MDBS REPORT DEFINITION REFERENCE MANUAL

The MDBS RDL MANUAL

Version 3.08

Micro Data Base Systems, Inc.

P. O. Box 248

Lafayette, Indiana 47902

USA

Telex: 209147 ISE UR

(312) 303-6300 (in Illinois)

December 1985

Rev. 85A

3255328

NEW RELEASES, VERSIONS, AND A WARNING

Any programming endeavor of the magnitude of the MDBS software will necessarily continue to evolve over time. Realizing this, Micro Data Base Systems, Inc., vows to provide its users with updates to **this version** for a nominal handling fee.

New versions of MDBS software will be considered as separate products. However, bona fide owners of previous versions are generally entitled to a preferential rate structure.

Finally, each copy of our software is personalized to identify the licensee. There are several levels of this personalization, some of which involve encryption methods guaranteed to be combinatorially difficult to decipher. Our products have been produced with a very substantial investment of capital and labor, to say nothing of the years of prior involvement in the data base management area by our principals. Accordingly, we are seriously concerned about any unauthorized copying of our products and will take any and all available legal action against illegal copying or distribution of our products.

Table of Contents

I. OVERVIEW 1
A. Organization 1
B. Introduction to RDL. 1

II. DEFINITIONS 5
A. Designers and End Users. 5
B. Reports and Pages. 5
C. Report Details 5
D. Prompt Variables 6
E. Computed Variables and Functions 6
F. Grouping: Headers and Footers 7
G. Sorting. 8
H. Relationships. 8

III. MDBS.RDL SPECIFICATIONS 9
A. Notational Conventions 9
B. RDL Sections 10
Identification Section. 12
Prompt Section. 14
Output Control Section. 17
Function Definition Section 20
Computed Variable Section 26
Report Detail Section 28
Group Section 34

IV. USING THE MDBS.RDL SOFTWARE 43
A. Overview 43
B. The RDL Analyzer 43
C. Compiling the Generated Program(s) 45
D. Executing the Compiled Program(s). 46

V. RDL ANALYZER DIAGNOSTICS. 47

VI. RDL EXAMPLES 53
Example 1. 56
Example 2. 61
Example 3. 70
Example 4. 80
Example 5. 91

FIGURES
Figure I-1. The Role of RDL. 4
Figure III-1. Notation 10
Figure VI-1. Sample Schema 53
Figure VI-2. Sample DDL Specification. 54

APPENDICES
Appendix A. RDL Keywords A-1
Appendix B. The RDL Syntax B-1

INDEX Index-1

I. OVERVIEW

A. Organization

The MDBS RDL Manual describes the optional RDL module of the MDBS III data base management system.* It is assumed that the reader is familiar with the postrelational, extended-network schemas supported by MDBS. Knowledge of the other MDBS languages for data base processing (e.g., DML, QRS, IDML) is not required in order to use the Report Definition Language. However, familiarity with QRS will facilitate the process of learning to use RDL, since RDL is designed to be consistent with QRS.

Chapter I of this manual concludes with an introduction to the role of RDL in application development. The RDL constructs and features available for designing a report are explained in Chapter II. Details of the RDL syntax are formally presented in Chapter III. Chapter IV explains how to use the RDL Analyzer and the final chapter discusses diagnostics that can arise during RDL analysis.

B. Introduction to RDL

The Report Definition Language (RDL) is another aid that helps magnify the productivity of application system developers who use MDBS. A developer uses the RDL to formally specify the characteristics of a desired report and the prompting behavior of a program that generates that report. An RDL specification is input to the RDL Analyzer which checks the specification for consistency and syntactic correctness. If no errors are detected, the RDL Analyzer automatically generates a complete program which behaves in accordance with the RDL specifications. The resultant program contains the DML commands needed to generate the desired report. These are embedded within the control structures of a C host language.

The role of RDL in application system development and usage is illustrated in Figure I-1. RDL enables a report designer who knows nothing about DML and C to produce application programs written in C and containing all necessary DML logic for extracting and displaying desired data from an MDBS data base. A report designer does need to know the schema of the data base being used. The generated program handles all report formatting as specified with the RDL and can route the report to a printer (using preprinted forms, if desired), a console, or a disk file. The generated program will also prompt an end user for various information as indicated in the RDL specification. An end user's responses serve to initialize variables in the generated program at execution time, thereby allowing the end user to place conditions on the retrieval of data for the report and to control the appearance of the report.

---

* MDBS III Version 3.06 (or greater).

An RDL specification can have up to seven distinct sections for specifying the behavior of the program that is to be generated:

Identification section
Prompt section
Output control section
Function definition section
Computed variable section
Report detail section
Group section

The sections must appear in this sequence. All specifications within a section are made in a free-form manner. The major characteristics of each section are summarized here.

The identification section is used to declare the name of the file or files that will hold the generated program. It is also used to indicate the name of the data base which the generated program will utilize, a user id for that data base, and that user's password.

A prompt variable is used to gather information from an end user. For each prompt variable declared in the RDL prompt section, the generated application program will prompt the end user to enter some data which will be used by the program to determine the ultimate appearance and content of the report being produced. For instance, the data base name, user id, and password might be treated as prompt variables. When specifying a prompt variable, the report designer gives its name, type, size, and the prompt message that is to be presented on the console to an end user. The designer can also specify a default value for each prompt variable. Thus, prompt variables serve as parameters to the generated application program.

The output control section is used to specify output page depth, left and top margins, pauses in output display, and the device to which the output report is to be routed. These can be either explicitly declared or they can be handled as prompt variables, enabling the end user of the generated program to determine these output traits.

A report designer can optionally specify one or more functions that are to be used within the generated program. These are declared in the RDL function definition section. A function can have multiple input parameters. Within the definition of a function local variables can be declared. These, together with global variables (e.g., prompt variables) and data items, can be used within assignment statements, if-then-else control structures, and test-case control structures inside the body of the function. Each function is specified so that it returns a single value.

Developer-defined functions, together with built-in statistical functions (e.g. SUM, AVERAGE), can be used to declare computed variables. This is accomplished in the computed variable section. Each computed variable is given a name, type and size. In addition, a formula is specified, indicating how the item's value is to be determined. The formula consists of a function and its arguments. For built-in functions, an optional conditional clause can be specified. For example, FOR (YTDEARN<10000) SUM (YTDEARN) is a formula that says the computed variable's value will be the sum of all values of the YTDEARN data item (in the data base) which are less than 10000.

The report detail section defines the content of the report. In this section the developer indicates which data items (from the data dictionary or screen dictionary), computed variables, prompt variables, and expressions involving any of the former three are to have their value(s) appear in the report. The physical position of each of these in the output report is specified. This section also has an optional FOR clause and a required THRU clause like those that appear in a QRS statement. Furthermore, an optional SORT clause can be specified to cause the report details to be sorted on various criteria.

The group section allows a developer to specify page breaks and various kinds of groupings of report details. Page breaks are stated in terms of data items. Whenever a data item's value changes, the next report detail is forced to the next report page. Grouping means that all report details having some specified characteristics in common appear as a group in the output report. Headers and footers automatically accompany each group, if desired. Also, automatic sorting of groups can be requested.

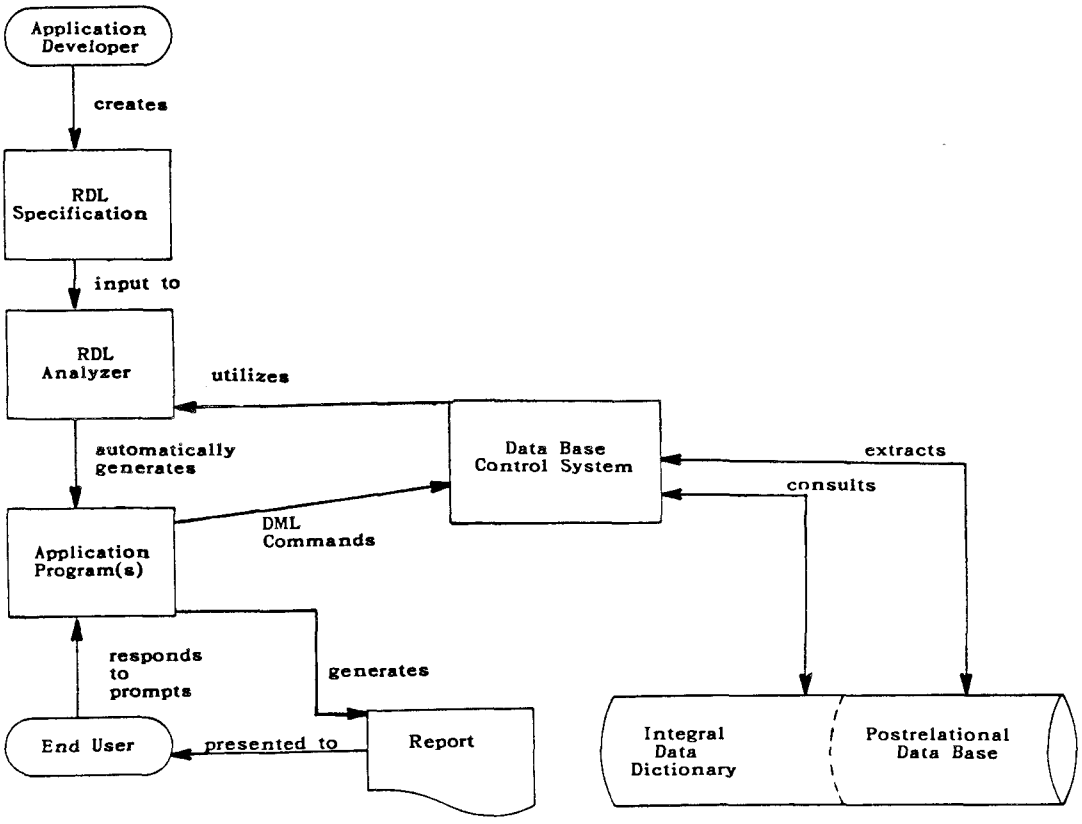


Figure I-1. The Role of RDL

II. DEFINITIONS

A. Designers and End Users

A report designer creates an RDL specification which defines the appearance and content of a report. The RDL Analyzer is a program that accepts an RDL specification as input and generates a program (or two programs, in some cases). After a generated program has been compiled, it can be executed by an end user. After prompting the end user for various information, the executing program produces a report for the end user. In the event that two programs were generated by the RDL Analyzer, the first program will produce an intermediate file which is then utilized by the second program to produce the final report.

Since the RDL Analyzer produces C source code programs, a designer with C programming expertise can modify generated programs if desired.

B. Reports and Pages

A report consists of one or more pages. A page consists of one or more horizontal lines of output. The report designer controls the maximum number of lines per report page. The left margin and top margin for a report's pages are also determined by the designer. The area to the left of the left margin on each page is blank. Similarly, nothing appears above a page's top margin. On each page, the first line beneath the top margin will (by default) contain the date and the page number. If desired, the report designer can suppress this automatic display of date and page number. A report can be routed to a printer, console screen, and/or disk file.

C. Report Details

A report typically contains many report details. All report details within a report have the same structural appearance, but they differ in terms of content. Each report detail consists of one or more lines containing literals and/or data values. Literals are titles, labels, and other strings of characters that do not change from one report detail to another. Data values can differ for different report details. Data values can be obtained from a data base and can be obtained from computations performed by the generated program. The positions (both horizontal and vertical) of literals and data values within a report detail are controlled by the report designer.

D. Prompt Variables

The designer can declare variables whose values are to be determined by an end user when the generated program begins executing. These are called prompt variables, because the program prompts the end user to acquire each variable's value. For each prompt variable, the designer specifies what the prompt message will be. A default response can also be declared for each prompt variable.

E. Computed Variables and Functions

The simplest kind of report detail displays data values that the generated program obtained from prompt variables or has retrieved from the data base. Beyond this, the designer can define computed variables whose values are automatically calculated (and displayed) by the generated program for each report detail. A computed variable is defined in terms of a function and function arguments. This can be a built-in function or a special function created by the report designer.

The seven built-in functions compute various statistics: COUNT, SUM, MIN, MAX, AVERAGE, STDDEV, and PERCENT. When a statistical function is used to define a computed variable, it can have one or two arguments. These arguments are data items.

If two arguments are used, then the second argument restricts the scope of the statistic's computation. For instance, AVERAGE (MOSAL,DNUM) is the average of monthly salaries for consecutive report details with the same department number. Normally, this function would be included in the footer defined for department groups (footers are described in the next section). When a report is generated, each group of details for a department number would be followed by a footer showing the average monthly salary for the department's employees. If this function is included instead in the definition for report details, then a department's average will appear in each of the details generated for that department group.

If only one argument is used, the scope of values used in computing the statistic depend on whether the computation occurs in a report detail, a group footer, a page footer or a report footer. In a report detail, the statistic is computed with all values* of the data item argument that have been retrieved. In a group (page or report) footer, the computation uses only those values* retrieved for that group (page or report). For instance, the value of AVERAGE (MOSAL) in a report detail is the average of all monthly salaries for employees that appear in the report. In a page footer, it is the average of all salaries on the page, and so forth.

---

* If no values have been retrieved, each statistic is set to 0.

A designer can optionally specify conditions for built-in functions. To define a computed variable whose value is the average department salary for employees over 30 years of age, the designer specifies

```
FOR (AGE>30) AVERAGE (MOSAL,DNUM).
```

The report designer can alternatively define computed variables in terms of special functions. A special function is created by the designer. It is essentially a program module that computes a value. Inputs to the function are specified as arguments when the function is invoked. Each argument corresponds to a parameter used within the function's body. Local working variables can also be declared within the function's body. In addition to assignment statements, if-then-else and test-case statements can be used within a function. A return statement is used to exit from a function with the function's value.

F. Grouping: Headers and Footers

The designer can request that report details be grouped in various ways. A major aspect of grouping is that a header and a footer can be specified to automatically accompany each group of report details. A header or footer can be designed to consist of one or more lines containing literals and/or data values. The data values can be values of data items, prompt variables, or computed variables.

The header for a group appears immediately before the group's report details. Normally, the header is designed to contain descriptive information introducing a group of details. A footer for a group appears just after the group's details. It is usually designed to contain information summarizing a group of details.

The simplest kind of grouping is to group by report. This causes all of a report's details to be treated as a group. The header and footer specified for this kind of grouping appear at the start and finish of the report, respectively. Report details can also be grouped by page. Since each group consists of a page of report details, the header and footer specified for page grouping serve as page headers and footers, appearing at the start and finish of each page.

Another kind of grouping allows report details to be separated into groups based on the values of a specified data item (or computed variable). As the generated program creates each new report detail, a check is made to see whether the specified data item (or computed variable) value has changed since the last report detail. If it has changed, then the newly created report detail begins a new group of report details. As with report grouping and page grouping, each resulting group can be prefaced by a header and followed by a footer.

The designer can specify multiple levels of grouping. The grouping of one level occurs within each group produced by the next higher (i.e., broader) level of grouping. For instance, suppose that report details consist of employee information and that three levels of grouping have been declared by the designer. At the highest level

there is grouping by company, then grouping by division, and finally grouping by department. The generated program produces an output report having employees grouped by company. Within each company group, the employee details are grouped by the division in which they work. Within each division, employees are grouped by department.

G. Sorting

A designer can specify two types of sorting: sorting of report details and sorting of groups. If a designer has requested sorting of report details, then the generated program will output report details in sorted order. If grouping (other than by report or by page) has been requested for the report, then the sorting of report details occurs within each of the lowest level groups. When specifying a grouping (other than by report or by page) the designer can request a sorting of the resultant groups.

In general sorting can be based on the values of one or more data items and/or computed variables. Any mixture of ascending and descending directions can be specified. For instance, the sorting of employee details (within each division) might be based on ascending job codes and descending monthly salary. The sorting of department groups (within a division) might be based on ascending department code. Within each company, division groups could be ordered by ascending division names. Finally, company groups might be arranged in terms of descending (i.e., highest to lowest) annual sales.

H. Relationships

Just as important as the data items used in generating a report, are the relationships among those data items. Because of the inherent flexibility of MDBS's postrelational approach to data structuring, a schema allows many relationships to exist among a group of selected data items. Set names are used to indicate the semantics of various relationships. The report designer can choose which relationships are to be used by the generated program in producing its report.

III. MDBS.RDL SPECIFICATIONS

A. Notational Conventions

After the content and layout for a report has been determined, the report can be formally specified with the Report Definition Language. An RDL source specification consists of several sections. Each section consists of various kinds of clauses; some of which are optional and some of which are required. These clauses are presented on a section-by-section basis. Discussion of those clauses that can be regarded as advanced features are denoted in the left margin by a vertical bar. The complete syntactic description of all clauses appears in Appendix B. Figure III-1 shows the meanings of the notations used.

A clause is composed of one or more of the following terms: keywords, identifiers, file names, strings, integers, real numbers, formats, user names, passwords, procedures.

Keywords are denoted by upper case letters. All other terms in a clause are in lower case. This upper case vs. lower case distinction is for documentation purposes only. Upper and lower case can be mixed in an actual RDL source specification. A complete list of RDL keywords appears in Appendix A.

Identifiers used in describing an RDL section are denoted by id-1, id-2, id-3, etc. Identifiers are selected by the report designer. An identifier consists of from one to eight alphanumeric characters. An RDL keyword can be used as an identifier if it is immediately preceded by the ITEM keyword. If the ITEM keyword is not inserted prior to an identifier, then that identifier should not be an RDL keyword.

File names used in describing an RDL section are denoted by file-1, file-2, file-3, etc. File names are chosen by the report designer; they must be fully qualified file names within the host operating system.

Integers used in describing an RDL section are denoted by int-1, int-2, int-3, etc.

Real numbers used in describing an RDL section are denoted by real-1, real-2, real-3, etc.

Formats are used to control the output appearances of numeric expressions. Denoted by fmt, a format is enclosed in a matching pair of quotes. Formats are specified in the C languages printf style. The manual that accompanies the C compiler should have descriptions of permissible printf formats.

Strings of characters are denoted by string-1, string-2, etc. Each string within an RDL section has a unique number as its suffix. The actual string values are chosen by the report designer. A string value must fit within a pair of matching double quotes on one line of the RDL source specification.

User names, denoted by usr, are one to sixteen character (alphanumeric) names.

Passwords, denoted by pass, are one to twelve character (alphanumeric) strings.

Procedures used in describing an RDL section are denoted by procedure-1, procedure-3, etc. A procedure is a program consisting of statements of various kinds: assignment, if-then-else, test-case, break, and return.

___	An underlined expression must appear.
[ ]	Zero or one of the alternatives within the brackets must be used.
{ }	Exactly one of the alternatives within the braces must appear.
{ }*	One or more of the alternatives within the braces must appear.

Figure III-1. Notation

B. RDL Sections

An RDL source specification consists of up to seven kinds of sections. These sections must appear in the order given below. Clauses within a section are stated in a free format (i.e., column positioning is unimportant for the RDL Analyzer). Upper case and lower case can be used interchangeably. More than one clause can appear on a line; a clause (but not a term) can be split over more than one line; unless otherwise indicated, optional clauses within a section can appear in any order. Clauses are separated by blanks. Comments can appear between any two terms of clauses within an RDL source specification.

The kinds of RDL sections and their ordering are as follows:

dentification Section
Prompt Section (optional)
Output Control Section (optional)
Function Definition Section (optional)
Computed Variable Section (optional)
Report Detail Section

Group Section (optional)

Comments in an RDL source specification

The characters /* located anywhere in the RDL source specification indicates that all text that follows is a comment, until the comment is terminated by the characters */

Comments can extend over many lines.

Examples:
```
/* THIS IS A COMMENT */

/*****
 *
 *  RDL SOURCE SPECIFICATION
 *
 *****/

/*
    FUNCTION DEFINITIONS
*/
```


IDENTIFICATION SECTION

The identification section must be present. This section is used to identify which MDBS data base will be accessed by the generated program. This will be referred to as the target data base. In addition, the report designer uses this section to specify

- a user name and password, to confirm that the designer does have access to the data base,
- the names of files that are to hold the generated program.

The identification section's clauses should be stated in the order shown below.

a) Data base clause DATABASE FILE NAME IS "file-1"

The indicated file-1 must be the physical file that holds the main area of the target data base. The data dictionary in this main area is used by the RDL Analyzer during program generation. File-1 is a fully qualified file name for the host operating system.

```
Examples:  DATABASE FILE NAME IS "B:DEB"
           DATABASE NAME "INVEN/DB:1"
           DATABASE "/usr/db/customer.db"
```

b) User clause (optional) USER IS {usr / "usr"} WITH {pass / "pass"}

The indicated usr must be a bona fide user for the target data base and pass must be that user's password. If this clause is omitted, the designer must specify the user and password on the operating system command line when the RDL Analyzer is invoked. Such an omission is desirable in security-sensitive circumstances, because it means that a valid name and password will not appear in the RDL specification. If use or pass contains a blank, then quotes must be used. If quotes are not used for usr or pass then all its alphabetic characters are converted to upper case.

```
Examples:  USER IS "Alvin Wade" WITH SECRET
           USER GC15 "DJL"
           USER KnowledgeMaster WITH MDBS
           USER deb WITH "mon reve"
```


c) Program clause (optional) PROGRAM FILE IS "file-2"

The RDL Analyzer will generate one or two C source code programs, plus two command files. One command file will compile the source program(s). The other will execute the compiled program(s).

The indicated file-2 name can be up to seven characters long and can be qualified by a drive/directory. It will be appended with 1 (and 2) to yield the name(s) of the file(s) holding the generated source program(s). A .C extension is used for source code files. The command file for compilation uses the file-2 name with a C prefix. The command file that executes has the name indicated by file-2.

If the program clause is omitted, file-2 is assumed to be RPG.

