MCBA®

Mini-Computer Business Applications Inc.

Digital Equipment Corporation

MCBA®
2441 Honolulu Avenue, Montrose, California 91020

SOFTWARE REFERENCE MANUAL

CUSTOMER ORDER PROCESSING

DIBOL

Release 7 for RT-11 DIBOL (RT)
Release 7 for RSTS/E DIBOL (RSTS)
Release 7 for VAX DIBOL (VAXD)

OCTOBER, 1985

TABLE OF CONTENTS

SECTION PAGE

SECTION PAGE

4. TECHNICAL DOCUMENTATION

This section is organized by application module and includes any Screen Formats, Program Specifications and Report Formats. These documents are used by programmers to maintain and modify the package.

5. FILE DEFINITIONS

This section contains the File Definitions in alphabetical order for all files used by this package. See the System Utilities Software Reference Manual for File Definitions of the system-wide data files (COMPNY, CONAME, DEVICE, MESARA, SECURE and SPLFIL).

SECTION PAGE

Checked by _____ Date 10 Dec 87

ALL PACKAGES
INSTALLATION INSTRUCTIONS
DIBOL AUG-84

HOW TO USE THIS MANUAL

This Software Reference Manual is organized into sections:

1. Installation Instructions
2. Technical Notes - Common to all MCBA Packages
3. Technical Notes - Specific to this Package
4. Technical Documentation
5. File Definitions

At the front of this manual there is a Table of Contents listing all sections. Subsections are listed under sections, and where the subsections are broken up into separate documents (notably in Section 4) these are listed under the subsection title.

Page numbers are of the form:

S.ss.pp

where "S" is the section number (1 through 5),
"ss" is the subsection number within a section, and
"pp" is the page number within a subsection.

Section 1 contains full instructions on how to compile the source code to create executable (runnable) program files. Sections 2 and 3 provide information on how the source code is constructed, and various naming conventions. For your convenience when working with more than one MCBA package, Section 2 contains Technical Notes that are identical in all packages for this operating system/programming language version. Section 3 contains those Technical Notes that pertain to the specific package. This way, you will not have to read through what you read in a prior package, yet it is available to refer to later. This data should be read and understood before any modification is attempted.

When you first receive an MCBA package, you should work through Section 1, referring to and reading Section 2 and 3 as needed until the executable code for your package is fully installed. You will then be ready to load your current data. At that point, you should refer to the User's Manual for this package to start using the software.

Section 4 contains detailed program specifications. In Section 4, Technical Documentation, each application has a subsection which is numbered to correspond with the selection number of the application as listed on the package menu (except that applications on the second screen of the menu have subsection numbers that follow in sequence).

Within each subsection for an application, there are specific documents which give you detailed information covering that application. Standard documents for an application are:

Screen Formats
Program Specifications
Report Formats

Not all of these documents will appear in every application subsection.

Section 5 starts with a very informational section explaining how the data files are structured. Following this, each file is defined in full detail with useful notes and descriptions.

We hope that you will find this MCBA package and its documentation useful in saving you time and money. It was designed and written with you in mind.

MINIMUM SYSTEM REQUIREMENTS

System Hardware Requirements

The VAX/VMS implementation of MCBA's Accounting/Distribution/Manufacturing system is designed specifically for DEC's VAX computers with the following minimum configuration:

1. 512 kbytes of memory.
2. Hard disk storage. (See estimates of required disk space below)
3. 1600 bpi tape drive or removable hard-disk for back-up.
4. Printer capable of printing 132 column reports.
5. VMS operating system, version 3.4 or later.
6. Vax-11 Dibol language version 2.0 or later.
   (As of this printing, version 2.1 is the latest version)

The actual disk storage requirements vary for each package within the system. There are a total of 16 possible packages in the MCBA Accounting/Distribution/Manufacturing system. If you wanted to run the entire MCBA system on one computer, you would need at least 50 megabytes of free space on the hard disk. The following estimates of space will help you decide your disk requirements:

1. Each package (e.g. I/M, COP, G/L) requires on the average of 2.1 mbytes for the executable programs and an average of .5 mbytes for the source code. The smallest package, BOMP, requires 1.0 mbytes for the executable code and .2 mbytes for the source code; the largest package, PR, requires over 3.0 mbytes for the executable code and .75 mbytes for the source code.

2. Each package's data files will add an additional 1 mbyte of space. This is a rough estimate, your own requirements will vary. See the User's Manual section entitled "Disk Space Required for Programs and Data" for more precise information.

3. MCBA's Utility programs add 1.5 mbytes.

SYSGEN PARAMETERS

System-wide parameters that may be in need of change are:

LOCKIDTBL should be no less than 256
RESHASHTBL should be one-fourth of LOCKIDTBL

For more information on these parameters and changing them by using Digital's SYSGEN utility, read Digital's System Management Operations Guide, Chapter 10, "System Parameters".

Resource Control

It is suggested that each user running MCBA's packages have the following limits set in the User Authorization file:

PRCLM should be no less than 5
ENQLM should be no less than 448
FILLM should be no less than 26
ASTLM should be no less than 25
TQELM should be no less than 25
SHRFILLM should be no less than 16

Setting Up the DIBOL Compiler and Runtime Library

If you have not already installed the DIBOL compiler and runtime library, please read the "VAX DIBOL Release Notes and Installation Guide" in the VAX DIBOL Manual.

During compilation and linking of MCBA software, it is desirable to have the DIBOL compiler and the DIBOL runtime library installed in memory as shareable images. If the available memory is limited, or the DIBOL compiler is used infrequently, install only the DIBOL runtime library, because this is used each time the executable code is run.

To install the DIBOL runtime library and compiler, type:

```
RUN SYS$SYSTEM:INSTALL
INSTALL>SYS$SYSTEM:DIBOL83/OPEN/SHARE/HEADER
INSTALL>SYS$SHARE:DBLRTL/OPEN/SHARE/HEADER
INSTALL>^Z
```

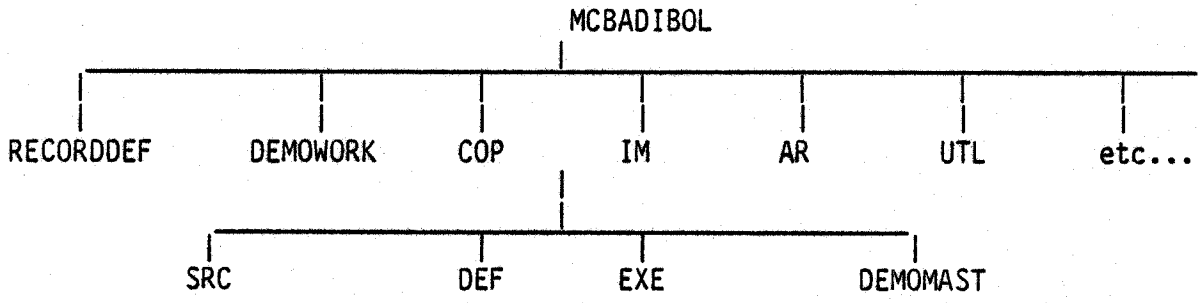
This should be inserted into the system manager's "SYSTARTUP.COM" file.

HOW TO INSTALL THE PACKAGE

Restoring from Distribution Media

To correctly restore the software you have received and save the directory structure MCBA has created, you must use Digital's BACKUP utility.

Files created by BACKUP are called save sets. The media you have received consists of two save sets that contain your software. The first contains your source and executable code. The second contains your sample data files. The restore operation will separate packages into different subdirectories. The save set being restored are "tree-structured" in the following manner:



In the above example, we have shown the directory structure for the Inventory Management package only. Each package has exactly the same structure. These installation instructions are applicable to all packages. Reference is made throughout to a "two or three" character package code. These are defined in the following table:

Package Code Table

Package Name	Package Code
MCBA Utilities	UT
Accounts Payable	AP
Accounts Receivable	AR
General Ledger	GL
Payroll	PR
Fixed Assets and Depreciation	AD
Customer Order Processing	COP
Purchase Order and Receiving	POR
Inventory Management	IM
Bill of Material Processor	BMP
Shop Floor Control	SFC
Job Costing	JC
Standard Product Costing	SPC
Standard Product Routing	SPR
Labor Performance	LP
Material Requirements Planning	MRP

The contents of each directory for each package follow the example below, in all cases, "xxx" refers to the package code:

1. [MCBADIBOL] - Resides in the master file directory as a user file directory. It contains only directories.

2. [MCBADIBOL.DEMOWORK] - This directory is used to hold the working set of demo data files. It is empty as sent to you.

3. [MCBADIBOL.RECORDDEF] - This directory is the recommended central location for all the record definitions for all packages that are installed. It is empty as sent to you.

4. [MCBADIBOL.xxx] - Resides in the MCBA directory. Its only contents are the directories below it.

5. [MCBADIBOL.xxx.SRC] - Contains the source code and build batches for the package. All the files in this directory have an extension that is the same as the standard two- or three-character package code used throughout these manuals.

6. [MCBADIBOL.xxx.DEF] - Contains the library of file definitions required to compile the source code of a package. All these files have the file extension ".DEF".

7. [MCBADIBOL.xxx.EXE] - Contains the executable code for a package that will run under the latest version of DIBOL that MCBA supports as an environment for its packages.

8. [MCBADIBOL.xxx.DEMOMAST] - Contains a master set of sample data files that comprise the package's contribution to the fictitious "MCBA Demonstration Furniture Company." These files all have the extension ".MC".

   There will also be a command file named "RESTOR.xxx". This command file will copy the master set of demo data files (.MC) for this package that are in this directory to a working set of demo data files in the [MCBADIBOL.DEMOWORK] directory. The files are renamed with the extension .MCB in the process.

   The [MCBADIBOL.UTL.DEMOMAST] directory also includes a RESTOR.ALL command file that restores the entire sample database.

You may choose any directory structure you wish. However, the Software Reference Manual and the User's Manual will refer to the above structure in examples, and it is recommended that you follow these directions in restoring the save set from your distribution medium:

1. Assign the logical "MCBAIN" to the physical device on which your distribution media is to be mounted,

For example, for magtape operations type:

```
ASSIGN MTA0:  MCBAIN:
```

(Use the physical device name of your tape drive in place of MTA0 if it is different).

For RK07 disk operations, type:

```
ASSIGN DMA0:  MCBAIN:
```

(Use the physical device name of your disk drive in place of DMA0 if it is different).

Assign the logical "MCBAOUT" to the physical device to which you are restoring. For example, type:

```
ASSIGN DRA0:  MCBAOUT:
```

(Use the physical device name of your disk drive in place of DRA0 if it is different).

2. Physically mount and put on-line the first volume of the distribution media you received.

3. Allocate the device that contains the distribution medium (ie. only allow access by yourself) by typing:

```
ALLOCATE MCBAIN:
```

4. Logically mount the drive with the foreign option by typing:

```
MOUNT/FOREIGN MCBAIN:
```

5. File Ownership and Protection - The files on the save sets that you have received all have the following protections: System = RE, Owner = RE, Group = RE, and World = RE.

   If you run the BACKUP utility as specified below, the owner (UIC code) of the files will be the account that the person performing the commands is logged into, and the file protections will NOT be changed. In order to access any data files, both directory and file protections must be: SYSTEM=RWE, OWNER=RWED, GROUP=RWED, WORLD=RWE.

   If you wish to change the file protections of the files after you restore, you should use the SET PROTECTION command as specified in your VAX Command Language User's Guide.

6. In order to verify the completeness of your order, you should get a backup listing of the files contained on the shipping media by typing:

```
BACKUP/LIST=SYS$PRINT MCBAIN:
```

   The files will be listed on the system printer.

7. Begin the restore process by typing:

```
BACKUP MCBAIN:/REPLACE/SELECT=([SHIP.DIBOLSRC...]*.*;*) -
MCBAOUT:[MCBADIBOL...]*.*;*
```

(Note that the typing of the "-" at the end of the first line will cause VAX to respond with the prompt, "$_". Then the second line would be typed.)

In order to do this process and create first level directories, you must have "SYSPRV" or "BYPASS" user privilege. If you do not have this privilege you cannot proceed. Consult your system supervisor or VAX System Manager's guide.

This restore process will take some time to complete. Its duration depends upon the number of packages of software on the distribution media.

The above procedure completes the restoration of your source and executable code. To restore the sample data files, now type, for magtape distribution media:

```
BACKUP/NOREWIND MCBAIN:/REPLACE/SELECT=([SHIP.DIBOL...]*.*;*) -
MCBAOUT:[MCBADIBOL...]*.*;*
```

For disk distribution, do not type the "/NOREWIND" option.

This procedure will be shorter than the first restore operation.

8. After the restore is complete, dismount and deallocate the device that contained the distribution media by typing the following:

```
DISMOUNT MCBAIN:
DEALLOCATE MCBAIN:
```

Setting Up Logical Directory Assignments to Build a Package

MCBA provides you with build batches that will compile and link your source code into executable modules. These build batches use the logical names listed below:

SRC =	The location of the package source code about to be built.
UTL =	The location of the MCBA utility and subroutine source code about to be built.
DEF =	The location of the record definition files that accompanied your shipment of MCBA source code.
OBJ =	The intended location of the temporary object code (created with a ".TMP" extension during the compile process and deleted after the build is complete.)
EXE =	The intended location of the package's executable code.
UT =	The intended location of the MCBA utility programs' executable code, the utility data files, and the utility object libraries, UTIL.OLB and UTIL20.OLB.

Using the recommended directory structures, these would be made by typing:

```
ASSIGN MCBAOUT:[MCBADIBOL.xxx.SRC] SRC:
ASSIGN MCBAOUT:[MCBADIBOL.UTL.SRC] UTL:
ASSIGN MCBAOUT:[MCBADIBOL.xxx.DEF] DEF:
ASSIGN MCBAOUT:[MCBADIBOL.OBJ] OBJ:
ASSIGN MCBAOUT:[MCBADIBOL.xxx.EXE] EXE:
ASSIGN/GROUP MCBAOUT:[MCBADIBOL.UTL.EXE] UT:
```

where "xxx" is the two- to three-character package code.

Note that the UT: logical assignment is made at the system level. This is the only logical assignment that is used in actually running the programs.

The DEF: logical here is set for an individual package. However, we actually recommend that all record definitions be moved into one account. The reason is that many packages use the same record definitions due to the many interfaces possible. Storing all the .DEF files in one location provides a central location from which any changes can be made without fear of overlooking the same change in a record definition library of another package.

MCBA's recommendation is that you copy all record definition files for each package to the [MCBADIBOL.RECORDDEF] account. Do so by typing:

```
COPY MCBAOUT:[MCBADIBOL.xxx.DEF]*.DEF;* -
MCBAOUT:[MCBADIBOL.RECORDDEF]*.*;*
    and
COPY MCBAOUT:[MCBADIBOL.UTL.DEF]*.DEF;* -
MCBAOUT:[MCBADIBOL.RECORDDEF]*.*;*
```

CAUTION: This command assumes that version numbers of identical file names are the same, thus causing no copy to occur. This is most important if modifications to a record definition have been made. It will also ensure that only one copy of each file exists, thus conserving disk space and speeding file access.

To make the DEF: assignment, type:

```
ASSIGN/GROUP MCBAOUT:[MCBADIBOL.RECORDDEF] DEF:
```

This centralization of record definition files has the added benefit of being able to make this assignment once at the system level for all packages and operations.

The location of the OBJ: directory is totally at the user's discretion. One way of proceeding would be to create a special directory, type:

```
CREATE/DIR MCBAOUT:[MCBADIBOL.OBJ]
```

and then to assign this directory the OBJ: logical, type:

```
ASSIGN/GROUP MCBAOUT:[MCBADIBOL.OBJ] OBJ:
```

This directory assignment would then stay in effect permanently.

The MCBA-provided build batches all have a commented out section that, if uncommented, will interactively prompt the user for the same assignments mentioned above. If you do a lot of compiling and linking of MCBA packages, you may want to uncomment these lines and simply respond to the prompts when initiating the build process. This will provide the side benefit of your not forgetting to leave out a crucial assignment and have your build batch fail.

Compiling and Linking System Utility Programs and Subroutines

The MCBA system has one set of utility programs and two different sets of subroutines. It also has a set of security system data files.

All the application packages are linked with the set of subroutines that have a .MAN file extension and that have been compiled into an object library called UTIL.OLB.

The system utility programs are programs that are shipped with each source shipment and demo. These programs include the master application menu and a number of utility programs that can be run from the back menu of the master menu. The programs and subroutines that comprise this group all have the file extension .UTL. The subroutines with the .UTL file extension are an advanced set of subroutines that incorporate enhancements over the .MAN subroutines. They are compiled into the object library called UTIL20.OLB.

Executable code already exists in the directory [MCBADIBOL.UTL.EXE]. If you intend no modifications to the source code, and you are running the same version of the DIBOL run-time system as indicated in the Minimum System Requirements for VAX/VMS section of this manual, you do not need to build the system utilities.

A build command file, BUILDUTL.UTL, is provided with your software to create both subroutine libraries and the fully executable system utility programs. It is executed in the same manner as the package specific build batches detailed in the following section.

If this is not your first installation of an MCBA package, then you also do not need to build the system utilities. This could be done by entering:

```
@UTL:BUILDUTL.UTL
```

This command file does the following:

1. Executes the command file COMPMAN.MAN. This compiles the .MAN subroutines into the UTIL.OLB library and puts the library into the logical directory UT:.

2. Executes the command file COMPUTL.UTL. This compiles the .UTL subroutines into the UTIL20.OLB library and puts the library into the logical directory UT:. It also compiles the system utility programs.

3. Executes the command file LINKUTL.UTL. This links the system utility programs with the UTIL20.OLB library and puts the executable code into the logical directory UT:.

4. Deletes the temporary object code created in logical directory OBJ:

Compiling and Linking Packages

Each package has a build batch with the file name BUILDxxx.xxx, where "xxx" refers to the two- or three-character package code.

These build batches may be run interactively from the monitor prompt by typing (after checking the assignments for DEF, SRC, and EXE):

```
@SRC:BUILDxxx.xxx
```

or they may be run in batch mode by typing:

```
SUBMIT SRC:BUILDxxx.xxx
```

The build process is automatic. If the build is in interactive mode, the terminal will display the progress. If the build is in batch mode, a print-out will be done once the build is completed, showing the commands executed and the results. Building a package takes somewhere between 20-60 minutes, depending upon the number of users on the system and the size of the package being built. As sent from MCBA, each of these build batches performs the following:

1. Runs the COMPMAN.MAN command file to compile the package subroutines into the UTIL.OLB library and puts the library into the logical directory UT:.

2. Runs the COMPxxx.xxx command file to compile the package programs.

3. Runs the LINKxxx.xxx command file to link the package programs with the UTIL.OLB library.

4. Deletes the temporary object code created in logical directory OBJ:.

Note that these build batches do NOT create the system utility programs or subroutines. This is done by running the BUILDUTL.UTL command file mentioned above.

To activate interactive assigning of logical directory assignments, uncomment the lines indicated within the build batch. Make any additional modifications as necessary.

All build batches have the command SET ON at their beginning. This will cause the batch to halt if any error is encountered.

Setting Up the Sample Data Base

If you wish to set up the sample data base provided with your source shipment, you should now execute the RESTOR command files referred to previously under item 8 in the subsection "Restoring from Distribution Media".

To do this, for each package you are installing, type:

```
@MCBAOUT:[MCBADIBOL.xxx.DEMOMAST]RESTOR.xxx
```

where "xxx" represents the package code.

