num;0001;00;0001;"      ;
0050;CIO;LOCAL;0143;02 ;LOOP_ERROR_COUNT      ;(05)9         ;Num;0005;00;0001;0      ;
0051;CO ;LOCAL;0148;02 ;AKTUELLE_AKTSTUFE     ;X(1)          ;Chr;0001;00;0001;"      ;
0052;IO ;LOCAL;0149;02 ;GLOBAL_OPERATOR       ;X(2)          ;Chr;0002;00;0001;' ='   ;
0053;   ;LOCAL;0151;02 ;VALOREN_I             ;(05)9         ;Num;0005;00;0001;0      ;
0054;   ;LOCAL;0156;02 ;VALOREN_O             ;(05)9         ;Num;0005;00;0001;0      ;
0055;   ;LOCAL;0161;02 ;BESTAND_I             ;(05)9         ;Num;0005;00;0001;0      ;
0056;   ;LOCAL;0166;02 ;BESTAND_O             ;(05)9         ;Num;0005;00;0001;0      ;
0057;   ;LOCAL;0171;02 ;TABELLEN_I            ;(05)9         ;Num;0005;00;0001;0      ;
0058;   ;LOCAL;0176;02 ;TABELLEN_O            ;(05)9         ;Num;0005;00;0001;0      ;
```

```
0059;  ;LOCAL;0181;02 ;MARCHZINSEN       ;(11)9.(2)9     ;Num;0013;02;0001;       ;
0060;  ;LOCAL;0194;02 ;MARCHZ_MELDUNG  ;X(1)          ;Chr;0001;00;0001;'0'    ;
0061;  ;LOCAL;0195;02 ;PARM_FAELL          ;X(10)          ;Chr;0010;00;0001;"      ;
0062;O  ;LOCAL;0205;02 ;PARM_SICHERST   ;(8)9          ;Num;0008;00;0001;"      ;
0063;  ;LOCAL;0213;02 ;PARM_LZPERB        ;X(10)          ;Chr;0010;00;0001;"      ;
0064;  ;LOCAL;0223;02 ;PARM_LZPERV        ;X(10)          ;Chr;0010;00;0001;"      ;
0065;  ;LOCAL;0233;02 ;PARM_EVERF          ;X(10)          ;Chr;0010;00;0001;"      ;
0066;  ;LOCAL;0243;02 ;PARM_STKNOM       ;(13)9.(2)9     ;Num;0015;02;0001;       ;
0067;  ;LOCAL;0258;02 ;PARM_ZSATZ          ;(07)9.(5)9     ;Num;0012;05;0001;       ;
0068;  ;LOCAL;0270;02 ;PARM_WAEK           ;999V9999       ;Num;0007;04;0001;       ;
0069;  ;LOCAL;0277;02 ;PARM_WAEE           ;999            ;Num;0003;00;0001;       ;
0070;  ;LOCAL;0280;02 ;PARM_USANZ          ;X(1)           ;Chr;0001;00;0001;"      ;
0071;  ;LOCAL;0281;02 ;INDEX_GEFUNDEN   ;X(1)           ;Chr;0001;00;0001;'0'    ;
0072;  ;LOCAL;0282;02 ;I_COUNTER           ;(05)9          ;Num;0005;00;0001;0      ;
0073;  ;LOCAL;0287;02 ;CONVERSIONS_FELD_1  ;X(11)          ;Chr;0011;00;0001;"      ;
0074;  ;LOCAL;0298;02 ;CONVERSIONS_FELD_2  ;X(11)          ;Chr;0011;00;0001;"      ;
0075;  ;LOCAL;0309;02 ;CONVERSIONS_FELD_3  ;X(12)          ;Chr;0012;00;0001;"      ;
0076;CO ;LOCAL;0321;02 ;KEY_FELD              ;X(33)          ;Chr;0033;00;0001;"      ;
0077;O  ;LOCAL;0354;02 ;ONE                   ;(08)9          ;Num;0008;00;0001;       ;
0078;P  ;LOCAL;0362;02 ;BESTAND_DB        ;(05)9          ;Num;0005;00;0001;1      ;
0079;P  ;LOCAL;0367;02 ;TABELLEN_DBA     ;(05)9          ;Num;0005;00;0001;2      ;
0080;P  ;LOCAL;0372;02 ;TABELLEN_DB       ;(05)9          ;Num;0005;00;0001;3      ;
0081;P  ;LOCAL;0377;02 ;VALOREN_DB        ;(05)9          ;Num;0005;00;0001;4      ;
0082;  ;LOCAL;0382;02 ;FETAB                ;X(30)          ;Chr;0030;00;0010;'AAAA',';
0083;  ;LOCAL;0682;02 ;EOF                   ;X(1)           ;Chr;0001;00;0001;' '    ;
0084;  ;LOCAL;0683;02 ;I                     ;(05)9          ;Num;0005;00;0001;       ;
0085;  ;LOCAL;0688;02 ;J                     ;(05)9          ;Num;0005;00;0001;       ;
0086;O  ;LOCAL;0693;02 ;IFE                   ;(05)9          ;Num;0005;00;0001;       ;
0087;O  ;LOCAL;0698;02 ;HERKUNFT          ;X(10)          ;Chr;0010;00;0001;"      ;
0088;  ;LOCAL;0708;02 ;BEDINGUNG          ;X(15)          ;Chr;0015;00;0001;'0000111;
0089;  ;LOCAL;0723;02 ;BEDINGUNG_ABFRAGE  ;(05)9     ;Num;0005;00;0001;       ;
0090;  ;LOCAL;0728;02 ;BEDINGUNG_1       ;(05)9          ;Num;0005;00;0001;16384  ;
0091;  ;LOCAL;0733;02 ;BEDINGUNG_2       ;(05)9          ;Num;0005;00;0001;08192  ;