Examples: PROGRAM FILE IS "INV"
PROGRAM "B:SALES"

PROMPT SECTION

The prompt section is optional. It is used to define the prompts which the generated program will present to an end user. Each prompt is defined in terms of

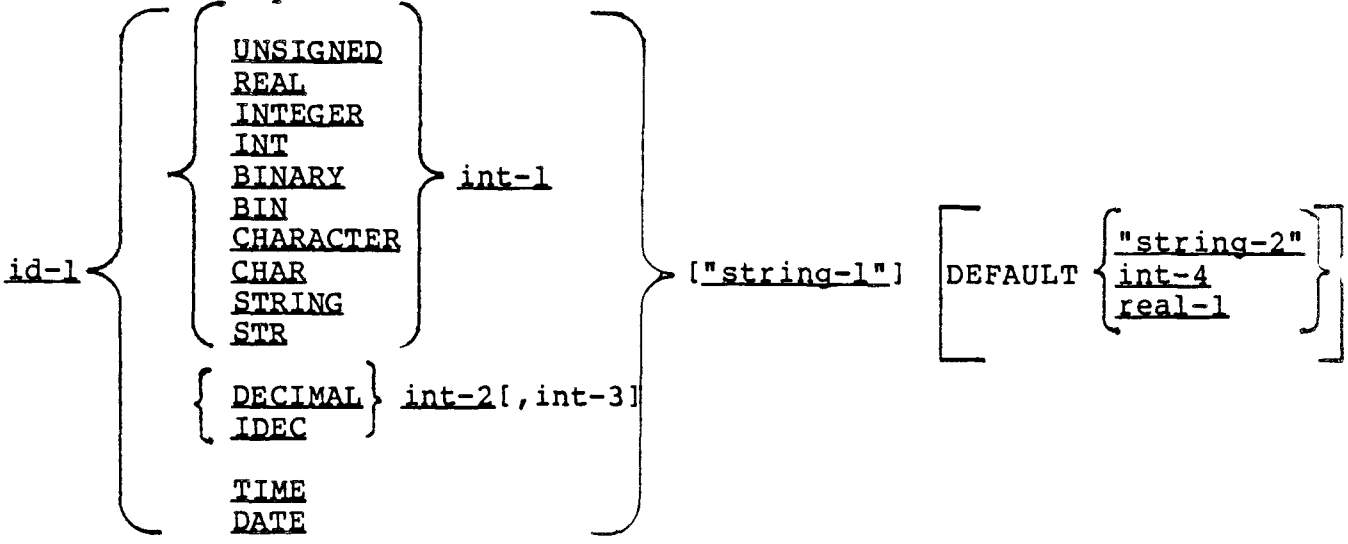
- a prompt variable which will hold the end user's response,
- the type and size of the prompt variable,
- a prompt message which will be presented to the end user,
- a default value for the prompt variable

a) Prompt clause PROMPT VARIABLES ARE

This clause is required if you intend to define prompts. It is followed by one or more prompt definition clauses.

Examples: PROMPT VARIABLES ARE
PROMPT

b) Prompt definition clause



A prompt definition begins wth id-1, which is the name of the prompt variable being defined. It cannot be a data item. This variable is declared to be one of nine types: unsigned, real, integer, binary, character, string, internal decimal (idec), time, or date. With the exception of time and date variables, a size can be specified. For a time prompt variable, the end user will be prompted to enter nine characters of the form hhh:mm:ss to indicate hours (hhh), minutes (mm) and seconds (ss). For a date type, the end user is prompted for ten characters of the form mm/dd/yyyy to indicate a month (mm), day (dd) and year (yyyy). An internal decimal size consists of two integers. The first, int-2, indicates the number of digits accepted for an idec prompt variable. The number of digits to the right of the decimal point is optionally specified by int-3. If int-3 is omitted, then 0 is assumed. For all other types, int-1 indicates the number of bytes used to hold the prompt variable's value. For a 1 or 2 byte integer, a user will be

allowed to enter up to 7 digits (including the sign). Up to 12 can be entered for larger integers. Up to 25 digits (including the sign and decimal point) can be entered for a real variable. Minimum and maximum sizes were discussed in the preceding chapter.

If the optional string-1 is specified, it serves as a prompt message which is output by the generated program. The end user sees this message on the console screen just prior to the place where the variable's value is to be entered.

As another option, a default value can be declared for a prompt variable. This value is an integer for unsigned or integer types, a real number for idec or real types, and a string for any of the other type. If a default value has been declared for a prompt variable, then that value is displayed in square brackets next to the prompt message (if any). An end user who desires the default value, needs only to press the Return (Enter) key when prompted for a value.

UFILE, UNAME and UWORD are three special prompt variables. If these are defined in the prompt section, they must appear in the above sequence and must precede the definitions of any other prompt variables. Types and sizes are not specified for these special prompt variables. The values that an end user supplies for these prompts are immediately used by the generated program to attempt to open the data base. If the end user has responded with valid values, then the data base is opened with the read/write access codes assigned to the indicated user name. The rest of the prompts are then presented. If an end user has responded with invalid values, then execution of the generated program is aborted.

Omission of these three special prompt variables from the RDL specification means that the file, user name, and password specified by the designer (for RDL analysis) appear in the source code of the generated program. As a result, the generated program opens the data base with the designer-specified file, user name, and password. The end user therefore has the read/write access privileges associated with the designer-specified user name.

UDATE, UPAGE and UTIME are also special prompt variables. Types and sizes are not specified when these prompt variables are declared. The value that an end user supplies for UDATE is used as the date that automatically appears at the top of each report page. If UDATE is not declared as a prompt variable, then the generated program obtains the date from the machine clock. The value that an end user supplies for UPAGE is the page number that automatically appears at the top of the report's first page. If UPAGE is not declared as a prompt variable, then 1 is the number of the report's first page. The value that an end user supplies for UTIME is used to "time-stamp" the report if UTIME is specified in the PLACE clause for a detail, header or footer.

Examples:

```
UNAME "USER NAME:"
BDATE DATE "BEGINNING DATE: "
EDATE DATE "ENDING DATE: "
MAXSAL IDEC 8,2 "MAXIMUM SALARY OF INTEREST IS "
DNAME CHAR 12 "DEPARTMENT NAME?"
CCAP REAL 4 "Estimated cost of capital: "
UCOM STR 75
   "Enter comments to be included on each report page:"
CDATE DATE "TODAY'S DATE "
CTIME TIME "CURRENT TIME ="
ODEV CHAR 7 "OUTPUT DEVICE (S or P)?" DEFAULT "PRINTER"
PSIZ INT 1 "How many lines per page?" DEFAULT 55
UPAGE "Starting page?" DEFAULT 5
UTIME "What time is it?"
```


OUTPUT CONTROL SECTION

The output control section is used to govern the general appearance of reports produced by the generated program. In this section a report's page size, left margin width, and top margin depth can be specified. The automatic date and page numering for each page can be suppressed. There are also clauses to indicate output pauses and the device (printer, screen, disk file) to which output from the generated program is to be sent. The output control section and any of the six clauses in this section are optional. The clauses in this section can be specified in any order.

a) Page size clause (optional) PAGE SIZE {int-1 / id-1} LINES

This clause is used to specify the maximum number of lines that will appear on each page of the report. If int-1 is specified, then that integer is "hard-wired" into the generated program. Alternatively, a previously defined prompt variable, id-1, can be specified. This allows an end user of the generated program to alter the page size as desired, via a response to the id-1 prompt. If the optional page size clause is omitted, a default of 24 lines per page is used.

If the identification section's screen clause is present, then the page size clause has no effect.

Examples:
```
PAGE SIZE IS 20
PAGE 55 LINES
PAGE PSIZ
```

b) Left Margin clause (optional) LEFT MARGIN {int-2 / id-2}

This clause is used to specify the location of the left margin for report pages. The number of spaces to the left of this margin is indicated either by int-2 (an integer) or by a previously defined prompt variable (id-2). The placements of report details within a page are made relative to the left margin. If the optional left margin clause is omitted, a left margin of five spaces is used.

When windows or screens are placed on a page, the left margin is ignored.

Examples:
```
LEFT MARGIN 2
LEFT 8
LEFT LPROMPT
```


c) Top margin clause (optional) TOP MARGIN { int-3 / id-3 }

The top margin clause is used to specify the location of the top margin for report pages. The number of blank lines above this margin is indicated either by int-3 (an integer) or by a previously defined prompt variable (id-3). The placement of lines within a page is made relative to the top margin. Omission of this optional clause results in a top margin of two blank lines.

Examples: TOP MARGIN 0
TOP 4
TOP TPROMPT

d) Device clause (optional)

DEVICE { PRINTER / SCREEN / "file-1" / id-5 }

Reports can be routed to a console screen, printer, or disk file. If PRINTER, SCREEN, or file-1 is specified, then that choice is hardwired into the generated program. Alternatively, a previously defined prompt variable (id-5) having an STR type can be specified (the variable's size is automatically set to be 10). This enables an end-user to choose one or more devices for each run of the generated program. The end user must respond to the prompt by entering some combination of the letters S, P, F. If printer output is desired, a P must be entered. A response of S causes output to be routed to the console screen. For disk file output, the end user's response must have an F. This causes the end user to be prompted for a fully qualified file name. If the device clause is not present, SCREEN is assumed.

Examples: DEVICE PRINTER
DEVICE "B:REPT.T"
DEVICE DEVPROM

e) Pause clause (optional) PAUSE AFTER PAGE

If this optional clause is present, output from the generated program will pause after each page. The page depth is controlled by the page depth clause. This pause occurs only for screen and printer output devices. Output resumes when the end user presses the Return (Enter) key.

Examples: PAUSE
PAUSE AFTER PAGE

f) Date/page suppression clause (optional) SUPPRESS DATE PAGE

The generated program will automatically output a date and page number on the top line of each report page, with the exception of the report header pages (if any). Either or both of these can be suppressed with the date/page suppression clause. If neither DATE nor PAGE is specified in this clause, both are suppressed. If the identification section contains a screen clause, then both date and page number are automatically suppressed.

Examples:
```
SUPPRESS PAGE
SUPPRESS DATE
SUPPRESS DATE PAGE
SUPPRESS
```

g) Form alignment clause (optional) ALIGNMENT PAGES { int-4 / id-6 }

This clause causes the generated program to print out a specified number of empty pages, for purposes of aligning preprinted forms (using the printer line feed control). The number of pages can be stated explicitly (int-4) by the report designer. Alternatively, the designer can indicate that the value of the id-6 prompt variable is to be used to control the number of alignment pages.

Examples:
```
ALIGNMENT PAGES 3
ALIGNMENT 1
ALIGNMENT ALPGS
```


FUNCTION DEFINITION SECTION

The function definition section is optional. If it is present in an RDL specification, it consists of one or more function definitions. Any of the functions defined in this section can later be used when defining computed variables in the next section. Each function is defined in terms of three clauses: function identification, parameter typing, and function computation. These clauses must appear in the order shown here.

a) Function identification clause

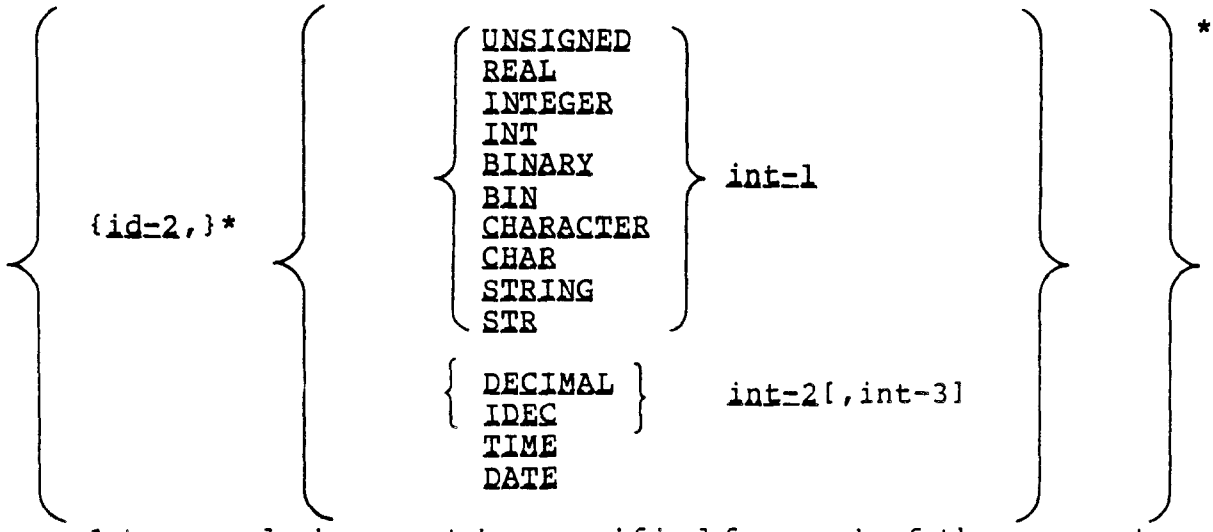
FUNCTION id-1 [({id-2,}*1)] type-1 size-1

In this clause a function is given a name (id-1), a type and a size. Because a function has a numeric value, type-1 must be INT, REAL, IDEC or UNSIGNED. With the exception of IDEC, the associated size-1 is an integer indicating how may bytes of storage are used to hold the function's value. For IDEC, size-1 is an integer indicating how may digits are in the function's value. Optionally, a second integer can be specified for the idec type to declare the number of digits to the right of the decimal point (if this is omitted, 0 is assumed).

If desired, a function can be declared to have one or more parameters. The names (e.g., id-2) of these formal parameters are separated by commas and enclosed within a pair of parentheses. A comma should not be used following the last parameter.

Examples:
```
FUNCTION COMPUTAX (P1) REAL 4
FUNCTION RTRNRATE (X1,X2,X3,X4,X5,I) IDEC 9,7
FUNCTION DATECVRT INT 4
```

b) Parameter typing clause



A type and size must be specified for each of the parameters declared in the function identification clause. The nine permissible types are unsigned, real, integer, binary, character, string, idec, time and date. No size is specified for time and date parameters. An idec parameter size can be specified in terms of two integers: int-2 is the total number of digits and int-3 is the number of digits to the right of the decimal point. If int-3 is omitted, its value is assumed to be 0. For all other types, int-1 indicates the number of bytes used to hold the parameter's value. Minimum and maximum sizes were discussed in the preceding chapter.

Examples:
```
P1 IDEC 4,1
I REAL 4  X1,X2,X3,X4 IDEC 7,2  X5 IDEC 10,2
PCODE CHAR 8
```

c) Function computation clause { procedure-1 }

This clause consists of a procedure enclosed in a matching pair of braces. The procedure is composed of local variable declarations followed by a sequence of statements. A local variable is used only within the procedure in which it is declared. Local variables for a procedure are not mandatory, but can be useful for holding temporary results. A local variable is declared in exactly the same manner as a parameter (see the parameter typing clause above).

After a procedure's local variables have been declared, a sequence of statements is specified which indicates how to compute the function's value. Five kinds of statements are allowed: assignment, if-then-else, test-case, break, and return. Any of these kinds of statements can be nested within an if-then-else or test-case statement. A return statement must be present. If the procedure execution does not exit through a return statement, then the function's value is undefined. Each statement within a procedure is terminated by a semi-colon (;).

i) Assignment statement

An assignment statement has the form:

variable = expression-1;

where the variable is a local variable which has been declared earlier in the procedure and the expression is an arithmetic expression. The permissible arithmetic operators in an expression are +, -, *, /, % (modulo), and unary-. The permissible operands are numbers, prompt variables, local variables, data items, parameters, and parenthesized arithmetic expressions. When an expression is evaluated, the following operator precedence is observed within each matching pair of parentheses (if any):

unary-, *, /, %, +, -

Expressions in the innermost parentheses pairs are evaluated first.

Examples:

```
TEM4=3+P1;
VAR=PROMPT3/32+MOSAL;
TAX=BASE+(YRSAL-BASE)*PRCNT;
```

ii) If-then-else statement

An if-then-else statement has the form:

IF (condition-1) THEN { statements-1 } ELSE { statements-2 }

If the specified condition is met, then the statements of statements-1 are executed. In cases where ELSE is used, the statements of statements-2 are executed when the specified condition is not satisfied.

A condition is enclosed within a matching pair of parentheses. It is composed of one or more relational expressions, each of which evaluates to true or false. Any one of the following relational operators can be used in a relational expression:

=, <>, >=, <=, >, <

Operands in a relational expression can be numeric or character constants, prompt variables, local variables, parameters, data items, or arithmetic expressions.

Wildcard string and symbol match characters (* and $, respectively) are allowed in character constants. Where a * appears within a quoted character constant, any string of zero, one or more symbols will match the *. Where a $ appears in a quoted sequence of characters, any single symbol will match it. In addition, a character class can be specified within a character constant by enclosing a group of characters within square brackets []. The character class position in the constant is matched by any of the characters in the group. For example, "L[e,o]t" matches Let or Lot; "[A-M]*" matches any character sequence beginning with A, B, ..., or M; "$[a-c,p]" matches any two characters having a, b, c, or p as the second character.

There is one additional operator beyond those noted above: the IN operator. The operand following IN consists of one or more constants enclosed in a matching pair of square brackets. The IN expression evaluates to true if the value of the left operand is equal to one of the constants of the right operand. Thus DEPT IN [5,7,13,2] is true if the value of DEPT is 5, 7, 12, or 2. NAME IN ["A*","*E","*S[O,E]N"] is true if the value of NAME begins with A or ends with E, SON or SEN.

Within condition-1, logical operators can be applied to relational expressions. When condition-1 is evaluated, the following precedence of logical operators is observed:

NOT, AND, OR, XOR

Parentheses can be used to control the order of evaluation within the condition. The contents of innermost parentheses are evaluated first.

Statements within an if-then-else statement can be any of the five kinds of statements permitted in a function computation clause. However, a break statement can be used only if the if-then-else statement is nested within a test-case statement.

Examples:

```
IF (PCODE=125A34) THEN {TEMP=P2+12} ELSE {TEMP=P2};
IF (P12>=PROMPT3 AND P7<13) THEN {TEMP7=TEMP7+1;T12=T12-PROMPT3};
IF (MOSAL<1000) THEN {IF(P4>1.2) THEN {TEMP=MOSAL/500}
   ELSE {TEMP=MOSAL/400}} ELSE {TEMP=MOSAL};
IF (PCODE=125$34) THEN {TEMP=P2+12} ELSE {TEMP=P2};
```

iii) Test-case statement

A test-case statement has the form:

TEST (expression-2) { {CASE constant-1: statements-3;}* [OTHERWISE: statements-4] }

As soon as a case constant which equals the test expression's value is detected, the statements for that case and statements of all subsequent cases (including the default case) are executed. However, if one of these is a break statement (described below), then the remaining statements within this test-case statement are skipped.

The test expression can be a prompt variable, local variable, data item, parameter, or arithmetic expression. It is enclosed in a matching pair of parentheses. Following expression-2, is a sequence of cases enclosed in a matching pair of braces. Each case is specified in terms of a constant, which is compared to the value of expression-2 when the test-case statement is evaluated. Following the case constant is a sequence of one or more statements. These statements (and those of subsequent cases) are executed for the first case constant which satisfies the test expression. The statements associated with a case can be any of the five kinds of statements permitted in a function computation clause. A default case, denoted by OTHERWISE, can optionally be specified. If it is included, then statements-4 are executed even though none of the case constants preceding it satisfy the test expression.

Example:

```
TEST (P1) {CASE 0: RAISE=YTDEARN*.05;BREAK;
           CASE 1: RAISE=YTDEARN*.1;BREAK;
           OTHERWISE: RAISE=0}
```

iv) Break statement

A break statement has the form:

```
BREAK;
```

When a break statement is encountered in statements-3 or statements-4, execution immediately breaks away to the statement following the end of the (innermost) test-case statement containing the break.

Example: BREAK;

v) Return statement

A return statement has the form:

RETURN (expression-3);

When a return statement is encountered in the course of executing a function's procedure, execution of the procedure halts and the value of expression-3 is returned as the function's value. The expression should have a numeric value: being a numeric constant, data item, parameter, local variable, prompt variable, or an arithmetic expression involving any of these.

Examples:
```
RETURN(5);
RETURN(PCODE);
RETURN((MOSAL/1000)*1.03);
```


COMPUTED VARIABLE SECTION

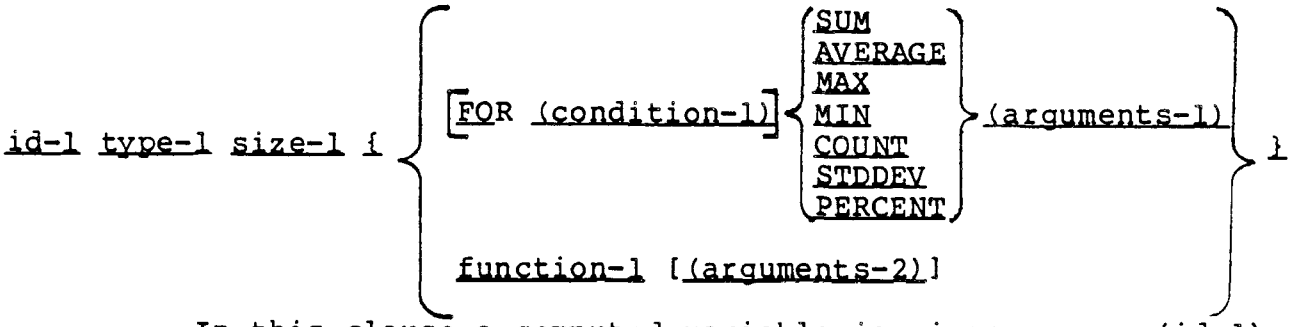
This optional section is used to define one or more computed variables. Each such variable is given a name, type, size. In addition, there is a declaration of how the variable's value is to be computed. This computation is stated in terms of a function defined in the previous section or a built-in statistical function.

a) Computed variable clause COMPUTED VARIABLES ARE

If a computed variable section exists, this must be its first clause. It can be followed by one or more computation clauses.

Examples: COMPUTED VARIABLES ARE
COMPUTED

b) Computation clause



In this clause a computed variable is given a name (id-1), a type and a size. It cannot have the same name as a data item. If a computed variable has a numeric value, then type-1 must be INT, REAL, IDEC or UNSIGNED. If a computed variable has a non-numeric value, then type-1 must be CHAR. With the exception of IDEC, the associated size-1 is an integer indicating how may bytes of storage are used to hold the computed variable's value. For IDEC, size-1 is an integer indicating how may digits are in the computed variable's value. Optionally, a second integer can be specified for the idec type to declare the number of digits to the right of the decimal point (if this is omitted, 0 is assumed).

The nature of the computation used to determine the computed variable's value is indicated by a function and its arguments (if any). The function is either a built-in statistical function (COUNT, SUM, MIN, MAX, AVERAGE, STDDEV, PERCENT) or a function (function-1) defined in the previous section of the RDL specification. If function-1 was defined with parameters, then one argument must be specified for each parameter. Prompt variables, previously declared computed variables, data items, and numeric constants can be used in the argument list (arguments-1, arguments-2). The arguments in argument-1 or arguments-2 are separated by commas and surrounded by a matching pair of parentheses.

The optional FOR clause can be used to specify conditions on a computation involving a built-in statistical function. A condition is enclosed within a matching pair of parentheses. It is composed of one or more relational expressions, each of which evaluates to true or false. Any one of the following relational operators can be used in a relational expression:

=, <>, >=, <=, >, <

Operands in a relational expression can be numeric or character constants, prompt variables, local variables, parameters, data items, or arithmetic expressions that do not involve functions defined in the RDL functionn definition section.

Wildcard string and symbol match characters (* and $, respectively) are allowed in character constants. Where a * appears within a quoted character constant, any string of zero, one or more symbols will match the *. Where a $ appears in a quoted sequence of characters, any single symbol will match it. In addition, a character class can be specified within a character constant by enclosing a group of characters within square brackets []. The character class position in the constant is matched by any of the characters in the group. For example, "L[e,o]t" matches Let or Lot; "[A-M]*" matches any character sequence beginning with A, B, ..., or M; "$[a-c,p]" matches any two characters having a, b, c, or p as the second character.