As a convenience, there is a single command file, RESTOR.ALL, that will restore all your data files for all packages. To execute this, type:

```
@MCBAOUT:[MCBADIBOL.UTL.DEMOMAST]RESTOR.ALL
```

This will cause all the data files for each package and also all the data files for packages that interface to it to be copied from the DEMOMAST directory to the DEMOWORK directory. The command file replaces any existing data files of the same name with the new ones, so you need not worry about multiple versions of the same file being created.

If this is your first installation of an MCBA system, you must also type:

```
@MCBAOUT:[MCBADIBOL.UTL.DEMOMAST]RESTOR.UTL
```

This restores the security system data files.

Default logical directory assignments can also be made by typing:

```
@MCBAOUT:[MCBADIBOL.UTL.DEMOMAST]ASSIGN.MCB
```

This command file makes all necessary assignments to properly run the MCBA provided sample data base.

User's Manual Installation Instructions

The balance for the installation of MCBA source code is performed using the installation instructions contained within the User's Manual for this package.

MINIMUM SYSTEM REQUIREMENTS

Hardware Requirements

The RT-11 implementation of the MCBA system is designed for the following minimum configuration:

1. Digital PDP-11 or Micro-11 computer
2. 160 Kilobytes of main memory (256 KB or more recommended)
3. 1600 bpi tape drive, floppy diskette or RL02 hard disk
4. 3-4 megabytes free disk space for each installed MCBA package
5. 132-column printer

MCBA packages are coded and linked to require a maximum of 32 kbytes memory, not including the run-time system. 160 Kb is guaranteed sufficient for only one user to run without swapping. In minimal configurations it might be sufficient for two users.

Notes on Disk Space Requirements

There are a total of 15 application packages plus one common utilities package in the MCBA system. The actual disk storage requirements for each package varies. The entire system (consisting of executable code, source code, data files and system work files) would require 50-70 megabytes of hard disk needs, exact figures for each package can be obtained from the User's Manual section entitled "Disk Space Required for Programs and Data". For a rough estimate of space required, use the following information:

1. Each application package (e.g. utilities, I/M, COP, G/L) requires an average of 2.0 megabytes for the executable code. The smallest package, Bill of Material Processor (BOMP), requires 0.5 megabytes. The largest package, Payroll, requires 3.0 megabytes.

2. The source code for each package requires an average of 0.6 megabytes of disk storage per package.

   NOTE: The source code does not have to be resident on the disk in order to run the programs.

3. Data files will require an additional 0.5 to 1.0 megabytes per package. However, because of the highly variable needs of individual companies, this should only be used as a rough estimate.

4. Sort work space should be reserved for the system and should be approximately 30% of your largest file.

5. The MCBA system allows reports to be permanently spooled to disk files for later printing. If you intend to make use of this feature, space will have to be allocated for these files. See the sections in the User's Manual entitled "Spoolers - Special Notes" and "Company File Maintenance" for a full explanation of the MCBA Spooler and its space requirements.

Software Requirements

The RT-11 version of the MCBA system written in DIBOL is supported under two different run-time/operating system combinations:

1. CTS-300 Version 8.1 (RT-11 Version 5.1) with the XMTSD run-time
2. TSX-Plus Version 5.0 with DBL Version 2.2 run-time

CTS-300 is a product of Digital Equipment Corporation. Digital refers to it as a "packaged software system" including the RT-11 operating system, the DIBOL compiler and run-time system and other time-shared DIBOL utilities. There are three run-time systems supplied running DIBOL programs: Single User DIBOL (SUD), Time-shared DIBOL (TSD), and Extended Memory TSD (XMTSD). MCBA packages require the XMTSD run-time system. Neither SUD nor TSD supply enough memory to run MCBA programs.

TSX-Plus is a product of S & H Computer Systems of Nashville, Tennessee. It is a multi-user operating system running from the RT-11 monitor. It provides many facilities not available in CTS-300, including the ability to access up to 4 mbytes of main memory and support for up to 31 users.

DBL is a product of Digital Information System Corporation (DISC) of Sacramento, California. The DBL language is a superset of the DIBOL language. Version 2.2 DBL is not entirely compatible with DIBOL-83. Version 4 of DBL (under development as of this writing) will restore full compatibility with DIBOL-83.

Release 7 of the MCBA system only uses those language elements common to both DIBOL-83 and DBL v2.2. However, future enhancements and patches to this product will make full use of the DIBOL-83 language, thus possibly removing compatibility with DBL v2.2.

TSX-Plus will run programs compiled with the DIBOL compiler. However, no record-locking facilities are provided, and the DIBOL subroutine, TNMBR, does not return a valid terminal number for non-console terminals. Therefore, MCBA packages are not supported under TSX-Plus when compiled under the DIBOL compiler.

Any attempt to run MCBA packages under earlier versions of CTS-300, TSX-Plus, or DBL than mentioned above may cause errors and is not supported.

RT-11/CTS-300 Sysgen Considerations

In RT-11 v5.1, the following answers to the sysgen dialogue are recommended:

1. Use the CTS-300 dialogue.

2. Only the extended memory monitor is required.

3. The size of the output buffer should be 134.

4. Answer "yes" for high speed ring buffer support.

5. Answer "yes" for BATCH support.

All other questions can be answered in response to individual needs and do not affect the functioning of MCBA packages.

For the CTS-300 v8.1 sysgen dialogue, take into consideration the following:

1. Only a time-shared system needs to be built.

2. Total messages stored in memory - no terminal should ever have more than one message pending at any one time. Only the sort programs and programs that access the CTS-300 spooler will send messages.

3. Programs to be run - no MCBA programs run detached. MCBA makes use of the CTS-300 spooler, which does run detached.

4. Total channels to be used - the average MCBA program will open 4-5 files at any one time. Only a very few open more than 8.

5. Extended memory must be used.

6. Number of files open at any one time - in a mutli-user environment, it is likely different programs will have at least one file open in a shared mode. The larger the number of users, the more this will occur.

7. Number of files open for update - it is unusual for any program to have more than two files opened for update.

8. ISAM files are used, but infrequently. Usage may increase in the future. The number of ISAM volumes per file is dependent on response to file initialization questions. However, MCBA does not recommend that any of its ISAM files be multi-volume. Thus the minimum, two, can be entered.

9. DDT is not necessary, unless you intend to do independent software development.

10. Implicit job startup must be used.

All other questions can be answered according to your needs.

Modification to LINK.SAV

The RT-11 Installation Guide has a section entitled, "Choosing Software Customizations". The manual states that for DIBOL, the patch detailed in section 2.7.7, "Changing the Size of LINK's Library Module List" must be installed. If you forget to do this, attempts to execute MCBA's build batches may result in a "?LINK-F-Library list overflow, increase size with /P" error message.

TSX-Plus Sysgen Considerations

The following parameters, which must be set in TSGEN.MAC, closely impact the execution of MCBA programs:

1. HIMEN: Should be 64. This number must take into consideration the space required by the run-time system.

2. DFLMEM: Can be as low as 32 if a shared run-time system is being used. Otherwise, it should be 64.

3. MAXCSH and NMFCSH: Directory caching should be used. Actual values will vary depending on how you choose to dispose MCBA data files.

4. MXSPAC: MCBA programs require 12 activation characters per line. Since you should not be masking TSX-Plus anomalies under DBL, this is all that is needed.

5. MAXSF: The average MCBA program will open 4-5 files at any one time. Only a few open more than 8. A safe number is four times the number of terminals on line at any one time. As the number of terminals increase, the multiplier of four can be decreased, since each opened file will be increasingly likely to have already been opened by another program. The minimum should be 30.

6. MAXSFC: A safe number is five times the number of terminals on line at any one time. The minimum should be 30.

7. MXLBLK: Should be 7.

8. MAXMC, MSCHRS, and MAXMSG: All three can be zero. MCBA programs running under TSX-Plus/DBL do not make use of the SEND or RECV language statements.

9. DBL should be installed as a shared run-time system.

DBL Version 2.2 Sysgen Considerations

All mandatory patches must be installed at least through patch 40. Patch 40 alters the form of the .INCLUDE statement to bring it into conformity with the DIBOL-83 version. Optional patches PCH01.MAC and PCH08.MAC MUST be installed.

The following items refer to the DBL generation itself:

1. A shared run-time system should be generated.

2. MCBA programs do not use the DBL SEND or RECV statements in its code. Whichever message facility fits your needs may be selected.

3. TSX anomalies should NOT be masked.

4. It is not necessary to remember the last STOP device or extension.

5. The new standard form of RENAM must be used.

6. Default blocking factors should be honored.

7. ISAM support is required. MCBA build batches assume that ISAM has been sysgenned into the DBL run-time system.

8. A stack size of 64 must be sysgened. (This is used by the MRP package.)

ALL PACKAGES
INSTALLATION INSTRUCTIONS FOR DIBOL RT-11
DIBOL FEB-85

HOW TO INSTALL THE PACKAGE

The programs and sample data files comprising your source shipments have been put onto your distribution media via the RT-11 PIP utility. All files are protected.

Each source license distribution contains:

1. One Software Reference Manual for each package ordered
2. One User's Manual for each package ordered
3. Source code consisting of:
   a. Program source code for each package
   b. Program source code for the system utility programs and subroutines
   c. A record definition library
4. Sample data files for each package ordered
5. Sample security system data files

In addition, if this is your first MCBA order, you will also receive a Utilities Software Reference Manual.

The Installation Instructions below are applicable to all packages. Reference is made throughout to a "two- or three-character package code". These are defined in the following table:

Package Code Table

Package Name	Package Code
MCBA Utilities	MAN and UTL
Accounts Payable	AP
Accounts Receivable	AR
General Ledger	GL
Payroll	PR
Fixed Assets and Depreciation	AD
Customer Order Processing	COP
Purchase Order and Receiving	POR
Inventory Management	IM
Bill of Material Processor	BMP
Shop Floor Control	SFC
Job Costing	JC
Standard Product Costing	SPC
Standard Product Routing	SPR
Labor Performance	LP
Material Requirements Planning	MRP

No attempt has been made to separate the source code of multi-package shipments into any sort of logical disk subsets. However, you may find it convenient to

do so in order to speed up file access speeds. Program naming conventions are as follows:

1. Source code programs all have the same extension as the package code defined above. In addition, MCBA utilities also include some programs with the extensions .ALL and .TEC.

2. Record definitions all have the extension ".DEF". It is recommended record definitions of all packages reside in the same directory. However, you may determine which record definitions belong to which package by referring to a file called LSTDEF.xxx, where "xxx" is the package code. Many record definitions are used in more than one package.

3. Sample security system data files all have the extension ".DDE".

4. Sample application data files all have the extension ".MC" and ".ONE", except ISAM data files, which have extensions ".TSX", ".TS1", ".CTS", and ".CT1".

Restoring from Distribution Media

The MCBA distribution media should not be used as work disk(s). Your first step, therefore, is to create a duplicate or backup copy of your distribution media.

1. Assign logical "SRC" to the physical device containing your distribution medium. For example, for magtape operations, type:

```
ASSIGN MT0: SRC:
```

2. Assign logical "DST" to the physical device containing the backup copy. For example, if the destination was to your second RL02 drive, you would type:

```
ASSIGN DL1: DST:
```

3. Mount your distribution medium and write-protect the disk.

4. Copy the contents of the distribution medium to the backup device by typing:

```
COPY SRC:*.* DST:*.*
```

   The original protected file status will remain in place. If you have received a multi-volume shipment, or have existing files from a prior shipment, you may receive some warnings that certain files were not copied due to the existence of a protected file of the same name on the destination disk. This is perfectly normal.

   If you wish to remove the protection, type:

```
UNPROTECT DST:*.*
```

5. Remove the first distribution volume.

6. Repeat steps 2 through 5 until all distribution volumes have been copied.

7. You may verify the completeness of your order by getting a directory listing of your newly created disk. Type:

   DIRECTORY/PRINTER/ORD:TYP/COL:4 DST:

   You can then compare this with listing files provided with the distribution. These are named LSTSRC.xxx for the source code and LSTDEF.xxx for the record definitions, where "xxx" is the two- or three-character package code.

Setting Up Logical Directories and Assignments

MCBA provides you with build batches that will compile and link your source code into executable modules. These build batches use the logical names listed below:

SRC =	The location of the package source code about to be built.
UTL =	The location of the MCBA utility and subroutine source code about to be built.
DEF =	The location of the record definition files that accompanied your shipment of MCBA source code.
OBJ =	The intended location of the package temporary object code (created with a ".TMP" extension during the compile process and deleted after the build is complete.)
EXE =	The intended location of the package's executable code.
UT =	The intended location of the MCBA utility programs' executable code, the utility data files, the utility object libraries, UTIL.OBJ and UTIL20.OBJ, and subroutine object files. The object files have the extension .TM1 for the MAN subroutines and .TM2 for the UTL subroutines. The utility data files have the extension .DDF with the exceptions of CONAME.MCB and CONAME.ONE.

At MCBA, we use the logical disk subsetting feature to separate the programs in the following manner:

1. Place all .DEF files (logical assignment DEF) in a unique logical disk. The entire 16 package system contains over 800 .DEF files taking up nearly 1200 blocks of disk space.

2. Place all .MAN, .UTL, and .TEC files (logical assignment UTL) in a unique logical disk. There are approximately 130 files taking up over 1300 blocks of disk space and comprise the utilities source shipment.

3. Place all utility executable and object code created by the build batches (logical assignment UT) in a unique logical disk. There are approximately 150 files taking over 2200 blocks of disk space.

4. Logical OBJ should be on a disk with at least 5000 free blocks and a clean directory with 31 segments. Even though the temporary object files placed in this directory are deleted at the end of a build batch, the directory entry remains. If you build several packages without squeezing the disk (via the RT-11 SQUEEZE command), you could encounter "Device Full" or "Directory Full" error messages even though there is free space on the disk.

5. Source and executable code for individual packages must be consolidated, since only 8 logical disks are allowed. This should be done at your convenience.

   The MCBA-provided build batches for TSX-Plus will interactively prompt the user for the same assignments mentioned above. For the CTS-300 build batches run under the control of the BATCH processor, these assignments must be made manually.

6. Although not required to build your packages, it is a good idea to also separate the sample data files that are provided. All files that have extensions .MC, .DDE, .TSX, .CTS, .TS1, .CT1 and .ONE plus all files with the filename RESTOR.xxx should be placed on this device. For all packages, there are approximately 140 files requiring nearly 3000 blocks of disk space.

   This device contains the files referred to as the master set of sample data files throughout the Software Reference and User Manuals.

Compiling and Linking System Utility Programs and Subroutines

The MCBA system has one set of utility programs and two different sets of subroutines. It also has a set of security system data files.

All the application packages are linked with the set of subroutines that have a .MAN file extension and that have been compiled into an object library called UTIL.OBJ.

The system utility programs are programs that are shipped with each source shipment and demo. These programs include the master application menu and a number of utility programs that can be run from the back menu of the master menu. The programs and subroutines that comprise this group all have the file extension .UTL. The subroutines with the .UTL file extension are an advanced set of subroutines that incorporate enhancements over the .MAN subroutines. They are compiled into the object library called UTIL20.OBJ.

Build command files are provided with your software to create both subroutine libraries and the fully executable system utility programs. They are executed in the same manner as the package specific build batches detailed in the following section.

If this is not your first installation of a Release 7 MCBA package, then you will not need to build the system utilities.

To build the utilities and subroutines under TSX-Plus, you must execute the command file TSXBLD.UTL. This is done by typing:

```
IND UTL:TSXBLD.UTL
```

where UTL: is assigned as described above.

This command file does the following:

1. Asks the user if he wishes to create a log file of the build batch. If he answers "Y", the name of the log file is asked. The default is the printer, but any valid file name can be entered.

2. Asks the user if he wishes to make logical assignments now. If yes, then the user is asked to enter the physical device corresponding to the logicals DEF, UT, and OBJ. (UTL must already have been manually assigned.)

3. Asks the user if he wishes to create the package subroutine library, UTIL.OBJ, and, if so, if he wishes to delete the .MAN subroutines' object code (created with a .TM1 extension).

4. Asks the user if he wishes to create the utility subroutine library, UTIL20.OBJ, and, if so, if he wishes to delete the utility program object code (created with .TMP and .DBL extensions) and the .UTL subroutines' object code (created with a .TM2 extension).

5. Asks the user if he wishes to link the utility programs.

6. Executes TSXCMP.MAN if yes answered to #3 above. This compiles the .MAN subroutines and stores the object code under the extension .TM1. It then creates the UTIL.OBJ library from the .TM1 object code. Both the object code and the library are placed in the UT: logical device.

7. Executes TSXCMP.UTL if yes answered to #4 above. This compiles the .UTL subroutines and stores the object code under the extension .TM2. It then creates the UTIL20.OBJ library from the .TM2 subroutine object code. The utility programs are also compiled, and the object code stored with the extension .TMP in the OBJ: logical device.

8. Executes TSXLNK.UTL if yes answered to #5 above. This links the utility programs with the subroutines and library created by TSXCMP.UTL and places the executable code in the UT: logical device.

9. Deletes those object files that were requested to be deleted.

To build the utilities and subroutines under CTS-300, a BATCH command file called CTSBLD.UTL is provided. The BATCH processor must be used instead of the indirect command file processor because the GSORT program that creates the sort programs must have direct operator input. The indirect command file processor does not have a facility that will allow such input to be provided from a command file, whereas the BATCH processor does.

Follow these steps:

1. Manually enter the logical device assignments referred to in the previous section.

2. All device handlers required must be loaded. Assuming logical subsets are used, you would type:

```
LOAD LP,BA,LD
```

Any other physical devices being used, such as DL, DM or RM, would also be loaded with a similar command. Logical devices such as SRC and UT do not have to be loaded.

3. Assign the printer the logical device LOG by typing:

```
ASSIGN LP: LST:
ASSIGN LP: LOG:
```

4. The BATCH processor cannot be run from XMTSD, but must be run from the RT-11 monitor. Invoke it by typing:

```
R BATCH
```

5. It will respond with an "*" prompt. Ensure the printer (LP) is on line. Then type:

```
UTL:CTSBLD.UTL
```

The printer will record the progress of the build batch. When complete, the words "END BATCH" will appear on the terminal.

The CTSBLD.UTL command file itself performs exactly the same steps as the TSXBLD.UTL command file, except it calls command files named CTSCMP.MAN, CTSCMP.UTL and CTSLNK.UTL. However, no interactive prompting is done. The temporary subroutine object code is not automatically deleted, but the utility program object code is deleted.

Compiling and Linking Packages

Each package has two build batches with the file names, TSXBLD.xxx and CTSBLD.xxx, where "xxx" is the two- or three-character package code. These build batches create executable code to run under the TSX-Plus and CTS-300 operating systems, respectively.

To execute the TSX-Plus command file, type:

```
IND SRC:TSXBLD.xxx
```

This command file does the following:

1. Asks the user if he wishes to create a log file of the build batch. If he answers "Y", the name of the log file is asked. The default is the printer, but any valid file name can be entered.

2. Asks the user if he wishes to make logical assignments now. If yes, then the user is asked to enter the physical device corresponding to the logicals UTL, DEF, EXE, UT, and OBJ. (SRC must already have been manually assigned.)

3. Asks the user if he wishes to create the package subroutine library, UTIL.OBJ, and, if so, if he wishes to delete the .MAN subroutines' object code (created with a .TMl extension).

4. Asks the user if he wishes to compile the package source code, and, if so, if he wishes to delete the object code (created with a .TMP extension).

5. Asks the user if he wishes to link the package programs.

6. If the user elects not to create the UTIL.OBJ library and wishes to link the package, the build batch checks to be sure that both the UTIL.OBJ library and GSORT.SAV exist. If they don't, then the build batch displays a message and halts.