0092;  ;LOCAL;0738;02 ;BEDINGUNG_3       ;(05)9          ;Num;0005;00;0001;04096  ;
0093;  ;LOCAL;0743;02 ;BEDINGUNG_4       ;(05)9          ;Num;0005;00;0001;02048  ;
0094;  ;LOCAL;0748;02 ;BEDINGUNG_5       ;(05)9          ;Num;0005;00;0001;01024  ;
0095;  ;LOCAL;0753;02 ;BEDINGUNG_6       ;(05)9          ;Num;0005;00;0001;00512  ;
0096;  ;LOCAL;0758;02 ;BEDINGUNG_7       ;(05)9          ;Num;0005;00;0001;00256  ;
0097;  ;LOCAL;0763;02 ;BEDINGUNG_8       ;(05)9          ;Num;0005;00;0001;00128  ;
0098;  ;LOCAL;0768;02 ;BEDINGUNG_9       ;(05)9          ;Num;0005;00;0001;00064  ;
0099;  ;LOCAL;0773;02 ;BEDINGUNG_10      ;(05)9          ;Num;0005;00;0001;00032  ;
0100;  ;LOCAL;0778;02 ;BEDINGUNG_11      ;(05)9          ;Num;0005;00;0001;00016  ;
0101;  ;LOCAL;0783;02 ;BEDINGUNG_12      ;(05)9          ;Num;0005;00;0001;00008  ;
0102;  ;LOCAL;0788;02 ;BEDINGUNG_13      ;(05)9          ;Num;0005;00;0001;00004  ;
0103;  ;LOCAL;0793;02 ;BEDINGUNG_14      ;(05)9          ;Num;0005;00;0001;00002  ;
0104;  ;LOCAL;0798;02 ;BEDINGUNG_15      ;(05)9          ;Num;0005;00;0001;00001  ;
0105;  ;LOCAL;0803;02 ;AKTUELLES_DATUM  ;X(8)           ;Chr;0008;00;0001;"      ;
0106;IO ;LOCAL;0811;02 ;CT                    ;(05)9          ;Num;0005;00;0020;       ;
0107;  ;LOCAL;0911;02 ;CTT                   ;X(23)          ;Chr;0023;00;0020;'ANZAHL ;
0108;O  ;LOCAL;1371;02 ;DATUM                 ;X(8)           ;Chr;0008;00;0001;       ;
0109;IO ;LOCAL;1379;02 ;ZEIT                  ;X(12)          ;Chr;0012;00;0001;       ;
0110;C  ;LOCAL;1391;02 ;DB_NR                 ;(05)9          ;Num;0005;00;0001;       ;
0111;I  ;LOCAL;1396;02 ;IO_BE                 ;X(2000)        ;Chr;2000;00;0001;       ;
0112;I  ;LOCAL;3396;02 ;IO_VA                 ;X(2000)        ;Chr;2000;00;0001;       ;
0113;CIO;LOCAL;5396;02 ;IO_TA                 ;X(2000)        ;Chr;2000;00;0001;       ;
0114;  ;LOCAL;7396;02 ;DATUM_I              ;X(8)           ;Chr;0008;00;0001;"      ;
0115;CO ;LOCAL;7404;02 ;PLIRETCODE      ;(08)9          ;Num;0008;00;0001;0      ;
0116;CIO;LOCAL;7412;02 ;FIXED_ERROR_COUNT  ;(05)9    ;Num;0005;00;0001;0      ;
0117;  ;BASED;0000;01 ;IO_TA_BASED        ;               ;Str;2000;00;0001;       ;
0118;P  ;BASED;0000;02 ;IO_TAA                ;X(2000)        ;Chr;2000;00;0001;       ;
0119;  ;LOCAL;0000;01 ;XLITERAL_TAB     ;               ;Str;0123;00;0001;       ;
0120;I  ;LOCAL;0000;05 ;XLIT007               ;X(026)         ;Chr;0026;00;0001;'0000000;
```

```
0121;I  ;LOCAL;0026;05 ;XLIT010                ;X(008)        ;Chr;0008;00;0001;'99.99.9;
0122;I  ;LOCAL;0034;05 ;XLIT011                ;X(012)        ;Chr;0012;00;0001;'12:34:5;
0123;I  ;LOCAL;0046;05 ;XLIT012                ;X(009)        ;Chr;0009;00;0001;'1234567;
0124;I  ;LOCAL;0055;05 ;XLIT013                ;X(004)        ;Chr;0004;00;0001;'9900'  ;
0125;I  ;LOCAL;0059;05 ;XLIT014                ;X(008)        ;Chr;0008;00;0001;'AB.CD.E;
0126;I  ;LOCAL;0067;05 ;XLIT015                ;X(006)        ;Chr;0006;00;0001;'EFCDAB';
0127;I  ;LOCAL;0073;05 ;XLIT016                ;X(004)        ;Chr;0004;00;0001;'9910'  ;
0128;I  ;LOCAL;0077;05 ;XLIT017                ;X(004)        ;Chr;0004;00;0001;'9909'  ;
0129;I  ;LOCAL;0081;05 ;XLIT018                ;X(008)        ;Chr;0008;00;0001;'ABCDEFG;
0130;I  ;LOCAL;0089;05 ;XLIT019                ;X(010)        ;Chr;0010;00;0001;'AB.CD.E;
0131;I  ;LOCAL;0099;05 ;XLIT020                ;X(008)        ;Chr;0008;00;0001;'B_VPFWE;
0132;C  ;LOCAL;0107;05 ;XLIT022                ;X(004)        ;Chr;0004;00;0001;'6789'  ;
0133;I  ;LOCAL;0111;05 ;XLIT023                ;X(008)        ;Chr;0008;00;0001;'B_TAGEF;