There is one additional operator beyond those noted above: the IN operator. The operand following IN consists of one or more constants enclosed in a matching pair of square brackets. The IN expression evaluates to true if the value of the left operand is equal to one of the constants of the right operand. Thus DEPT IN [5,7,13,2] is true if the value of DEPT is 5, 7, 12, or 2. NAME IN ["A*","*E","*S[O,E]N"] is true if the value of NAME begins with A or ends with E, SON or SEN.

Within condition-1, logical operators can be applied to relational expressions. When condition-1 is evaluated, the following precedence of logical operators is observed:

NOT, AND, OR, XOR

Parentheses can be used to control the order of evaluation within the condition. The contents of innermost parentheses are evaluated first. The condition-1 and arguments-1 specifications together can contain references to as many as 10 variables.

Examples:

```
EMPCNT INT 2 {FOR (YTDEARN >13000.0) COUNT (LASTNAME)}
EMPCNT INT 2 {FOR (DEPNO IN [13,2,9]) COUNT (LASTNAME)}
TAX IDEC 8,2 {COMPUTAX (YRLSAL)}
IRR IDEC 5,3 {RTRNRATE (200,2200,2400,2600,28000,20000)}
JDATE INT 4 {DATECVRT()}
```

REPORT DETAIL SECTION

This section is used to specify the detailed contents of the report. It consists of five kinds of clauses which must appear in the order shown below. These clauses are used to indicate

- where to position output lines on a report page
- the conditions under which data is to be retrieved (and thus displayed) from the data base
- the relationships that are to be used in retrieving data from the data base
- the sorting that is to be performed for retrieved data

This section cannot be omitted from an RDL specification.

a) Report clause REPORT

The report detail section must begin with the word REPORT. This is followed by one or more place clauses.

Example: REPORT

b) Place clause

PLACE {[UNIQUE] [UNDERLINED] expression-1 [FORMAT "fmt"] [@ int-1]}*
[SKIP {int-2 [LINES BEFORE [int-3 LINES AFTER]] / int-3 LINES AFTER}]

Many place clauses can be specified. Each place clause defines a line of output. One or more expressions can be assigned to an output line. Expression-1 is a numeric constant, a string of characters enclosed in double quotes, a prompt variable, a computed variable, a data item, or an arithmetic expression. When a data item is involved in an expression, it can be prefaced by its record type name and a period (.). Multiple data items of the same name may exist in the record types involved in relationships specified in the relationship clause (described below). In this case, the record type name must be used to preface these data items.

The value of expression-1 is formatted according to the indicated fmt and placed int-1 spaces from the report page's left margin. The int-1 that is specified must be in the range from 0 through 252. Optionally, int-2 lines can be skipped before the output line and/or int-3 lines can be skipped after the output line. If neither BEFORE nor AFTER is specified, BEFORE is assumed. Thus, an int-2 of 1 results in single spacing, an int-2 of 2 produces double spacing, and so forth. If this option is omitted, single spacing is assumed.

If the FORMAT option is omitted, a default format will be used. All default formats left justify expression values. The default format that is used for an expression depends on the type and size of its values. It also depends on how many positions are available for displaying its value (before the next expression's starting point on the line). In the following chart this number of positions is indicated by x.

Type	Size	Default Format
UNSIGNED	1 or 2	"%-xu"
UNSIGNED	3 or 4	"%-xlu"
REAL	any	"%-x.0f"
INTEGER	1 or 2	"%-xd"
INTEGER	3 or 4	"%-xld"
BINARY	any	"%-xo"
CHARACTER	y characters	"%-x.ys"
STRING	any	"%-x.xs"
IDEC	z,y	"%-x.yf"
TIME	---	"%-x.9s"
DATE	---	"%-x.10s"

The proper integer values for x and y are automatically determined, depending on value length and/or defined sizes of the expression's variables. Examples are shown in the generated programs that appear in Chapter VI.

There are two further options. Each expression in the place clause can be qualified by UNIQUE and/or UNDERLINED. The UNIQUE qualifier causes successive duplicate values of the expression to be suppressed in the output report. For instance, if five successive report details have the same expression value, then that value will be displayed only in the first of these details. The UNDERLINED qualifier causes values of an expression to be underlined in the output report.

Examples:
```
PLACE "NAME" @ 2  "ADDRESS" @ 20
PLACE JDATE @ 30 SKIP 2
PLACE FNAME @ 12
      UNIQUE LNAME @ 23
      "AGE" @ 40
      AGE @ 44 SKIP 4 AFTER
PLACE UNDERLINED MOSAL @ 18 YRLSAL @ 28 TAX @ 38
```

Special Case: Placing a Repeating Data Item

A repeating data item can have multiple values. Using the above form of the place clause, will cause these values to be placed one beneath the other in a column indicated by int-1. This column of values begins after int-2 lines are skipped and is followed by skipping int-3 lines. If UNIQUE is specified, only the first value of a succession of two or more duplicate values is output.

Greater control over the placement of a repeating data item's values is provided by an alternative form of the place clause. For a repeating data item,

```
@ int-1
```

in the place clause syntax can be replaced by

```
[int-4 ACROSS [FROM (int-5,int-6)]  @ int-7 [{,int-8}*]
 DOWN         [FROM (int-9,int-10)] @ int-11                ]
```

Here, int-4 is a positive integer indicating how many of the repeating data item's values are to be placed on each line. For example, if there are 14 values and 5 ACROSS is specified, then the values are placed on three consecutive lines (with only four values on the third line). The first value on each line begins in the column indicated by int-7, the second in the column indicated by int-9, and so forth. Columns must be in the range from 0 through 252. If fewer than int-4 column starting positions are specified, the remaining columns are automatically positioned so that an inter-column spacing of 2 results. If more than int-4 starting positions are specified, then an error message results.

Regardless of whether DOWN or ACROSS is specified, the report designer can optionally request that only some of the repeating values be output. This is accomplished by stating which value to begin with (as indicated by int-5 or int-9) and which value to end with (as indicated by int-6 or int-10). For instance, FROM (9,14) means that the ninth through fourteenth values of the repeating data item will be output. If int-6 (for ACROSS) or int-10 (for DOWN) exceeds the maximum replications of a repeating data item, then the maximum is used.

Examples:

```
PLACE PASTJOBS 3 ACROSS @ 5,25,43
PLACE LINE 2 ACROSS FROM (1,4) @ 2,52
PLACE LINE DOWN FROM (3,5) @ 12
```


c) Conditional clause (optional)

FOR (condition-1)

The conditional clause is optional. It states the conditions that must be met by the data items included in place clauses and mapped to frames of windows (and screens) included in display clauses. A report detail (as specified by the place and for display clauses) is generated only for data values that satisfy condition-1.

A condition is enclosed within a matching pair of parentheses. It is composed of one or more relational expressions, each of which evaluates to true or false. Any one of the following relational operators can be used in a relational expression:

=, <>, >=, <=, >, <

Operands in a relational expression can be numeric or character constants, prompt variables, local variables, parameters, data items, or arithmetic expressions that do not involve functions defined in the RDL function definition section.

Wildcard string and symbol match characters (* and $, respectively) are allowed in character constants. Where a * appears within a quoted character constant, any string of zero, one or more symbols will match the *. Where a $ appears in a quoted sequence of characters, any single symbol will match it. In addition, a character class can be specified within a character constant by enclosing a group of characters within square brackets []. The character class position in the constant is matched by any of the characters in the group. For example, "L[e,o]t" matches Let or Lot; "[A-M]*" matches any character sequence beginning with A, B, ..., or M; "$[a-c,p]" matches any two characters having a, b, c, or p as the second character.

There is one additional operator beyond those noted above: the IN operator. The operand following IN consists of one or more constants enclosed in a matching pair of square brackets. The IN expression evaluates to true if the value of the left operand is equal to one of the constants of the right operand. Thus DEPT IN [5,7,13,2] is true if the value of DEPT is 5, 7, 12, or 2. NAME IN ["A*","*E","*S[O,E]N"] is true if the value of NAME begins with A or ends with E, SON or SEN.

Within condition-1, logical operators can be applied to relational expressions. When condition-1 is evaluated, the following precedence of logical operators is observed:

NOT, AND, OR, XOR

Parentheses can be used to control the order of evaluation within the condition. The contents of innermost parentheses are evaluated first.

Examples:

```
FOR (MOSAL > 1000)
FOR (AGE < 32 AND MOSAL < 2000)
FOR (DNUMBER IN [12,5,18,20] AND ADDRESS = "*CHICAGO*")
```

d) Relationship clause

THRU { [-] [>] set-1, } *

The relationship clause is required. It enables the report designer to concisely specify the data interrelationships that should exist among data values in the desired report. Two RDL specifications that differ only in their relationship clauses will typically yield different reports. To use the relationship clause, examine all data items included in the RDL specification. In the schema, place a check mark on every record type that contains at least one of these data items. In order to produce the report, the generated program will have to examine occurrences of each of the checked record types. Thus the relationship clause should specify a sequence of sets, indicating which relationships among checked record types are to be used by the generated program.

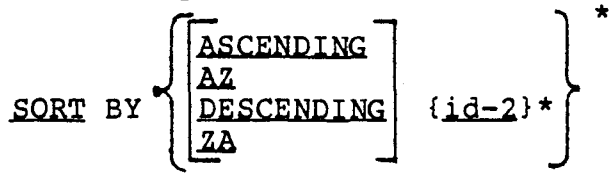
The RDL Analyzer can generate a program capable of processing either upstream (member to owner) or downstream (owner to member) through a set. If the sequence of relationships requires an upstream movement for some set, then that set's name is prefaced with the > symbol. If report details are specified in terms of a computed variable involving a built-in function, then the second argument (if any) of that function should not be a data item reached via an upstream relationship through a 1:1 or 1:N set.

The sequence of relationships in this clause must begin with a SYSTEM-owned set and must connect all checked record types. A relationship loop is not allowed; that is, the sequence of sets must not enter the same record type more than once. Placing a negative sign (-) in front of a set causes the generated program to examine the set's member (or owner, for upstream processing) occurrences in reverse order.

Examples:

```
THRU IDEP,HAS,POSSESS
THRU IDEP,HAS,>FILLEDBY,NEEDS
THRU IDEP,-HAS,->FILLEDBY,NEEDS
```


e) Sort clause (optional)



This optional clause is used to enforce a desired sorting of the report details. The sorting can be based on one or more data items and/or computed variables. Any mixture of ascending (AZ) and descending (ZA) directions can be specified. If no direction is stated, then ascending is assumed. Omission of the sort clause means that the order of report details is determined by the set orders of those sets included in the relationship clause.

Examples:

```
SORT BY LASTNAME
SORT ASCENDING ID DESCENDING MOSAL
SORT AZ ID ZA MOSAL
SORT ZA MOSAL AZ LASTNAME ID
```

GROUP SECTION

The group section is optional. If it is omitted, report details appear one after another with no interruption other than page breaks forced by the page depth. The group section can be used to specify more elaborate structuring of report details within the report. This includes

- automatic page breaks based on changes in data item values
- grouping of report details based on data item values
- specification of group headers and footers
- sorting of groups

Nested groupings are supported.

a) Page break clause (optional) PAGE BREAK AT id-1

If this optional clause is present, it must be the first clause of the group section. The generated program will force a page break to occur whenever the value of id-1 changes; id-1 is a data item. Page breaks are also caused whenever the maximum report page depth is reached.

b) Group clause (optional)

GROUP BY { REPORT / PAGE / id-2 }

One or more group clauses can be specified. A group clause specifies how report details are to be grouped. There are three kinds of grouping: by report, by page, and by id-2 (a data item). REPORT means that all report details are considered to constitute a group. PAGE means that all report details which fit on a report page constitute a group. If id-2 is specified, then grouping is based on the value of id-2. All report details with the same value for id-2 are in the same group.

Each group clause can optionally be followed by a header clause and a footer clause (in that order). The header clause is used to specify header information that will precede each group of report details. Similarly, a footer clause specifies footer information that will appear after each group of report details. For grouping by id-2, a sort clause can be specified after the footer to indicate how the groups themselves are to be sorted.

If more than one group clause appears in an RDL specification, then multi-level grouping is used. Each group clause (other than the first) specifies the grouping of report details within groups defined by the preceding group clause. For instance, the first group clause may specify grouping by company. If the second specifies grouping by division, then report details within each company group are grouped by division. If a third level involes grouping by department, then report details within each division are grouped by department. Page grouping should not be used more than once in an RDL specification. If report grouping is specified, it must be the first group clause. If page grouping is specified, its group clause must precede all group clauses other than report grouping.

Examples:

```
GROUP BY REPORT
GROUP PAGE
GROUP COMPANY
GROUP BY DIVISION
```

c) Header clause (optional)

HEADER IS

If a header clause is specified for a grouping, then it must immediately follow the .group clause and it must be immediately followed by one or more place clauses. The place information forms the header for each group of report details.

Place clause:

PLACE {expression-1 [FORMAT "fmt"] @ int-1}*

[SKIP { int-2 [LINES BEFORE [int-3 LINES AFTER]] / int-3 LINES AFTER }]

Many place clauses can be specified. Each place clause defined a line of output. One or more expressions can be assigned to an output line. Expression-1 is anumeric constant, a string of characters enclosed in double quotes, a prompt variable, a computed variable, a data item, or an arithmetic expression. The value of expression-1 is formatted according to fmt and is placed int-1 spaces from the report page's left margin. The value of int-1 must be in the range from 0 through 252.

If the FORMAT option is omitted, a default format will be used. All default formats left justify expression values. The default format that is used for an expression depends on the type and size of its values. It also depends on how many positions are available for displaying its value (before the next expression's starting point on the line). In the following chart this number of positions is indicated by x.

Type	Size	Default Format
UNSIGNED	1 or 2	"%-xu"
UNSIGNED	3 or 4	"%-xlu"
REAL	any	"%-x.0f"
INTEGER	1 or 2	"%-xd"
INTEGER	3 or 4	"%-xld"
BINARY	any	"%-xo"
CHARACTER	y characters	"%-x.ys"
STRING	any	"%-x.xs"
IDEC	z,y	"%-x.yf"
TIME	---	"%-x.9s"
DATE	---	"%-x.10s"

The proper integer values for x and y are automatically determined, depending on value length and/or defined sizes of the expression's variables. Examples are shown in the generated programs that appear in Chapter VI.

Optionally, int-2 lines can be skipped before the output line and/or int-3 lines can be skipped after the output line. If neither BEFORE nor AFTER is specified, BEFORE is assumed. Thus, an int-2 of 1 results in single spacing, an int-2 of 2 produces double spacing, and so forth. If this option is omitted, single spacing is assumed.

Examples:
```
PLACE "NAME" @ 2  "ADDRESS" @ 20
PLACE JDATE @ 30 SKIP 2
PLACE FNAME @ 12
      LNAME @ 23
      "AGE" @ 40
      AGE @ 44 SKIP 4 AFTER
PLACE MOSAL @ 18 YRLSAL @ 28 TAX @ 38
```


Special Case: Placing a Repeating Data Item

A repeating data item can have multiple values. Using the above form of the place clause, will cause these values to be placed one beneath the other in a column indicated by int-1. This column of values begins after int-2 lines are skipped and is followed by skipping int-3 lines. If UNIQUE is specified, only the first value of a succession of two or more duplicate values is output.

Greater control over the placement of a repeating data item's values is provided by an alternative form of the place clause. For a repeating data item,

@ int-1

in the place clause syntax can be replaced by

```
[int-4 ACROSS [FROM (int-5,int-6)]  @ int-7 [{,int-8}*]]
[DOWN         [FROM (int-9,int-10)] @ int-11             ]
```

Here, int-4 is a positive integer indicating how many of the repeating data item's values are to be placed on each line. For example, if there are 14 values and 5 ACROSS is specified, then the values are placed on three consecutive lines (with only four values on the third line). The first value on each line begins in the column indicated by int-7, the second in the column indicated by int-9, and so forth. Columns must be in the range from 0 through 252. If fewer than int-4 column starting positions are specified, the remaining columns are automatically positioned so that an inter-column spacing of 2 results. If more than int-4 starting positions are specified, then an error message results.

Regardless of whether DOWN or ACROSS is specified, the report designer can optionally request that only some of the repeating values be output. This is accomplished by stating which value to begin with (as indicated by int-5 or int-9) and which value to end with (as indicated by int-6 or int-10). For instance, FROM (9,14) means that the ninth through fourteenth values of the repeating data item will be output. If int-6 (for ACROSS) or int-10 (for DOWN) exceeds the maximum replications of a repeating data item, then the maximum is used.

Examples:

```
PLACE PASTJOBS 3 ACROSS @ 5,25,43
PLACE LINE 2 ACROSS FROM (1,4) @ 2,52
PLACE LINE DOWN FROM (3,5) @ 12
```


d) Footer clause (optional)

FOOTER IS

If a footer clause is specified for a grouping, then it must immediately follow the place clauses for the header (if a header clause exists for the group). A footer clause is immediately followed by one or more place clauses. The place information forms the footer for each group of report details.

Place clause:

PLACE {expression-2 [FORMAT "fmt"] @ int-12}*

[SKIP { int-13 [LINES BEFORE [int-14 LINES AFTER]] / int-14 LINES AFTER }]

Many place clauses can be specified. Each place clause defined a line of output. One or more expressions can be assigned to an output line. Expression-2 is a numeric constant, a string of characters enclosed in double quotes, a prompt variable, a computed variable, a data item, or an arithmetic expression. The value of expression-1 is formatted according to the indicated fmt and is is placed int-12 spaces from the report page's left margin.

If the FORMAT option is omitted, a default format will be used. All default formats left justify expression values. The default format that is used for an expression depends on the type and size of its values. It also depends on how many positions are available for displaying its value (before the next expression's starting point on the line). In the following chart this number of positions is indicated by x.

Type	Size	Default Format
UNSIGNED	1 or 2	"%-xu"
UNSIGNED	3 or 4	"%-xlu"
REAL	any	"%-x.0f"
INTEGER	1 or 2	"%-xd"
INTEGER	3 or 4	"%-xld"
BINARY	any	"%-xo"
CHARACTER	y characters	"%-x.ys"
STRING	any	"%-x.xs"
IDEC	z,y	"%-x.yf"
TIME	---	"%-x.9s"
DATE	---	"%-x.10s"

The proper integer values for x and y are automatically determined, depending on value length and/or defined sizes of the expression's variables. Examples are shown in the generated programs that appear in Chapter VI.

Optionally, int-13 lines can be skipped before the output line and/or int-14 lines can be skipped after the output line. If neither BEFORE nor AFTER is specified, BEFORE is assumed. Thus, an int-13 of 1 results in single spacing, an int-13 of 2 produces double spacing, and so forth. If this option is omitted, single spacing is assumed.

```
Examples:  PLACE "NAME" @ 2  "ADDRESS" @ 20
           PLACE JDATE @ 30 SKIP 2
           PLACE FNAME @ 12
                 LNAME @ 23
                 "AGE" @ 40
                 AGE @ 44 SKIP 4 LINES AFTER
           PLACE MOSAL @ 18 YRLSAL @ 28 TAX @ 38
```

Special Case: Placing a Repeating Data Item

A repeating data item can have multiple values. Using the above form of the place clause, will cause these values to be placed one beneath the other in a column indicated by int-14. This column of values begins after int-15 lines are skipped and is followed by skipping int-16 lines. If UNIQUE is specified, only the first value of a succession of two or more duplicate values is output.

Greater control over the placement of a repeating data item's values is provided by an alternative form of the place clause. For a repeating data item,

```
@ int-12
```

in the place clause syntax can be replaced by

```
[int-15 ACROSS [FROM (int-16,int-17)]  @ int-18 [{,int-19}*]
 DOWN [FROM (int-20,int-21)] @ int-22]
```

Here, int-15 is a positive integer indicating how many of the repeating data item's values are to be placed on each line. For example, if there are 14 values and 5 ACROSS is specified, then the values are placed on three consecutive lines (with only four values on the third line). The first value on each line begins in the column indicated by int-18, the second in the column indicated by int-19, and so forth. If fewer than int-15 column starting positions are specified, the remaining columns are automatically positioned so that an inter-column spacing of 2 results. If more than int-15 starting positions are specified, then an error message results.

Regardless of whether DOWN or ACROSS is specified, the report designer can optionally request that only some of the repeating values be output. This is accomplished by stating which value to begin with (as indicated by int-16 or int-20) and which value to end with (as indicated by int-16 or int-20). For instance, FROM (9,14) means that the ninth through fourteenth values of the repeating data item will be output. If int-16 (for ACROSS) or int-21 (for DOWN) exceeds the maximum replications of a repeating data item, then the maximum is used.

Examples:

```
PLACE PASTJOBS 3 ACROSS @ 5,25,43
PLACE LINE 2 ACROSS FROM (1,4) @ 2,52
PLACE LINE DOWN FROM (3,5) @ 12
```


e) Sort groups clause (optional)

SORT GROUPS BY { [ASCENDING / AZ / DESCENDING / ZA] {id-3}* }*

This clause can appear as the last clause related to a group clause, providing the grouping is not by report or by page. The use of id-2 in a group clause produces groups of report details. These groups or "chunks" of report details can be sorted based on their values for id-3, which is either a data item or computed variable. Multiple sort criteria can be specified in any mix of ascending and descending directions. If no direction is specified, ascending is assumed.

The sort groups clause does not affect the ordering of report details within a group; nor does it affect the ordering of lower level groups within each of groups being sorted.

If the grouping for which a sort clause is specified is at a lower level than another grouping, then the sorting of the lower level groups takes place within each of the higher level groups.

Examples:

```
SORT GROUPS BY DESCENDING DEPTSALE
SORT BY COMPTYPE,COMPNAME
SORT ZA DEPTSALE AZ DEPTNAME
```


This page intentionally left blank.

IV. USING THE MDBS.RDL SOFTWARE

A. Overview

The MDBS.RDL software is provided on several files. It consists of the RDL Analyzer, an RDL library, and the RDL sort-merge processor. The RDL Analyzer is used to convert RDL specifications into interactive C language programs. The RDL specifications are normally prepared with a word processor or text editor (the MDBS DDL Analyzer, for instance). The RDL library is used during the compilation of the generated C programs. The RDL sort-merge processor performs all sorting (if any) needed by the generated programs.

B. The RDL Analyzer

The RDL Analyzer consists of a file named RDL. In some environments, additional files named RDL1, ..., RDL6 may also be provided. RDL is an executable file having the appropriate extension for executable files under the host operating system. The other files (if provided) are overlays needed by RDL during the Analyzer's execution. These files have OVL extensions. RDL, all of its overlays, and the main data base area should normally be on the default drive/directory when executing the RDL Analyzer.

The RDL Analyzer is executed by entering the following operating system command line:

RDL filename

where **filename** is the name of a file containing an RDL source specification. If no extension is specified for this file, an RDL extension is assumed. If the **filename** is not qualified by a drive/directory indicator, the default drive/directory is assumed.