7. Executes TSXCMP.MAN if yes answered to #3 above. This compiles the .MAN subroutines and stores the object code under the extension .TM1. It then creates the UTIL.OBJ library from the .TM1 object code. Both the object code and the library are placed in the UT: logical device.

8. Executes TSXCMP.xxx if yes answered to #4 above. This compiles the package source code and stores the object code under the extension .TMP in the OBJ: logical device.

9. Executes TSXLNK.xxx if yes answered to #5 above. This links the package object code with the subroutines and library created by TSXCMP.MAN and places the executable code in the EXE: logical device.

10. Deletes those object files that were requested to be deleted.

To execute the CTS-300 BATCH command file, follow these steps:

1. Manually enter the logical device assignments referred to in the previous section.

2. All device handlers required must be loaded. Assuming logical subsets are used, you would type:

```
LOAD LP,BA,LD
```

   Any other physical devices being used, such as DL, DM or RM would also be loaded with a similar command. Logical devices such as SRC and UT do not have to be loaded.

3. Assign the printer the logical device LOG by typing:

```
ASSIGN LP: LST:
ASSIGN LP: LOG:
```

4. The BATCH processor cannot be run from XMTSD, but must be run from the RT-11 monitor. Invoke it by typing:

```
R BATCH
```

5. It will respond with an "*" prompt. Ensure the printer is on line. Then type:

```
SRC:CTSBLD.xxx
```

The printer will record the progress of the build batch. When complete, the words "END BATCH" will appear on the terminal.

The CTSBLD.xxx command file itself performs exactly the same steps as the TSXBLD.xxx command file, except it calls command files named CTSCMP.MAN, CTSCMP.xxx and CTSLNK.xxx.

Setting Up the Sample Data Base

If you have not already done so, you should now separate the sample data base as described in item #6 in the prior section entitled "Setting Up Logical Assignments and Directories". The working copy of the sample data base can be on the same device as the master set. Each set (both master and working copies) requires 3000 blocks (1.5 Mb) of disk space.

If you wish to set up the sample data base provided with your source shipment, execute the RESTOR.ALL command file. It is shipped with your utility source code. Execute it under either TSX-Plus or CTS-300 by typing:

```
IND IN:RESTOR.ALL
```

where "IN:" is the device containing the master set of data files.

This command file performs the following steps:

1. Prompts the operator for the device location of the master set of sample data files sent with the shipment. These are the files with the .MC and .DDE file extensions. This device is then assigned the logical name IN:.

2. Prompts the operator for the device location of the working set of sample data files. This device is then assigned the logical name OUT:.

   CAUTION: If you are running under TSX-Plus/DBL, this device should not contain data files for any other company. The reason is that DBL ISAM files all contain the same extension, regardless of the Company code.

3. Asks the operator if he/she wishes to restore the security system data files. If the answer is yes, prompts the operator for the device location of the utility programs and assigns it the logical name UT:.

4. Asks the user if the demo is running under DBL/TSX-Plus or CTS-300.

5. Executes the following commands:

```
     COPY/NOPROTECTION IN:*.MC OUT:*.MCB
     COPY/NOPROTECTION IN:*.DDE UT:*.DDF

For TSX-Plus/DBL
     COPY/NOPROTECTION IN:*.TSX UT:*.MCM
     COPY/NOPROTECTION IN:*.TS1 UT:*.MC1

For CTS-300
     COPY/NOPROTECTION IN:*.CTS UT:*.MCB
     COPY/NOPROTECTION IN:*.CT1 UT:*.MC1
```

If a working installation already exists, care should be taken in running this command file. Any changes that have been made to either sample data files or security files will be reversed.

User's Manual Installation Instructions

The remaining installation of MCBA's executable code is performed using the installation instructions contained within the User's Manual for this package.

This page intentionally left blank.

MINIMUM SYSTEM REQUIREMENTS

Hardware Requirements

The RSTS/E implementation of the MCBA system is designed for the following minimum configuration:

1. Digital PDP-11 or Micro-11 computer
2. 256 kilobytes of main memory (recommend 1 megabyte or more)
3. 1600 bpi tape drive or RL02 hard disk drive (for media transfer)
4. 3-4 megabytes free disk space for each installed MCBA package
5. 132-column printer

MCBA programs require a maximum of 64 kilobytes of memory (including the runtime system and resident library) plus the memory required for the operating system. Therefore, the minimum main memory of 256 kilobytes is not recommended for more than 2 or 3 users.

Notes on Disk Space Requirements

There are a total of 15 application packages plus one common utilities package in the MCBA system. The actual disk storage requirements for each package varies. The entire system (consisting of executable code, source code, data files and system work files) would require 50-70 megabytes of hard disk needs. Exact figures for each package can be obtained from the User's Manual section entitled "Disk Space Required for Programs and Data". For a rough estimate of space required, use the following information:

1. Each application package (e.g. utilities, I/M, COP, G/L) requires an average of 2.0 megabytes for the executable code. The smallest package, Bill of Material Processor (BOMP), requires 0.5 megabytes. The largest package, Payroll, requires 3.0 megabytes.

2. The source code for each package requires an average of 0.6 megabytes of disk storage per package.

   NOTE: The source code does not have to be resident on the disk in order to run the programs.

3. Data files will require an additional 0.5 to 1.0 megabytes per package. However, because of the highly variable needs of individual companies, this should only be used as a rough estimate.

4. Sort work space should be reserved for the system and should be approximately 30% of your largest file.

5. The MCBA system allows reports to be permanently spooled to disk files for later printing. If you intend to make use of this feature, space will have to be allocated for these files. See the sections in the User's Manual

entitled "Spoolers - Special Notes" and "Company File Maintenance" for a full explanation of the MCBA Spooler and its space requirements.

Software Requirements

The Release 7.0 MCBA system is supported on RSTS/E version 8.0 (patch level G) or later. It is supported on RSTS DIBOL version 5.1a or later.

Any attempt to run MCBA packages under earlier versions than those specified may cause errors and is not supported.

Attempts have been made in the past to run MCBA packages under the Micro RSTS environment. Unfortunately, Digital does not provide a version of DIBOL for Micro RSTS and such attempts, although eventually successful, required changes in the MCBA software. MCBA cannot directly support any software problems resulting from running in the Micro RSTS environment.

RSTS and DIBOL Sysgen Considerations

This section deals with those sysgen options which are required (or recommended) for use with the MCBA system.

RSTS V8.0

1. Both RSX and DCL runtimes must be generated. DIBOL requires RSX and there are several command files and batch scripts which require DCL to properly run. RSX and DCL should be added as resident runtimes in your startup command file START.CTL.

2. When RSTS is first started, certain parameters are set and detached jobs are started via your START.CTL startup file. The following jobs need to be inserted (if not there already) into your START.CTL file.

   OPSER
   QUEMAN
   SPOOL
   BATCH
   ERRLOG (optional but recommended)
   MESMAN

   SWAP MAX should be 32KW
   JOB MAX should be at least the number of terminals + 10.

3. Terminal characteristics - your terminal should be set to VT52 or VT100 (under V9.0, VT200 is also supported). Terminal width should be 134 or greater (for display of reports in 132 column mode).

4. The full spooling package (as opposed to the micro RSTS spooling package) is recommended for DIBOL installation.

5. The following programs (with extension .TSK) must reside on SY:[1,2]

   PIP
   TKB
   LBR
   ATPK

Dibol 5.1a

1. DIBOL version 5.1a must be installed. You must have at least installed the DMS version of DIBOL.

2. DIBOLD, the DMS resident library, must be installed (via UTILTY). This is usually done for you automatically when DIBOL is installed, but you must put it into your startup command file START.CTL.

3. The system-wide logical LB: must be assigned to the library directory containing the libraries DMSUSL.OLB and DMSOSL.OLB.

4. The following programs (with extension .TSK) must reside on SY:[1,2]

   DICOMP
   DMSORT

   DMSORT.TSK must be a privileged program (set protection to this file to 232) to allow interface to the Message Management utility.

5. ISAM must be supported.

HOW TO INSTALL THE PACKAGE

The programs and sample data files comprising your source shipments have been put onto your distribution media via the RSTS/E PIP utility (or via the DCL COPY command.

Each source license distribution contains:

1. One Software Reference Manual for each package ordered
2. One User's Manual for each package ordered
3. Source code consisting of:
   a. Program source code for each package
   b. Program source code for the system utility programs and subroutines
   c. A record definition library
4. Sample data files for each package ordered
5. Sample security system data files

In addition, if this is your first MCBA order, you will also receive a Utilities Software Reference Manual.

To successfully complete the following installation instructions, you must be logged into a privileged account. This will give you the system privilege necessary for some of the steps, including building the packages.

The Installation Instructions below are applicable to all packages. Reference is made throughout to a "two- or three-character package code". These are defined in the following table:

Package Code Table

Package Name	Package Code
MCBA Utilities	MAN and UTL
Accounts Payable	AP
Accounts Receivable	AR
General Ledger	GL
Payroll	PR
Fixed Assets and Depreciation	AD
Customer Order Processing	COP
Purchase Order and Receiving	POR
Inventory Management	IM
Bill of Material Processor	BMP
Shop Floor Control	SFC
Job Costing	JC
Standard Product Costing	SPC
Standard Product Routing	SPR
Labor Performance	LBP
Material Requirements Planning	MRP

The distribution media is divided into subdirectories which are RSTS accounts. The following account naming convention has been followed and is recommended for your installation.

All executable programs are on accounts [200,x]
All sample data files are on the account [201,1]
All source files are on accounts [202,x]
All record definition files are on accounts [203,x]
All CNL (compile and link) files are on accounts [204,x]
Temporary object files will be placed on account [200,20] for all packages

where 'x' is a different number for each package as defined below (the 2-3 character code is the package code as defined earlier):

UTL + MAN = 0
AR = 1
IM = 2
BMP = 3
COP = 4
AP = 5
GL = 6
PR = 7
SFC = 8
JC = 9
SPC = 10
SPR = 11
MRP = 12
POR = 13
LBP = 14
AD = 15
RW = 16

For example, the Accounts Receivable source code would reside on account number [202,1] on your distribution media and the record definition files for the Customer Order Processing package would reside on account [203,4].

In addition, the programs within each individual package have a unique file extension which corresponds to the above 2-3 character package code. For example, all Accounts Receivable source files have the extension ".AR" and all Customer Order Processing files have the extension ".COP". A few more extensions exist.

1. All record definition files have the extension ".DEF"
2. All utility data files have the initial extension ".DDE" and are later renamed to extension ".DDF".
3. All sample data files (except for the ORDHDR and ORDLIN ISAM Index files) have the extension ".MC" which later is copied onto extension ".MCB". The ORDHDR and ORDLIN have two files each; one with extension ".MC" and the other ".M1", which later become ".MCB" and ".MC1", respectively.

Restoring from Distribution Media

The MCBA distribution media should not be used as work disk(s). Your first step, therefore, is to create a duplicate or backup copy of your distribution

media, using whatever method is standard for your installation. Be sure your distribution media is write protected.

The next step depends upon whether you intend to use the recommended MCBA directory structure or not. If you do not, then you must manually create accounts on your destination disk and then copy all files from the distribution media onto those accounts individually. If you do use the recommended directory structure, proceed with the following numbered steps:

1. Make the following logical assignments as system logicals (via UTILTY for RSTS V8.0 or via ASSIGN/SYSTEM for RSTS V9.0):

   IN: = the device of the distribution media (for example DL1: or MT0:)
   OUT: = the destination device (ex: DM0:, or even SY:)

   these logicals MUST be made before continuing. They should not contain account numbers. For example, to assign IN to tape drive MT0: and OUT to disk drive DR0: (you will substitute the appropriate drives for your site), type:

   For RSTS V8.0

```
RUN SY:[1,2] UTILTY
ADD LOGICAL MT0:IN
ADD LOGICAL DR0:OUT
^Z
```

   For RSTS V9.0

```
ASSIGN/SYSTEM MT0: IN
ASSIGN/SYSTEM DR0: OUT
```

2. If you have a multi-volume shipment, you must first mount the disk or tape that contains the Utilities package.

3. If you have RSTS V8.0 installed, you will use ATPK to run the command file. Type:

```
@IN:[202,0]INSTV8.UTL
```

   If you have RSTS V9.0 installed, you will use DCL command language to run the command file. Type:

```
@IN:[202,0]INSTV9.UTL
```

   The command file will automatically create the account structure on the OUT device.

4. For each distribution volume (including the volume you have mounted for the previous step), mount that volume and type:

```
COPY/PROTECTION=0  IN:[*,*]*.* OUT:[*,*]
```

5. Once the copies are completed, you may compare your shipment directories to the contents of the LSTSRC.xxx and LSTDEF.xxx files contained in each package source directory (where "xxx" is the 2-3 character package code). Each source directory (as defined above) should contain the same files as contained in the LSTSRC file and each file definition directory (again as defined above) should contain the same files as contained in the LSTDEF file. For example, OUT:[202,1]LSTSRC.AR lists all files which belong in account [202,1] and OUT:[202,1]LSTDEF.AR lists all files which belong in account [203,1].

6. If you plan to perform any package modifications involving changing file sizes, you may want to create a new directory and move all record definition files from all packages into that directory. Each package in the distribution is accompanied by record definition files which reside in their own unique directories. Many record definitions are duplicated across packages. If you put all record definition files for all packages into one directory (and assign it the logical "DEF" as defined below), then you will not only eliminate duplicate files and save space, you will not need to make duplicate changes to the many copies of the record definition in the various directories. This directory will get large, however, as there are more than 500 record definition files for the entire MCBA system. Also, the logical "DEF" is assigned physically in each package build batch and would need to be changed by you before you would be able to access the single "DEF" directory with the builds.

Setting Up Logical Assignments

MCBA provides you with build batches that will compile and task build your source code into executable programs. These build batches use the logical names listed below:

SRC = The location of the package source code about to be built.
UTL = The location of the MCBA utility program and subroutine source code about to be built, plus the utility data files with extension ".DDE".
DEF = The location of the record definition files for the source code about to be built.
OBJ = The intended location of the package temporary object code (all except the utility subroutines are created with the ".TMP" extension during compilation and then deleted after the task build is completed).
xxx = The package 2-3 character code which becomes the intended location of the package's executable code after building. Ex: "AR" will contain the AR package executable code.
UT = The intended location of the MCBA utility programs executable code, the utility data files (renamed to extension ".DDF"), and the utility object libraries UTIL.OLB and UTIL20.OLB which are used during task building of all packages.
CNL = The intended location of the CNL (compile and link) command files. These files are an option for building the package under ATPK. This assignment is not needed if you intend to use the BATCH method of building the package.

These logical names are made as system-wide logicals (using the UTILITY program and ADD LOGICAL command for RSTS V8.0 or the ASSIGN/SYSTEM command for RSTS V9.0). If you use the recommended directory structure, the following assignments should be made:

```
UTL = dev:[202,0]
UT  = dev:[200,0]
OBJ = dev:[200,20]
```

where "dev:" is the disk device on which you placed your source. The disk containing the OBJ directory should have at least 5000 empty blocks available.

The logical xxx (such as "AR" or "COP") depends upon the package(s) being built and should correspond to the recommended directories for the package executable code defined at the beginning of this section. For example, (and using "dev" for the disk device as above) if you were building the AR package, you would set:

```
AR = dev:[200,1]
```

Of course, any other package would use a directory structure similarly (differing by the last digit only).

Two logicals, SRC and DEF are assigned within the build files themselves as local variables (so that multiple builds can be chained together without reassigning SRC and DEF manually - all other logicals can be shared). The following commands are contained in the BLDxxx.xxx batch file for (as an example) the AR package (i.e. BLDAR.AR):

```
ASSIGN DSK:[202,1] SRC
ASSIGN DSK:[203,1] DEF
```

Note that these assignments correspond to the directory structure defined at the beginning of the installation procedure. Note also the use of a new device logical "DSK". This logical must be assigned to the device (device only, without the account) containing the source code and record definition files for the package. It must be a system logical. So if, for example, you are using device DRO as the device containing the source and record definitions, you would type

(for RSTS V8.0)

```
RUN SY:[1,2]UTILTY
ADD LOGICAL _DRO: DSK
^Z
```

(for RSTS V9.0)

```
ASSIGN/SYSTEM _DRO: DSK
```

Finally, there are two more logical names used by the build batches. They are LB and SY. Both are normally setup at system startup time. SY is the system disk and LB is the location of various library files (such as the DIBOL library DMSUSL.OLB).

Compiling and Task Building System Utility Programs and Subroutines

The MCBA system has one set of utility programs and two different sets of subroutines. It also has a set of security system data files.

All the application packages are built with the set of subroutines that have a .MAN extension on the source files. These subroutines are then put into the library file UTIL.OLB.

The system utility programs are programs that are shipped with each source shipment and demo. These programs include the master system menu and a number of utility programs that can be run from the back menu of the master menu. The programs and subroutines that comprise this group all have the file extension .UTL. The subroutines with the .UTL file extension are an advanced set of subroutines that incorporate enhancements over the .MAN subroutine. They are compiled into the object library called UTIL20.OLB. Build command files are provided with your software to create both of the subroutine libraries and the executable system utility programs.

If this is not your first installation of a Release 7.0 MCBA package, then it is likely that you have already built the utility programs and subroutines and need not rebuild them (you can skip to the section entitled Compiling and Task Building Packages). If this is your first installation, then perform the following steps:

1. To build the .UTL utilities programs and subroutines, type the command

   SUBMIT UTL:BLDUTL.UTL

   This will run the build command file using BATCH as a detached job. Once complete, a log file BLDUTL.LOG will be written to the disk with the results of the build.

   The BLDUTL.UTL batch file does the following:

   1. Assigns the logical name DEF to DSK:[203,0]
   2. Copies the .DDE system data files from the UTL directory to the UT directory with the extension .DDF and with the protection 63 (no access by anyone). This protection is for external access only, since the MCBA packages are built with special privilege (232).
   3. Compiles the .UTL subroutines, placing the temporary object files on the directory OBJ with the file extension .TM2.
   4. Creates the library UT:UTIL20.OLB from the .TM2 object files.
   5. Compiles the system utility programs, placing the temporary object files on the directory OBJ with the file extension .TMP.
   6. Task builds the system utility programs, placing the executable programs on the UT directory with the file extension .TSK.
   7. Sets the protection code for the system utility programs to (232) (special privilege).
   8. Deletes all .TM2 and .TMP object files from the OBJ directory.

2. To build the .MAN SUBROUTINES (into the UTIL.OLB library), type the command

   SUBMIT UTL:BLDMAN.MAN

This will produce the BLDMAN.LOG log file upon completion. (Since BATCH is being used, both of the above commands may be run at the same time if you want to queue the batch for overnight processing.) Total execution time of the build is 1-2 hours, depending upon your system. Upon completion, check the log file(s) for any errors or warnings (there should be none).

The BLDMAN.MAN batch file does the following:

1. Assigns the logical name DEF to DSK:[203,0]
2. Compiles the .MAN subroutines, placing the temporary object files on the directory OBJ with the file extension .TM1.
3. Creates the library UT:UTIL.OLB from the .TM1 object files.
4. Deletes all .TM1 object files from the OBJ directory.

Compiling and Task Building Packages

Each package has its own build batch file. It assumes that you have already built the utilities and subroutines as defined in the previous steps. You may, if you wish, queue up a number of the package build batches at once. Since the logicals UT, UTL, and OBJ are the same for each package, the logicals SRC and DEF are assigned internally by each batch, and the "xxx" package logical is unique to each package. You must NOT submit the batches for simultaneous batch processing. Instead, the builds can be sequential (that is, one starts after the previous build finishes). This is because each build deletes all .TMP files on the OBJ: directory upon completion of the build. This would delete any .TMP's created by another build batch as well. (Also, system performance would degrade seriously if more than one build were to be run simultaneously.)