0134;   ;LOCAL;0119;05 ;LITXEND               ;X(004)        ;Chr;0004;00;0001;'ENDE'  ;
0135;   ;LOCAL;0000;01 ;XCONSTANT_TAB ;              ;Str;0055;00;0001;       ;
0136;I  ;LOCAL;0000;05 ;XCON3                  ;(5)9          ;Num;0005;00;0001;3      ;
0137;I  ;LOCAL;0005;05 ;XCON4                  ;(5)9          ;Num;0005;00;0001;4      ;
0138;I  ;LOCAL;0010;05 ;XCON5                  ;(5)9          ;Num;0005;00;0001;5      ;
0139;I  ;LOCAL;0015;05 ;XCON6                  ;(5)9          ;Num;0005;00;0001;6      ;
0140;C  ;LOCAL;0020;05 ;XCON612                ;(5)9          ;Num;0005;00;0001;612    ;
0141;I  ;LOCAL;0025;05 ;XCON12                 ;(5)9          ;Num;0005;00;0001;12     ;
0142;C  ;LOCAL;0030;05 ;XCON200                ;(5)9          ;Num;0005;00;0001;200    ;
0143;I  ;LOCAL;0035;05 ;XCON8                  ;(5)9          ;Num;0005;00;0001;8      ;
0144;I  ;LOCAL;0040;05 ;XCON100                ;(5)9          ;Num;0005;00;0001;100    ;
0145;C  ;LOCAL;0045;05 ;XCON60                 ;(5)9          ;Num;0005;00;0001;60     ;
0146;   ;LOCAL;0050;05 ;CONXEND                ;(5)9          ;Num;0005;00;0001;999    ;
0147;P  ;LOCAL;0000;01 ;SSA_BE                 ;              ;Str;0052;00;0001;       ;
0148;   ;LOCAL;0000;03 ;SEGMENT                ;X(8)       ;Chr;0008;00;0001;'BESTAND;
0149;   ;LOCAL;0008;03 ;KEYNAME                   ;X(9)          ;Chr;0009;00;0001;'(BESKEY;
0150;O  ;LOCAL;0017;03 ;OP                    ;X(2)          ;Chr;0002;00;0001;' ='    ;
0151;O  ;LOCAL;0019;03 ;KEY                    ;X(32)         ;Chr;0032;00;0001;''      ;
0152;   ;LOCAL;0051;03 ;FILLER                 ;X(1)          ;Chr;0001;00;0001;'       ;
0241;   ;PROC ;0000;01 ;PROC_X_P2715_002           ;              ;Str;0000;00;0001;       ;
0242;   ;PROC ;0000;01 ;PROC_X_P2715_003           ;              ;Str;0000;00;0001;       ;
0243;   ;PROC ;0000;01 ;PROC_X_P2715_004           ;              ;Str;0000;00;0001;       ;
0244;   ;PROC ;0000;01 ;PROC_X_P2715_005           ;              ;Str;0000;00;0001;       ;
0245;   ;PROC ;0000;01 ;PROC_FEHLER                ;             ;Str;0000;00;0001;       ;
0246;   ;PROCS;0000;77 ;PCB0                   ;POINTER        ;Ptr;0008;00;0001;       ;
0247;   ;PROCS;0008;77 ;JFE                    ;(05)9         ;Num;0005;00;0001;       ;
0248;   ;PROC ;0000;01 ;PROC_FEHLERMELDUNG  ;              ;Str;0000;00;0001;       ;
0249;CO ;PROCS;0000;77 ;FEHLER_INDEX         ;(05)9      ;Num;0005;00;0001;       ;
0250;C  ;PROCS;0005;77 ;MAX_INDEX            ;(05)9         ;Num;0005;00;0001;30     ;
0251;O  ;PROCS;0010;77 ;FEHLERTEXT           ;X(60)         ;Chr;0060;00;0030;       ;
0252;   ;PROC ;0000;01 ;PROC_DB_AKTION       ;              ;Str;0000;00;0001;       ;
0253;C  ;PROCS;0000;77 ;DB_NR                ;(05)9         ;Num;0005;00;0001;       ;
0254;   ;PROCS;0005;77 ;SSA_PTR              ;POINTER        ;Ptr;0008;00;0001;       ;
0255;P  ;PROCS;0013;77 ;DB_FUNKTION          ;X(4)          ;Chr;0004;00;0001;       ;
0256;P  ;PROCS;0017;77 ;DB_SCHLUESSEL        ;X(*)          ;Chr;0000;00;0001;       ;
0257;C  ;PROCS;0017;77 ;AKTION               ;X(1)          ;Chr;0001;00;0001;       ;
0258;O  ;PROCS;0018;77 ;FOUR                 ;(08)9         ;Num;0008;00;0001;4      ;
```

6.26 Data Reference Table

```
001 P P2715              P PCB_TAA
001 P P2715              P PCB_TA
001 P P2715              P PCB_VA
001 P P2715              P COIBM
001 P P2715              P PLIXOPT
001 P P2715              P PLIXHD
002 P P2715
001 P P2715              O ONE
001 P P2715              O TWO
```

```
001 P P2715              O THREE
001 P P2715              I XCON3