If the User clause is omitted from the RDL specification's Identification section, then the RDL Analyzer will prompt for a user name and associated password. If a valid user/password is entered the Analyzer will continue its processing; otherwise, processing halts. RDL Analyzer processing can also be halted at any time by pressing the hard-interrupt key (e.g., Escape, Break, Control-C under many operating systems).

There are three optional arguments that can appear in any order following **filename** on the RDL command line. These are

-Uuser
-Ppass
-Ddb

where **user** is a bona fide user name, **pass** is a valid password, and **db** is the (fully qualified) name of the file containing the data base's main area. If -Uuser appears, it overrides the user (if any) declared in the RDL specification and suppresses the RDL Analyzer's prompt for a user name. If -Ppass appears, it overrides the password (if any) declared in the RDL specification and suppresses the RDL Analyzer's prompt for a password. If -Ddb appears, it overrides the file specified in the Data base clause of the Identification section.

A fourth optional argument that can appear anywhere following **filename** on the RDL command line is

-Bnnnnn

where **nnnnn** is an integer number indicating how many bytes are to be allocated as a buffer for use by the MDBS Data Management System during RDL analysis. If this argument is omitted, approximately half of the available memory is allocated. In cases where the RDL specification is large, the -B option can be used to allocate a smaller buffer thereby leaving more room for RDL analysis.

In some environments, a -I option can be used on the command line that loads the MDBS Data Management System. See Chapter VI of the MDBS System Specific Manual for your environment. If the -I option can be used for DMS/SMS loading in your environment, it can also appear as an option on the command line used to invoke the RDL Analyzer. Before a program generated by the RDL Analyzer can be executed, the DMS/SMS must be loaded (see Section D in this chapter). In cases where the -I option is not used for DMS/SMS loading, the -I option should not appear on the command line that invokes the RDL Analyzer. If the -I option is used for DMS/SMS loading it must also be used when invoking the RDL Analyzer. The interrupt number specified with -I must be the same in both places.

If the RDL Analyzer detects an error in the RDL specification, then an error message is displayed and processing halts. Chapter V describes the possible error conditions. In cases where the Analyzer detects no errors, a message similar to the following is displayed upon completion of the program generation:

```
RDL analysis complete:  "rpg1.c" and "rpg2.c" generated
Enter "crpg" to compile generated program(s)
Enter "rpg" to execute compiled program(s)
```

If the RDL specification's Identification section contains a Program clause, then the file name indicated in that clause replaces the letters rpg in the names of the files generated by the RDL Analyzer. For relatively simple reports, the rpg2.c program file is not needed and therefore is not generated. The three or four generated files are output to the default drive or working directory, unless a drive/directory qualifier is declared with the file name specified in a Program clause. Such a drive/directory qualifier will determine the drive to which the generated files are written.

C. Compiling the Generated Program(s)

The RDL Analyzer produces a command file which, when invoked, will compile and link the generated program(s). If no Program clause appeared in the Identification section, then the RDL Analyzer reports that this command file's name is crpg. Whatever its name is, it is invoked by entering the compilation command that the RDL Analyzer showed in double quotes. Type

```
crpg
```

for instance. Before invoking this command file, be sure that the generated program(s), RDLIB, STDIO.H and (if necessary) your C compiler/linker software are on the default drive/directory. One or two additional files (SETJMP.H and/or RPGLOB.H) may have been provided with the RDL software. If they are provided, be sure that they too are on the default drive/directory prior to compilation.

RDLIB is a library provided as part of the MDBS.RDL software. The C compiler that is used must be the C compiler with which MDBS interfaces under the host operating system. The SETJMP and RPGLOB (if provided) are C "include" files. The STDIO file is provided with the C compiler.

Each compiled program will have the same name as its source code file, except that the c extension is replaced by the normal extension for an executable program under the host operating system. The compiled program(s) resides on the default drive or working directory.

D. Executing the Compiled Program(s)

Prior to executing any application program (regardless of whether it was generated by the RDL Analyzer or written using traditional methods) MDBS must have been installed and, depending on the environment, its data management system must be resident in main memory. Instructions for accomplishing this appear in the MDBS System Specific Manual for C under your operating system. If MDBS is not resident, the message "Please load MDBS Data Management System" will be issued. This message can also be caused by inconsistent use of the -I option (i.e., the interrupt number used for loading DMS is not the same as that specified when invoking the RDL Analyzer).

Because the generated programs are in C, the C compatible MDBS interface must be used. For instance, the generated programs will not run with the MDBS data management system that interfaces to COBOL. Also prior to executing an application program, the data base itself must of course be on-line.

The RDL Analyzer produces a command file which, when invoked, will execute the compiled program(s). If no Program clause appeared in the Identification section, then the RDL Analyzer reports that this command file's name is rpg. Whatever its name is, it is invoked by entering the execution command that the RDL Analyzer showed in double quotes. Type

```
rpg
```

for instance. Before invoking this command file, be sure that the compiled program(s) and RDLSM are available for execution (e.g., on the default drive/directory). RDLSM is an executable program provided as part of the MDBS.RDL software. In cases where multiple programs are generated by the RDL Analyzer, RDLSM is used during execution to handle all file sorting/merging activities on data extracted from the data base.

Because the generated program invokes DML commands, it is possible that certain command status numbers can result from abnormal conditions during the course of execution. In such a case, the command status number is displayed and execution terminates. The meanings of command status numbers are documented in the **MDBS DMS Manual.**

If a generated program is interrupted in the midst of execution, the data base is not closed. This does not harm the integrity of the data base. However, when you attempt to re-execute the report program, the message "Database already open" will appear and the data base will be closed. The report program can then be executed as usual.

V. RDL ANALYZER DIAGNOSTICS

The RDL Analyzer is able to detect a wide range of errors in RDL specifications. When an error is detected, the Analyzer ceases processing at that point and displays an error message. These diagnostics and their possible causes are described in this chapter. Because the RDL Analyzer contains portions of MDBS.DMS, it is also capable of producing a few command status messages. The command status numbers are documented in the MDBS DMS Manual.

both operands of relational expression are constants or identical

A relational expression cannot have constant operands on both sides of the relational operator (e.g., <, >, =, etc.), nor can it have the same operand on both sides.

cannot mix place across with place down

A Place clause cannot have both DOWN and ACROSS terms following the expression.

cannot sort by

The indicated term cannot be used in a sort clause. Data items and computed variables are the only valid sort keys.

computed variable cannot depend on itself

A computed variable cannot be defined in terms of itself.

computed variable definition must start with '{'

In a Computation clause, the declaration of how the computed variable's value is to be determined must begin with {.

computed variable definition must terminate with '}'

In a Computation clause, the declaration of how the computed variable's value is to be determined must end with }.

computed variables are not allowed in FOR clause

Computed variables are not permitted in a conditional expression (i.e., a logical expression beginning with the FOR keyword).

conditional clause must contain a relational operator

A conditional expression must contain at least one relational operator (e.g., >, <, =, etc.).

conditional expression too long

A conditional expression (i.e., beginning with FOR) cannot exceed 512 characters.

current line exceeds 255 characters

A line in the RDL specification cannot exceed 255 characters. For very long literals, use multiple Place clauses.

data type not recognized

An unrecognizable type has been specified for a function or variable.

database access not allowed

An invalid user/password combination was specified. A -B option was specified with a buffer size that is physically too small or too large.

database and program names must be in quotes

The name of a file holding the main data base area or a file specified in a Program clause must be enclosed in double quotes.

default value's type incompatible

The type of a specified default value is incompatible with the type of the prompt variable.

expecting PROMPT/FUNCTION/COMPUTED/ or REPORT section

One of these sections must appear following the Data base section. When used, they must appear in the correct order: Prompt, Function, Computed variable, Report details.

FOR clause too complex

The RDL Analyzer encountered a conditional expression (i.e., a logical expression beginning with the FOR keyword) that is too complex to successfully process.

format specification is located after item being formatted

In a PLACE clause, a format specification cannot appear prior to the expression whose values are to be formatted.

format specification must be a string

A format specification must follow the FORMAT keyword. It must be enclosed in a matching pair of double quotes.

function definition must start with '{'

A function computation clause must begin with {.

function definition must terminate with '}'

A function computation clause must be terminated by }.

function not defined or not built-in

A function referenced in the Computation clause is undefined or a non-built-in function is used following a conditional (i.e., FOR) expression.

group HEADER or FOOTER expected

Neither a HEADER nor FOOTER was specified for a group's Place clause.

improper conditional clause

The conditional expression (i.e., a logical expression beginning with the FOR keyword) does not have a valid syntax.

IN operator must occur within conditional clause

The IN operator cannot be used outside of a conditional (i.e., FOR) expression.

in report detail, SORT clause follows THRU clause

A Sort clause cannot follow the Thru clause.

incorrect SORT syntax

Some word other than AZ, ASCENDING, ZA, or DESCENDING was used to specify sort ordering.

insufficient room in memory

The RDL Analyzer does not have enough room to execute for an RDL specification of this size. Use a smaller specification (e.g., eliminate non-essential Place clauses) or a smaller -B option on the command line that invokes the RDL Analyzer.

invalid arithmetic expression

An invalid arithmetic expression was encountered (the parentheses are not balanced, there is an operator-operand mismatch, a keyword is improperly used, etc.).

invalid IN usage

The operand to the right of the IN operator must begin with [and end with].

invalid name

An invalid name was specified for a prompt variable, function, function parameter, or computed variable. Names must be alphanumeric and cannot begin with a numeral.

invalid record type

The indicated record type is not valid for this data base.

invalid syntax in path clause

A delimiter other than '>', ',', '-' or a blank has been specified in the Thru clause.

item is not in record type

The indicated data item does not exist in the record type that was used to qualify its name.

item is not unique within specified path

The report references a data item that exists in more than one record type involved in the relationships specified in the Thru clause. Qualify the data item by prefacing it with its record type name.

item is not within the specified path

The RDL specification refers to a data item that is not in any record types lying along the path specified in the Thru clause.

item used in computed variable definition is not recognized

A non-existent data item, prompt variable or computed variable was referenced when defining a computed variable.

left parenthesis expected before argument list

A function's arguments must be enclosed in parentheses.

mismatched curly brackets

Unbalanced curly brackets ({,}) have been used.

mismatched parenthesis

Unbalanced parentheses have been used.

missing closing parenthesis or incorrect number of arguments

A function's arguments must be enclosed in parentheses and the correct number of arguments must be specified.

must group by database item

Only data items can be used as the basis for grouping.

must sort by database item or computed variable

Only data items and/or computed variables can be used as the basis for sorting.

no items specified

The RDL specification does not involve any data items from the indicated data base.

no sets specified in THRU clause

The Thru clause must include one or more set names.

number expected

A number must be used with specifying coordinates, how many lines to skip, or the size of a variable.

output device not recognized

The output device must be either SCREEN, PRINTER, or a quoted file name.

page size/left/top margin should be numeric

Page size, left margin, and right margin must be numeric.

percent computed variable cannot be placed in group section

A computed variable defined in terms of the built-in percent function cannot be referenced in the Group section.

rdl source file not present

An RDL source file was not specified on the command line used to invoke the RDL Analyzer or the specified file is not present on the appropriate drive.

REPORT section is required

The Report details section is required in every RDL specification. It begins with the REPORT keyword.

set is invalid

Every set specified in the Thru clause must exist in the data base schema. The first set specified must be owned by system.

set is not connected within specified path

A completely connected line cannot be drawn through all sets specified within the Thru clause.

set is superfluous to report writing

The indicated set is not needed to produce this report. Eliminate it from the RDL specification.

string must terminate with " on current line

A string constant cannot be broken across multiple lines and it must terminate with ".

THRU clause is required

The Thru clause must appear at the end of the Report details section.

token not recognized

The most recently specified token (e.g., keyword, variable, special symbol, etc.) is unrecognizable. It is misspelled, out of correct syntactic order, etc.

too many column coordinates

When using ACROSS in a Place clause, too many coordinates were specified for the number of values to be placed. For instance, more than 5 coordinates were specified for 5 ACROSS.

too many computed variables used in current computed variable definition

No more than 10 computed variables can be referenced within a computed variable definition.

too many items specified

No more than 102 data items can appear in an RDL specification.

too many operands/operators

The number of operators/operands make the RDL specification too complex to be processed.

verb PLACE expected

Each Place clause must begin with the PLACE keyword.

*** and / are not unary operators**

These arithmetic operators must have 2 operands.

VI. RDL EXAMPLES

This chapter contains several sample RDL specifications. Each is used to describe a desired report to be obtained from a data base having the schema shown in Figure VI-1. The formal DDL specification for this schema appears in Figure VI-2. The RDL Analyzer was used to process each of the RDL specifications appearing in this Chapter. The generated program(s) for each was compiled and executed against a loaded data base. In this chapter, the resultant output report for each example RDL specification is reproduced alongside that specification. For the first two examples the generated source program is also shown. All of these examples were run under the PCDOS operating system and therefore used the MDBS interface to Lattice C (also known as MicroSoft C). In other environments the RDL specifications and resultant reports are identical to those shown here. However, the generated source code may differ slightly from the two examples shown here due to differences in C compilers.

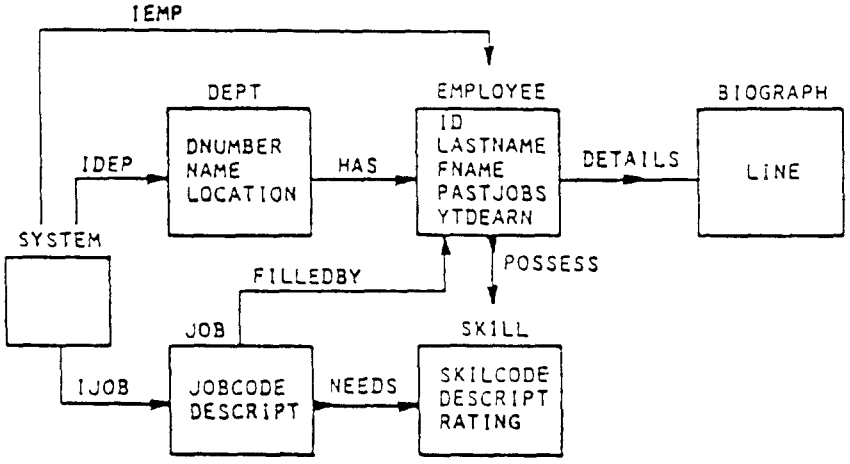


Figure VI-1. Sample Schema


```
/*****
 *
 *  sample data base description (advanced level)
 *
 *****/

db jobs         file "jobs.db"
                size 50 pages, page size 512


/******************  DEFAULT DEFINITION  ************************/

default for char is 8
default for bin is 250
default for str is 50
default for unsigned is 2


/******************  USER DEFINITION  ***************************/

user "user" with "pass" read access (a-p) write access (a-p)
user "gary" with "rush" read access (a-p) write access (a-p)


/*********************  AREA DEFINITION  ************************/

area name is job1
     file name is "job1.db"
     size is 20 pages
     page size is 512  /* Note:  this page size would not be
                               allowed in Version 3a because
                               the main area's page size is 1024 */
     read access is (a-p)  write access is (a-p)


area job2       file "job2.db"
                size 20 pages
                read (a-p)  write (a-p)


/******************  RECORD TYPE DEFINITION  ********************/

record DEPT
     in job5  key is NAME nodup
     title is "this is a title"
     read access p  write access (a,b)
          item DNUMBER int 1
                       range is 1 to 42
                       syn is DNO
          item is NAME    char 12   encrypted
                       write access b
          item LOCATION  str 35     syn LOC
                         range "A" to "ZZZZZZ"


record employee in job1, key is ID nodup
     read access (a,d)  write access (a,p)
          item ID char 11  encrypted range is "0" to "999999999"
          item LASTNAME char 20
          item FNAME     str 12       range "Aa" "Zz"
          item PASTJOB   str 25  occurs 3 times
          item YTDEARN idec 7,2


record BIOGRAPH in job2
     read access b  write access p
          item LINE  str  occurs 5 times
```

Figure VI-2. Sample DDL Specification

```
record SKILL in (job5,job2) calc key is SKILCODE nodup
     read access (b,d)  write access f
          item SKILCODE unsigned syn SC  range 0 to 3000
          item DESCRIPT str 55
          item RATING real 2              range 0.0 to 4.0


 record job   /* in area of member of NEEDS */
     read access b  write access (a,p)
 item jobcode unsigned 1
range 1 to
200
          item DESCRIPT  str



/************************  SET DEFINITION  ************************/

set IEMP, type 1:n, retention fixed
     read access d  write access p
          owner SYSTEM
          member  EMPLOYEE,  order sorted by ascending  (lastname, Fname)
               insertion auto


set POSSESS, type n:m
     read access (a,d)  write access (f,p)
          owner EMPLOYEE, order sorted by Az ID
          member SKILL


set DETAILS, type 1:1, fixed
     read access (b,d)  write access p
          owner EMPLOYEE
          member BIOGRAPH auto


set FILLEDBY, type 1:n
     read access (a-d)  write access p
          owner job
          member EMPLOYEE fifo auto


set NEEDS n:m
     read access b  write access (f,p)
          owner JOB sorted za JOBCODE
          member SKILL sorted az (RATING,SKILCODE)


set HAS 1:n
     read access (a,b)  write access a
          owner DEPT
          member EMPLOYEE lifo auto

 set IDEP 1:n   read access b
      owner SYSTEM   member DEPT auto


 set Ijob 1:n   read access b
      owner SYSTEM   member job auto fifo

 END
```

Figure VI-2. Sample DDL Specification (Continued)

This page intentionally left blank.

Example 1

RDL source specification (page 58)

Generated C source program (pages 59 and 60)

One-page output report (page 61)

User responses to prompt variables:

NLINE - user pressed Enter key (⟵┘) resulting in the default of 30
OUTDEV - user entered SP

```
*************************************************************************************
*                                                                                   *
*                                SAMPLE RDL SOURCE 1                                *
*                                                                                   *
*************************************************************************************

/*******************************
 *
 *      database section
 *
 *******************************/

        database file name is "jobs.db"
        user "user" with "pass"

/*******************************
 *
 *      prompt variables
 *
 *******************************/

        prompt
                NLINE   int   2
                "No. of lines per page ?"  default  30

                OUTDEV  char  7
                "printer, screen, or disk file ?"

/*******************************
 *
 *      output control section
 *
 *******************************/

        page size is NLINE lines per page
        device is OUTDEV
        pause

/*******************************
 *
 *      report section
 *
 *******************************/

        report is
                place unique    FNAME           @ 1
                      underlined PASTJOB  2 across  @ 10, 22
                      underlined LINE     2 across from (2, 5) @ 34, 50
                      " "                       @ 74

        for (FNAME in [ "Bob", "J[a,e]nnie", "D*", "C*"])

        thru

                -iemp details

/*******************************
 *
 *      group section
 *
 *******************************/

        group by PAGE
            header
                place "EMPLOYEE FILE"   @ 33    skip  3 after
                place "=============================================================================="
                                        @  1
                place "First Name"      @  2
                      "Past Jobs"       @  13
                      "Vita"            @  36
                place "=============================================================================="
                                        @  1    skip  3 after
```

```
/* analyzer, MDBS Inc, version 3.08 */
#define ... a
#include          "rpglob.h"
struct  hdrarr  {
        short     0, 1, 2, 3, 4;
        };
int     outctrl[6] = { 0, 2, 5, 0, 0, 0};
char    RV_1[13];          /* FNAME     */
char    RV_2[5][51];       /* LINE      */
char    RV_3[3][26];       /* FASTJOB   */
short   NLINE     ;
char    OUTDEV  [11];
char    UDATE   [10];
short   UPAGE     ;
char    UTIME   [9];
int     ind 1;
int     ind 2;
short   rtab1, otab1;
short   rtab2, otab2;
char    grpbrk[2];
char    brkflag[2];
char    utrm1[13];
char lngtyp = '\0';
char ftab1[1], ftab2[1], ftab3[1];
main()
{
register          int       e0;

if (e0 = setjmp(&env)){
        errexit(e0);
        }
umssip(&env, 0);
if (dbinit(0))
        errexit(0);
strcpy(opbll.urwst,   "r");
strcpy(opbll.ufile,   "jobs.db");
strcpy(opbll.usernm,  "user");
strcpy(opbll.upassw,  "pass");
prmt();
_prompt("No. of lines per page ?[30]"
,&NLINE    , 2, sizeof(NLINE    ), 0, 1);
_prompt("printer, screen, or disk file ?"
, OUTDEV   , 1, 11, 0, 1);
ckdevice(OUTDEV   , &rfname);
if(e0 = dbopn(&opbll))
        errexit(e0);
getime(UDATE, UTIME);
outctrl[0] = NLINE    ;
opendev(rfname);
rpstart = TRUE;
rpend = FALSE;
pradjust();
pagebrk = FALSE;
curline = outctrl[0] + 1;
otab1 = rtab1 = 0;
for (e0 =  flm("IEMP     "); e0 == 0; e0 = fpm("IEMP     ")){
        rtab1++;
        e0 = gfc("FNAME    ",  RV_1);
        e0 = gfc("FASTJOB ",  RV_3);
        if (!(((strc(1, 13, RV_1, "Bob") ||
strc(1, 13, RV_1, "J[a,e]nnie") ||
strc(1, 13, RV_1, "D*") ||
strc(1, 13, RV_1, "C*")))))    continue;
        otab2 = rtab2 = 0;
        e0 = som("DETAILS  , IEMP     ");
        for (e0 =  ffm("DETAILS "); e0 == 0; e0 = fnm("DETAILS ")){
                rtab2++;
                e0 = gfc("LINE     ",  RV_2);
                update();
                pdetail();
                rpstart = FALSE;
                }
        }
rpend = TRUE;
pgbreak();
clsdev();
dbcls();
}
pdetail()
{
pgcheck(1, outctrl[2] + 0);
curpos.c_olp = 1;
punderscore(10 - curpos.c_olp, 11);
curpos.c_olp = 10 + 12 - 1;
punderscore(22 - curpos.c_olp, 11);
curpos.c_olp = 22 + 12 - 1;
punderscore(34 - curpos.c_olp, 15);
curpos.c_olp = 34 + 16 - 1;
punderscore(50 - curpos.c_olp, 23);
curpos.c_olp = 50 + 24 - 1;
mprintf("\r");
pgcheck(0, outctrl[2] + 0);
if (strncmp(RV_1, utrm1, 12) || outctrl[4]) {
        mprintf("%-9.9s",RV_1);
        strncpx(utrm1, RV_1 , 12);
        outctrl[4] = 0;
        }
else  {
        pblank(9);
        }
mprintf("%-12.12s",RV_3[0]);
mprintf("%-12.12s",RV_3[1]);
mprintf("%-16.16s",RV_2[1]);
mprintf("%-16.16s",RV_2[2]);
mprintf("%-1.1s"," ");
pgcheck(1, outctrl[2] + 0);
curpos.c_olp = 1;
punderscore(10 - curpos.c_olp, 11);
curpos.c_olp = 10 + 12 - 1;
punderscore(22 - curpos.c_olp, 11);
curpos.c_olp = 22 + 12 - 1;
punderscore(34 - curpos.c_olp, 15);
curpos.c_olp = 34 + 16 - 1;
punderscore(50 - curpos.c_olp, 23);
curpos.c_olp = 50 + 24 - 1;
mprintf("\r");
pgcheck(0, outctrl[2] + 0);
if (strncmp(RV_1, utrm1, 12) || outctrl[4]) {
        mprintf("%-9.9s",RV_1);
        strncpx(utrm1, RV_1 , 12);
        outctrl[4] = 0;
        }
```

```
else  {
        pblank(9);
        }
mprintf("%-12.12s",RV_3[2]);
pblank(12);
mprintf("%-16.16s",RV_2[3]);
mprintf("%-16.16s",RV_2[4]);
mprintf("%-1.1s"," ");
}
update()
{
if (rpstart) {
        }
otab1 = rtab1;
otab2 = rtab2;
}
pgfooter()
{
        skipline(1);
        _pgpause();
}
pdp()
{
if (!rpend || r_pfoot) {
        UPAGE++;
        mprintf("%10.10s", UDATE);
        pblank(55);
        mprintf("PAGE %d\n", UPAGE);
        curline++;
        }
}
pgheader()
{
        pgcheck(1, outctrl[2] + 32);
        mprintf("%-13.13s","EMPLOYEE FILE");
        skipline(3);
        pgcheck(1, outctrl[2] + 0);
        mprintf("%-68.68s","==========================================================
================");
        pgcheck(1, outctrl[2] + 1);
        mprintf("%-11.11s","First Name");
        mprintf("%-23.23s","Past Jobs");
        mprintf("%-4.4s","Vita");
        pgcheck(1, outctrl[2] + 0);
        mprintf("%-68.68s","==========================================================
================");
        skipline(3);
}
```

10/07/1983 PAGE 1

EMPLOYEE FILE

First Name	Past Jobs	Vita
Donald	system analy, group manage	single, born in Lebanon,
	programmer,	M.S. Management, member MMSOLN
Jannie	accountant, IRS investig	single, born in Tulahoma
	bank examine	Ph.D. Creative A, member SSED
Carl	magician, comedian	married, born in Remingto
		XXXXXXXXXXXXX, member READ-FOR-
Charles	clerk, cashier	single, born in Dayton,
	programmer,	B.A. Mathematics
Deanna	karate instr, FBI field in	single, born in stinking
	animal train	M.S. Psycology, member Homemaker
Bob	singer, telephone op	married, born in Spitbroo
	talk show ho	B.S. Physical ed, member IIKMN
David	police offic, dancer	single, born in Bondsbro
	boxer	M.S. Civil engin, member JUIMKL

This page intentionally left blank.