For each package that you intend to build, make sure that the DSK and xxx (where "xxx" is the 2-3 character package code) logicals are properly assigned as defined above. Also, make sure that UT and OBJ have the same assignments used in the build of the utilities above. Then type the command

```
SUBMIT DSK:[202,z]BLDxxx.xxx
```

where "z" is the number corresponding to the package being built (as defined at the beginning of this section) and "xxx" is in both cases the 2-3 character package code for the package being built. The command file will be run detached using BATCH and will write a log file to the disk (on your default directory) with the name BLDxxx.LOG once completed. Each package takes between 1 and 4 hours to build, depending upon your system and the size of the package. Upon completion, check the log file for any errors or warnings (there should be none).

The BLDxxx.xxx batch file does the following:

1. Assigns the logical name DEF to DSK:[203,x] and SRC to DSK:[202,x].
2. Compiles the package programs, placing the temporary object files on the directory OBJ with the file extension .TMP.
3. Task builds the package programs, placing the executable programs on the "xxx" directory with the file extension .TSK.
4. Sets the protection code for the package programs to (232) (special privilege).
5. Deletes all .TMP object files from the OBJ directory.

CNL Build Procedure

The previous release of the MCBA RSTS/E DIBOL system provided build Compile and Link (abbreviated CNL) command files. These are files that run under ATPK and may be run all at once or each separately to compile and link the package (or individual programs). Part of that system is still provided with the release 7.0 system. However, it is not supported beyond RSTS V9.0 since ATPK is not to be supported either after that version.

The CNL builds use the same logical assignments (i.e., UT, UTL, OBJ, SRC, DEF, DSK, and xxx) as the build batch files documented above, with the following modifications:

1. The SRC and DEF logicals must be assigned as system logicals the same as with UT, UTL, OBJ, etc. No internal assignments are made in CNLs. They should be assigned to the same directories as above, just make the assignments as system logicals.

2. The logical CNL must also be assigned. This is the directory containing the CNL files for each package and is DSK:[204,x] where DSK is assigned above and "x" is the digit corresponding to the package as defined at the beginning of the installation procedure. So, for example, to assign CNL for the AR package, you would type:

   (for RSTS V8.0)

   ```
   RUN SY:[1,2]UTILTY
   ADD LOGICAL DSK:[204,1] CNL
   ^Z
   ```

   (for RSTS V9.0)

   ```
   ASSIGN/SYSTEM DSK:[204,1] CNL
   ```

   Then, execute an individual CNL by typing:

   ```
   @CNL:prgnam.CNL
   ```

where "prgnam" is the name of the executable program in that package that you want to compile and link.

If you want to build the entire package, type

```
@CNL:xxxBLD.CMD
```

where "xxx" is the package code for that package. This command will execute individually the "@CNL:prgnam.CNL" commands for all programs within the package.

Each CNL command file does the following:

1. If it is a sort file, runs the UT:GSORT program and creates the sort control file. Then compiles the sort control file together with UTL:SORT.MAN, creating a temporary object file on the OBJ directory.

2. If it is not, a sort file compiles the program (plus any package subroutines which are used by the program) from the SRC directory and places the temporary object files on the OBJ directory.

3. Task builds the program, placing the executable program on the "xxx" directory with the extension .TSK. If the task build requires an overlay structure, the appropriate overlay description file (with extension .ODL) is read from the SRC directory.

4. Deletes any .TMP files created on the OBJ directory during the build.

5. Sets the protection code for the executable program to 232.

Setting Up the Sample Data Base

If you followed the recommended directory structure, your sample data files reside upon directory account [201,1] with the extension ".MC". These files must be copied to extension ".MCB" before they can be used. RESTOR.xxx command files have been provided for this process. All RESTOR.xxx files use the internal logicals IN and OUT for the copy. Note that these two logicals were used previously for other purposes and MUST be reassigned before proceeding, since it is certain that they are different. IN and OUT will refer to the disk:[account] for the source and destination of the copy. The disk is that device you originally assigned to OUT during the earlier part of the installation. The account for IN is [201.1] and the account for OUT may also be [201,1] unless you wish to place the data files on some other directory. The following examples use the disk DR0: as the drive you originally assigned to OUT. If your drive is different, be sure to substitute its name whenever DR0: is used.

If you are using RSTS V8.0, you would use UTILTY to make the assignments

```
RUN SY:[1,2]UTILTY
REMOVE LOGICAL OUT
REMOVE LOGICAL IN
ADD LOGICAL DR0:[201,1]IN
ADD LOGICAL DR0:[201,1]OUT
^Z
```

If you are using RSTS V9.0, you would use the commands

```
ASSIGN/SYSTEM DR0:[201,1] IN
ASSIGN/SYSTEM DR0:[201,1] OUT
```

Then, once the assignments have been made, you have the choice of restoring all data files for all packages, or else restoring only the data files for a select package. The command file RESTOR.ALL is for all packages. To use it, type the following.

```
RUN SY:[1,2]PIP
@IN:RESTOR.ALL
^Z
```

The package command file RESTOR.xxx (where "xxx" is the 2-3 character package code) is for restoring only those data files which are unique to that package (including a RESTOR.UTL command for the MCBA system utility .DDF data files). To execute each one of them, type

```
RUN SY:[1,2]PIP
@IN:RESTOR.xxx
^Z
```

where "xxx" is again the 2-3 character package code for the desired package.

NOTE: The above RESTOR commands must be run from privileged accounts, otherwise you may get protection violation errors.

This page intentionally left blank.

RELEASE NOTES - CUSTOMER ORDER PROCESSING, VAX-11 DIBOL RELEASE 7

Release 7 of MCBA's Accounting, Distribution and Manufacturing system addresses problems created by Digital's VAX-11 DIBOL, Release 2.1, as well as providing enhancements to the Customer Order Processing package itself.

Enhancements for the COP Package

1. Terms discounts are now interactively maintainable by the operator, instead of being hard-coded within the source code. Associated with each Terms code is an invoice due date, a terms discount percentage, and a discount due days. These are maintained in the Accounts Receivable package and are accessed wherever applicable in COP.

2. A new Tax Codes file has been created that allows up to three different taxing authorities in A/R. This data provides default sales tax amounts in the billing cycle. Reports have been changed to show these distributions.

3. A new Ship-to file has been added that is interactively maintained by the user. Each customer now has a virtually unlimited number of ship-to addresses.

4. A new Ship-via file has been added that is interactively maintained by the user.

5. The above enhancements make it unnecessary to make any source code modifications in order to run the COP package standardly without modifications.

6. All reported problems in function have been addressed.

Enhancements for All Packages

1. DIBOL, Release 2.1 has a new subroutine WAIT. This serves a different purpose than MCBA's subroutine WAIT. MCBA's packages will run effectively as they are currently linked, but will prohibit use of DEC's new subroutine for programmers wishing to customize our software. MCBA has renamed its WAIT subroutine to WATE and globally changed all occurrences of XCALL WAIT to XCALL WATE in its source code.

2. DIBOL, Release 2.1, has changed the algorithm by which their subroutine TNMBR returns a value when XCALLed. The new value can have a value as high as 997, as opposed to the previous high of 99. The work to expand the receiving field from a D2 to a D3 was minor. Unfortunately, MCBA's security system is heavily terminal-dependent, and records in the MESARA file are accessed directly by terminal number. Spooled reports also are stored in files that have the terminal number encoded in the file name.

Substantial recoding would have had to be done in order to address this directly and would have gone counter to MCBA's attempts to produce DIBOL code that is portable across operating systems. It would also have made necessary conversion programs to retain all spooled reports.

Instead, we have created a new subroutine, TTNO, and substituted all occurrences of XCALL TNMBR with XCALL TTNO. We have also created a new file, TTXREF, that provides a cross reference between the number supplied by TNMBR and the terminal specific information stored in the MESARA file.

This cross reference is transparent to the user. Its only effect is that MCBA terminal numbers are assigned in order of first log on to the MCBA system and are not tied in any significant manner to the terminal number by which the VAX operating system recognizes each terminal.

3. The Software Reference Manual and User's Manual have been completely reformatted for ease of use. In the past, the User's Manual was merely an abstract from the Software Reference Manual. Now they are two completely different documents.

   The Software Reference Manual details how to produce executable code from source code and contains information on an application by application basis useful to the programmer or DP professional in modifying or simply understanding how the MCBA system is constructed. The Technical Notes section has been expanded and broken into two sections: one system specific and the other package specific.

   The User's Manual details how to install an MCBA package once executable code has been obtained. Front end installation instructions are more explicit. All operator instructions include sample screens that depict exactly what the user will see on the screen with sample data. Data entry field descriptions have been enhanced and put in a new format. And an index has been provided for ease in locating information on specific subjects.

4. Code has been standardized to prepare for use of DIBOL-83 language enhancements in all future enhancements and patches.

RELEASE NOTES - CUSTOMER ORDER PROCESSING, DIBOL RT-11 RELEASE 7

Release 7 of MCBA's Accounting, Distribution and Manufacturing system addresses problems created by Digital's DIBOL-83, as well as provides enhancements to the Customer Order Processing package itself. In addition the Security System utilities have been significantly enhanced.

Enhancements for the COP Package

1. Terms discounts are now interactively maintainable by the operator, instead of being hard-coded within the source code. Associated with each Terms code is an invoice due date, a terms discount percentage, and a discount due days. These are maintained in the Accounts Receivable package and are accessed wherever applicable in COP.

2. A new Tax Codes file has been created that allows up to three different taxing authorities in A/R. This data provides default sales tax amounts in the billing cycle. Reports have been changed to show these distributions.

3. A new Ship-to file has been added that is interactively maintained by the user. Each customer now has a virtually unlimited number of ship-to addresses.

4. A new Ship-via file has been added that is interactively maintained by the user.

5. The above enhancements make it unnecessary to make any source code modifications in order to run the COP package standardly without modifications.

6. All reported problems in function have been addressed.

Enhancements for All Packages

1. Logging in and out of the MCBA system has been speeded up from 10-20 seconds to 1-3 seconds.

2. The back menu has been reorganized to display applications in their most common order of use.

3. The user now has an interactive option of using either the MCBA DIBOL sort, Digital's new macro sort, or S & H's RTSORT.

4. Control-C abort is now interactively maintainable.

5. The CTS-300 spooler is now used. This will significantly speed up processing under CTS-300.

6. The security system setup has been enhanced. File access codes and device assignments can now be set by package as well as individually.

7. The Software Reference Manuals and User's Manuals have been completely reformatted for ease of use. In the past, the User's Manual was merely an abstract from the Software Reference Manual, Now, they are two completely different documents.

   The Software Reference Manual details how to produce executable code from source code and contains information on an application by application basis useful to the programmer or DP professional in modifying or simply understanding how the MCBA system is constructed. The Technical Notes section has been expanded and broken into two sections - one system specific and the other package specific.

   The User Manual details how to install an MCBA package once executable code has been obtained. Front end installation instructions are more explicit. All operator instructions include sample screens that depict exactly what the user will see on the screen with sample data. Data entry field descriptions have been enhanced and put in a new format. And an index has been provided for ease in locating information on specific subjects.

8. DIBOL-83 has a new subroutine WAIT. This serves a different purpose than MCBA's subroutine WAIT. MCBA's packages will run effectively as they are currently linked, but will prohibit use of DEC's new subroutine for programmers wishing to customize our software. MCBA has renamed its WAIT subroutine to WATE and globally changed all occurrences of XCALL WAIT to XCALL WATE in its source code.

9. In order to maintain source code compatability with other operating system versions of DIBOL-83, all occurrences of an external call to the TNMBR subroutine have been replaced with a new MCBA subroutine, TTNO.

10. Code has been standardized to prepare for use of DIBOL-83 language enhancements in all future enhancements and patches.

RELEASE NOTES - CUSTOMER ORDER PROCESSING, DIBOL RSTS/E RELEASE 7

Release 7 of MCBA's Accounting, Distribution and Manufacturing system addresses problems created by Digital's DIBOL-83, as well as provides enhancements to the Customer Order Processing package itself. In addition the Security System utilities have been significantly enhanced.

Enhancements for the COP Package

1. Terms discounts are now interactively maintainable by the operator, instead of being hard-coded within the source code. Associated with each Terms code is an invoice due date, a terms discount percentage, and a discount due days. These are maintained in the Accounts Receivable package and are accessed wherever applicable in COP.

2. A new Tax Codes file has been created that allows up to three different taxing authorities in A/R. This data provides default sales tax amounts in the billing cycle. Reports have been changed to show these distributions.

3. A new Ship-to file has been added that is interactively maintained by the user. Each customer now has a virtually unlimited number of ship-to addresses.

4. A new Ship-via file has been added that is interactively maintained by the user.

5. The above enhancements make it unnecessary to make any source code modifications in order to run the COP package standardly without modifications.

6. All reported problems in function have been addressed.

Enhancements for All Packages

1. Logging in and out of the MCBA system has been speeded up from 10-20 seconds to 1-3 seconds.

2. The back menu has been reorganized to display applications in their most common order of use.

3. The user now has an interactive option of using either the MCBA DIBOL sort or Digital's macro sort utility.

4. Control-C abort is now interactively maintainable.

5. The security system setup has been enhanced. File access codes and device assignments can now be set by package as well as individually.

6. The Software Reference Manuals and User's Manuals have been completely reformatted for ease of use. In the past, the User's Manual was merely an abstract from the Software Reference Manual, Now, they are two completely different documents.

   The Software Reference Manual details how to produce executable code from source code and contains information on an application by application basis useful to the programmer or DP professional in modifying or simply understanding how the MCBA system is constructed. The Technical Notes section has been expanded and broken into two sections - one system specific and the other package specific.

   The User Manual details how to install an MCBA package once executable code has been obtained. Front end installation instructions are more explicit. All operator instructions include sample screens that depict exactly what the user will see on the screen with sample data. Data entry field descriptions have been enhanced and put in a new format. And an index has been provided for ease in locating information on specific subjects.

7. DIBOL-83 has a new subroutine WAIT. This serves a different purpose than MCBA's subroutine WAIT. MCBA's packages will run effectively as they are currently linked, but will prohibit use of DEC's new subroutine for programmers wishing to customize our software. MCBA has renamed its WAIT subroutine to WATE and globally changed all occurrences of XCALL WAIT to XCALL WATE in its source code.

8. In order to maintain source code compatability with other operating system versions of DIBOL-83, all occurrences of an external call to the TNMBR subroutine have been replaced with a new MCBA subroutine, TTNO.

9. Code has been standardized to prepare for use of DIBOL-83 language enhancements in all future enhancements and patches.

GENERAL MCBA SYSTEM APPROACH

MCBA accounting and manufacturing packages are completely interactive, operator oriented, with stress on ease of operator use, full audit trails and completeness of the accounting and manufacturing functions. Each application package is driven (that is, accessible) by a main package menu (e.g. the A/R menu, or BOMP menu, etc.) which is arrived at via the Master menu, also called the MCBA Manufacturing/Distribution menu. The Master menu lists all packages and selecting one will take the operator to that package's main menu. This main menu lists all the applications of the package that would be used in normal everyday processing. It is a separate program which makes the applications of the particular package accessible to the user. All applications appear on this menu except for a few isolated special programs which are not part of the usual running of the package. Examples of such programs would be initialization programs which are only run at the end of the year to close out the books and prepare for the new year.

Each package is modular in design. That is, each separate function in the package is done by a separate program (rather than having one large program that does everything).

The typical flow would be:

The operator runs the Master menu program (MSMENU) and from the Master menu, he selects the package he wishes to use. The Master menu program then chains to (transfers control to) the package's main menu. The operator then selects an application from the package menu. Very often, the program which drives (controls) a particular selected application is another menu program, which displays its own menu for that application, such as ADD, CHANGE, DELETE, PRINT-OUT. The user then chooses which function of the application he wishes to perform, and the application menu program chains to another program which performs this function. This is basically what is meant by the modular approach.

Another technique that is used extensively in MCBA packages is the transaction file approach. A typical package usually has one main file which holds the most important data of the package. In A/R this is the A/R Open Item file (containing the accounts of all customers); in G/L it is the Year-to-Date General Ledger file, showing all account activity for the current year. Rather than allowing the user to make changes to these very sensitive files directly, any new data to be recorded in them must be done via a temporary transaction file. The data is entered, via the CRT screen, into the temporary file, within which it can be added to, changed, or deleted at will. None of this affects the permanent file yet. The data in the transaction file can be printed out in the form of an edit list as many times as desired, and thoroughly inspected for correctness and completeness. When the data in the temporary transaction file is determined to be complete and correct, the user then posts this data to (updates) the permanent main file (by selecting the POST selection on the menu

for this application). The data is then transferred from the temporary file to the permanent file. A Transaction Register or Journal is printed (e.g., Sales Journal) which is the final hard-copy audit trail document; and finally, the temporary file is completely cleared out.

COMPILING AND LINKING

Compiling

Compilation of MCBA programs is normally accomplished by means of command files. Each command file compiles all programs and subroutines associated with one MCBA package.

Each operating system version of DIBOL packages has a different format, but the formats within each operating system are consistent.

For VAX/VMS, the standard format is:

```
$ DIBOL/NOSTANDARD/OPTIMIZE/OBJ=OBJ:prgnam.TMP SRC:prgnam.xxx
```

For CTS-300, the standard format is:

```
.R DICOMP
*OBJ:prgnam.TMP=SRC:prgnam.xxx/O
```

For TSX-Plus, the standard format is:

```
.R DBL
*OBJ:prgnam.TMP=SRC:prgnam.xxx/R:DBG
```

For RSTS/E, the standard format is:

```
RUN SY:[1,2]DICOMP
*OBJ:prgnam.TMP=SRC:prgnam.xxx/O
```

In all the above, the following references are standard: "OBJ" refers to the logical directory containing the temporary object files with the .TMP extension. "SRC" refers to the logical directory containing the source code for the "xxx" package, where "xxx" is the two- or three-character package code. "prgnam" refers to any program or subroutine.

Utilities and subroutines use a slightly different format. There are two for each operating system, one for the .MAN subroutines and another for the .UTL programs and subroutines.

For VAX/VMS:

```
$ DIBOL/NOSTANDARD/OPTIMIZE/OBJ=OBJ:prgnam.TMP UTL:prgnam.MAN
$ DIBOL/NOSTANDARD/OPTIMIZE/OBJ=OBJ:prgnam.T20 UTL:prgnam.UTL
```

For CTS-300:

```
.R DICOMP
*UT:prgnam.TM1=UTL:prgnam.MAN/O
*UT:prgnam.TM2=UTL:prgnam.UTL/O
```

For TSX-Plus:

```
.R DBL
*UT:prgnam.TM1=UTL:prgnam.MAN/R:DBG
*UT:prgnam.TM2=UTL:prgnam.UTL/R:DBG
```

For RSTS/E:

```
RUN SY:[1,2]DICOMP
*OBJ:prgnam.TM1=UTL:prgnam.MAN/O
*OBJ:prgnam.TM2=UTL:prgnam.UTL/O
```

Utility Libraries

All utility subroutines are compiled into libraries. Naming conventions are the same across all operating systems:

1. .MAN subroutines are compiled into the library UTIL.

2. .UTL subroutines are compiled into the library UTIL20.

Library file extensions conform to the default for the operating system. For VAX/VMS and RSTS/E, it is .OLB. For CTS-300 and TSX-Plus, it is .OBJ.

Linking

Separate build command files accomplish the final linking of object code into executable code.