001 P P2715              O FOUR
001 P P2715              I XCON4
001 P P2715              O FIVE
001 P P2715              I XCON5
001 P P2715              O SIX
001 P P2715              I XCON6
001 P P2715              O PBEST
001 P P2715              I IO_BE
001 P P2715              O PVAL
001 P P2715              I IO_VA
001 P P2715              O PTAB
001 P P2715              I IO_TA
001 P P2715              I CONVERSION
001 P P2715              I LISTE
002 L PGM_START
001 L PGM_START         O CT
001 L PGM_START         O DATUM
001 L PGM_START          I XLIT010
001 L PGM_START         O ZEIT
001 L PGM_START         O ZEIT
001 L PGM_START          I XLIT011
001 L PGM_START          I ZEIT
001 L PGM_START          I XLIT012
001 L PGM_START         O PLIRETCODE
001 L PGM_START         O KEY_FELD
001 L PGM_START         P TABELLEN_DB
001 L PGM_START         P KEY_FELD
001 L PGM_START         C PCB.STATUS
001 L PGM_START         O PLIRETCODE
001 L PGM_START         I XCON12
001 L PGM_START         O PDAT
001 L PGM_START         I IO_TA
001 L PGM_START         O AKTUELLE_AKTSTUFE
001 L PGM_START         I DATUMREC.AKTSTUFE
001 L PGM_START         O KEY_FELD
001 L PGM_START          I XLIT013
001 L PGM_START          O GLOBAL_OPERATOR
001 L PGM_START          P TABELLEN_DBA
001 L PGM_START          P KEY_FELD
001 L PGM_START          C PCB.STATUS
001 L PGM_START          C IO_TA
001 L PGM_START          C KEY_FELD
001 L PGM_START          O PLIRETCODE
001 L PGM_START          I XCON12
001 L PGM_START          O PDAT
001 L PGM_START          I IO_TA
001 L PGM_START          O DATUM
001 L PGM_START           I XLIT014
001 L PGM_START           I ABR9900.MASCHOV
001 L PGM_START           I XLIT015
001 L PGM_START           I LISTE
001 L PGM_START          C AKTUELLE_AKTSTUFE
001 L PGM_START          O KEY_FELD
001 L PGM_START          I XLIT016
001 L PGM_START          O SUCHKEY
001 L PGM_START          I XLIT016
001 L PGM_START          O KEY_FELD
001 L PGM_START          I XLIT017
001 L PGM_START          O SUCHKEY
001 L PGM_START          I XLIT017
```

```
001 L PGM_START         O IO_TA
001 L PGM_START         I SUCHKEY
001 L PGM_START         O GLOBAL_OPERATOR
001 L PGM_START         P TABELLEN_DBA
001 L PGM_START         P KEY_FELD
001 L PGM_START         C PCB.STATUS
001 L PGM_START         C IO_TA
001 L PGM_START         C SUCHKEY
001 L PGM_START         O PLIRETCODE
001 L PGM_START          I XCON4
001 L PGM_START         O KEY_FELD
001 L PGM_START         O GLOBAL_OPERATOR
001 L PGM_START         P BESTAND_DB
001 L PGM_START         P KEY_FELD
001 L PGM_START         C PCB.STATUS
001 L PGM_START         C IO_BE
001 L PGM_START         O PLIRETCODE
001 L PGM_START          I XCON8
001 L PGM_START         O PDAT
001 L PGM_START          I IO_BE
001 L PGM_START         O PARM_SICHERST
001 L PGM_START          I XLIT018
001 L PGM_START          I DATUMREC.SICHERST
001 L PGM_START          I XLIT019
002 L LOOP
001 L LOOP              C PCB.STATUS
001 L LOOP              P BESTAND_DB
001 L LOOP              C PCB.STATUS
001 L LOOP              C B_ABRTAGW
001 L LOOP              O KEY_FELD
001 L LOOP              I B_VALNR
001 L LOOP              I B_VALZUS
001 L LOOP              I B_ZUGEH
001 L LOOP              I B_EIGENT
001 L LOOP              I B_NL
001 L LOOP              I B_RESKEYBE
001 L LOOP              P VALOREN_DB
001 L LOOP              P KEY_FELD
001 L LOOP              C PCB.STATUS
001 L LOOP              O PLIRETCODE
001 L LOOP              I XCON8
001 L LOOP              O HERKUNFT
001 L LOOP              I XLIT020
001 L LOOP              C S_SNCD