Example 2

RDL source specification (page 64)

Generated C source specification (pages 65 and 66)

Five-page output report (pages 67 through 71)

User response to prompt variable:

OUTDEV - user entered SP

```
*****************************************************************************
*                                                                           *
*                            SAMPLE RDL SOURCE 2                            *
*                                                                           *
*****************************************************************************

/*******************************
 *
 *      database section
 *
 *******************************/

        database is "jobs.db"
        user "user" with "pass"

/*******************************
 *
 *      prompt section
 *
 *******************************/

        prompt variable

        OUTDEV string 10
               "Output device ?"

/*******************************
 *
 *      environment section
 *
 *******************************/

        page size is 50 lines per page
        device is OUTDEV

/*******************************
 *
 *      computed variable section
 *
 *******************************/

        computed variable

                ERNTTL   idec    7,2
                       {  sum(YTDEARN)      }

/*******************************
 *
 *      report section
 *
 *******************************/

        report is
                place unique FNAME        @ 1
                      unique LASTNAME     @ 10
                      unique SKILCODE     @ 24
                             YTDEARN      @ 34
                             PASTJOB      @ 47 skip 1 after

          thru
                idep has possess

/*******************************
 *
 *      group section
 *
 *******************************/

        page break at DNUMBER

        group by PAGE
            header
                place "EMPLOYEE FILE"   @ 33    skip  1 line
                place "=================================================================="
                                        @  1    skip 2 lines
                place "First"           @  1
                      "Skill code"      @  23
                      "Salary"          @  34
                      "Past Jobs"       @  46
                place "=================================================================="
                                        @  1    skip  2 after

        group by DNUMBER
            header
                place "Department number" @  1
                        DNUMBER           @  20  skip 2 lines after

            footer
                place "Department earning total : " @   1
                        ERNTTL                      @   29
                                                    skip 3
```

```
/* RDL analyzer, MDBS Inc, version 3.08 */
#define F_trm(a)        a
#include        "rpglob.h"
struct  hdrarr  {
        int     x0,x1,x2,x3,x4;
        };
int     outctrl[6] = { 50, 2, 5, 0, 0, 0};
short   RV_1;   /* DNUMBER  */
char    RV_2[3][26];    /* PASTJOB  */
float   RV_3;   /* YTDEARN  */
unsigned        RV_4;   /* SKILCODE */
char    RV_5[20];       /* LASTNAME */
char    RV_6[13];       /* FNAME    */
float   ERNTTL  [1];
char    OUTDEV  [11];
char    UDATE   [10];
short   UPAGE   ;
char    UTIME   [9];
int     indx1;
short   rtab1, otab1;
short   rtab2, otab2;
short   rtab3, otab3;
short   btrm2;
char    grpbrk[3];
char    brkflag[3];
char    utrm1[13];
char    utrm2[20];
unsigned        utrm3;
char lngtyp = '\0';
char ftab1[1], ftab2[1], ftab3[1];
main()
{
register        int     e0;

if (e0 = setjmp(&env)){
        errexit(e0);
        }
dmssjp(&env, 0);
if (dbinit(0))
        errexit(0);
strcpy(opblk.urwst,  "r");
strcpy(opblk.ufile,  "jobs.db");
strcpy(opblk.usernm, "user");
strcpy(opblk.upassw, "pass");
prmt();
_prompt("Output device ?"
, OUTDEV  , 1, 11, 0, 1);
chdevice(OUTDEV  , &rfname);
if(e0 = dbopn(&opblk))
        errexit(e0);
gettime(UDATE, UTIME);
opendev(rfname);
rpstart = TRUE;
rpend = FALSE;
pradjust();
pagebrk = FALSE;
curline = outctrl[0] + 1;
otab1 = rtab1 = 0;
for (e0 =  ffm("IDEF    "); e0 == 0; e0 = fnm("IDEF    ")){
        rtab1++;
        e0 = gfc("DNUMBER ", &RV_1);
        otab2 = rtab2 = 0;
        e0 = som("HAS     , IDEF    ");
        for (e0 =  ffm("HAS     "); e0 == 0; e0 = fnm("HAS     ")){
                rtab2++;
                e0 = gfc("PASTJOB ",  RV_2);
                e0 = gfc("YTDEARN ", &RV_3);
                e0 = gfc("LASTNAME",  RV_5);
                e0 = gfc("FNAME   ",  RV_6);
                otab3 = rtab3 = 0;
                e0 = som("POSSESS , HAS     ");
                for (e0 =  ffm("POSSESS "); e0 == 0; e0 = fnm("POSSESS ")){
                        rtab3++;
                        e0 = gfc("SKILCODE", &RV_4);
                        ubrk();
                        abrk2();
                        if ((!rpend) && ckpgbrk(grpbrk, 2))
                                pgbreak();
                        bbrk2();
                        update();
                        pdetail();
                        rpstart = FALSE;
                        }
                }
        }
rpend = TRUE;
ubrk();
abrk2();
pgbreak();
clsdev();
dbcls();
}
pdetail()
{
indx1 = 0;
while (indx1 < 3) {
        pgcheck(1, outctrl[2] + 0);
        if (strncmp(RV_6, utrm1, 12) || outctrl[4]) {
                mprintf("%-9.9s",RV_6);
                strncpx(utrm1, RV_6 , 12);
                }
        else    {
                pblank(9);
                }
        if (strncmp(RV_5, utrm2, 20) || outctrl[4]) {
                mprintf("%-14.14s",RV_5);
                strncpx(utrm2, RV_5 , 20);
                }
        else    {
                pblank(14);
                }
        if ((RV_4 != utrm3) || outctrl[4]) {
                mprintf("%-10u",RV_4);
                utrm3 = RV_4;
                outctrl[4] = 0;
                }
```

```
        else  {
                pblank(10);
                }
        mprintf("%-13.2f",RV_3);
        mprintf("%-25.25s",RV_2[ind%1]);
        ind%1++;
        }
        skipline(1);
}
ubrk() {
        brkflag[2] = (RV_1 != btrm2) ;
}

abrk2()
{
if (( brkflag[2]  || rpend) && (!rpstart)) {
        pgcheck(3, outctrl[2] + 0);
        mprintf("%-28.28s","Department earning total : ");
        mprintf("%-13.2f",ERNTTL  [0]);
        ERNTTL  [0] = 0;
        grpbrk[2] = TRUE;
        }
}
bbrk2()
{
if ( brkflag[2]  || rpstart) {
        pgcheck(1, outctrl[2] + 0);
        mprintf("%-19.19s","Department number");
        mprintf("%-5d",RV_1);
        skipline(2);
        btrm2 = RV_1;
        }
}
update()
{
if (rpstart) {
        ERNTTL  [0] = 0;
        }
if ((rtab2 != otab2) || (rtab1 != otab1))
        {
        ERNTTL  [0] += RV_3;
        }
otab1 = rtab1;
otab2 = rtab2;
otab3 = rtab3;
}
pdp()
{
if (!rpend || r_pfoot) {
        UPAGE++;
        mprintf("%10.10s", UDATE);
        pblank(55);
        mprintf("PAGE %d\n", UPAGE);
        curline++;
        }
}
pgheader()
{
        pgcheck(1, outctrl[2] + 32);
        mprintf("%-13.13s","EMPLOYEE FILE");
        pgcheck(2, outctrl[2] + 0);
        mprintf("%-68.68s","==========================================================
================");
        pgcheck(1, outctrl[2] + 0);
        mprintf("%-8.8s","First");
        mprintf("%-14.14s","Last Name");
        mprintf("%-11.11s","Skill code");
        mprintf("%-12.12s","Salary");
        mprintf("%-9.9s","Past Jobs");
        pgcheck(1, outctrl[2] + 0);
        mprintf("%-68.68s","==========================================================
================");
        skipline(2);
}
```


10/07/1983 PAGE 1

EMPLOYEE FILE

First	Last Name	Skill code	Salary	Past Jobs
Department number 1				
Gould	Mcmullan	2508	13000.00	park manager
			13000.00	sales clerk
			13000.00	security deputy
Leo	Gramstad	2745	10000.00	apartment manager
			10000.00	handuman
			10000.00	carpenter
Bob	Ammes	1011	9000.00	singer
			9000.00	telephone operator
			9000.00	talk show host
Deanna	Anderson	1130	11000.00	karate instructor
			11000.00	FBI field investigator
			11000.00	animal trainer
David	Ahrendt	1100	15000.00	police officer
			15000.00	dancer
			15000.00	boxer
		1030	15000.00	police officer
			15000.00	dancer
			15000.00	boxer
		1001	15000.00	police officer
			15000.00	dancer
			15000.00	boxer

Department earning total : 58000.00

10/07/1983 PAGE 2

EMPLOYEE FILE

First	Last Name	Skill code	Salary	Past Jobs
Department number 6				
Erica	Poe	2542	24000.00	technical advisor
			24000.00	production supervisor
			24000.00	system analyzer

Department earning total : 24000.00

10/07/1983 PAGE 3

EMPLOYEE FILE

First	Last Name	Skill code	Salary	Past Jobs
Department number 5				
Silver	Knight	1764	21000.00	clerk
			21000.00	postal carrier
			21000.00	author
Charles	Grupenhoff	2563	12000.00	clerk
			12000.00	cashier
			12000.00	programmer

Department earning total : 33000.00

10/07/1983 PAGE 4

EMPLOYEE FILE

First	Last Name	Skill code	Salary	Past Jobs
Department number 4				
Carl	Gudman	2653	11200.00	magician
			11200.00	comedian
			11200.00	
Jannie	Howard	2326	42000.00	accountant
			42000.00	IRS investigator
			42000.00	bank examiner

Department earning total : 53200.00

10/07/1983 PAGE 5

EMPLOYEE FILE

First	Last Name	Skill code	Salary	Past Jobs
Department number 2				
Richard	Mayerstain	2502	9900.00	computer operator
			9900.00	bartender
			9900.00	
		2306	9900.00	computer operator
			9900.00	bartender
			9900.00	
Kirk	Kommick	1852	14000.00	market analyst
			14000.00	nurse
			14000.00	civil engineer
Aaron	Galahad	2305	11300.00	programmer
			11300.00	volunteer fireman
			11300.00	pastry chef
Burton	Backoff	1203	32000.00	system engineer
			32000.00	programmer
			32000.00	cost controller
Francine	Franzmeier	2029	23000.00	system programmer
			23000.00	accountant
			23000.00	project analyst

Department earning total : 90200.00

This page intentionally left blank.

Example 3

RDL source specification (pages 74 and 75)

Eight-page output report (pages 76 through 83)

User response to prompt variable:

OUTDEV - user entered SP


```
*****************************************************************************************
*                                                                                       *
*                                SAMPLE RDL SOURCE 3                                    *
*                                                                                       *
*****************************************************************************************

/*******************************
 *
 *      database section
 *
 *******************************/

       database is "jobs.db"
       user name is "user" with "pass"

/*******************************
 *
 *      prompt section
 *
 *******************************/

       prompt

              OUTDEV    string    7
                      " printer or screen ? "

/*******************************
 *
 *      output control section
 *
 *******************************/

       page size is 50 lines per page
       device is OUTDEV
       pause

/*******************************
 *
 *      computed variable section
 *
 *******************************/

       computed variables are

              P1        idec    5,2
              {    percent(YTDEAR, DNUMBER)    }

              EMPTTL    int     2
              {     count(ID)     }

              S1        idec    11,2
              {     stddev(YTDEARN)     }

/*******************************
 *
 *      report section
 *
 *******************************/

       report is
              place underlined unique DNUMBER          @ 3
                    unique FNAME                       @ 9
                    unique YTDEARN - (0.1 * YTDEARN)   @ 19
                    P1                                 @ 32
                    EMPTTL                             @ 52
                           skip 1 after

              thru idep has
```

```
/*****************************
 *
 *      group section
 *
 *****************************/

        page break at DNUMBER

        group by REPORT
            footer is
                place "Total number of employees:" @ 10
                      EMPTTL                       @ 38  skip 5
                place "Standard deviation of year-to-date earning:" @ 10
                      S1                           @ 57  skip 3

        group by PAGE
            header is
                place "EMPLOYEE FILE"   @ 33   skip 1 line
                place "=============================================================================="
                                        @ 1    skip 2 lines
                place "Dept."           @ 2
                      "First Name"      @ 8
                      "Earning"         @ 19
                      "Percent Earning" @ 32
                      "Company employee total" @ 52
                place "=============================================================================="
                                         @ 1    skip 1 lines after

            footer is
                place "Employee total : "   @  30
                      EMPTTL                @  48   skip 5

        group by DNUMBER
            header is
                place "Employee Listing for Department #"  @ 1  skip 2 lines
                place  DNUMBER                             @ 36 skip 0 before 2 after

            footer is
                place " standard deviation for department"  @ 5
                      DNUMBER                               @ 41
                      ":"                                   @ 45
                      S1                                    @ 47 skip 3 lines

                place   S1                                  @ 45 skip 2
```

10/10/1983 PAGE 1

EMPLOYEE FILE

Dept.	First Name	Earning	Percent Earning	Company employee total
Employee Listing for Department # 1				
1	Kari	4860.00	0.08	25
	Gould	11700.00	0.18	25
	Luke	7650.00	0.12	25
	Leo	9000.00	0.14	25
	Bob	8100.00	0.13	25
	Deanna	9900.00	0.15	25
	David	13500.00	0.21	25

standard deviation for department 1 : 2900.60

2900.60

Employee total : 7

10/10/1983 PAGE 2

EMPLOYEE FILE

Dept.	First Name	Earning	Percent Earning	Company employee total

Employee Listing for Department # 7

7____	Royal	16200.00	0.46	25
_____	Karen	18900.00	0.54	25

standard deviation for department 7 : 1500.00

1500.00

Employee total : 2

10/10/1983 PAGE 3

EMPLOYEE FILE

Dept.	First Name	Earning	Percent Earning	Company employee total
Employee Listing for Department # 6				
6____	Erica	21600.00	0.51	25
_____	Otto	20700.00	0.49	25

standard deviation for department 6 : 500.00

500.00

Employee total : 2

10/10/1983 PAGE 4

EMPLOYEE FILE

Dept.	First Name	Earning	Percent Earning	Company employee total
Employee Listing for Department # 5				
5____	Mary	28800.00	0.49	25
_____	Silver	18900.00	0.32	25
_____	Charles	10800.00	0.18	25

standard deviation for department 5 : 8178.56

8178.56

Employee total : 3

10/10/1983 PAGE 5

EMPLOYEE FILE

Dept.	First Name	Earning	Percent Earning	Company employee total

Employee Listing for Department # 4

4____	Carl	10080.00	0.21	25
_____	Jannie	37800.00	0.79	25

standard deviation for department 4 : 15400.00

15400.00

Employee total : 2

10/10/1983 PAGE 6

EMPLOYEE FILE

Dept.	First Name	Earning	Percent Earning	Company employee total
Employee Listing for Department # 3				
3____	Savie	46800.00	0.62	25
____	Ralph	28800.00	0.38	25

standard deviation for department 3 : 10000.00

10000.00

Employee total : 2

10/10/1983 PAGE 7

EMPLOYEE FILE

Dept.	First Name	Earning	Percent Earning	Company employee total
Employee Listing for Department # 2				
2____	Ann	12600.00	0.12	25
_____	Richard	8910.00	0.09	25
_____	Kirk	12600.00	0.12	25
_____	Aaron	10170.00	0.10	25
_____	Burton	28800.00	0.28	25
_____	Francine	20700.00	0.20	25
_____	Donald	10800.00	0.10	25

standard deviation for department 2 : 7423.71

7423.71

Employee total : 7

```
Total number of employees:  25

Standard deviation of year-to-date earning:     11196.58
```


This page intentionally left blank.

Example 4

RDL source specification (pages 86 and 87)

Nine-page output report (pages 88 through 96)

User responses to prompt variables:

OUTDEV - user entered SP
PAGESIZE - user pressed Enter key resulting in the default of 50
LFTMGIN - user pressed Enter key resulting in the default of 5


```
*****************************************************************************************
*                                                                                       *
*                                SAMPLE RDL SOURCE 4                                    *
*                                                                                       *
*****************************************************************************************

/*******************************
 *
 *      database section
 *
 *******************************/

        database is "jobs.db"

/*******************************
 *
 *      prompt variable section
 *
 *******************************/

        prompt

                OUTDEV          string  7
                        "Screen or Printer ?"

                PAGESIZE        int     2
                        "Number of lines per page : " default   50

                LFTMGIN         int     2
                        "Left margin : " default        5

/*******************************
 *
 *      output control section
 *
 *******************************/

        page size is PAGESIZE lines per page
        left margin is LFTMGIN
        top margin is 4
        device is OUTDEV
        pause

/*******************************
 *
 *      function definition section
 *
 *******************************/

        function findraise(pl)  idec    5,2
                 pl     int
                 {
                 if (pl > 1)
                        return(YTDEARN * 0.10);
                 else
                        return(YTDEARN * 0.06);
                 }

/*******************************
 *
 *      computed variable section
 *
 *******************************/

        computed variables are

            EMPTTL    int       1
                      {  count(ID)   }

            ERNTTL    idec      7,2
                      {  sum(YTDEARN)   }

            SKILNO    int       1
                      {  count(SKILCODE)   }

            RAISE      idec       5,2
                      {  findraise(SKILNO)   }

            HIJOBCODE  int      1
                      {  max(JOBCODE)   }

            LOSKILL    int      2
                      {  min(SKILCODE)   }
```

```
/*******************************
 *
 *      report section
 *
 *******************************/

        report is
                place unique DNUMBER + 3  @ 3
                      unique ID           @ 9
                      unique JOBCODE      @ 23
                      unique SKILCODE     @ 33
                      PASTJOB             @ 43
                      unique RAISE        @ 65

                thru
                      idep has
                      possess >filledby details

/*******************************
 *
 *      group section
 *
 *******************************/

        page break DNUMBER

        group by PAGE
                header
                place "FOR INTERNAL CIRCULATION: via proper channel"
                                        @ 20    skip 2
                place "-------------------------------------------------------------------------------"
                                        @ 1     skip 3 lines
                place "Dept"            @ 1
                      "Soc-Sec-No."     @ 8
                      "Jobcode"         @ 22
                      "Skillcode"       @ 32
                      "Past jobs"       @ 24
                      "Sugg. Raise"             @ 64
                place "-------------------------------------------------------------------------------"
                                        @ 1     skip 1 line before 2 after

        group by DNUMBER
                header is
                        place "Department number :" @ 1
                              DNUMBER               @ 25

                footer is
                        place "Department employee total : "    @ 10
                              EMPTTL                            @ 39  skip 2 lines
                        place "Department salary total : "      @ 10
                              ERNTTL                            @ 39  skip 2 lines
                        place "Department skill count : "       @ 10
                              SKILNO                            @ 39
                        place "Department highest jobcode : "   @ 10
                              HIJOBCODE                         @ 39
                        place "Department lowest skilcode : "   @ 10
                              LOSKILL                           @ 39

                sort by descending HIJOBCODE

        group by ID
                header
                        place "Personal information for " @ 10
                              FNAME                       @ 40
                              LASTNAME                    @ 50    skip 2 line
                        place LINE                        @ 12    skip 1 after

                footer
                        place "Skill count is "   @ 15
                              SKILNO              @ 31      skip 2 lines
                        place "Low skill code is" @ 15
                              LOSKILL             @ 33

                sort by ascending HIJOBCODE SKILNO
```



```
10/10/1983                                                    PAGE 1

                FOR INTERNAL CIRCULATION: via proper channel

-------------------------------------------------------------------------
Dept   Soc-Sec-No.   Jobcode   Skillcode    Past jobs          Sugg. Raise
-------------------------------------------------------------------------

Department number :     6

        Personal information for      Erica     Poe
          age 36
          married
          born in Plainfield, New Jersey
          Ph.D. Management
          member VIPPL

  9     332-55-4195   51        2542       technical advisor     1440.00
                                           production supervisor
                                           system analyzer

             Skill count is  1
             Low skill code is 2542

        Department employee total :  1
        Department salary total :    24000.00
        Department skill count :     1
        Department highest jobcode : 51
        Department lowest skilcode : 2542
```

```
10/10/1983                                                        PAGE 2

                   FOR INTERNAL CIRCULATION: via proper channel

---------------------------------------------------------------------------
Dept   Soc-Sec-No.   Jobcode   Skillcode    Past jobs           Sugg. Raise
---------------------------------------------------------------------------

Department number :      5

        Personal information for      Charles   Grupenhoff
          age 29
          single
          born in Dayton, Ohio
          B.A. Mathematics

  8     562-73-1263   41         2563        clerk                 720.00
                                             cashier
                                             programmer

           Skill count is  1
           Low skill code is 2563

        Personal information for      Silver    Knight
          age 30
          married
          born in Biggers, Arkansas
          XXXXXXXXXXXXXX
          member EAT-FOR-HEALTH

        376-46-8364   42         1764        clerk                 1260.00
                                             postal carrier
                                             author

           Skill count is  1
           Low skill code is 1764

        Department employee total :  2
        Department salary total :    33000.00
        Department skill count :     2
        Department highest jobcode : 42
        Department lowest skilcode : 1764
```

```
10/10/1983                                                    PAGE 3

                FOR INTERNAL CIRCULATION: via proper channel

-------------------------------------------------------------------------
Dept   Soc-Sec-No.   Jobcode   Skillcode   Past jobs          Sugg. Raise
-------------------------------------------------------------------------

Department number :     4

        Personal information for      Jannie     Howard
          age 34
          single
          born in Tulanoma, Tennesse
          Ph.D. Creative Accounting
          member SSED

  7     342-89-1111   30        2326        accountant            2520.00
                                            IRS investigator
                                            bank examiner

             Skill count is  1
             Low skill code is 2326

        Personal information for      Carl       Gudman
          age 42
          married
          born in Remington, Virginia
          XXXXXXXXXXXXX
          member READ-FOR-PLEASURE-BOOK-CLUB

        659-56-9876   31        2653        magician              672.00
                                            comedian

             Skill count is  1
             Low skill code is 2653

         Department employee total :  2
         Department salary total :    53200.00
         Department skill count :     2
         Department highest jobcode : 31
         Department lowest skilcode : 2326
```