For VAX/VMS, the standard format is:

```
$ LINK/NOMAP/NOTRACEBACK/EXE=EXE:prgnam.EXE -
$_OBJ:prgnam.TMP,UT:UTIL/LIB,SYS$LIBRARY:DBLRTL/OPT
```

For CTS-300, the standard format is:

```
.R LINK
*EXE:prgnam.TSD=OBJ:prgnam.TMP,UT:UTIL,SY:TDIBOL/B:100000
*^C
.R REDUCE
*EXE:prgnam.TSD/N
*^C
```

For TSX-Plus, the standard format is:

```
.R LINK
*EXE:prgnam.SAV=OBJ:prgnam.TMP,UT:UTIL,SY:DLIB
```

For RSTS/E, the standard format is:

```
RUN $TKB
TKB>xxx:prgnam.TSK=OBJ:prgnam.TMP
```

```
TKB>UT:UTIL/LB
TKB>LB:DMSUSL/LB
TKB>/
Enter Options:
TKB>LIBR=DIBOLD:RO
TKB>//
```

Utility programs with the .UTL source code extension are linked with the following standard formats:

For VAX/VMS:

```
$ LINK/NOMAP/NOTRACEBACK/EXE=UT:prgnam.EXE -
$_OBJ:prgnam.T20,UT:UTIL20/LIB,SYS$LIBRARY:DBLRTL/OPT
```

For CTS-300:

```
.R LINK
*UT:prgnam.TSD=OBJ:prgnam.TMP,UT:UTIL20,SY:TDIBOL/B:100000
```

For TSX-Plus:

```
.R LINK
*UT:prgnam.SAV=OBJ:prgnam.TMP,UT:UTIL20,SY:DLIB
```

For RSTS/E:

```
RUN $TKB
TKB>xxx:prgnam.TSK=OBJ:prgnam.TM2
TKB>UT:UTIL20/LB
TKB>LB:DMSUSL/LB
TKB>/
Enter Options:
TKB>LIBR=DIBOLD:RO
TKB>//
```

Program Size and Overlays

Under VAX/VMS, program size is of no concern, since VAX/VMS is a virtual memory operating system. Thus, if an additional subroutine needs to be linked in with the main program module, the following format is standard:

```
$ LINK/NOMAP/NOTRACEBACK/EXE=EXE:prgnam.EXE-
$_OBJ:prgnam.TMP,OBJ:sub1.TMP,OBJ:sub2.TMP,UT:UTIL/LIB,
$_SYS$LIBRARY:DBLRTL/OPT
```

where "sub1" and "sub2" are subroutines.

Under other operating systems, each program is limited to a 64 kbytes program load size, regardless of the available free memory. Included within this 64 kbytes is the space required by the run-time system. In fact, we are effectively limited to 32 kbytes as the maximum program load size.

Thus, larger programs must be broken into smaller main modules and two or more externally called subroutines. These subroutines are placed in what is known as an overlay segment. In many cases, the necessary reduction in load size can be accomplished by simply pulling some of the subroutines residing in the UTIL library into an overlay segment, without breaking up the main program into subroutines.

Using the above example, the typical CTS-300 overlay linkage would be:

```
.R LINK
*EXE:prgnam.TSD=OBJ:prgnam.TMP,UT:UTIL,SY:TDIBOL/B:100000/C
*OBJ:sub1.TMP/O:1/C
*OBJ:sub2.TMP/O:1
```

The same overlay for TSX-Plus:

```
.R LINK
*EXE:prgnam.SAV=OBJ:prgnam.TMP,UT:UTIL,SY:DLIB/C
*OBJ:sub1.TMP/O:1/C
*OBJ:sub2.TMP/O:1
```

Under RSTS/E, overlays are accomplished using ODL (overlay description language) files. When a file must be overlayed, the format is:

```
RUN $TKB
TKB>xxx:prgnam.TSK=SRC:prgnam/MP
Enter Options:
TKB>LIBR=DIBOLD:RO
TKB>//
```

and the SRC:prgnam.ODL file will contain an overlay description such as below. Note that overlays under the Task Builder are different than overlays under RT-11 or other systems, so that subroutines may not necessarily reside in the same overlay regions as above. The Task Builder Reference Manual should be referred to before attempting overlays under RSTS.

```
     .ROOT    OBJ1-U1-*(S1,S2)
OBJ: .FCTR    OBJ:prgnam.TMP
S1:  .FCTR    OBJ:sub1.TMP-U1-L1
S2:  .FCTR    OBJ:sub2.TMP-U1-L1
U1:  .FCTR    UT:UTIL/LB
L1:  .FCTR    LB:DMSUSL/LB
     .END
```

Whether or not an overlay is required can be determined by consulting the appropriate build batch for the package and operating system version. The Utilities Software Reference Manual contains more information on subroutine object size and overlay techniques.

Compiling and Linking Sort Programs

Sort programs under VAX/VMS are compiled in the same manner as other programs. Under other operating systems, the sort program provided with the source distribution is actually a sort control file. The sort control file is

processed by the GSORT utility program to generate a piece of a data division. The sequence for CTS-300, TSX-Plus, and RSTS/E, respectively, is:

a.
```
.RUN UT:GSORT
     INFILE  = SRC:prgnam.xxx
     OUTFILE = OBJ:prgnam.DBL
```

b.
```
.R DICOMP
*OBJ:prgnam.TMP=OBJ:prgnam.DBL,UTL:SORT.MAN/O

.R DBL
*OBJ:prgnam.TMP=OBJ:prgnam.DBL,UTL:SORT.MAN/R:DBG

RUN SY:[1,2]DICOMP
*OBJ:prgnam.TMP=OBJ:prgnam.DBL,UTL:SORT.MAN/O
```

c.
```
.R LINK
*EXE:prgnam.TSD=OBJ:prgnam.TMP,UT:UTIL,SY:TDIBOL/B:100000

.R LINK
*EXE:prgnam.SAV=OBJ:prgnam.TMP,UT:UTIL,SY:DLIB

RUN $TKB
TKB>xxx:prgnam.TSK=OBJ:prgnam.TMP
TKB>UT:UTIL/LB
TKB>LB:DMSUSL/LB
TKB>/
Enter Options:
TKB>LIBR=DIBOLD:RO
TKB>//
```

This page intentionally left blank.

EXTERNAL SUBROUTINE LIBRARIES

Both "UTIL" and "UTIL20" are libraries created during the compilation of MCBA utility source code subroutines. The UTIL library contains the ".MAN" subroutines and the UTIL20 library contains the ".UTL" subroutines. Both libraries are used in the linking procedure to help resolve global references with object programs. The UTIL library is used in linking most MCBA packages, while UTIL20 is used in linking MCBA utility (.UTL) programs.

Refer to the System Utilities Software Reference Manual for more information on the utility subroutines.

The following subroutines exist in the UTIL subroutine library (not necessarily in this order):

ADDTE, ANYCN, BDATE, DSPLY, ENVRN, FFILE, FILES, FRMAT, GDATE, GETAC, INPUT, INPT3, IO, IOS, ISIO, LEFTJ, LINFD, LPOFF, LPON, LPOUT, MESAG, MMENU, MOUNT, NSMNU, OFILE, OPENF, OUTPT, PGCHN, PGMND, PRCSN, PRSPL, RDATE, SCALE, SERCH, SRCHQ, SNMSG, STENO, TERID, TMENU, TTNO, and WATE

The following subroutines exist in the UTIL20 subroutine library (again not necessarily in this order):

ANYCN, DSPLY, ENVRN, FFILE, FILES, FRMAT, GETAC, GLACC, INPUT, INPT3, IO, IOS, ISIO, LEFTJ, LINFD, LPOFF, LPON, LPOUT, MESAG, MMENU, MOUNT, NSMNU, OFILE, OPENF, OUTPT, PGCHN, PRSPL, RDATE, SERCH, SRCHQ, SNMSG, STENO, TERID, TMENU, TTNO, and WATE.

There are a few exceptions by operating system:

A. DBL/TSX-Plus libraries do not include the ENVRN routine.
B. CTS-300 libraries include RTSRT.

These libraries are not interchangeable because the function and content of each subroutine differ and they contain different subroutines.

"FILES" - The File Handling Subroutine

Throughout the MCBA accounting and manufacturing packages, in most applications, files are OPENed, CLOSEd and protected via the external subroutine "FILES".

Exact use of the routine is detailed in the source code of the "FILES" subroutine and in the System Utilities Software Reference Manual. Also refer to the use of "XCALL FILES ..." throughout the source code of almost all programs for many examples.

Briefly, here is how "FILES" works:

When "FILES" accesses a data file, it first accesses the disk resident Device Assignment Table.

The Device Assignment Table (DEVICE.DDF) is a system utility data file which is an integral part of the Security System. It is used to determine the file's location as well as its current usage status. It is structured as follows:

There are exactly 200 records. Every data file used in any of the MCBA Accounting and Manufacturing packages has been allocated a unique record in this file. A program accessing a particular file via "FILES" will query the record in the DEVICE.DDF file corresponding to it, by using the relative record number of the entry in DEVICE.DDF for this file. For example, the A/R Open Item file in the Accounts Receivable package is called AROPEN. The entry in DEVICE.DDF for AROPEN is record #3. (See the document entitled "Device Table Assignments", section 3 of this manual, for a list of this package's files and their relative record numbers in the DEVICE.DDF file.)

Each record of the DEVICE.DDF file contains the disk name of the data file, the logical directory name for the drive and/or directory where the file resides and the current usage status of the file. The Usage Status field tells how many users have the file open. When set to "99", it means one user has the file opened exclusively. Each record actually has room for eight logical directory names and eight usage statuses to correspond to the eight companies whose data files can be processed.

The FILES subroutine does all the security checking necessary to ensure that the file in question is being accessed with the user's file access privileges.

Depending on the option specified by the programmer, "FILES" attempts to either 1) open, 2) open and protect--do not display message if in use, 3) protect (with no open or close), 4) close and unprotect, 5) open without changing the status, 6) increment user count without opening the file, 7) close and delete, and 8) open and protect--display message if in use.

Depending on the status of the file being processed and the current user's access privileges, "FILES" is either successful or unsuccessful at executing the option it attempts to perform, and returns a parameter indicating whether or not it has been successful.

If "FILES" is successful, the program simply proceeds. If "FILES" is not successful, certain messages are displayed on the CRT and the user has several options, depending on the specific application.

"FILES" can get confused if the user aborts a program (with CTRL/C) or if a program aborts because of a fatal error. Basically what happens in such a case is that the status of certain files is left "in use" or "protected" and never gets reset automatically as it would have had the program continued to its natural completion. The solution for this is to run the Clear File Status Flags application on the System Functions menu.

RECORD DEFINITIONS

This section is applicable for licensees of MCBA source code only.

MCBA source code makes extensive use of the DIBOL ".INCLUDE" statement when referring to data files within the Data Division. This means that, during compilation of source code, other files, defined within the source code via the ".INCLUDE" statement, may be compiled into the object module.

The primary benefit of this feature is that it is extremely easy to modify a record definition for a file by modifying the file definition (contained in a separate file from the source code). After recompilation, you can be assured that the change made will be recognized throughout the package or system.

Naming Conventions

All Record Definition files are named according to a specific convention:

1. The first two characters are "RD" (for Record Definition) when the record definition applies to a file contained within the DEVICE.DDF file (which is the main utility file which keeps track of the data files and their locations within the system).

   The only files not within DEVICE.DDF are MCBA utility data files: DEVICE.DDF, SECURE,DDF, MESARA.DDF, TTXREF.DDF, COMPNY.DDF, and SPLDIR.DDF. For these, the first two characters will be something other than "RD" (i.e. "SPL" stands for record definitions for SPLDIR, "MES" stands for record definitions for MESARA, "TTX stands for TTXREF, "DEV" stands for DEVICE, "SEC" stands for SECURE, and "CMP" stands for COMPNY).

2. For Record Definitions starting with RD, the next three characters are numbers which represent that file's position within the DEVICE.DDF file. (Ex: "RD001" refers to a record definition for CUSMAS, which is the first name in DEVICE.DDF.)

3. For record definitions starting with RD the last character is a letter, which distinquishes one form of definition for the record from another. (Ex: "RD001A.DEF" is the primary record definition for CUSMAS, while "RD001B.DEF" is the record definition for the control record of CUSMAS, etc.)

4. The extension for ALL record definitions is ".DEF".

5. The logical name for the device/directory for ALL record definitions is "DEF:".

Characteristics of Record Definition Files

The contents of these files include a description of the RD file, the name of the data file to which it applies, the record length, and the field names and sizes for the file.

The RD file may apply to the entire record, or only part of a record (which would be used as an overlay definition for a primary record definition).

It may be a single field name which defines the record length. This is used within programs which initialize (or create) the file to define the record length. Size calculations are based on the record length defined by this field.

Field names used within each set of record definitions do not duplicate the names in any other record definition of any other file, though record definitions for the same file may contain the same spelling of certain fields if they are never used in the same program.

Each package contains a file which lists all the Record Definition files used when compiling that package. This file is titled "LSTDEF.xxx" (LISTDEF.xxx for VAX/VMS version), where "xxx" is the package source code extension.

Installing New Record Definitions

Every package source code shipment is accompanied by the Record Definition files used within that package. These files should be placed on the same directory as the Record Definition files shipped with other packages. Since some Record Definition files are used by more than one package, they are identical and can therefore be used interchangeably. Also, since all record definitions are sought on logical device "DEF:", it is best to centrally locate these files.

(NOTE: If you have previously modified record definitions after receiving them, then these record definitions should supercede the ones sent from MCBA. In this case, you should only copy the definitions which were not sent with any previous packages.)

If the definition of a file within a package is to be changed, ALL of the Record Definition files must be edited to reflect that change (i.e. to change the CUSMAS record length/field characteristics, etc., change all record definitions of other form "RD001x.DEF", where "x" is the character that specifies the different formats of records in the CUSMAS file).

If field lengths are changed and those changes are to be reflected in entry screens, print-outs, etc., then these changes must be manually entered into the source code.

The package is then recompiled/linked (as well as other packages which use those record definitions).

REPORT SEQUENCE NUMBERS

All printed reports are numbered sequentially from 1 to 999. The report sequence number appears on the top line of each page of the report as: SEQ# nnn. When number 999 is reached, the next printed report will have sequence number 1. The last sequence number used is stored in the second record of the CONAME file. This number is just an aid to the user to verify the sequence of his printed reports. This number can be used as a form of audit trail by incorporating the report sequence number of the posting register in the transaction data records that appeared on that register. For example, in the General Ledger package, the sequence number of the General Journal is actually stored in the Year-to-Date Transaction record for all the transactions that appeared on that Journal.

This page intentionally left blank.

STANDARD PROGRAM SPECIFICATIONS
STANDARD MASTER FILE MAINTENANCE

Function: Performs maintenance functions on a Master file and its separate Index file. These functions are add, change, delete and print-out.

Input: KBD
Master File
Index File

Files Updated: Master File
Index File (to the Master file)

Output: Master List

Enter Module From: System Menu

When Done Return To: System Menu

Programs in Module: * XXXMNT, XXXPRT, ORGXXX, SRTXID, XXXCNT

* For XXX substitute the first three letters of the name of the Master file; for X, in SRTXID, substitute the first letter of the name of the Master file. Since duplicate program names are not allowed throughout the MCBA system, exception occurs when necessary to create unique program names.

Program Functions and Notes:

PRELIMINARY

1. The Master file and its separate Index file are created by the file initialization program at package start-up. At this time, the Master file contains a control record (described in general in the File Definition section) as its first record, and the rest of the file is filled with dummy bracket records. The Index file is set up similarly, except its first record is just blanks. See the detailed description of the Standard Master file, Index file set up in the beginning of the File Definition section before reading any further here. The terms defined there will be freely used in this description.

2. The Standard Master File Maintenance module allows the following capabilities:

    1. Add Master Records
    2. Change Master Records
    3. Delete Master Records
    4. Print Master Records

3. Add, change and delete modes will usually be in one program (unless program size would be excessive). Print mode is in a separate program. There are also three other supporting programs in the module:

    a. A "Sort" program on the Index file;
    b. A "Reorganization" program, which physically purges records that are logically marked for deletion, in delete mode;

c. An "Update Counter" program which adjusts the control record of the Master file after the "Sort" program runs.

In the A/R package, the names of these programs are: SRTCID, ORGCUS and CUSCNT, respectively. These are similarly named in other packages.

XXXMNT

Contains add, change, and delete modes on file XXXMAS.YYY with Index file XXXIDX.YYY (YYY is the company extension for the source code program. For executable code YYY will be SAV, TSK, TSD, or EXE, depending upon the operating system).

1. The Master file and Index file are both opened in update mode, the control record is read and the value of ORGXXX is saved in BSEND (the binary search variable - see SERCH description), MAXXXX is saved in MAXCNT. (XXX is the value of the files DEVICE table position, for example RECOOl corresponds to the record count of the CUSMAS file.) MAXCNT will be used to test whether the addition of a new record will exceed the file's allocated size. The Master record in the MCBA Utilities Reference Manual is then unlocked, so that other users may access the file.

2. The subroutine MMENU is called (see separate description) to display the maintenance submenu and accept the user's selection.

3. If add, change or delete are selected, control stays within the XXXMNT program. If the print-out function is selected at this point, MMENU asks whether a sort of the Index file is desired first. Then control goes in one of two ways, depending on whether or not the "Sort" is requested:

(No Sort) XXXMNT ------ XXXPRT
(Sort) XXXMNT ------ SRTXID ------ XXXCNT ------ XXXPRT

If "END" is selected, control passes back to the main package menu (see step 8 for more details on what happens in this case).

4. If add, change, or delete mode is selected, the main data entry screen is displayed (see individual packages for details of each one). This is the screen which will accept all necessary data for a Master File record, in add mode, or display this data in change and delete modes. Note that there may be two or even three successive screens in add and change mode to accommodate all the data within one master record.

In change and delete modes, an asterisk is placed beside the key field to indicate that this field is the key field and must be entered before the program can search for the desired record.

5. Add Mode

A. If your security access to either the Master or the Index file is " " or "I", you will be given a message and denied use of the add mode. Add mode and delete mode require "U" (unlimited) access to the files.

B. The key field(s) are requested first, and a binary search is set up and performed (using the SERCH subroutine on the Index file) to determine whether this key is already on the Master file. If the search is successful and the key is already on file, entry is refused. If the search is unsuccessful, the new key may be added and entry of the remaining data for the record proceeds.

C. Entry of the remaining data is done using the INPUT subroutine and whatever validation and verification is called for in the specifications of the specific maintenance program, which vary from field to field and package to package (see the specific packages for details).

D. After all data fields have been entered, subroutine ANYCN is used to request changes to the data entered (see separate description for ANYCN in the MCBA Utilities Reference Manual, Subroutines Section). Any field on the screen may be changed at this point. If the key field is changed, the program does the exact same search as described in step 1) to once again verify that the changed key has not already been entered.

E. When there are no more changes, the following sequence of actions are taken by the program.

   a. The control record is read (and locked) so as to obtain the absolutely latest value of RECXXX (which may have changed from its value when the control record was last read due to other terminals also running this program);

   b. The Index file is searched sequentially from the ending point of the last search to verify that the key value is still not on file;

   c. If the key is still not on file (not put there by another terminal since the last time the file was checked), RECCNT is incremented;

   d. RECXXX is compared to MAXXXX, and if it is greater, a "FILE FULL" message is displayed and processing stops.

   e. If the file is not full, the Index File record is created using the key value entered and the new value of RECXXX becomes the value of IRCXXX (the pointer to the Master record).

   f. Then, the new Index record is written at record number RECXXX in the Index file; the updated control record (with new value of RECXXX) is written to the Master file; and the new Master record is written at record number RECXXX in the Master file;

   g. The internal I/O buffers must now be cleared to ensure that the new created record is actually written to the disk. This is done by first reading the first record in the file and then the last record (MAXxxx). Finally, the channel is UNLOCKed. This is done for both the index file and the master file.

   h. The record area for the Master file in the program is cleared, and the data entry screen is redisplayed in preparation for adding another record.