001 L LOOP              O B_VPFWET
001 L LOOP              O HFLOAT_1
001 L LOOP              I B_TAWERT
001 L LOOP              O HFLOAT_2
001 L LOOP              I S_VPFANS
001 L LOOP              I XCON100
001 L LOOP              O B_VPFWET
001 L LOOP              I HFLOAT_1
001 L LOOP              I HFLOAT_2
001 L LOOP              C S_VALART
001 L LOOP              C XLIT022
001 L LOOP              O BSTKNOM
001 L LOOP              I B_STKNOM
001 L LOOP              C S_SNCD
001 L LOOP              O KEY_FELD
001 L LOOP              I VALOR.C_OWAEC
001 L LOOP              P TABELLEN_DB
001 L LOOP              P KEY_FELD
```

001 L LOOP C PCB.STATUS
001 L LOOP O PLIRETCODE
001 L LOOP I XCON8
001 L LOOP C D_WAEE
001 L LOOP C B_TAWERT
001 L LOOP O HFLOAT_1
001 L LOOP I D_WAEK
001 L LOOP I D_WAEE
001 L LOOP O HFLOAT_2
001 L LOOP I BSTKNOM
001 L LOOP I S_DIV
001 L LOOP I B_TAWERT
001 L LOOP O B_RENDTA
001 L LOOP I HFLOAT_1
001 L LOOP I HFLOAT_2
001 L LOOP O B_TAWERT
001 L LOOP O KEY_FELD
001 L LOOP I VALOR.C_HWAEC
001 L LOOP P TABELLEN_DB
001 L LOOP P KEY_FELD
001 L LOOP C PCB.STATUS
001 L LOOP O KEY_FELD
001 L LOOP I VALOR.C_OWAEC
001 L LOOP P TABELLEN_DB
001 L LOOP P KEY_FELD
001 L LOOP C PCB.STATUS
001 L LOOP O PLIRETCODE
001 L LOOP I XCON8
001 L LOOP C D_WAEE
001 L LOOP C B_TAWERT
001 L LOOP O HFLOAT_1
001 L LOOP I D_WAEK
001 L LOOP I D_WAEE
001 L LOOP O HFLOAT_2
001 L LOOP I BSTKNOM
001 L LOOP I S_DIV
001 L LOOP I XCON100
001 L LOOP I B_TAWERT
001 L LOOP O B_RENDTA
001 L LOOP I HFLOAT_1
001 L LOOP I HFLOAT_2
001 L LOOP O B_TAWERT
001 L LOOP O B_RENDTA
001 L LOOP O HERKUNFT
001 L LOOP I XLIT023
001 L LOOP O BNOMSFRS
001 L LOOP I B_NOMSFR
001 L LOOP O HFLOAT_1
001 L LOOP I B_RENDTA
001 L LOOP O B_TAGEFA
001 L LOOP I BNOMSFRS
001 L LOOP I HFLOAT_1
001 L LOOP O B_ABRTAGW
001 L LOOP P BESTAND_DB
001 L LOOP C PCB.STATUS
001 L LOOP O PLIRETCODE
001 L LOOP I XCON12
001 L LOOP O TAWERT_Z
001 L LOOP I B_TAWERT
001 L LOOP O TAWERT_NEU
001 L LOOP I TAWERT_Z
001 L LOOP O LINECOUNTER

```
001 L LOOP              I LINECOUNTER
001 L LOOP              C LINECOUNTER
001 L LOOP              C XCON60
001 L LOOP              I LISTE
001 L LOOP              O COUNTER
001 L LOOP              I COUNTER
001 L LOOP              C AKTUELLE_AKTSTUFE
001 L LOOP              O KEY_FELD
001 L LOOP              I XLIT016
001 L LOOP              O SUCHKEY
001 L LOOP              I XLIT016
001 L LOOP              O KEY_FELD
001 L LOOP              I XLIT017
001 L LOOP              O SUCHKEY
001 L LOOP              I XLIT017
001 L LOOP              O IO_TA
001 L LOOP              I SUCHKEY
001 L LOOP              O GLOBAL_OPERATOR
001 L LOOP              P TABELLEN_DBA
001 L LOOP              P KEY_FELD
001 L LOOP              C PCB.STATUS
001 L LOOP              C IO_TA
001 L LOOP              C SUCHKEY
001 L LOOP              O PLIRETCODE
001 L LOOP              I XCON4
001 L LOOP              P TABELLEN_DBA
001 L LOOP              C PCB.STATUS
001 L LOOP              O PLIRETCODE
001 L LOOP              I XCON12
001 L LOOP              O PLIRETCODE
001 L LOOP              I XCON12
001 L LOOP              C UPDDAT
001 L LOOP              C PCB_BE
001 L LOOP              O PCB_PTR
001 L LOOP              I PCB_BE
001 L LOOP              O PLIRETCODE
001 L LOOP              I XCON12
002 P X_P2715_002
002 P X_P2715_002           O LOOP_ERROR_COUNT
002 P X_P2715_002           I LOOP_ERROR_COUNT
002 P X_P2715_002           C LOOP_ERROR_COUNT
002 P X_P2715_002           C XCON5
002 P X_P2715_002           P PLIRETCODE
002 P X_P2715_002           C PLIRETCODE
002 P X_P2715_002           C XCON4
002 P X_P2715_002           O IFE