10/10/1983 PAGE 4

FOR INTERNAL CIRCULATION: via proper channel

Dept	Soc-Sec-No.	Jobcode	Skillcode	Past jobs	Sugg. Raise

Department number : 2

Personal information for Francine Franzmeier
age 28
married
born in Fayetteville, SC
M.S. Computer Technology
Member XI SIGMA

5	126-35-7123	11	2029	system programmer accountant project analyst	1380.00

Skill count is 1
Low skill code is 2029

Personal information for Burton Backoff
age 37
married
born in Lexington, Mass
Ph.D. Mathematics
member KIWANA club

	873-56-1422	12	1203	system engineer programmer cost controller	1920.00

Skill count is 1
Low skill code is 1203

Personal information for Aaron Galahad
age 26
married
born in Chatanooga, Tennesse
B.S. English
member Writers' guild

	231-45-3219	13	2305	programmer volunteer fireman	678.00

10/10/1983 PAGE 5

FOR INTERNAL CIRCULATION: via proper channel

Dept	Soc-Sec-No.	Jobcode	Skillcode	Past jobs	Sugg. Raise
5	231-45-3219	13	2305	pastry chef	678.00

Skill count is 1
Low skill code is 2305

Personal information for Kirk Kommick
age 27
married
born in Chicago, Illinois
B.S. Engineering Science
member XILS

876-45-6219	14	1852	market analyst nurse civil engineer	840.00

Skill count is 1
Low skill code is 1852

Personal information for Richard Mayerstain
age 25
married
born in Lancaster, New York
XXXXXXXXXXXXXXXXXXX
member WNIU

145-63-3284	15	2502	computer operator bartender	594.00
		2306	computer operator bartender	990.00

Skill count is 2
Low skill code is 2306

Department employee total : 5
Department salary total : 90200.00
Department skill count : 6

10/10/1983 PAGE 6

FOR INTERNAL CIRCULATION: via proper channel

Dept	Soc-Sec-No.	Jobcode	Skillcode	Past jobs	Sugg. Raise

Department highest jobcode : 15
Department lowest skilcode : 1203

```
10/10/1983                                                    PAGE 7

                FOR INTERNAL CIRCULATION: via proper channel

-------------------------------------------------------------------------
Dept   Soc-Sec-No.   Jobcode   Skillcode    Past jobs          Sugg. Raise
-------------------------------------------------------------------------

Department number :     1

        Personal information for      David     Ahrendt
          age 27
          single
          born in Bondsbrook, New Jersey
          M.S. Civil engineering
          member JUIMKL

  4     123-34-9875   1          1100       police officer        900.00
                                            dancer
                                            boxer
                                 1030       police officer       1500.00
                                            dancer
                                            boxer
                                 1001       police officer
                                            dancer
                                            boxer

             Skill count is  3
             Low skill code is 1001

        Personal information for      Deanna    Anderson
          age 35
          single
          born in stinking creek, Tennesse
          M.S. Psycology
          member Homemakers' association

        653-34-7655   2          1130       karate instructor     660.00
                                            FBI field investigator
                                            animal trainer

             Skill count is  1
             Low skill code is 1130

        Personal information for      Leo       Gramstad
          age 37
          married
```

10/10/1983 PAGE 8

FOR INTERNAL CIRCULATION: via proper channel

Dept	Soc-Sec-No.	Jobcode	Skillcode	Past jobs	Sugg. Raise
	born in New York, New York				
	member IIJGOLM				
4	765-65-7531	3	2745	apartment manager handyman carpenter	600.00
	Skill count is 1 Low skill code is 2745				
	Personal information for Bob Ammes age 34 married born in Spitbrook, New Hampshire B.S. Physical education member IIKMN				
	237-87-7521		1011	singer telephone operator talk show host	540.00
	Skill count is 1 Low skill code is 1011				
	Personal information for Gould Mcmullan age 30 single born in Little Rock, Arkansas B.S. Management member VLKBO				
	562-44-8743	5	2508	park manager sales clerk security deputy	780.00
	Skill count is 1 Low skill code is 2508				

```
Department employee total :  5
Department salary total :    58000.00
Department skill count :     7
Department highest jobcode : 5
Department lowest skilcode : 1001
```

Example 5

RDL source specification (pages 98 through 100)

Fourteen-page output report (pages 101 through 114)

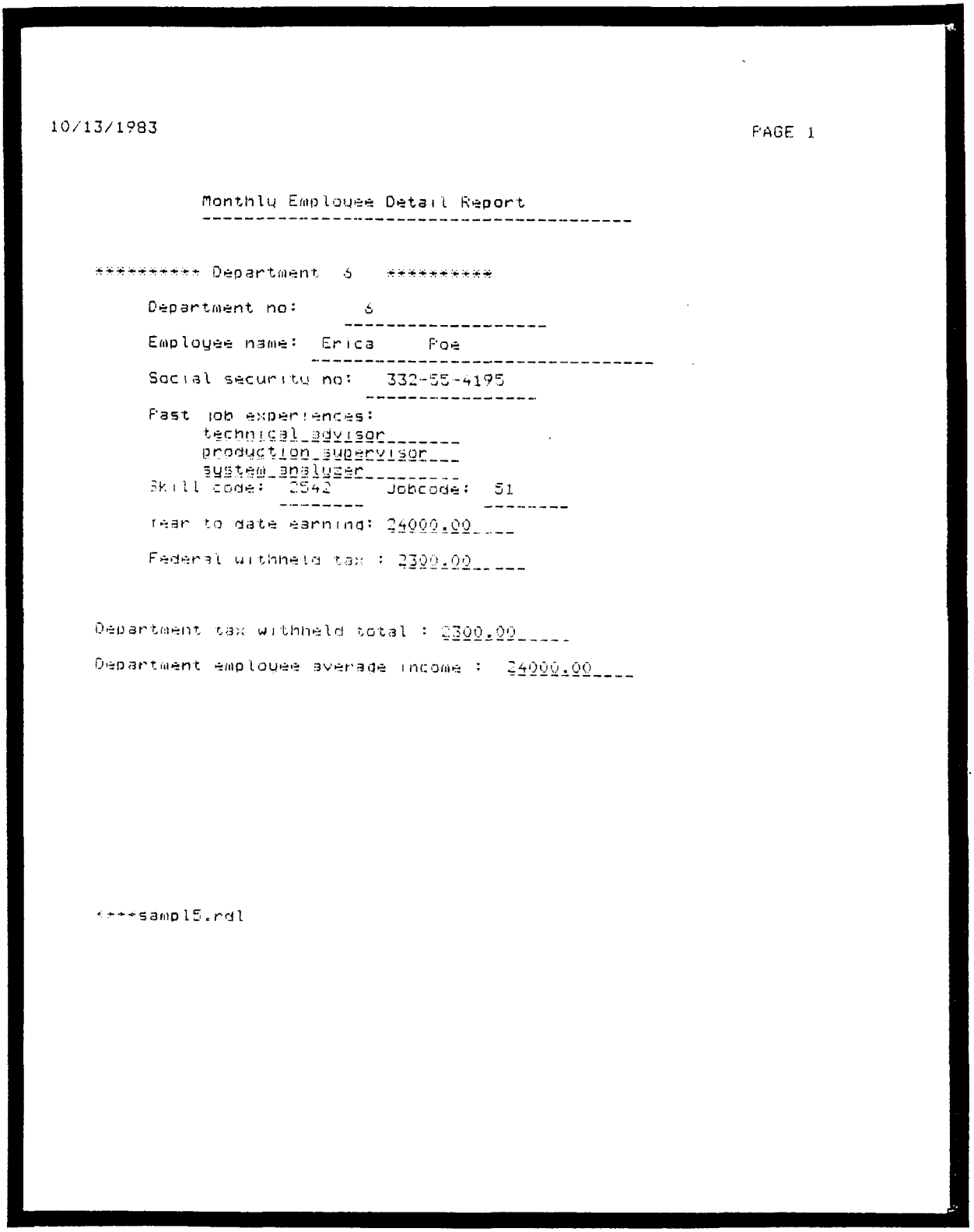
User responses to prompt variables:

UNAME - user pressed Enter key resulting in default
UWORD - user pressed Enter key resulting in default
LFTMGIN - user pressed Enter key resulting in default
MAXLINE - user pressed Enter key resulting in default
OUTDEV - user entered SP
TITLE - user pressed Enter key resulting in default
CONST1 - user entered 0
UDATE - user pressed Enter key resulting in default of computer's internal clock date
FOOTNOTE - user entered sampl5.rdl


```
*****************************************************************************************
*                                                                                       *
*                                SAMPLE RDL SOURCE 5                                    *
*                                                                                       *
*****************************************************************************************

/*******************************
 *
 *      database section
 *
 *******************************/

        database is "jobs.db"
        user name is "user" with "pass"

/*******************************
 *
 *      prompt variable section
 *
 *******************************/

        prompt variables are

                UNAME    string     10
                         "Enter user name : "
                         default "user"

                UWORD    string     10
                         "Enter user password : "
                         default "pass"

                LFTMGIN  int        2
                         "Page left margin is : "
                         default to 2

                MAXLINE  int        2
                         "Number of lines per page : "
                         default to 45

                OUTDEV   string     7
                         "printer or screen ? "

                TITLE    string     40
                         "Enter string for page title : "
                         default to "Monthly Employee Detail Report"

                CONST1   int        2
                         prompt with "Report is for DNUMBER > ? "
                         default 2

                UDATE    string     10
                         "Enter today's date : "

               FOOTNOTE    string    40
                         "Enter string for footnote : "

/*******************************
 *
 *      output control section
 *
 *******************************/

        page size is MAXLINE lines per page
        left margin is LFTMGIN
        top margin is 4
        device is OUTDEV

/*******************************
 *
 *      function definition section
 *
 *******************************/

        function  computax(p1)  real    4
                p1      idec
                {
                if (p1 < 1000.0)
                        return (0.0);
                else
                        return((p1 - 1000.0) * 0.1);
                }
```

```
/*****************************
 *
 *      computed variable section
 *
 *****************************/

        computed variables are
                EMPTTL  int     1
                        {  count(ID)    }

                ERNTTL  idec    7,2
                        {  sum(YTDEARN)    }

                AVGPAY  idec    7,2
                        {  average(YTDEARN)    }

                TAXWHLD idec    7,2
                        {  COMPUTAX(YTDEARN)    }

                TAXSUM  idec    7,2
                        {  sum(TAXWHLD)    }

                AVGTAX  idec    7,2
                        {  average(TAXWHLD)    }

/*****************************
 *
 *      report section
 *
 *****************************/

        report is
            place "Department no:" @ 10
                  DNUMBER          @ 30   skip 2 lines
            place "________________"      @ 28
            place "Employee name:" @ 10
                  FNAME            @ 26
                  LASTNAME         @ 36
            place "_____________________________________"  @ 25
            place "Social security no:"    @ 10
                  ID                       @ 32
            place "_______________"         @ 30
            place "Past job experiences:"  @ 10
            place  underlined PASTJOB                    @ 15
            place "Skill code:"     @ 10
                  SKILCODE          @ 23
                 "Jobcode:"         @ 32
                  JOBCODE           @ 42
            place "________" @ 22
                  "________" @ 41
                 "Year to date earning:"    @ 10
                 underlined YTDEARN         @ 32
            place "Federal withheld tax : " @ 10
                  underlined TAXWHLD        @ 33  skip 2

            for DNUMBER > CONST1

           thru
              idep has
              possess >filledby
```

```
/*******************************
 *
 *      group section
 *
 *******************************/

       page break at DNUMBER

       group by REPORT
           footer is
               place "SUMMARY"                                  @ 30
                                                          skip 5 lines
               place "***********"                              @ 28
               place "Total company employee income : "         @ 10
                     underlined ERNTTL                          @ 45
                                                          skip 3 lines
               place "Total company employee number : "         @ 10
                     underlined EMPTTL                          @ 45
                                                          skip 2 lines
               place "Average company employee income : "       @ 10
                     underlined AVGPAY                          @ 45
                                                          skip 2 lines
               place "Total federal tax withheld for employee : "
                                                                @ 10
                     underlined TAXSUM                          @ 50
                                                          skip 2 lines
               place "Average federal tax withheld : "          @ 10
                     underlined AVGTAX                          @ 45
                                                          skip 2 lines

       group by PAGE
           header is
               place TITLE                 @ 15 skip 3 lines
               place "_____________________________________"  @ 15
                                                  skip 2 after

           footer is
               place "*****"               @ 5 skip 4 lines
               place FOOTNOTE              @ 9 skip 0

       group by DNUMBER
           header is
               place "********** Department : "      @ 5
                     DNUMBER                 @ 28
                     "***********"           @ 32

           footer
               place "Department tax withheld total : "  @ 5
                     underlined TAXSUM    @ 37   skip 4 lines
               place "Department employee average income : "  @ 5
                     underlined AVGPAY @ 43  skip 2 lines
         sort by descending DNUMBER
```