6. Change Mode

A. As in add mode, the key is requested first and a search is set up and done on the Index file for a record with this key.

B. If the search is not successful, a message appears on the screen indicating that this particular record is not on file; if the search is successful, the Master File record is read, using the record number pointer obtained from the Index record. The variable BSMID is equal to the record number of the Index record. (Note that the SERCH subroutine automatically does a sequential search of the overflow area if it does not find the desired record using a binary search of the sorted portion of the Index file. Thus a record that was just added can immediately be called up in change mode, even though the file has not been sorted since the addition. See the description of the SERCH subroutine in the MCBA System Utilities Software Reference Manual, Subroutines Section for more details.)

C. The contents of the Master record are displayed on the screen. If our security access to the Master and Index files is "U" (unlimited), the ANYCN subroutine is called to request changes to this data. The value of the key field cannot be changed. All other fields may be changed.

If you have only "I" (inquiry only) access to the Master and Index files, the data from the record will be displayed and "PRESS RETURN" will be displayed at the bottom of the screen. No change to the records will be allowed, and the record will be unlocked. Also, the next step will not be performed.

D. When there are no more changes, the following sequence of actions is taken:

a. The changed Master record is written back to the Master file at its original location.

b. The logic given above under add mode, step 7g), to flush out the internal I/O buffers is done (the same lines of code are executed).

c. The record area for the Master file in the program is cleared and the data entry screen is redisplayed in preparation for changing another record.

7. Delete Mode

A. This mode operates the same as does change mode steps A and B. Also, delete mode will not be allowed when the user does not have "U" (unlimited) access to the Master and Index files.

B. The contents of the record are displayed on the screen, along with the message "RIGHT RECORD ?" (or a message to this effect). If the user answers "N", the program returns to step a.

C. If the user answers "Y", the following sequence of actions is taken:

a. The control record of the Master file is read; DELCNT is incremented by 1, and then the control record is written back out;

b. The first six characters of some chosen description field in the Master record to be deleted are set to "]]]DEL", and this record is written back to the Master file (this is how it is logically marked for deletion).

c. The Record Number Pointer field of the Index record for this Master record is set to zero; and the Index record is written back out, using the saved value of BSMID obtained from the search as its record number.

d. The message "RECORD DELETED" is displayed.

8. Ending Off

A. If the "END" key is pressed for the key value in add, change, or delete modes, the program loops back to step 2 above; and subroutine MMENU is called again to display the Maintenance submenu.

B. If the "END" key is pressed for the Maintenance submenu selection, the following sequence of actions is taken.

a. The control record is read once again;

b. If DELCNT is 50 (or sometimes 95) or greater, an attempt is made to protect the Master and Index files. If successful, the ORGXXX program is chained to, to automatically purge logically deleted records. Then, the Index file is sorted and the organized count and delete count are updated before returning to the main package menu.

c. If the conditions in b. are not met, then if (RECCNT-ORGCNT) is 50 or greater (meaning 50 new records have been added to the Master file since its Index was last sorted), then the Index file is protected, and the program chains to the SRTXID program after sending it the appropriate message. If the Index file cannot be protected, the program skips doing the sort. For more details on the message sent to the Sort program, see the separate Sort documentation in the MCBA Utilities Reference Manual.

d. If the Master file does not need to be either sorted or reorganized, the program simply chains to the main package menu.

Note that the actual comparison numbers, used to determine whether a sort reorganization is called for, may vary depending on the specific program.

9. Print

A. If print is selected from the Master File Maintenance submenu, the subroutine MMENU asks "SORT BEFORE PRINTING ?". The judgement to sort

or not is left up to the user. If a number of new master records have been added since the last time the Index file was sorted, these new records will appear at the end of the print out (out of order) if the Index file is not sorted at this point.

B. If a sort is requested, the Index file is protected using the FILES subroutine. The control record of the Master file is read, and the message to be sent to the Sort program is made up. This message is sent using the SNMSG subroutine (for a description of SNMSG, refer to the MCBA Utilities Reference Manual, Subroutines Section, and the program chains to the Sort (SRTXID). If the Index file cannot be protected (i.e. it is in use by another terminal), the program chains back to the main package menu and no print-out is done. When the Sort is complete, the Sort chains to the "Update Counter" program (XXXCNT), which then chains to the print-out program (XXXPRT) (see description of XXXCNT below).

XXXPRT

This is the actual print-out program, which is chained to by the Maintenance program (XXXMNT). It is always a separate program.

1. The print-out destination is selected (display, printout, spool file, etc. using the LPON subroutine, refer to the MCBA Utilities Reference Manual). The Master file and Index file are both opened; the Master File control record is read, and the search (SERCH) variable BSEND is set equal to ORGCNT.

2. The MCBA utility subroutine STENO is used to get the starting and ending key values for the print-out. If the RETURN key is pressed while in STENO, STRTNO (starting key) is set to spaces, and ENDNO (ending key) is set to "[[[", indicating that a print-out of the full Master file is to be done (if the requested key values are designated as numeric, the starting and ending numbers are set to all 0s and all 9s, respectively, for "ALL").

3. If STRTNO = ENDNO, only one Master record is requested; and a binary search of the index is done for this one record using the SERCH subroutine.

4. If a range of Master records (or "ALL") was selected, a generic search is performed to find the first valid record, then the range (or the entire Master file) is printed using the MCBA LPOUT subroutine.

5. As soon as this is fully printed, the printer is closed using the MCBA subroutine LPOFF; and the program goes back to step 1.

6. When the "END" key is pressed, the files are closed and the program chains back to the Maintenance program.

ORGXXX

This program physically purges logically deleted records from both the Master file and the Index. It is chained to automatically when the

Maintenance program senses that DELCNT in the control record of the Master file has exceeded a certain point (see the description of the XXXMNT program for more details on this).

The program operates as follows:

1. The control record of the Master file is read, and the values of ORGCNT and RECCNT are saved.

2. The Master file is read sequentially. Each time an undeleted Master record is found, its record number is placed in the "Records Array" (RECARR), which has space for 50-100 entries. The number of entries made into this array is kept track of by the variable CNT. If a deleted record is found, it is not put into this array.

3. If a record was marked for deletion, it is not written back out to the file. If it was not marked for deletion, it is placed in its entirety in a temporary holding array and is only written back out to the Master file when this temporary array is full (or the end of the Master file is encountered).

   The "Read" pointer and the "Write pointer to the Master file are kept separately, and what essentially happens is that the Master file is written back out over itself, with the deleted records missing.

   When the records are being physically rewritten, the "Write" pointer along with the key fields for the record are stored in a new array. This new array will become the rebuilt Index file. Each time the record array is written to the disk, this index array is filled. When the 50-100 records are all written, the index array is then written to the Index file. Both array counts are then reset.

4. This process of filling the record array, writing it to the disk as you fill the index array, then writing the index array to the disk, is kept up until all records have been read.

5. When this process is completed, first any unwritten records in the arrays are written to the disk. Then, bracket records are written into the file up to the previous record count record number. The control record in the Master file is reread, RECXXX is set to the new (purged) record count, DELXXX is set to 0, and ORGXXX is set to 1 (since the Index file is now unsorted).

6. The record count and organized count are sent via the SNMSG subroutine, and the SRTXID program is chained-to so as to fully sort the index.

7. In this manner, the Index file of a Master file reorganization occurs. This assures that the Index file, even if previously corrupted by a system crash, can be recreated from existing data.

SRTXID

This is a standard sort program which fully sorts the Index file to the Master file on the main key. The Maintenance program (XXXMNT) or the Reorganization program (ORGXXX) sends a message (using the SNMSG subroutine) containing the name of the program to chain to when the sort finishes (which is always the XXXCNT program, described below), and another message containing the name of the program that the XXXCNT should chain to once it completes. The sort routine reads and clears the first message and the XXXCNT reads the second message.

XXXCNT

This program simply sets ORGCNT equal to RECCNT in the control record of the Master file, and unprotects the Index file (using the FILES subroutine), after the sort (SRTXID). It reads the message (using SNMSG) for the next program to chain to, clears the message and chains to the program.

STANDARD PROGRAM SPECIFICATIONS
STANDARD TRANSACTION FILE ENTRY

Function: Allows additions, changes, and deletions of transactions to a Standard Transaction file. It also includes an Edit List print program.

Input: KBD
Master File
Master File Index
Possible Auxiliary Files

Files Updated: Transaction File

Output: Transaction File
Edit List

Enter Module From: Package Menu

When Done Return To: Package Menu

Programs in Module: Entry and Editing Program (sometimes split into separate programs for add, change, and delete modes)
Edit List Program

Program Functions and Notes:

GENERAL

The Transaction file approach is used for adding records to a major file, or changing fields on a major (or Master) file, when it is necessary to produce a hard-copy audit trail of such additions or changes. Such a hard-copy document is called a Register or a Journal.

In such a case, the MCBA package does not allow direct and immediate access to the major file, but requires the user to add to or change the major file via a Transaction File Entry, Editing and Posting procedure.

Using this method, the proposed additions and changes to the major file are first entered into a holding file called a Transaction file. The records in this file are called transactions. Within this file the transactions can be added to, changed, or deleted at will without yet affecting the major file in any way. A print-out of the entire contents of the Transaction file may be obtained at any time. This print-out is called a Transaction Edit List and is used to check the correctness and completeness of entries made to the transaction. The Edit List is not the hard-copy audit trail document.

When the user is satisfied that the data he has entered into the Transaction file is correct and complete, he then initiates the "Posting" procedure. The posting procedure is a job stream of several programs (different for different specific applications) which performs the following general functions:

a. Sorts the records in the Transaction file in the same order as the key of the major file that will be updated;

b. Prints the hard-copy audit trail document (the Transaction Register or Transaction Journal);

c. Updates the major file and any auxiliary files depending on the specific application (this is the actual posting step);

d. Clears all data records from the Transaction file and replaces them with dummy bracket records (thus restoring the Transaction file to the exact same condition it was in when it was first created).

For descriptive purposes, it will be assumed that the Entry and Editing portion of the Transaction module are each in separate programs, called "XXXENT" and "XXXEDT". This is the usual case. Posting varies from application to application and is described in detail under the individual application's Program Specifications.

XXXENT

This program handles the addition, alteration and deletion of transaction records on the Transaction file.

1. All necessary files are opened. The control record of any Master file to be used in conjunction with the entry of transactions is read and ORGCNT is saved in the BSEND variable for binary search purposes. The control record of the Transaction file is read and MAXREC for this file is saved in order for the program to be able to determine when this file is full.

2. The MCBA utility subroutine TMENU is used to display the Standard Transaction Entry, Editing and Posting menu, and to accept the user's selection of the mode he wishes to enter (the modes are: ADD, CHANGE, DELETE, PRINT EDIT LIST, POST).

   If "PRINT EDIT LIST" is selected, the program chains to the appropriate program to perform these functions. "ADD", "CHANGE" and "DELETE" modes are handled by the current program.

   If "POST" is selected, the user is first asked "ALL TRANSACTIONS OK TO POST ?". If "Y" is answered, the necessary files are protected (or just incremented if they will be used and/or updated but not fully reorganized). When transactions are to be directly merged into a Master file, the Master File Control record is read to get the number of available records. If this number is less than the number of records to be posted (record count of the Transaction file less than the deletion count less the control record), then a message is displayed and posting does not occur.

   If the exact number of records is not known (i.e. the number is something less than the number of transaction records), then the above step is performed in the Posting program or in the Merge-X routine. When all is successful, the first program in the posting stream begins.

3. For add, change, and delete modes the full data entry screen is displayed in preparation for the user to enter transaction data. In change and delete modes, an asterisk is displayed to the left of all the fields that are a part of the key of the Transaction file (this key varies with the specific application).

4. Add Mode

   Note: Add mode is allowed only for users with "U" (unlimited) access to the applicable transaction file.

   A. The data for a particular transaction is entered by the user per the Data Entry Specifications for the particular application. This varies to a very large degree from application to application, and usually involves various kinds of cross-checking between files, calculations, and automatic screen displays. Very often, the Index to a Master file will have to be searched to verify that a particular Master record key value is on file. The binary search option of the MCBA subroutine SERCH is usually used for this. This Master record key value is also usually a part of the key of the Transaction record.

   B. After all fields are entered, the MCBA subroutine ANYCN is used to accept changes to the data just entered.

   C. When there are no more changes, the control record of the Transaction file is read and its RECCNT is incremented by 1. The control record is written back out, and the new transaction is written out at the record location given by RECCNT.

      Note that if the incremented value of RECCNT is greater than the value of MAXREC (for the Transaction file), a message indicating that the Transaction file is full is displayed and the program exits back to the package menu after closing the files.

   D. The data is then cleared from the screen and the entry screen is redisplayed (the Transaction record area in the program is not usually cleared), in preparation for the next transaction to be entered.

5. Change Mode

   A. All the values of the key fields must be entered first (the fields that have an asterisk to the left). Then the Transaction file is searched sequentially from the beginning for a record matching the key fields just entered. The MCBA subroutine SERCH can be used in sequential mode in this case, since the Transaction file is not in any sorted order at this point.

   B. If a match is not found, a "TRANSACTION NOT ON FILE" message is displayed and the program returns to step a.

   C. The first matching record that is found (that is not marked for deletion) is displayed on the screen with the message "RIGHT TRX ?". If the user answers "N" to this, the search is continued for another

matching Transaction record. This continues until either the user answers "Y" to "RIGHT TRX ?" or the end of the Transaction file is encountered (in which case the program returns to step B). There can be multiple transactions on the file with the same key and it is up to the user to make certain he does not incorrectly enter the same transaction more than once.

D. When the correct transaction is found and displayed, if the user has "U" (unlimited) access privilege to the transaction file, the user is given the opportunity to change any of the non-key fields. The key fields cannot be changed in change mode. To accomplish this the user must delete the particular transaction and re-enter the corrected one in add mode.

If the user does not have "U" access, the transaction is displayed but no changes are allowed.

E. When there are no more changes to the Transaction record, the record is written back to the Transaction file. Then the screen is cleared, the entry screen is redisplayed, and the program goes back to step A.

6. Delete Mode

Note: If the user does not have "U" access to the transaction file, he cannot select delete mode.

A. Steps 5A, 5B, and 5C are performed, exactly as for change mode.

B. When the user answers "Y" to "RIGHT TRX ?", instead of the change logic, the delete logic is performed - the record is marked as (logically) deleted by inserting a string of zeros (usually six) in a designated field (which varies with the specific application). The record is then written back to the Transaction file, the control record is read to increment the deletion count, then is rewritten and a message appears on the screen "TRX DELETED".

C. The screen is then cleared; the entry screen is redisplayed, and the program goes back to step 6A.

7. Ending Off

A. If the "END" key is pressed for the key value in add, change, or delete modes, the XXXENT program goes back to step 2, and redisplays the Transaction Entry, Editing and Posting menu.

B. If the "END" key is then pressed for the menu selection, the XXXENT program chains back to the main package menu after closing the necessary files.

XXXEDT

This program prints an Edit List, showing all non-deleted transactions in the Transaction file, in the order in which they were entered. The report destination is selected, using the LPON subroutine.

The Transaction file is opened and read sequentially from the beginning of the file. Records logically marked as deleted are skipped. All other records are printed out using the LPOUT subroutine. When the end of the Transaction file is encountered (indicated by a dummy bracket record), various transaction totals that have been accumulated as individual records were printed, are printed out. The specific types of totals shown depend on the particular application. However, the number of (non-deleted) transactions on file are always shown.

POSTING

The posting option is denied users who do not have "U" (unlimited) access to all files which are updated in the posting stream. As mentioned at the beginning of this section, the posting logic varies widely with the specific application. However, the general sequence of steps is usually:

a. Protect or increment user count of files used in the posting procedure before beginning any processing. Also when possible, check to be sure enough space exists in the files to be posted to.

b. Sort the Transaction file;

c. Print the hard-copy audit trail document (Transaction Register or Transaction Journal);

d. Update the appropriate files;

e. Clear the Transaction file;

f. Unprotect and decrement user counts of all files used in the posting procedure.

g. Chain back to the Transaction Entry and Editing submenu.

This page intentionally left blank.

STANDARD PROGRAM SPECIFICATIONS
MERGE-X ROUTINE

Function: This program is a merge-in-place program used in the general posting job stream to merge Transaction records into a main file, in order, without using additional work space.

Input: Trx File
Main File

Files Updated: Main File

Output:

Enter Module From: Within posting stream

When Done Return To: Next program in posting job stream

Programs in Module: Merge-X program

Program Function and Notes:

Part of the posting job stream (i.e. the sequence of programs activated when "POST TRANSACTIONS" is selected from the Standard Transaction Entry, Editing and Posting submenu) is a program which merges the transaction records from the Transaction file into the main file (such as the A/R Open Item file in Accounts Receivable, or the A/P Open Item file in Accounts Payable). At this point, the Transaction file is assumed to be in sorted order on the same key as the key of the main file. Thus, only a simple merge is necessary to update the main file.

A standard technique, called the "Merge-X" technique, is used in the MCBA packages to accomplish this merge. This technique does a merge-in-place on the main file and does not require any temporary work files or additional storage space. The technique works as follows:

A. The Transaction file control record is read and a count of the number of records to be added to the main file is obtained. The number of records to be added equals the record count minus the deletion count minus 1 (to count the control record). Call this count NEWCNT.

B. Then the main file control record is read and NEWCNT + RECCNT is compared with MAXREC (that is, RECCNT and MAXREC for the main file) to ensure that there is room in the main file for the new transactions. If (RECCNT + NEWCNT) is greater than MAXREC, a "FILE FULL" message is displayed and the posting job stream is terminated. Note that none of the new transactions have been added to the main file when a "FILE FULL" condition is detected.

Note: The steps A and B may often occur at the beginning of the posting stream, when the actual number of posted records is known at that point. Steps A and B are for posting streams where source records in the Transaction file may not be added to the main file and the actual count is not known until the merge program. Also, in this case, the records in the Transaction file may need to be counted manually instead of using the Transaction record count.

C. Assuming that there is room in the main file to hold all the new Transaction records, RECCNT and ORGCNT in the control record of the main file are reset to their old values plus NEWCNT; and the control record is written back out (note that the main file is always in completely sorted order, so RECCNT = ORGCNT).

D. A word of explanation on the actual merge step first. The simplest kind of a merge would be to take the Transaction file and the main file and merge them into a third file big enough to accommodate the total number of records. This could be done by reading through both files sequentially from the beginning, comparing a record from each file for the lowest key and moving the low record to the next available location in the third file.

If this process was attempted without having a third file (merging both files into the main file and writing the main file over itself), Transaction records would over-write Main File records and main file data would be lost. Otherwise, the entire bottom end of the main file would have to be shifted each time another Transaction record was inserted into the main file. This would take excessive I/O time.

The way around this problem, but still not requiring a separate work file, is to start at the high-order end of each file and read them sequentially backwards, comparing the current Transaction record with current Main File record, and inserting the one with the higher key into the main file at the next available position.

The Transaction file has NEWCNT new records, while the main file starts out with RECCNT records. However, the high order record in the first comparison on the Transaction file and main file is inserted into the main file at record position (RECCNT + NEWCNT) which initially contains a dummy bracket record. Some valid data records of the main file will eventually get over-written by transaction records, but only after these main file records have been repositioned to a later point in the main file.