002 P X_P2715_002           P IFE
002 P X_P2715_002           P PCB_PTR
003 P X_P2715_003
003 P X_P2715_003           C XCON612
003 P X_P2715_003           O PLIRETCODE
003 P X_P2715_003           I XCON12
003 P X_P2715_003           O ONSOURCE
003 P X_P2715_003           I XLIT007
003 P X_P2715_003           I ONSOURCE
004 P X_P2715_004
004 P X_P2715_004           C FIXED_ERROR_COUNT
004 P X_P2715_004           C XCON200
004 P X_P2715_004           O PLIRETCODE
004 P X_P2715_004           I XCON12
004 P X_P2715_004           O FIXED_ERROR_COUNT
004 P X_P2715_004           I FIXED_ERROR_COUNT
```

```
004 P X_P2715_004          O ONSOURCE
008 P DB_AKTION             P DB_FUNKTION
008 P DB_AKTION             P DB_SCHLUESSEL
009 P DB_AKTION
008 P DB_AKTION             C DB_NR
008 P DB_AKTION             C BESTAND_DB
008 P DB_AKTION             O SSA_BE.OP
008 P DB_AKTION             I GLOBAL_OPERATOR
008 P DB_AKTION             O SSA_BE.KEY
008 P DB_AKTION             I DB_SCHLUESSEL
008 P DB_AKTION             C AKTION
008 P DB_AKTION             P FOUR
008 P DB_AKTION             P DB_FUNKTION
008 P DB_AKTION             P PCB_BE
008 P DB_AKTION             P IO_BE
008 P DB_AKTION             P SSA_BEU
008 P DB_AKTION             P THREE
008 P DB_AKTION             P DB_FUNKTION
008 P DB_AKTION             P PCB_BE
008 P DB_AKTION             P IO_BE
008 P DB_AKTION             P FOUR
008 P DB_AKTION             P DB_FUNKTION
008 P DB_AKTION             P PCB_BE
008 P DB_AKTION             P IO_BE
008 P DB_AKTION             P SSA_BE
008 P DB_AKTION             O PCB_PTR
008 P DB_AKTION             I PCB_BE
008 P DB_AKTION             C TABELLEN_DBA
008 P DB_AKTION             O SSA_TAA.OP
008 P DB_AKTION             I GLOBAL_OPERATOR
008 P DB_AKTION             O SSA_TAA.KEY
008 P DB_AKTION             I DB_SCHLUESSEL
008 P DB_AKTION             C AKTION
008 P DB_AKTION             P FOUR
008 P DB_AKTION             P DB_FUNKTION
008 P DB_AKTION             P PCB_TAA
008 P DB_AKTION             P IO_TA
008 P DB_AKTION             P SSA_TAAU
008 P DB_AKTION             P THREE
008 P DB_AKTION             P DB_FUNKTION
008 P DB_AKTION             P PCB_TAA
008 P DB_AKTION             P IO_TAA
008 P DB_AKTION             P FOUR
008 P DB_AKTION             P DB_FUNKTION
008 P DB_AKTION             P PCB_TAA
008 P DB_AKTION             P IO_TA
008 P DB_AKTION             P SSA_TAA
008 P DB_AKTION             O PCB_PTR
008 P DB_AKTION             I PCB_TAA
008 P DB_AKTION             C TABELLEN_DB
008 P DB_AKTION             O SSA_TA.OP
008 P DB_AKTION             I GLOBAL_OPERATOR
008 P DB_AKTION             O SSA_TA.KEY
008 P DB_AKTION             I DB_SCHLUESSEL
008 P DB_AKTION             C AKTION
008 P DB_AKTION             P FOUR
008 P DB_AKTION             P DB_FUNKTION
008 P DB_AKTION             P PCB_TA
008 P DB_AKTION             P IO_TA
008 P DB_AKTION             P SSA_TAU
008 P DB_AKTION             P THREE
008 P DB_AKTION             P DB_FUNKTION
```

```
008 P DB_AKTION              P PCB_TA
008 P DB_AKTION              P IO_TA
008 P DB_AKTION              P FOUR
008 P DB_AKTION              P DB_FUNKTION
008 P DB_AKTION              P PCB_TA
008 P DB_AKTION              P IO_TA
008 P DB_AKTION              P SSA_TA
008 P DB_AKTION              O PCB_PTR
008 P DB_AKTION              I PCB_TA
008 P DB_AKTION              C VALOREN_DB
008 P DB_AKTION              O SSA_VA.OP
008 P DB_AKTION              I GLOBAL_OPERATOR
008 P DB_AKTION              O SSA_VA.KEY
008 P DB_AKTION              I DB_SCHLUESSEL
008 P DB_AKTION              C AKTION
008 P DB_AKTION              P FOUR