10/13/1983 PAGE 1

Monthly Employee Detail Report

********** Department 6 **********

Department no: 6

Employee name: Erica Poe

Social security no: 332-55-4195

Past job experiences:
technical advisor
production supervisor
system analyzer

Skill code: 2542 Jobcode: 51

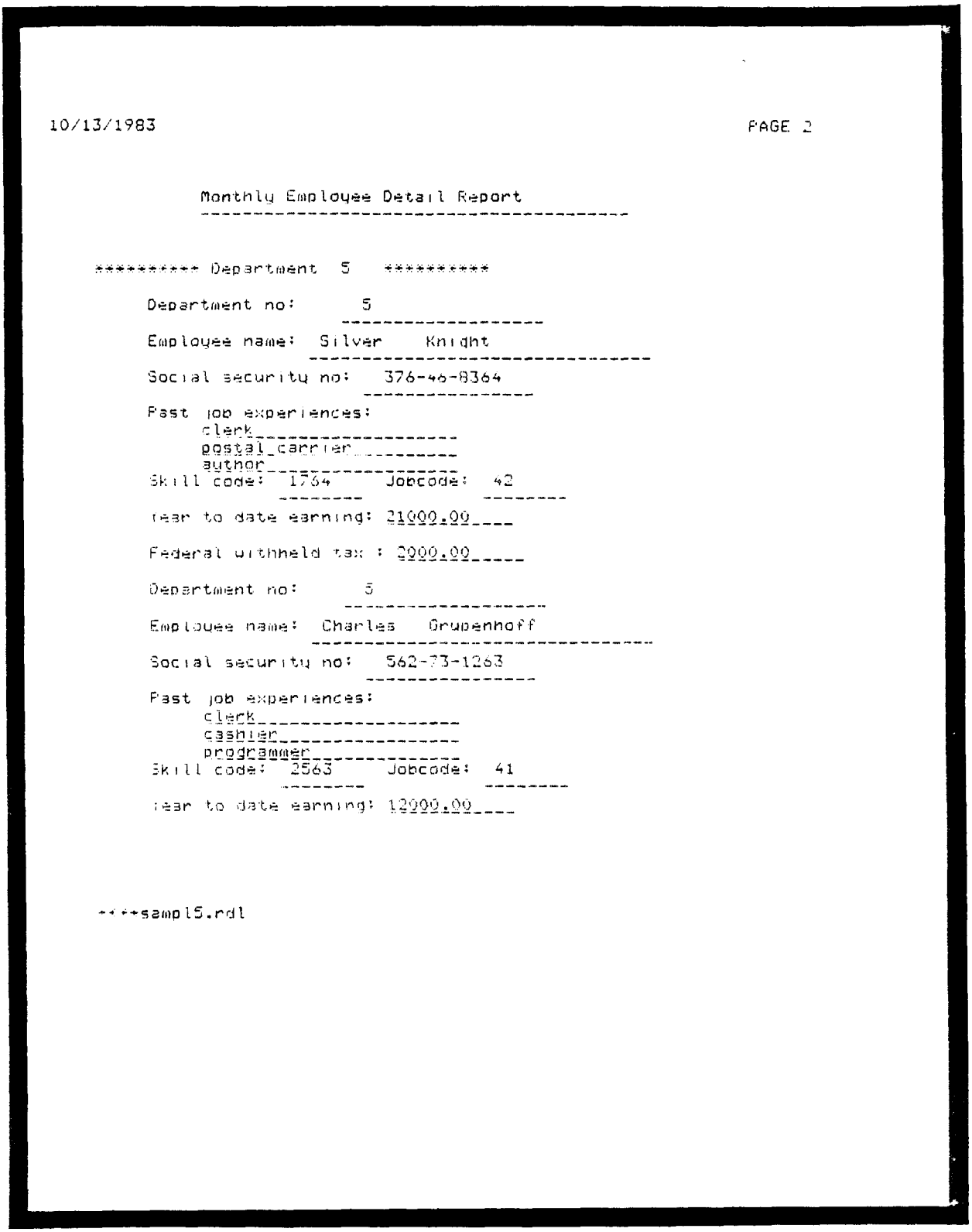
Year to date earning: 24000.00

Federal withheld tax : 2300.00

Department tax withheld total : 2300.00

Department employee average income : 24000.00

****sampl5.rdl



10/13/1983 PAGE 2

Monthly Employee Detail Report

********** Department 5 **********

Department no: 5

Employee name: Silver Knight

Social security no: 376-46-8364

Past job experiences:
clerk
postal_carrier
author
Skill code: 1734 Jobcode: 42

Year to date earning: 21000.00

Federal withheld tax : 2000.00

Department no: 5

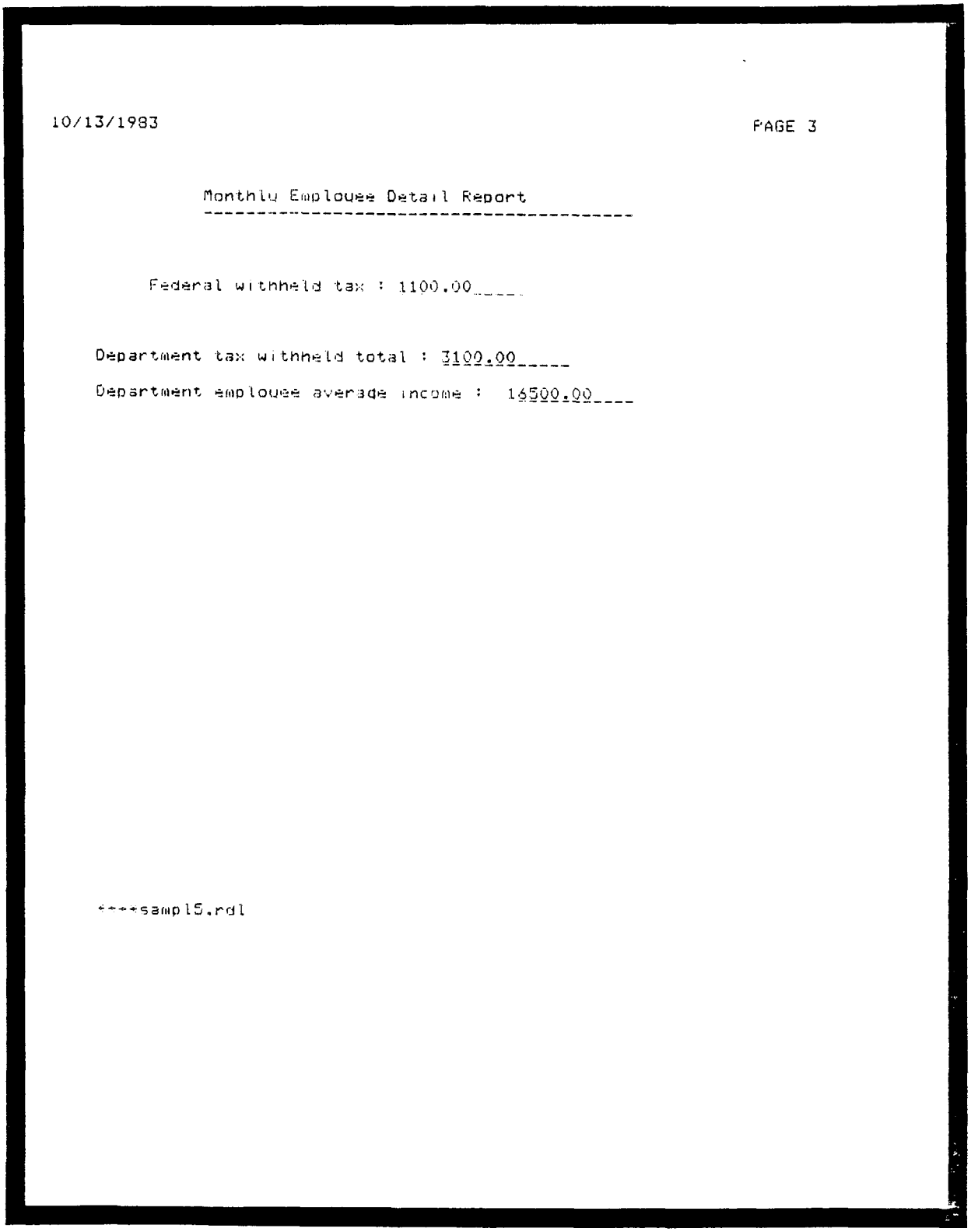
Employee name: Charles Grupenhoff

Social security no: 562-73-1263

Past job experiences:
clerk
cashier
programmer
Skill code: 2563 Jobcode: 41

Year to date earning: 12000.00

****sampl5.rdl



10/13/1983 PAGE 3

Monthly Employee Detail Report

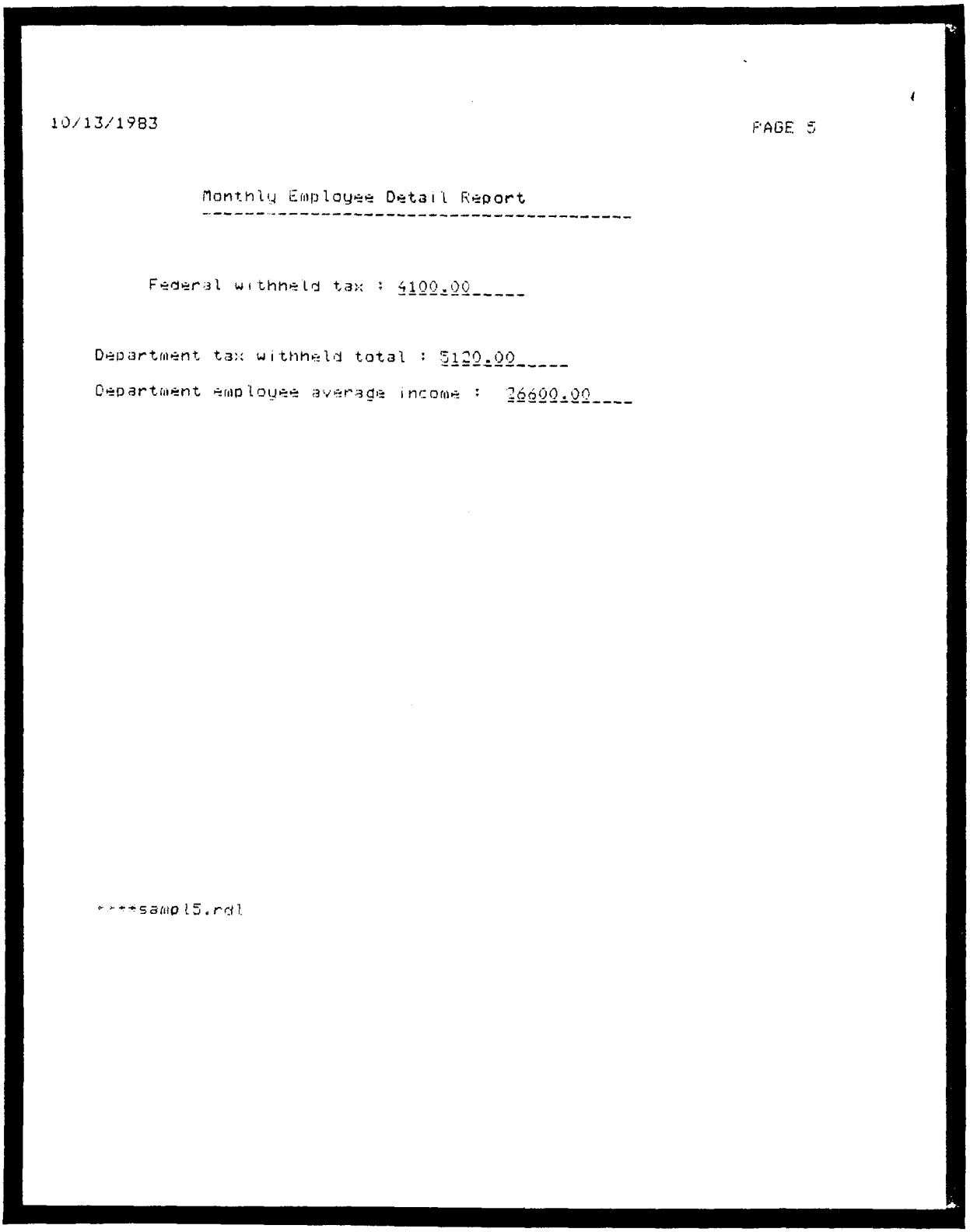
Federal withheld tax : 1100.00

Department tax withheld total : 3100.00

Department employee average income : 16500.00

****sampl5.rdl

```
10/13/1983                                                    PAGE 4

              Monthly Employee Detail Report
              ------------------------------

********** Department  4   **********

     Department no:      4
                        ------------------
     Employee name:  Carl       Gudman
                        --------------------------------
     Social security no:   659-56-9876
                        -----------------
     Past job experiences:
          magician_____________
          comedian_____________
                        --------
     Skill code:  3653      Jobcode:  31
                        --------
     Year to date earning: 11200.00

     Federal withheld tax : 1020.00_____

     Department no:      4
                        ------------------
     Employee name:  Jannie     Howard
                        --------------------------------
     Social security no:   342-89-1111
                        -----------------
     Past job experiences:
          accountant___________
          IRS_investigator_____
          bank_examiner________
                        --------
     Skill code:  2226      Jobcode:  30
                        --------
     Year to date earning: 42000.00____


****sampl5.rdl
```



10/13/1983 PAGE 5

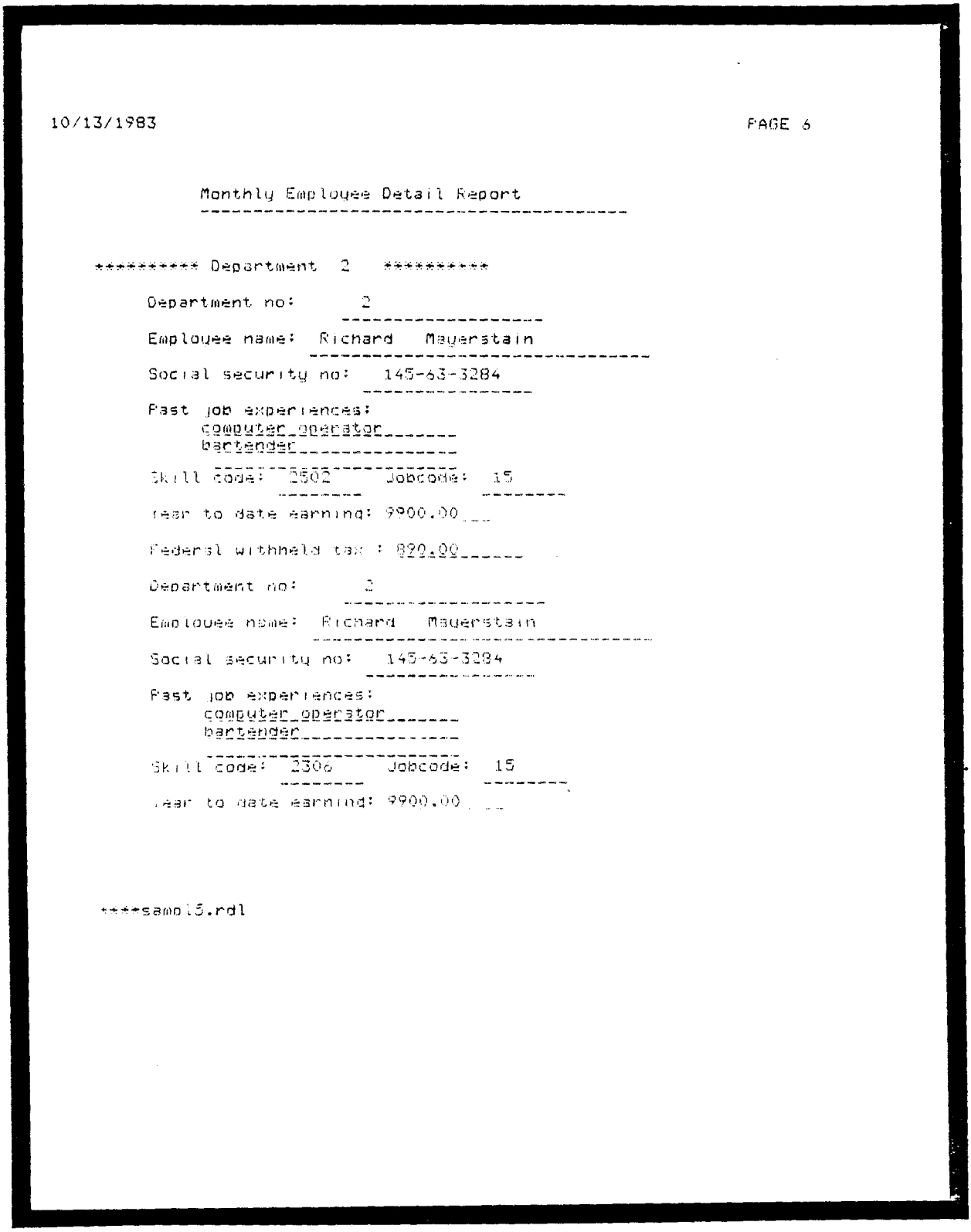
Monthly Employee Detail Report

Federal withheld tax : 4100.00_____

Department tax withheld total : 5120.00_____

Department employee average income : 26600.00____

****samp15.rdl



10/13/1983 PAGE 6

Monthly Employee Detail Report

********** Department 2 **********

Department no: 2

Employee name: Richard Mayerstain

Social security no: 145-63-3284

Past job experiences:
computer operator
bartender

Skill code: 2302 Jobcode: 15

Year to date earning: 9900.00

Federal withheld tax : 820.00

Department no: 2

Employee name: Richard Mayerstain

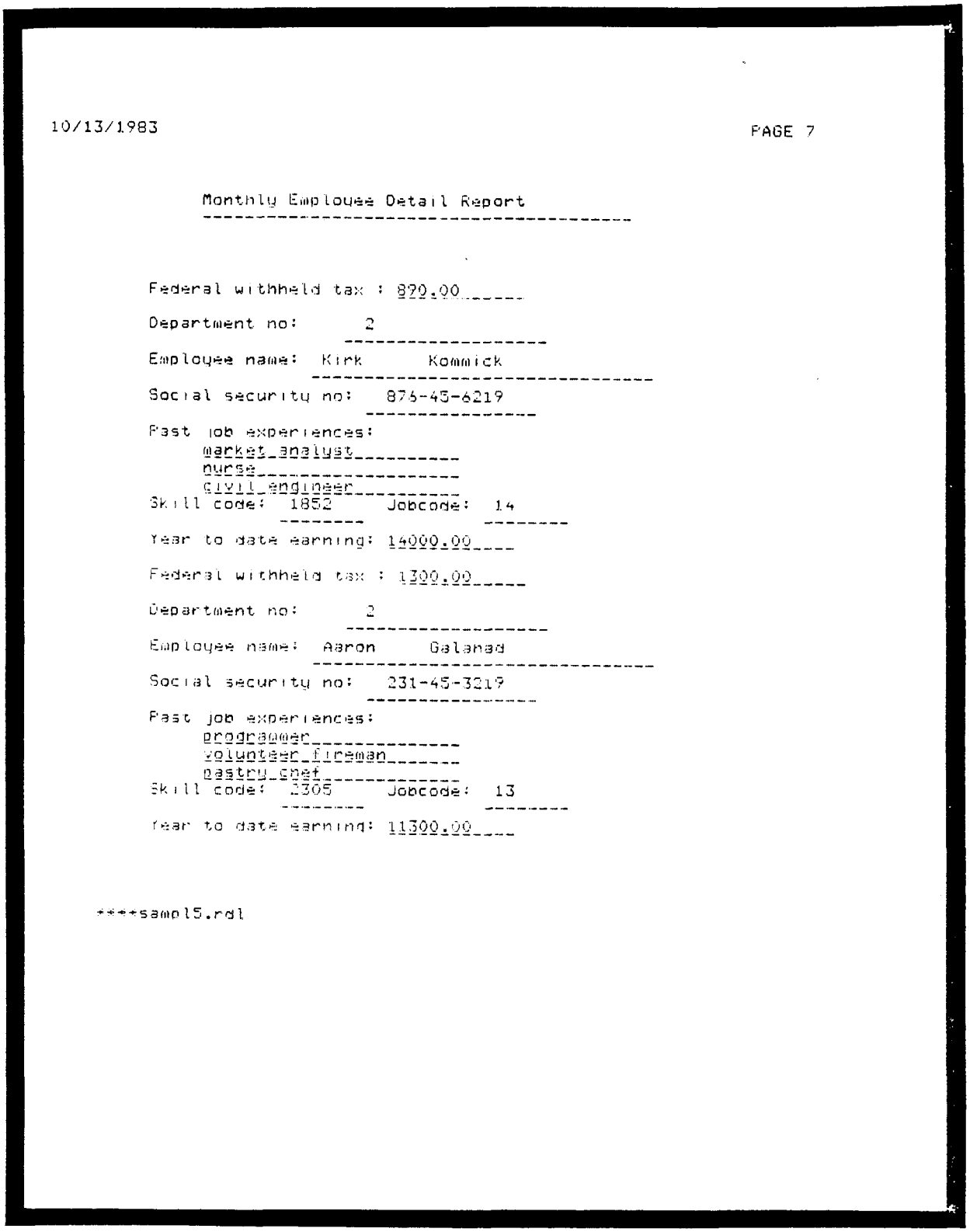
Social security no: 145-63-3284

Past job experiences:
computer operator
bartender

Skill code: 2306 Jobcode: 15

Year to date earning: 9900.00

****sample.rdl



10/13/1983 PAGE 7

Monthly Employee Detail Report

Federal withheld tax : 870.00

Department no: 2

Employee name: Kirk Kommick

Social security no: 876-45-6219

Past job experiences:
market analyst
nurse
civil engineer
Skill code: 1852 Jobcode: 14

Year to date earning: 14000.00

Federal withheld tax : 1300.00

Department no: 2

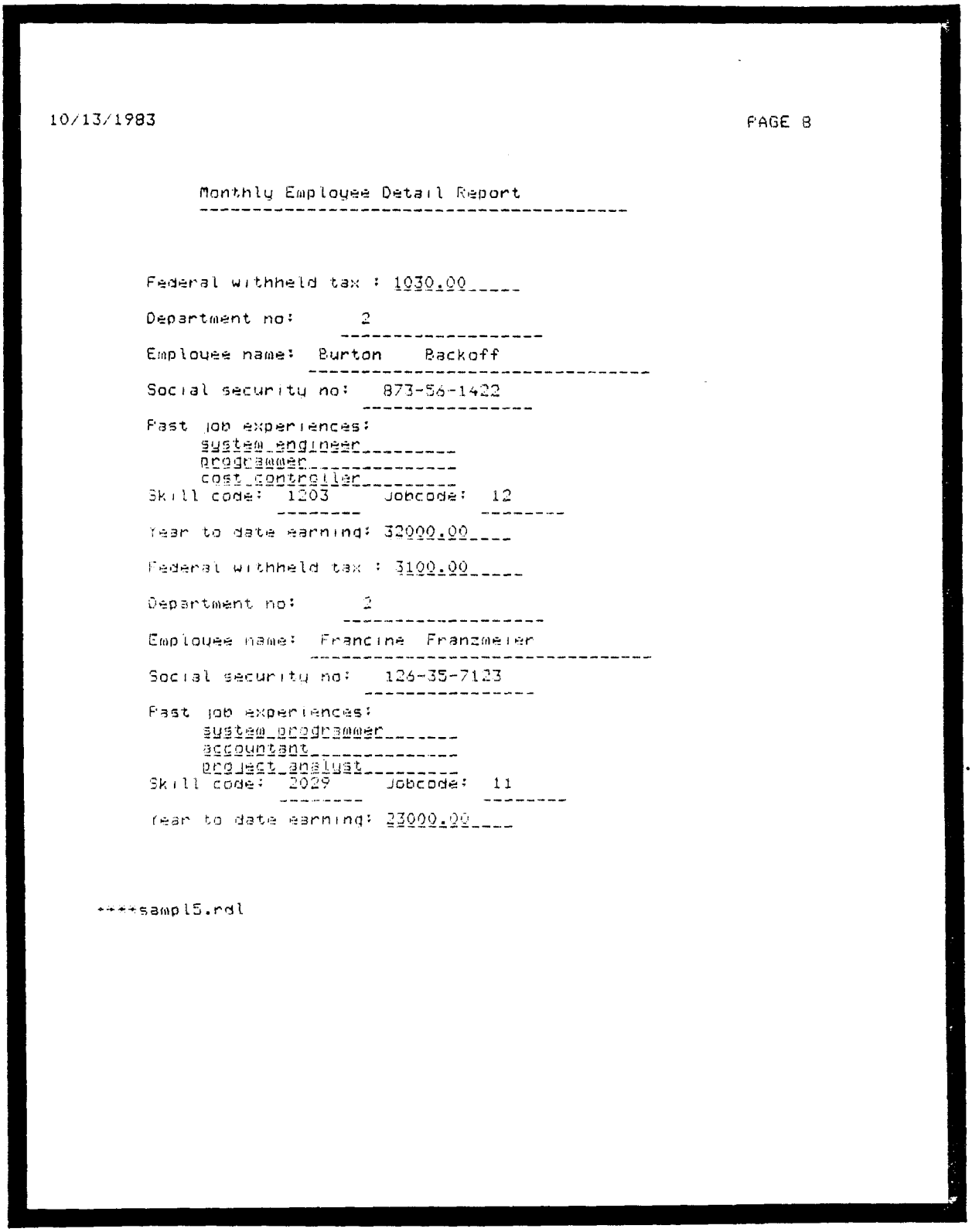
Employee name: Aaron Galanad

Social security no: 231-45-3219

Past job experiences:
programmer
volunteer fireman
pastry chef
Skill code: 2305 Jobcode: 13

Year to date earning: 11300.00

****sampl5.rdl



10/13/1983 PAGE 8

Monthly Employee Detail Report

Federal withheld tax : 1030.00

Department no: 2

Employee name: Burton Backoff

Social security no: 873-56-1422

Past job experiences:
system engineer
programmer
cost controller
Skill code: 1203 Jobcode: 12

Year to date earning: 32000.00

Federal withheld tax : 3100.00

Department no: 2

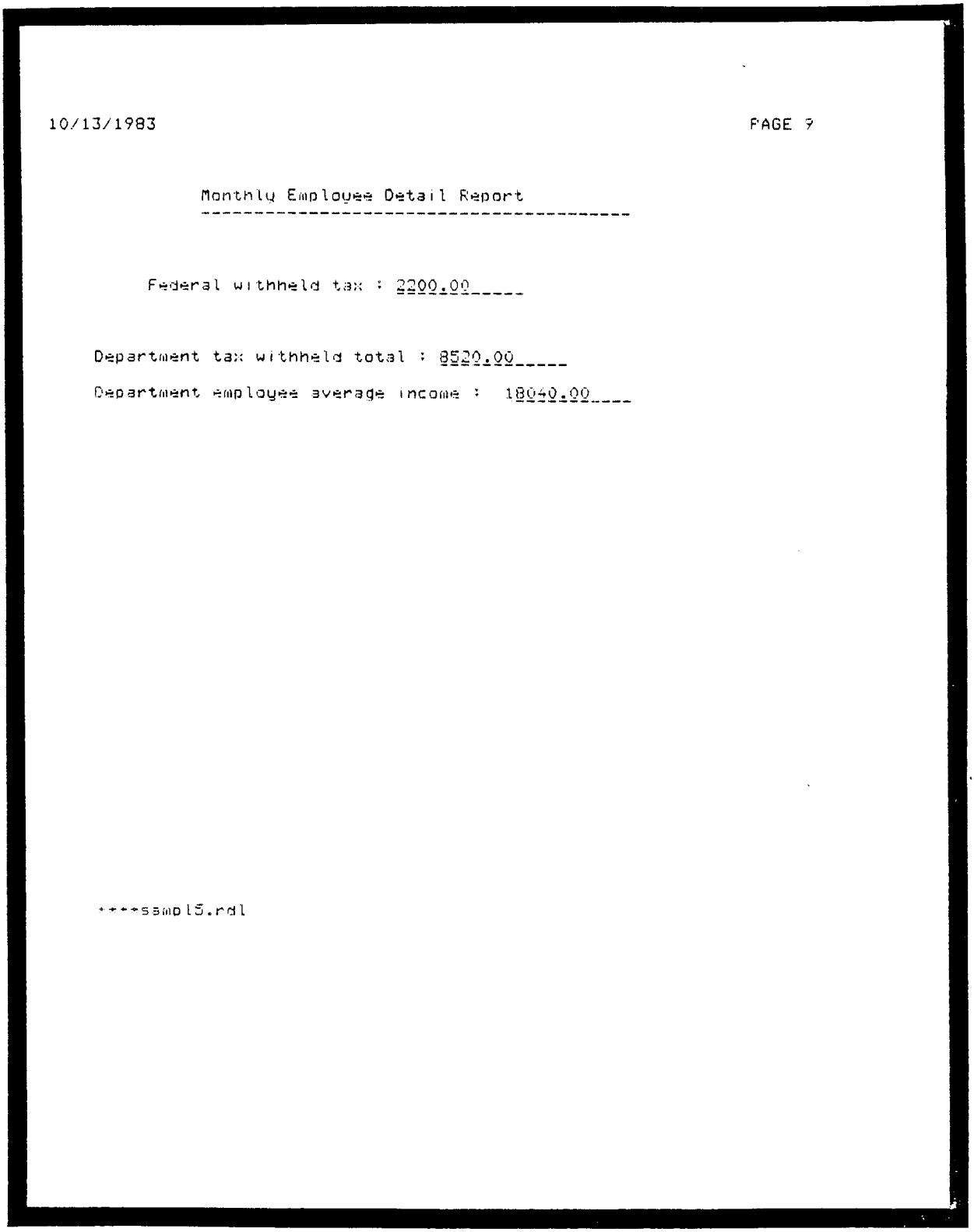
Employee name: Francine Franzmeier

Social security no: 126-35-7123

Past job experiences:
system programmer
accountant
project analyst
Skill code: 2029 Jobcode: 11