Thus a true merge-in-place is accomplished.

In addition, to prevent incorrect data and allow restart of merge routines after a system crash or program abort, each Transaction record, once posted, is rewritten back to the Transaction file with a "Record Posted" flag set. During the merge operation, records with this "Posted" flag set are ignored, so no duplicate posting will occur.

E. The program uses a buffered technique to accomplish the "Write" portion of step D. That is, the record selected in each comparison described in step D is not immediately written to the main file. It is first inserted into a buffer in the program. When this buffer finally fills, then the entire buffer (usually holding between 50 and 100 Main File records) is written to the main file all at once. This eliminates continuous alternation between reading and writing on the main file and greatly cuts down on physical I/O time.

This page intentionally left blank.

MANAGEMENT AIDS - FILE LISTINGS AND COMMAND FILES

For VAX/VMS

Three file listings are standardly included in every source shipment: LISTDEF.xxx, LISTSRC.xxx, and LISTEXE.xxx.

The LISTDEF.xxx file is a list of all .DEF files included with the "xxx" package. All these files are to be found in the [MCBADIBOL.xxx.DEF] directory. A complete recompile of the package will require all these files to be resident in the DEF: directory.

The LISTSRC.xxx file is a list of all the programs, build batches, sort control files and listing files included with your source shipment. All these files are to be found in the [MCBADIBOL.xxx.SRC] directory.

The LISTEXE.xxx file is a list of all the executable modules obtained after compiling and linking the "xxx" package. All these files will be found in the [MCBADIBOL.xxx.EXE] directory.

These files are provided both for general reference and as an aid in constructing command files for global copying or editing operations.

For RT-11

Four file listings are standardly included in every source shipment: LSTDEF.xxx, LSTSRC.xxx, LSTSAV.xxx and LSTTSD.xxx.

The LSTDEF.xxx file is a list of all .DEF files included with the "xxx" package. A complete recompile of the package will require all these files to be resident on the DEF: logical device.

The LSTSRC.xxx file is a list of all the programs, build batches, sort control files, listing files and command files included with your source shipment.

The LSTSAV.xxx file is a list of all the executable modules obtained after compiling and linking the "xxx" package under the DBL compiler to run under the TSX-Plus operating system.

The LSTTSD.xxx file is a list of all the executable modules obtained after compiling and linking the "xxx" package under the DIBOL compiler to run under the CTS-300 operating system.

These files are provided both for general reference and as an aid in constructing command files for global copying or editing operations.

Since RT-11 does not provide as complex a directory structure as other Digital operating systems, packages often have to be placed in the same directories. For source code this is not a major difficulty, since all source code programs and command files are differentiated by their file extension. For record

definitions and executable files, this is not the case. Three additional command files are provided to aid in copying these packages:

PIPDEF.xxx - is a command file to copy all record definitions used in the "xxx" package from device IN: to device OUT:.

PIPSAV.xxx - is a command file to copy all DBL/TSX-Plus executable code associated with the "xxx" package from device IN: to device OUT:.

PIPTSD.xxx - is a command file to copy all CTS-300 executable code associated with the "xxx" package from device IN: to device OUT:.

For RSTS/E

Three file listings are standardly included in every source shipment: LSTDEF.xxx, LSTSRC.xxx, and LSTTSK.xxx.

The LSTDEF.xxx file is a list of all .DEF files included with the "xxx" package. All these files are to be found in a [203,nnn] account. A complete recompile of the package will require all these files to be resident in the DEF: account.

The LSTSRC.xxx file is a list of all the programs, build batches, sort control files, listing files and command files included with your source shipment. All these files are found in a [202,nnn] account.

The LSTTSK.xxx file is a list of all the executable modules obtained after compiling and linking the "xxx" package. These are all the files that should reside on the [200,nnn] account.

These files are provided both for general reference and as an aid in constructing command files for global copying or editing operations.

MANAGEMENT AIDS - CONVERTING WAIT TO WATE

This new Release 7 of your package has already converted all occurrences of XCALL WAIT to XCALL WATE. (See the Release Notes in Section 1 for details.)

As a convenience to you, the TECO macro, WATE.TEC, has been included in the source directory for the utilities. You can use this in converting existing source code of your own.

Full instructions on its use are included as comments within the macro itself.

This macro will also allow you to convert WATE to WAIT, should you wish to alter the Release 7.0 MCBA code to prior conventions.

This page intentionally left blank.

INITIALIZE CUSTOMER ORDER PROCESSING FILES

This program (INITCP) may be run upon request to initialize the master, index and transaction type files in the COP package. Normally, these files are initialized only when the package is installed, (except for temporary Index files that are created and deleted within a specific application). It can also be used to create a single file at a later time.

The program first requests the logical directory assignment or physical device where the files about to be created are to reside, and the Company code extension. The three-character Company code is used as the extension for the data files about to be created, instead of .DDF. This is how files for different companies are distinguished. A screen is then displayed requesting the user to enter the number of records to which he desires each particular file to be pre-extended. If a nonzero record count is entered for the file, it will be created, pre-extended to its full size with right-bracket records (records made up entirely of the "]" character).

A record count of zero may be entered for any file. If a nonzero record count is entered for a file which already exists on the physical device specified for it, this file will be lost and will be replaced by a fresh file containing only right bracket records. Additionally, the files SLSSUM and SLSIDX are always created as a pair with the same number of records, since SLSIDX is the where-used index to the SLSSUM file.

For each file to be created, its size in blocks (of 512 bytes each) is calculated using the record size (see the various File Definitions) and desired number of records to be in the file. Two characters are added on to the record size for the end of record mark [(CR) (LF)], and two whole records are added to the record count entered: one for the control record (first record of the file) and one to ensure that the file will always have a final bracket record.

For the each file, the control record is first written out, with ORGCNT = RECCNT = 1 and MAXCNT equal to 1 greater than the number of records specified for the file; and DELCNT = 0 (see the various File Definitions, as well as the description of the SERCH utility in the MCBA Utilities documentation). The rest of the file is then filled out with bracket records. If the user requested X records for the file, then the file will actually have (X +2) records. (The SLSIDX file has a blank record as a spacer instead of a control record.)

The files are all created on the devices that have been previously specified for them in the DEVICE.DDF file, by using the Security System Maintenance application.

NOTE: The Order Header and Order Line files are not created here since they are ISAM files. They must be created separately using the ISAM utility. See the User's Manual for detailed instructions.

This page intentionally left blank.

CUSTOMER ORDER PROCESSING MENU

This program (CPMENU) displays the main menu for the Customer Order Processing package and allows the selection of all the major applications of the package. The user's selection is accepted in the form of a number. Then chains to the appropriate program to perform the selected application. It will also accept the END key to end off and return to the Master menu.

The Customer Order Processing applications that are not on this menu are accessible through selection #12, Special Functions.

This page intentionally left blank.

SPOOLER FILE NAMES

The disk file names for spooled reports in the Customer Order Processing package have the following format:

Exttss.ccc

"E" is the fixed designation for the MCBA Customer Order Processing package.

x is a one-character designation for the particular report. A list of these designations is given below.

tt is the terminal number from which the running program created the report.

ss is a Sequence Number field used to insure a unique disk file name in case the same report was spooled more than once by the same terminal for the same Company code.

ccc is the Company code for the company of this report.

The one-character designations ("x" above) for Customer Order Processing reports and the programs that create them are as follows:

A - Order Edit List	- ORDEDT
B - Billing Edit List	- BILEDT
C - Sales History Journal	- SLHJNL
D - Credit Memo Edit List	- CRMEDT
E - Sales History Journal - Credit Memos	- CSLHJL
F - Price List	- PRICES
G - Backorder Report by Customer - Stocked Items	- BOCUST
H - Backorder Report by Customer - Nonstocked Items	- BOCUST
I - Backorder Report by Customer - All Items	- BOCUST
K - Open Customer Orders by Customer - All	- BOCUST
L - Product Category Account File	- PDALST
M - Sales Analysis by Product Category	- SAPCAT
O - Purged Detail Sales History	- PRGSLH
P - Sales Comparison by Customer	- CUSSLS
Q - Sales Comparison by Customer and Product	- CPRSLS
R - Sales History by Product Category/by Product	- PRDSLS
S - Sales Comparison by Product and Customer	- PRCSLS
T - Detail Sales History Edit List	- SLHEDT
U - Backorder Report by Item - Stocked Items	- BOITEM
V - Backorder Report by Item - Nonstocked Items	- BOITEM
W - Backorder Reprot by Item - All Items	- BOITEM
X - Open Customer Orders by Item - Nonstocked Items	- BOITEM
Y - Open Customer Order by Item - All Items	- BOITEM
Z - Customer Ship-To Address List	- SHTPRT
1 - Ship-Via Codes List	- SHVPRT

As an example, the first Order Edit List spooled by terminal number 2 and logged-on to company MCBA's files would have the Spool file name:

EA0201.MCB

DEVICE TABLE ASSIGNMENTS

Record #	File Name	File Description
44	ORDHDR	Order Header File (ISAM)
45	ORDLIN	Order Line Item File (ISAM)
46	CRMHDR	Credit Memo Header File
47	CRMLIN	Credit Memo Line Item File
48	LINIDX	Picking Ticket Line Item Index (Temporary)
49	SAPIDX	Sales Analysis by Product Category Index (Temporary)
51	BAKORD	Back Order File (Temporary)
52	BOINDX	Back Order File Index (Temporary)
55	SLSHST	Detailed Sales History File
57	SLHWRK	Sales History Work File (Temporary)
58	SLSSUM	Sales Summary File
59	SLSIDX	Sales Summary File Index
60	COPCTL	COP Control File
69	PRDACT	Product Category Account File
88	SSVDSH	Purged Detail Sales History File
171	SHIPTO	Ship-to Address & Tax File
172	SHPVIA	Ship-via Codes & Descriptions File

This page intentionally left blank.

LIST OF PROGRAMS BY APPLICATION

(Programs in parentheses are subroutines of the main program.)

1. Initialize Customer Order Processing files

```
INITCP
```

2. Order Entry & Editing

```
OEMNUT
ORDADD (SCRN1, SCRN2, COMIT, OE1, OE2, OE4)
INQUIR (OE1, OE2)
CHANGE (ORDCN, LINCN, RECOM, OE1, OE4, OE6)
CANCEL (UNCOM, OE1)
ORDEDT
```

3. Print Picking Tickets

```
LINIDX
SRTLIX
ALNINV
PIKTIK (FNDPS)
```

4. Billing (Print Invoices)

```
BILLS (ORDBL, LINBL, BLMNU, OE1, OE3, OE4, OE7)
UNBILL (OE1)
BILEDT
ALNINV
INVOIC
POSTAR
POSTINV (BPOST)
SLHJNL
SRTSLH
PSTSLH
CLRLIN
CLRHDR
UNPRBL
```

5. Credit Memo Entry & Editing

```
CRMENT (CMSC1, CMSC2, CMSC3, CRMNU, CR1, CR2, CR3)
CRMCNC (CR40
CRMEDT
CRMINV
SRTCRH
CRHCNT
SRTCRL
```

ALNINV
PRTCRM
CRMAR
PSTCRM
CSLHJL
SRTSLH
PSTSLH
CLRCRH
CLRCRL
UNPRCM

6. Price Maintenance

PRCMNT

7. Mass Price Change

PRCCNG

8. Print Price List

PRICES

9. Print Order Status Reports

BAKORD
SRTBIT
BOITEM
SRTBCU
BOCUST
UNPRBO

10. Product Category Account Maintenance

PDAMNT
PDALST
SRTPDA
PDACNT
ORGPDA

11. Print Product Sales Analysis

ANALYS
STSAPC
SAPCAT

12. Sales History Programs

SSMENU
HSTSEL
SSUPD

SRTSIX)
SSCNT)
SSIUPD) Post and/or Purge Detailed Sales History
PRGSLH)
PURMSG)
UNPSLH)
CUSSLS
CPRSLS
PRDSEL)
SRTPIX) Print Sales Comparison by Product
PRDSLS)
PRCSEL)
SRTPRC) Print Sales Comparison by Product and Customer
PRCSLS)
CLRMSS
CLRYSS
ORGSLS
SSSFMN
BLDSLH
SRTBLD
SLHEDT
SSSET

13. Special Functions

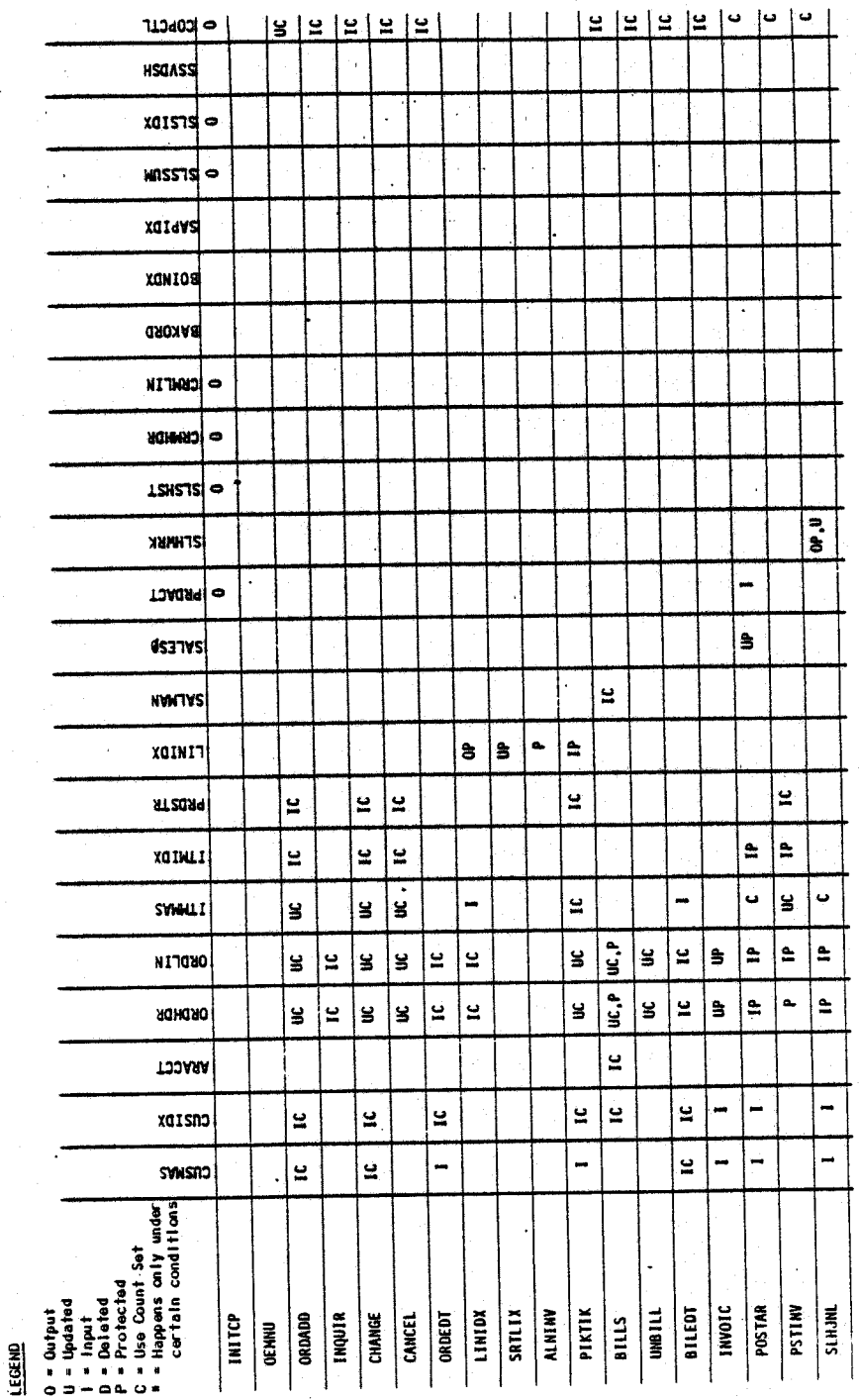
SPCFUN
RESET
TRADED
TERMSD
RSTCOM (COMSB)
CPFILS
CPSPOL
SHVMNT)
OPGSHV)
SHVCNT) Ship-via Maint
SRTSHV)
SHVPRT)

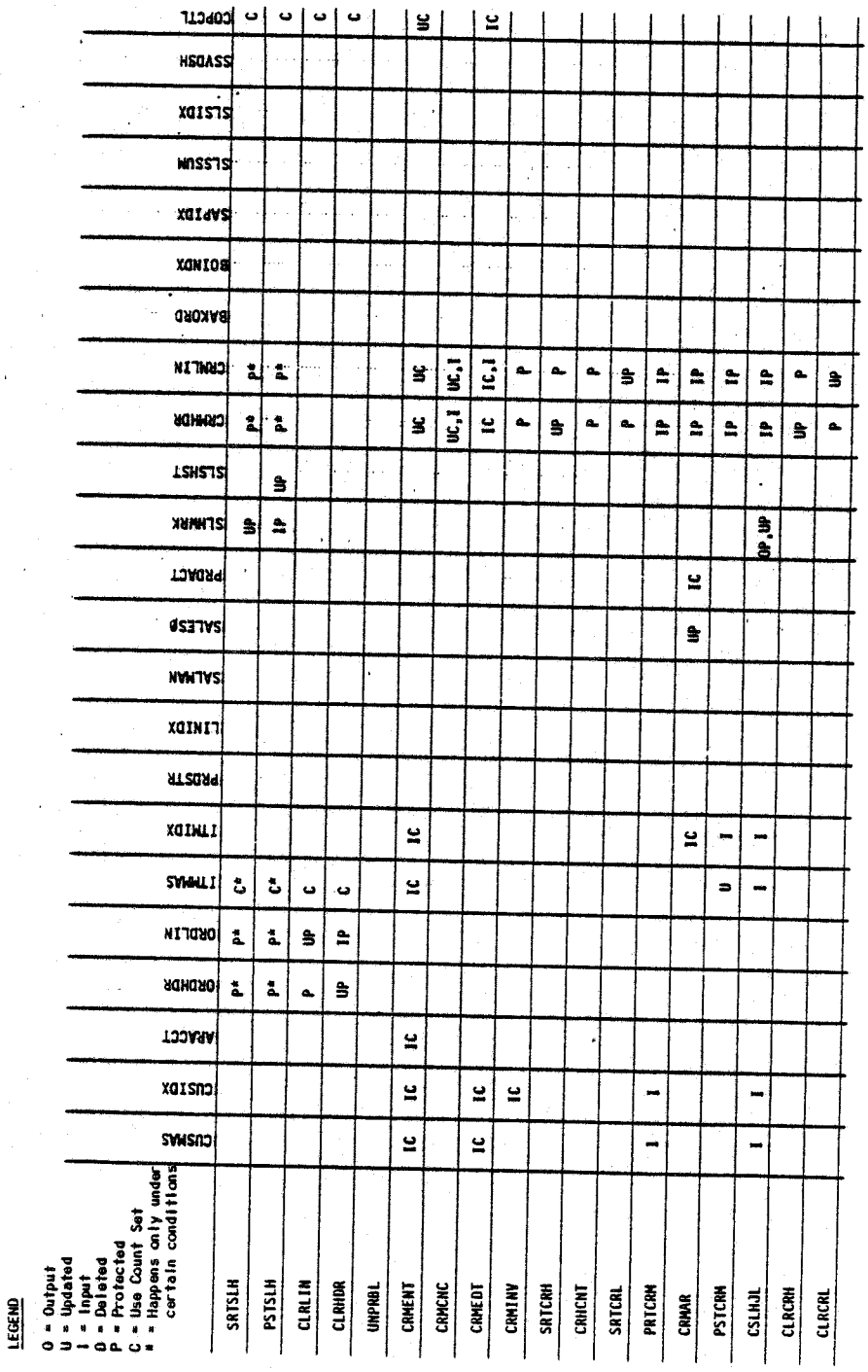
SHTMNT)
ORGSHT)
SRTSHT) Ship-to Maint
SHTPRT)
SHTCNT)