008 P DB_AKTION              P DB_FUNKTION
008 P DB_AKTION              P PCB_VA
008 P DB_AKTION              P IO_VA
008 P DB_AKTION              P SSA_VAU
008 P DB_AKTION              P THREE
008 P DB_AKTION              P DB_FUNKTION
008 P DB_AKTION              P PCB_VA
008 P DB_AKTION              P IO_VA
008 P DB_AKTION              P FOUR
008 P DB_AKTION              P DB_FUNKTION
008 P DB_AKTION              P PCB_VA
008 P DB_AKTION              P IO_VA
008 P DB_AKTION              P SSA_VA
008 P DB_AKTION              O PCB_PTR
008 P DB_AKTION              I PCB_VA
008 P DB_AKTION              O PLIRETCODE
008 P DB_AKTION              I XCON12
008 P DB_AKTION              O GLOBAL_OPERATOR
032 L PROGRAMM_ENDE
```

6.27 XML Class Documentation

```
<Class name = "PROC_FEHLER" type = "PROCS" >
  <Object name = "FEHLER_Object" type = "char[13  ]" />
  <Attributes>
    <Attr name = "PCB0                " type = "9(8)    " pos = "0000" lng = "0013" comment = "undefined"/>
    <Attr name = "JFE                 " type = "9(05)   " pos = "0008" lng = "0005" comment = "undefined"/>
  </Attributes>
  <ProcessingMethods>
    <Method name = "P2715_FEHLER" type = "processing" >
      <Parameters>
      </Parameters>
      <Successors>
        <NextMethod name = "P2715.PROCS.PROC_P2715.P2715_X_PUT_020" type = "PERF"/>
      </Successors>
    </Method>
  </ProcessingMethods>
</Class>
<Class name = "PROC_FEHLERMELDUNG" type = "PROCS" >
  <Object name = "FEHLERMELDUNG_Object" type = "char[1810]" />
  <Attributes>
    <Attr name = "FEHLER_INDEX " type = "9(05)   " pos = "0000" lng = "1810" comment = "undefined"/>
    <Attr name = "MAX_INDEX       " type = "9(05)   " pos = "0005" lng = "0005" comment = "undefined"/>
    <Attr name = "FEHLERTEXT      " type = "X(60)   " pos = "0010" lng = "0060" comment = "undefined"/>
  </Attributes>
  <ProcessingMethods>
```

```
      <Method name = "P2715_FEHLERMELDUNG" type = "processing" >
        <Parameters>
          <Param use = "Input" name = "FEHLER_INDEX" class = "PROC_FEHLERMELDUNG" type =
"int"/>
          <Param use = "Input" name = "MAX_INDEX" class = "PROC_FEHLERMELDUNG" type = "int"/>
        </Parameters>
        <Successors>
          <NextMethod name = "P2715.PROCS.PROC_P2715.P2715_X_PUT_021" type = "PERF"/>
          <NextMethod name = "P2715.PROCS.PROC_P2715.P2715_FEHLERM_END" type = "GOTO"/>
          <NextMethod name = "P2715.PROCS.PROC_P2715.P2715_X_PUT_022" type = "PERF"/>
          <NextMethod name = "P2715.PROCS.PROC_P2715.P2715_X_PUT_023" type = "PERF"/>
        </Successors>
      </Method>
    </ProcessingMethods>
  </Class>
  <Class name = "PROC_DB_AKTION" type = "PROCS" >
    <Object name = "DB_AKTION_Object" type = "char[211 ]" />
    <Attributes>
      <Attr name = "DB_NR              " type = "9(05)   " pos = "0000" lng = "0005" comment = "undefined"/>
      <Attr name = "SSA_PTR            " type = "9(8)    " pos = "0150" lng = "0008" comment = "undefined"/>
      <Attr name = "DB_FUNKTION    " type = "X(4)    " pos = "0158" lng = "0004" comment = "undefined"/>
      <Attr name = "DB_SCHLUESSEL" type = "X(40)   " pos = "0162" lng = "0040" comment =
"undefined"/>
      <Attr name = "AKTION             " type = "X(1)    " pos = "0202" lng = "0001" comment = "undefined"/>
      <Attr name = "FOUR               " type = "9(08)   " pos = "0203" lng = "0008" comment = "undefined"/>
    </Attributes>
    <ProcessingMethods>
      <Method name = "P2715_DB_AKTION" type = "processing" >
        <Parameters>
          <Param use = "InOut" name = "DB_FUNKTION" class = "PROC_DB_AKTION" type = "String"/>