Year to date earning: 23000.00

****sampl5.rdl



10/13/1983 PAGE 9

Monthly Employee Detail Report

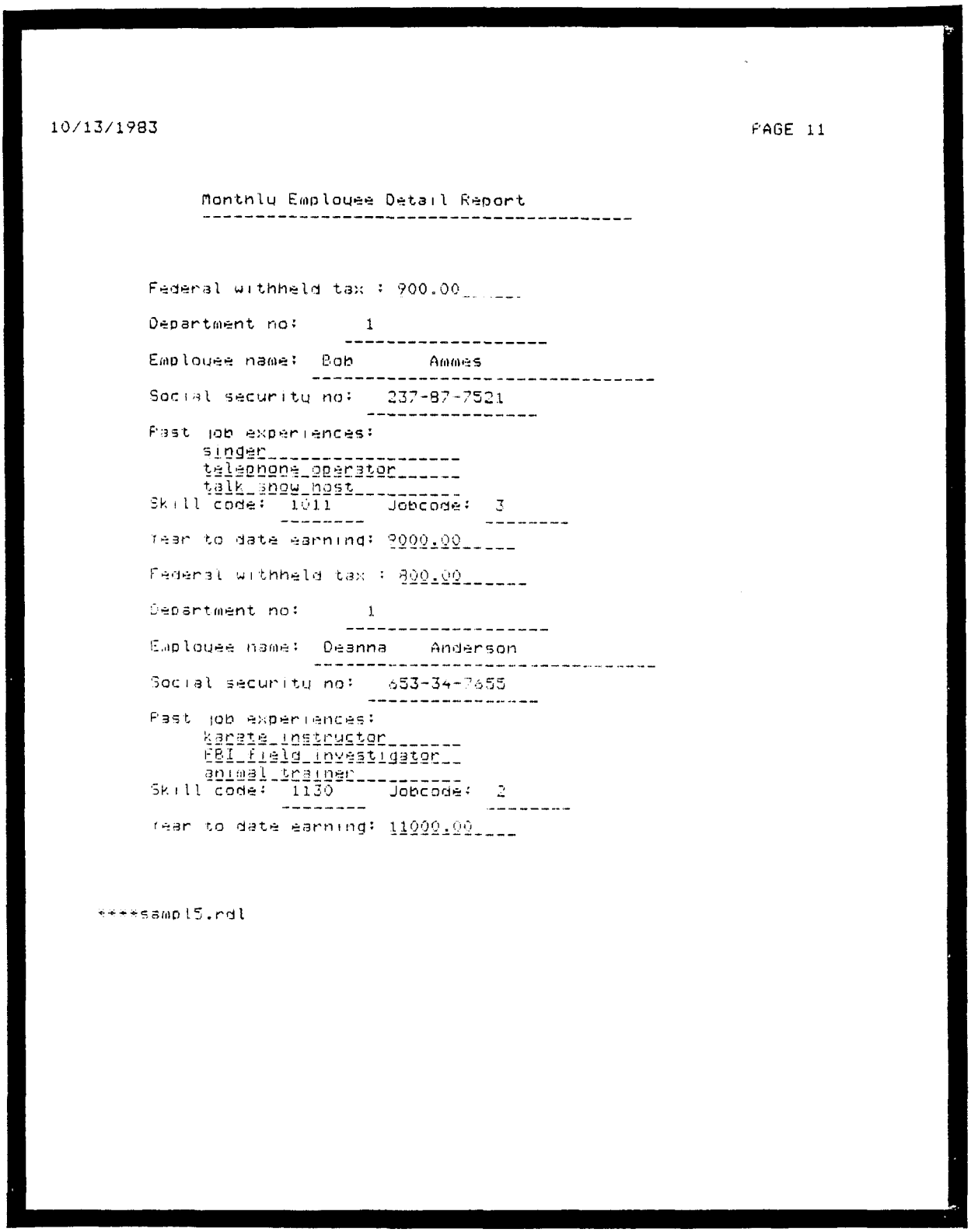
Federal withheld tax : 2200.00

Department tax withheld total : 8520.00

Department employee average income : 18040.00

****smpl5.rdl

```
10/13/1983                                                    PAGE 10

              Monthly Employee Detail Report
              ------------------------------

********** Department  1   **********

     Department no:      1
                        ------------------
     Employee name:  Gould      Mcmullan
                     ---------------------------------
     Social security no:   562-44-8743
                           ----------------
     Past job experiences:
          park_manager___________
          sales_clerk____________
          security_deputy________
     Skill code:  2500      Jobcode:  5
                  --------            --------
     Year to date earning: 13000.00____

     Federal withheld tax : 1200.00_____

     Department no:      1
                        ------------------
     Employee name:  Leo        Gramsted
                     ---------------------------------
     Social security no:   765-65-7531
                           ----------------
     Past job experiences:
          apartment_manager_______
          handyman_______________
          carpenter______________
     Skill code:  2745      Jobcode:  3
                  --------            --------
     Year to date earning: 10000.00____



****sampl5.rdl
```



10/13/1983 PAGE 11

Monthly Employee Detail Report

Federal withheld tax : 900.00

Department no: 1

Employee name: Bob Ammes

Social security no: 237-87-7521

Past job experiences:
singer
telephone operator
talk show host

Skill code: 1011 Jobcode: 3

Year to date earning: 9000.00

Federal withheld tax : 800.00

Department no: 1

Employee name: Deanna Anderson

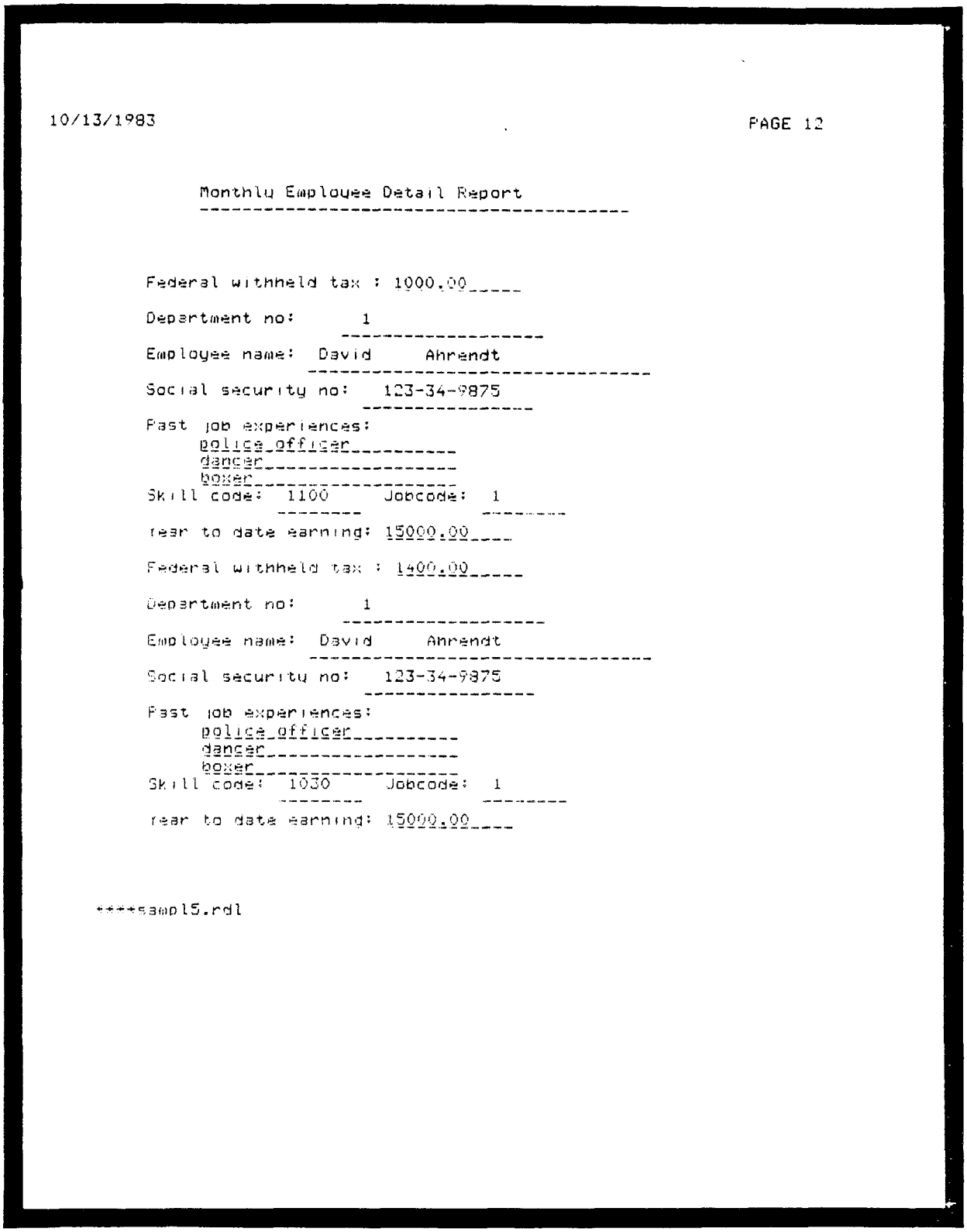
Social security no: 653-34-7655

Past job experiences:
karate instructor
FBI field investigator
animal trainer

Skill code: 1130 Jobcode: 2

Year to date earning: 11000.00

****sampl5.rdl



10/13/1983 PAGE 12

Monthly Employee Detail Report

Federal withheld tax : 1000.00

Department no: 1

Employee name: David Ahrendt

Social security no: 123-34-9875

Past job experiences:
police_officer
dancer
boxer
Skill code: 1100 Jobcode: 1

Year to date earning: 15000.00

Federal withheld tax : 1400.00

Department no: 1

Employee name: David Ahrendt

Social security no: 123-34-9875

Past job experiences:
police_officer
dancer
boxer
Skill code: 1030 Jobcode: 1

Year to date earning: 15000.00

****sampl5.rdl

10/13/1983 PAGE 13

Monthly Employee Detail Report

Federal withheld tax : 1400.00

Department no: 1

Employee name: David Ahrendt

Social security no: 123-34-9875

Past job experiences:
police_officer
dancer
boxer

Skill code: 1001 Jobcode: 1

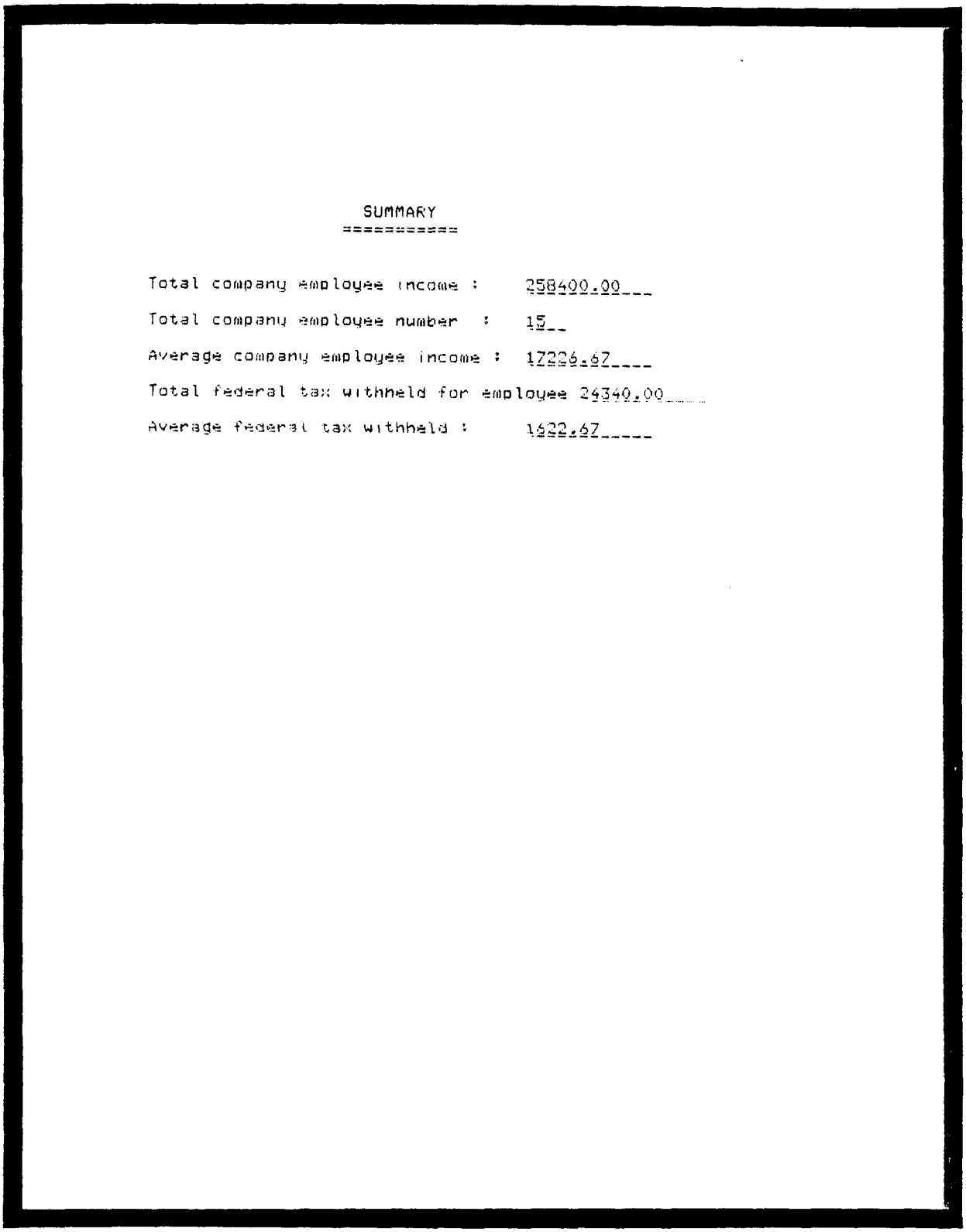
Year to date earning: 15000.00

Federal withheld tax : 1400.00

Department tax withheld total : 5300.00

Department employee average income : 11600.00

****sampl5.rdl



SUMMARY
==========

Total company employee income : 258400.00___

Total company employee number : 15__

Average company employee income : 17226.67____

Total federal tax withheld for employee 24340.00_____

Average federal tax withheld : 1622.67_____

APPENDIX A

The following are keywords in the Report Definition Language. Data items, record types, and sets in a data base being accessed through RDL cannot have any of these keywords as their names. Functions and user-defined variables should also avoid conflicts with these keywords.

ACROSS
AFTER
ALIGNMENT
AND
ARE
ASCENDING
AT
AVERAGE
AZ
BEFORE
BIN
BINARY
BREAK
BY
CASE
CHAR
CHARACTER
CLEAR
COMPUTED
COUNT
DATABASE
DATE
DECIMAL
DEFAULT
DESCENDING
DEVICE
DISPLAY
DOWN
ELSE
EQ
FILE
FOOTER
FOR
FROM
FUNCTION
GE
GROUP
GT
HEADER
IDEC
IF
INT
INTEGER
IS
LE
LEFT
LINES
LT
MARGIN
MAX
MIN
NAME
NE
NOT
OF
OR
OTHERWISE
PAGE
PAGES
PAUSE
PERCENT
PLACE
PRINTER
PROGRAM
PROMPT
REAL
REPORT
RETURN
SCREEN
SIZE
SKIP
SORT
STDDEV
STR
STRING
SUM
SUPPRESS
TEST
THEN
THRU
TIME
TOP
UNDERLINED
UNIQUE
UNSIGNED
USER
VARIABLES
WITH
XOR
ZA

APPENDIX B

See Chapter III for a detailed presentation of the RDL.

NOTATION:

<u> </u>	An underlined expression must appear.
[ ]	Zero or one of the alternatives within the brackets must be used.
{ }	Exactly one of the alternatives within the braces must appear.
{ }*	One or more of the alternatives within the braces must appear.

ORDERING OF RDL SECTIONS:

Identification Section
Prompt Section (optional)
Output Control Section (optional)
Function Definition Section (optional)
Computed Variable Section (optional)
Report Detail Section
Group Section (optional)

RDL SECTIONS:

1. Identification Section

DATABASE FILENAME IS "file-1"

[USER IS {usr / "usr"} WITH {pass / "pass"}]

[PROGRAM FILE IS "file-2"]

[SCREEN GROUP IS id-1]

2. Prompt Section (optional)

PROMPT VARIABLES ARE

{ id-1 { { UNSIGNED / REAL / INTEGER / INT / BINARY / BIN / CHARACTER / CHAR / STRING / STR } int-1
{ { DECIMAL / IDEC } int-2[,int-3]
{ TIME / DATE } } ["string-1"] [DEFAULT { "string-2" / int-4 / real-1 }] } *

3. Output Control Section

```
[PAGE SIZE {int-1} LINES]
           {id-1}

[LEFT MARGIN {int-2}]
             {id-2}

[TOP MARGIN {int-3}]
            {id-3}

[DEVICE {PRINTER  }]
        {SCREEN   }
        {"file-1" }
        {id-5     }

[PAUSE AFTER PAGE]

[SUPPRESS DATE PAGE]

ALIGNMENT PAGES {int-4}
                {id-6}
```

4. Function Definition Section (0, 1, or more per RDL specification)

```
FUNCTION id-1 [({id-2,}*)] type-1 size-1

{ {id-2,}* { { UNSIGNED  } int-1                          } }*
           { { REAL      }                                  }
           { { INTEGER   }                                  }
           { { INT       }                                  }
           { { BINARY    }                                  }
           { { BIN       }                                  }
           { { CHARACTER }                                  }
           { { CHAR      }                                  }
           { { STRING    }                                  }
           { { STR       }                                  }
           {                                                }
           { {DECIMAL} int-2[,int-3]                        }
           { {IDEC   }                                      }
           {  TIME                                          }
           {  DATE                                          }

{ procedure-1 }
```

5. Computed Variable Section (optional)

COMPUTED VARIABLES ARE

```
id-1 type-1 size-1 { [FOR (condition-1)] { SUM | AVERAGE | MAX | MIN | COUNT | STDDEV | PERCENT } (ARGUMENTS-1)
                   | function-1 [(arguments-2)] }*
```

6. Report Detail Section

REPORT

```
{ PLACE {UNIQUE UNDERLINED expression-1 [FORMAT "fmt"]
        [ @ int-1
        | int-4 ACROSS [FROM (int-5,int-6)] @ int-7 [{,int-8}*]
        | DOWN [FROM (int-9,int-10)] @ int-11 ]
        [SKIP { int-2 [LINES BEFORE [int-3 LINES AFTER]]
              | int-3 LINES AFTER } ] }*

[FOR ( condition-1 )]

THRU {[-][>]set-1,}*

[SORT BY { [ASCENDING | AZ | DESCENDING | ZA] {id-2}* }* ]
```


7. Group Section

```
[PAGE BREAK AT id-1]

{GROUP BY {REPORT}
          {PAGE  }
          {id-2  }

[HEADER IS{PLACE {UNIQUE UNDERLINED expression-1 [FORMAT "fmt"]                    }*}]
          {      [@ int-1                                                          ]  }
          {      [int-4 ACROSS [FROM (int-5,int-6)] @ int-7 [{,int-8}*]            ]  }
          {      [DOWN [FROM (int-9,int-10)] @ int-11                              ]  }
          {      [SKIP {int-2 [LINES BEFORE [int-3 LINES AFTER]]}]]}
          {            {int-3  LINES AFTER                       }  }

[FOOTER IS{PLACE {UNIQUE UNDERLINED expression-1 [FORMAT "fmt"]                    }*}]
          {      [@ int-14                                                         ]  }
          {      [int-4 ACROSS [FROM (int-18,int-19)] @ int-20 [{,int-21}*]        ]  }
          {      [DOWN [FROM (int-22,int-23)] @ int-24                             ]  }
          {      [SKIP {int-15 [LINES BEFORE [int-16 LINES AFTER]]}]]}
          {            {int-16  LINES AFTER                      }  }

[SORT GROUPS BY {[ASCENDING ]  {id-3}*  }*]*}
                {[AZ        ]           }  }
                {[DESCENDING]           }  }
                {[ZA        ]           }  }
```

INDEX

access codes 15
assignment statement. 22
break statement 24,25
C source code 1,5,13,43,45,46,53
character class matching. 23,27,31
clauses 10
clear console screen. 30,37,40
command file. 13,45,46
command status messages 46,47
comments. 10,11
computed variable section 2,3,10,26-27
computed variables. 3,6-8,20,26,28,31,33,36,41
conditions. 23,24,27,28,31
console screen output 1,5,18,30,35,38
control structures. 2,10,21-25
data base 2,4-6,12,28,43
data items. 6-8,25,26,28-30,31-33,35-37,39,41
data relationships. 8,28,32,33
data values 5-7
dates 5,13,15,19
disk file output. 1,18
DML 1,4,46,47
DMS/SMS loading 44,46
duplicate value suppression 29
end user. 5,6,14,15
errors detected by RDL Analyzer 44,47-52
expressions 3,22,25,28,36
footers 3,6,34,35,38-40
function definition section 2,10,20-25
function parameters 20,21,25
functions 2,3,6,20,21,25,26
group section 2,3,10,34-41
grouping. 3,6,34,35-41
headers 3,6,19,34,35-38
identification section. 2,10,12-13,17,19,30,37,45,46
if-then-else statement. 23-24
interaction 5,6,18
literals. 6,7
local variables 2,6,21,22,25
margins 2,5,11,18,28
MDBS Data Management System 1,5,8,44,46
operators, arithmetic 22
operators, logical. 23,27,31
operators, relational 23,27,31
output control section. 2,10,17-19
page. 5,17,18
page breaks 3,34

page numbers 5,13,15,19
page size 5,13,17,34
pause after page. 18
preprinted form alignment 19
printer output. 1,18
procedures. 9,10,21,25
program compilation 45
program execution 46
program files 4,12,13,44,59-60,65-66
program generation. 1,2,4,5,7,8,12,14,17,19,43-44
prompt section. 2,10,14-16
prompt variable 2,6,7,14-16,17-19,25,26,28,31,36
QRS 1
RDL Analyzer. 1,4,5,10,12,32,43-44,45,53
RDL library 43,45
RDL sections. 2,10
RDL specification 1,2,4,5,9,10,15,35,58,64,74-75,86-87,98-100
repeating data items. 29,30,36,37,39
report. 5
report designer 5-7,12,15,19
report detail section 2,3,10,28-33
report details. 3,5-7,17,28-33,34,35,41
report header pages 7,19
report layout 1,2,5,28-30,35-40
return statement. 25
schema. 1,32,53-55
screen dictionary 5,12,13,30,37,40
screen group. 12,13,30,37,40
SCREEN MASTER 1,5,12,46
screens 5,13,17,18,30,37,38,40
sets. 8,32,33
skipping lines. 28,29,36,38
SML 1,4,46
sort-merge processor. 43
sorting 3,7,8,28,33,35,41
spacing between columns 28,29,37,39
statistical functions 3,6,26
terms 10
test-case statement 24,25
THRU clause 3,32
underlining 29
unique qualifier. 29
user name 2,9,12,15,43
user password 2,9,12,15,43
wildcard matching 23,27,31
windows 5,13,17,18,30,37,38,40

DOCUMENTATION COMMENT FORM

MDBS Document Title:_____

We welcome and appreciate all comments and suggestions that can help us to improve our manuals and products. Use this form to express your views concerning this manual.

Please do not use this form to report system problems or to request materials, etc. System problems should be reported to **MDBS** by phone or telex, or in a separate letter addressed to the attention of the technical support division. Requests for published materials should be addressed to the attention of the marketing division.

Sender:

_____ _____
(name) (position)

_____ _____
(company) (telephone)

_____
(address)

_____
(city, state, zip)

COMMENTS:

Areas of comment are general presentation, format, organization, completeness, clarity, accuracy, etc. If a comment applies to a specific page or pages, please cite the page number(s).

_____

_____

_____

_____

_____

_____

_____

_____

Continue on additional pages, as needed. Thank you for your response.