          <Param use = "InOut" name = "DB_SCHLUESSEL" class = "PROC_DB_AKTION" type = "String"/>
          <Param use = "Input" name = "DB_NR" class = "WORK_P2715_AUTO" type = "int" />
          <Param use = "Input" name = "BESTAND_DB" class = "WORK_P2715_AUTO" type = "int"/>
          <Param use = "Output" name = "OP" class = "WORK_SSA_BE" type = "String" />
          <Param use = "Input" name = "GLOBAL_OPERATOR" class = "WORK_P2715_AUTO" type =
"String"/>
          <Param use = "Output" name = "KEY" class = "WORK_SSA_BE" type = "String" />
          <Param use = "Input" name = "DB_SCHLUESSEL" class = "PROC_DB_AKTION" type = "String"/>
          <Param use = "Input" name = "AKTION" class = "PROC_DB_AKTION" type = "char" />
          <Param use = "InOut" name = "FOUR" class = "WORK_P2715_AUTO" type = "int" />
          <Param use = "InOut" name = "PCB_BE" class = "PROC_P2715" type = "String" />
          <Param use = "InOut" name = "IO_BE" class = "WORK_P2715_AUTO" type = "String" />
          <Param use = "InOut" name = "SSA_BEU" class = "WORK_SSA_BEU" type = "String" />
          <Param use = "InOut" name = "THREE" class = "WORK_P2715_AUTO" type = "int" />
          <Param use = "InOut" name = "SSA_BE" class = "WORK_SSA_BE" type = "String" />
          <Param use = "Output" name = "PCB_PTR" class = "PROC_P2715" type = "String" />
          <Param use = "Input" name = "PCB_BE" class = "PROC_P2715" type = "String" />
          <Param use = "Input" name = "TABELLEN_DBA" class = "WORK_P2715_AUTO" type = "int" />
          <Param use = "InOut" name = "PCB_TAA" class = "PROC_P2715" type = "String" />
          <Param use = "InOut" name = "IO_TA" class = "WORK_P2715_AUTO" type = "String" />
          <Param use = "InOut" name = "SSA_TAAU" class = "WORK_SSA_TAAU" type = "String" />
          <Param use = "InOut" name = "IO_TAA" class = "BASE_IO_TA_BASED" type = "String" />
          <Param use = "InOut" name = "SSA_TAA" class = "WORK_SSA_TAA" type = "String" />
          <Param use = "Input" name = "PCB_TAA" class = "PROC_P2715" type = "String" />
          <Param use = "Input" name = "TABELLEN_DB" class = "WORK_P2715_AUTO" type = "int"/>
          <Param use = "InOut" name = "PCB_TA" class = "PROC_P2715" type = "String" />
          <Param use = "InOut" name = "SSA_TAU" class = "WORK_SSA_TAU" type = "String" />
          <Param use = "InOut" name = "SSA_TA" class = "WORK_SSA_TA" type = "String" />
          <Param use = "Input" name = "PCB_TA" class = "PROC_P2715" type = "String" />
          <Param use = "Input" name = "VALOREN_DB" class = "WORK_P2715_AUTO" type = "int"/>
          <Param use = "InOut" name = "PCB_VA" class = "PROC_P2715" type = "String" />
          <Param use = "InOut" name = "IO_VA" class = "WORK_P2715_AUTO" type = "String" />
```

```
            <Param use = "InOut" name = "SSA_VAU" class = "WORK_SSA_VAU" type = "String" />
            <Param use = "InOut" name = "SSA_VA" class = "WORK_SSA_VA" type = "String" />
            <Param use = "Input" name = "PCB_VA" class = "PROC_P2715" type = "String" />
            <Param use = "Output" name = "PLIRETCODE" class = "WORK_P2715_AUTO" type = "int"/>
            <Param use = "Input" name = "XCON12" class = "WORK_XCONSTANT_TAB" type = "int"/>
            <Param use = "Output" name = "GLOBAL_OPERATOR" class = "WORK_P2715_AUTO" type =
"String"/>
          </Parameters>
          <Successors>
            <NextMethod name = "P2715:MAIN.P2715_PLITDLI" type = "CALL"/>
            <NextMethod name = "P2715.PROCS.PROC_FEHLERMELDUNG.P2715_FEHLERMELDUNG"
type = "PERF"/>
          </Successors>
        </Method>
      </ProcessingMethods>
    </Class>
```

6.28 JavaDoc View of a Class