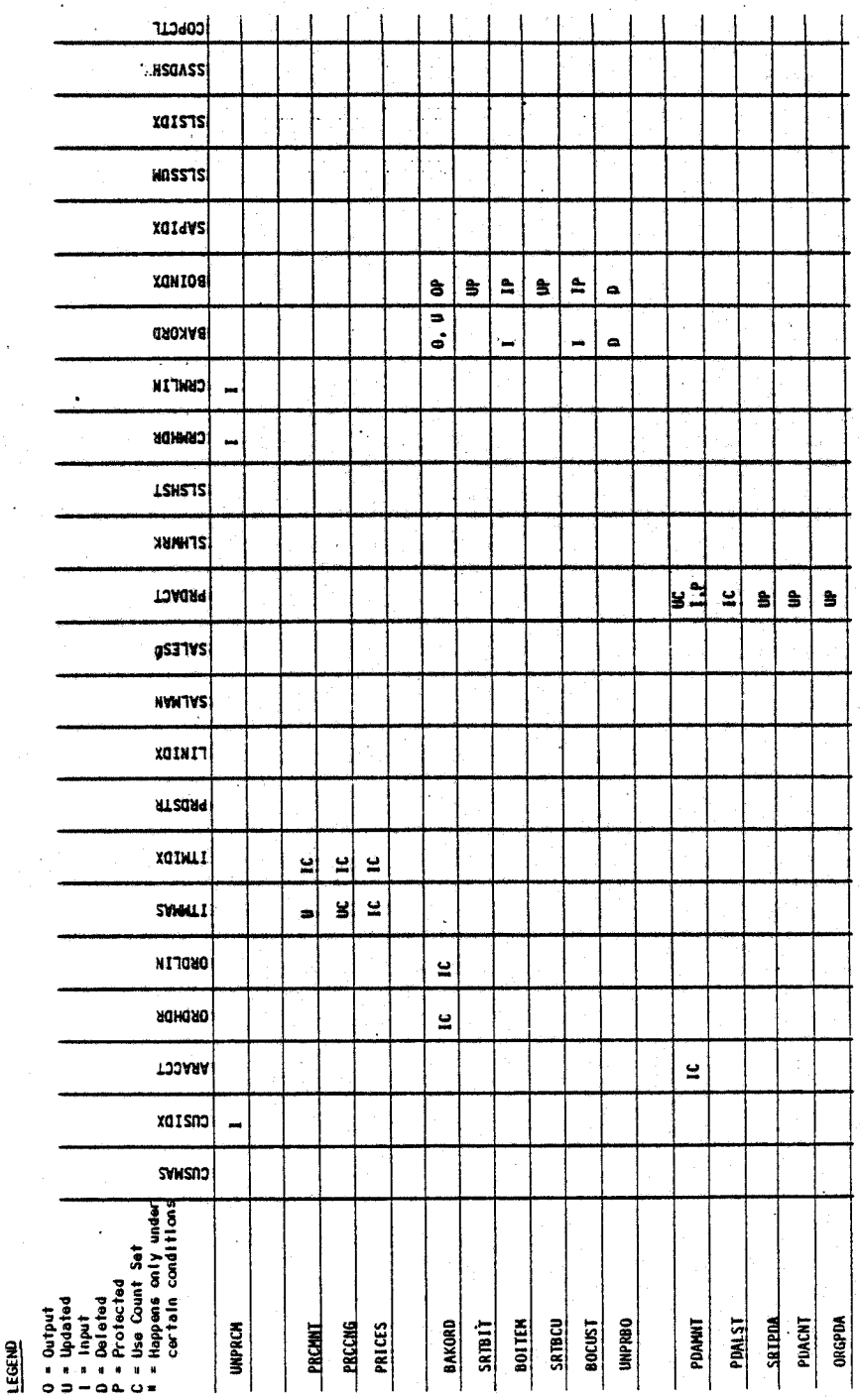
This page intentionally left blank.

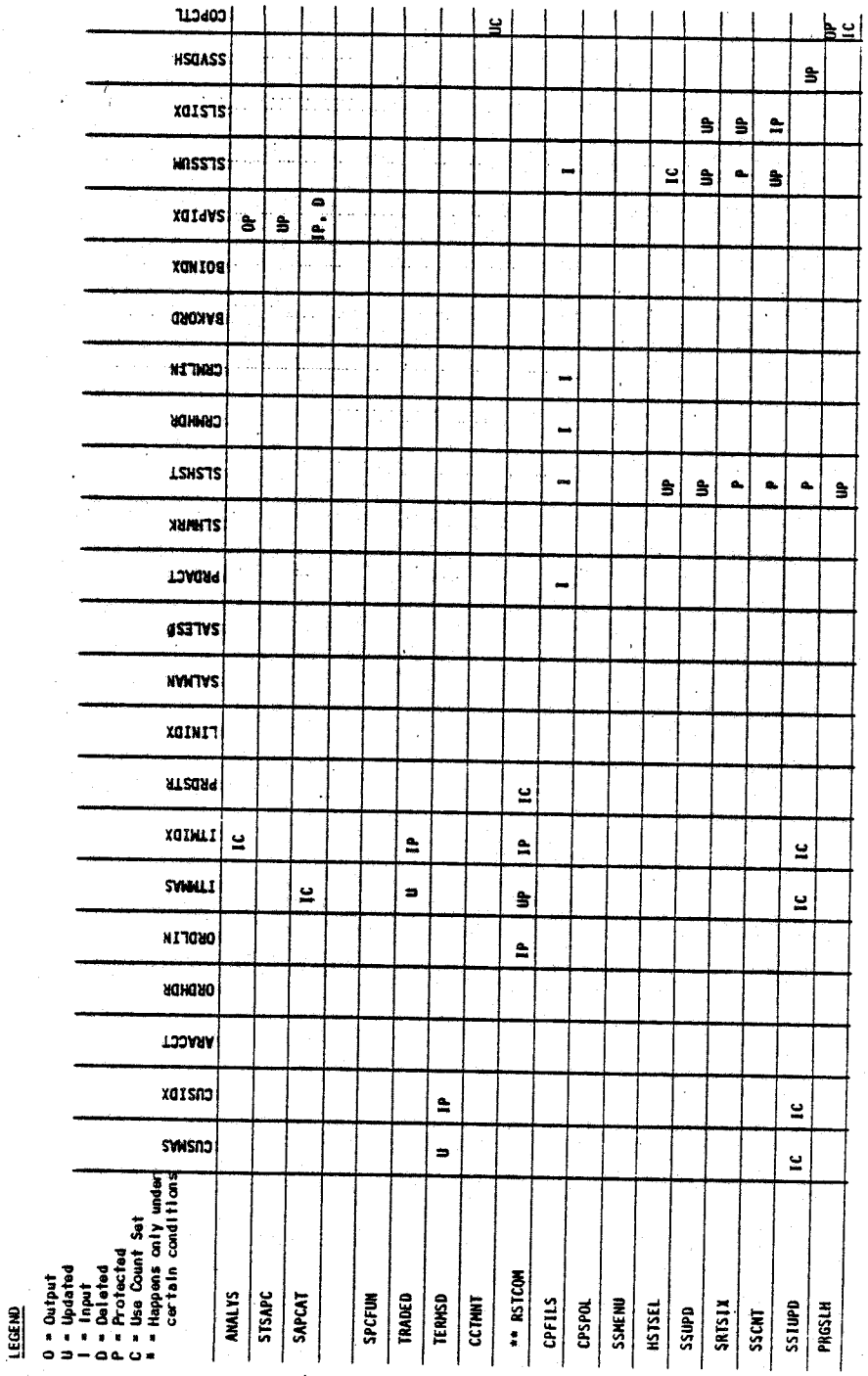
CUSTOMER ORDER PROCESSING PACKAGE
DIBOL SEP-84

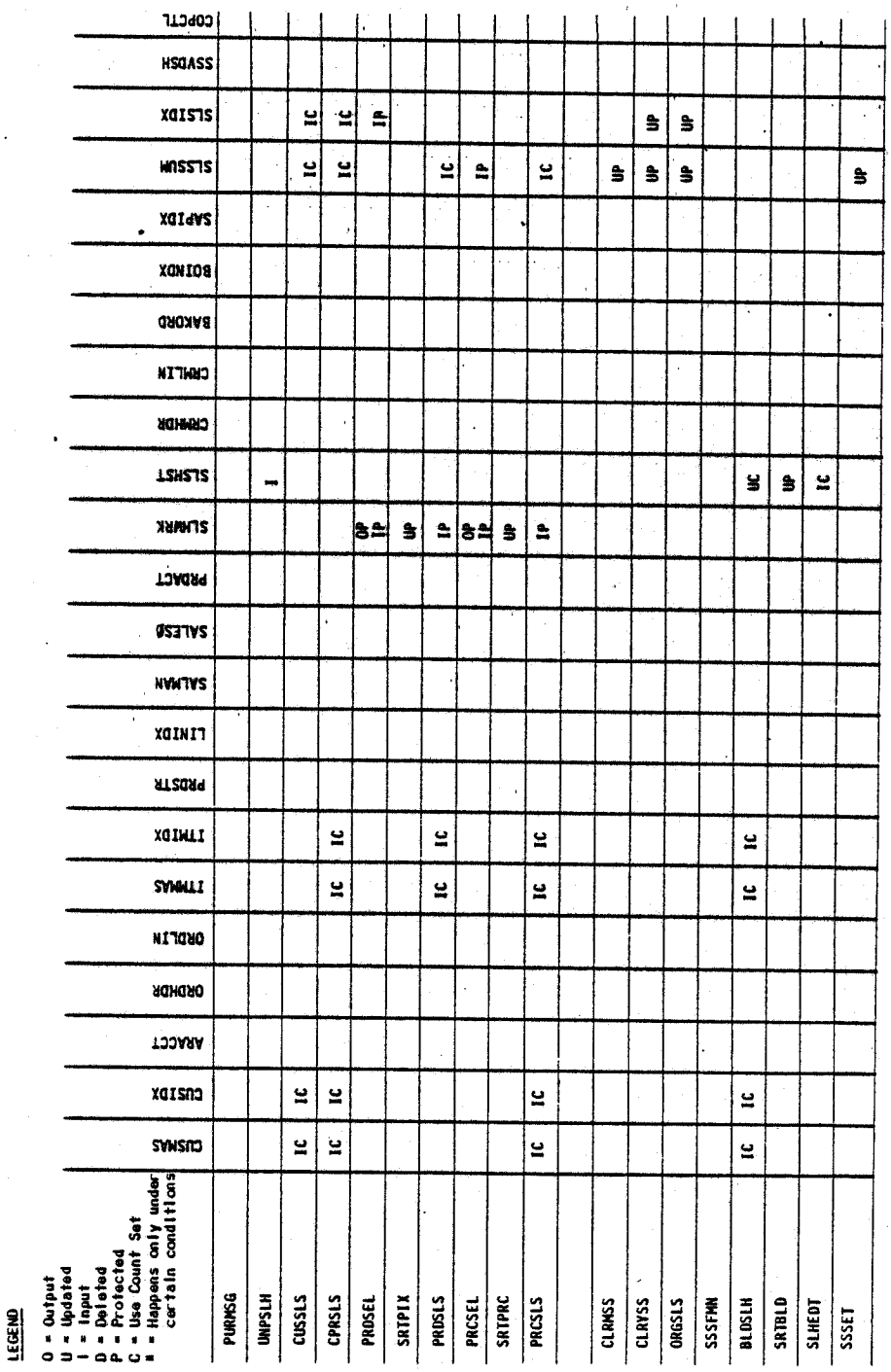
FILE USAGE MAP






LEGEND
O = Output
U = Updated
I = Input
D = Deleted
P = Protected
C = Use Count Set
* = Happens only under certain conditions
CUSMAS
CUSIDX
ARACCT
ORDHDR
ORDLIN
ITMMAS
ITMIDX
PRDSTR
LINIDX
SALMAN
SALESD
PRDACT
SLMARK
SLSHST
CRMHDR
CRMLIN
BAKORD
BOINDX
SAPIDX
SLSSUM
SLSIDX
SSYDSH
COPCTL
UNPRCM
PRCMNT
PRCCNG
PRICES
BAKORD
SRTBIT
BOITEM
SRTBCU
BOCUST
UNPRBO
PDAMNT
PDALST
SRIPDA
PUACNT
ORGPDA




LEGEND
O = Output
U = Updated
I = Input
D = Deleted
P = Protected
C = Use Count Set
* = Happens only under certain conditions
PURMSG
UNPSLH
CUSSLS
CPRSLS
PRDSEL
SRTPIX
PRDSLS
PRCSEL
SRTPRC
PRCSLS
CLRMSS
CLRYSS
ORGSLS
SSSFMM
BLDSLH
SRTBLD
SLHEDT
SSSET
CUSMAS
CUSIDX
ARACCT
ORDHDR
ORDLIN
ITMMAS
ITMIDX
PRDSTR
LINIDX
SALMAN
SALESD
PRDACT
SLWRK
SLSHST
CRMHDR
CRMLIN
BAKORD
BOINDX
SAPIDX
SLSSUM
SLSIDX
SSYDSH
COPCTL

This page intentionally left blank.

MODIFYING THE DEFAULT NUMBER OF PRICES PER ITEM

The Item Master file contains prices by customer type. These prices are stored in two arrays. The first price in the array is the base price for the item, and is entered via the Item Master File Maintenance application in the Inventory Management (I/M) package. The second and subsequent prices are entered via the Price Maintenance application in the Customer Order Processing (COP) package.

PRICCD - a 2-character alphanumeric field that contains the customer type.
PRICE - an 8-digit numeric field that contains the price for the item for this customer type. It has two decimal places.

The array sizes for these two fields must be identical. As shipped from MCBA, five prices can be entered for each item. The maximum number of prices that the system is designed to accommodate is 42.

Steps to Modify the Array

To change the prices array, you must make the same change to each of the two fields associated with prices, so that the dimension (size) of each field's array is the same. Even though the prices are maintained in the COP package, the changes are all made to the Item Master file.

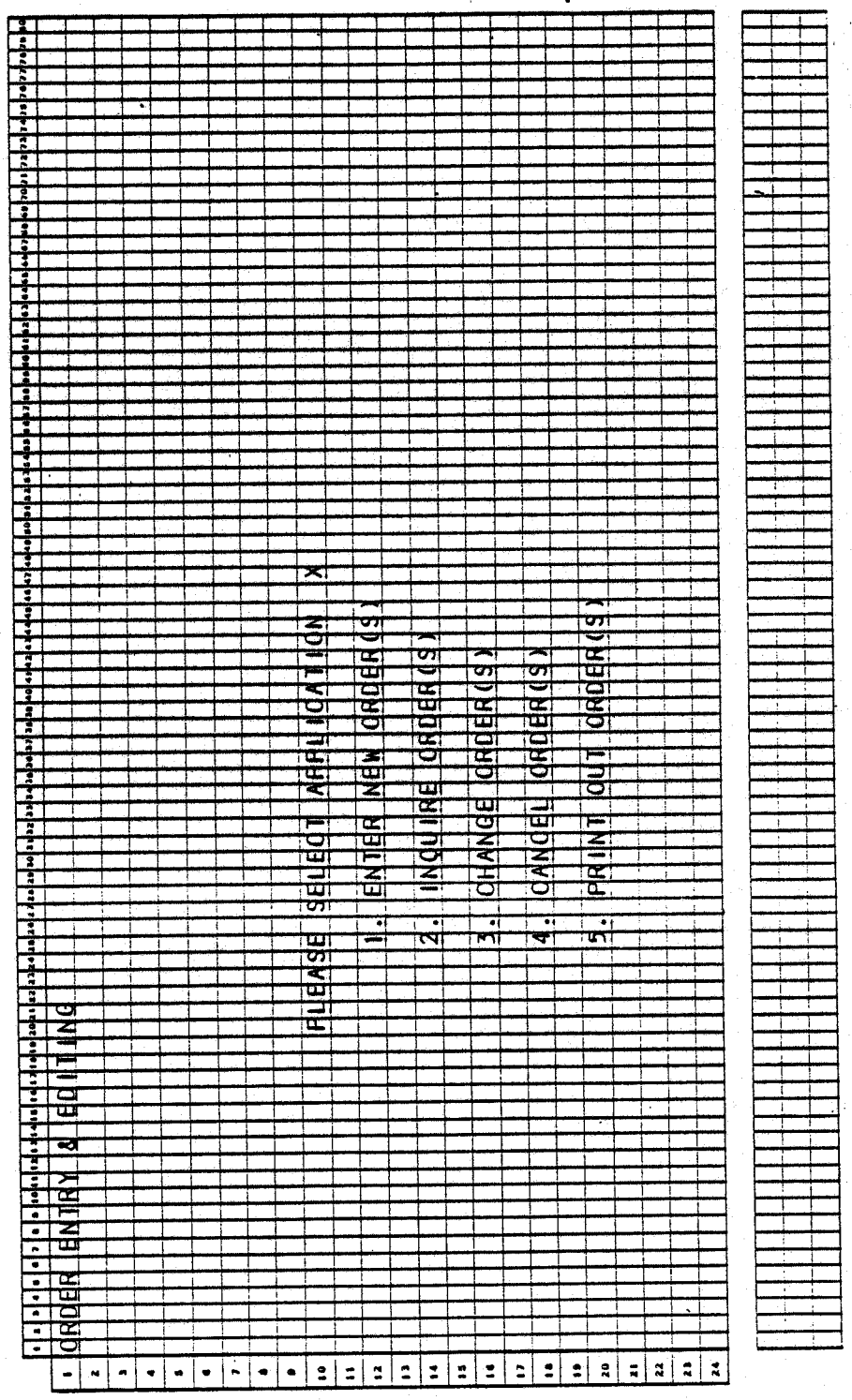
The necessary steps are outlined in detail in the I/M Software Reference Manual section entitled "Modifying the Default Number of Locations, Prices or Vendors".

This page intentionally left blank.

CUSTOMER ORDER PROCESSING PACKAGE
ORDER ENTRY & EDITING APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: OE:OEMNU



Program: ORDADD:SCRN1:OE1

```
ORDER ENTRY & EDITING
ADD

   1. ORDER #    XXXXXX                  2. DATE  XXXXXX

   3. CUST #     XXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXX

   4. SALESMN    XX      5. LOC    XX    6. SHIP VIA  X      7. PO #    XXXXXXXXXX

   8. DISCNT     XX      9. TERMS  X     9. COLL/PPD  X     11. JOB #   XXXXXXXXXX

  12. SHIP TO    XXXXXXXXXXXXXXXXXXXXXXXXX                  (1) 1 = XXXXXXXXXXXX

                 XXXXXXXXXXXXXXXXXXXXXXXXX                      2 = XXXXXXXXXXXX

                 XXXXXXXXXXXXXXXXXXXXX                          3 = XXXXXXXXXXXX

                 XXXXXXXXXXXXXXXXXXXXXXXXXXXXX                  4 = XXXXXXXXXXXX

  13. COMMENT    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX            5 = XXXXXXXXXXXX

                 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX       14. AR ACCT #  XXXX-XXX

FIELD # TO CHANGE  XX
```

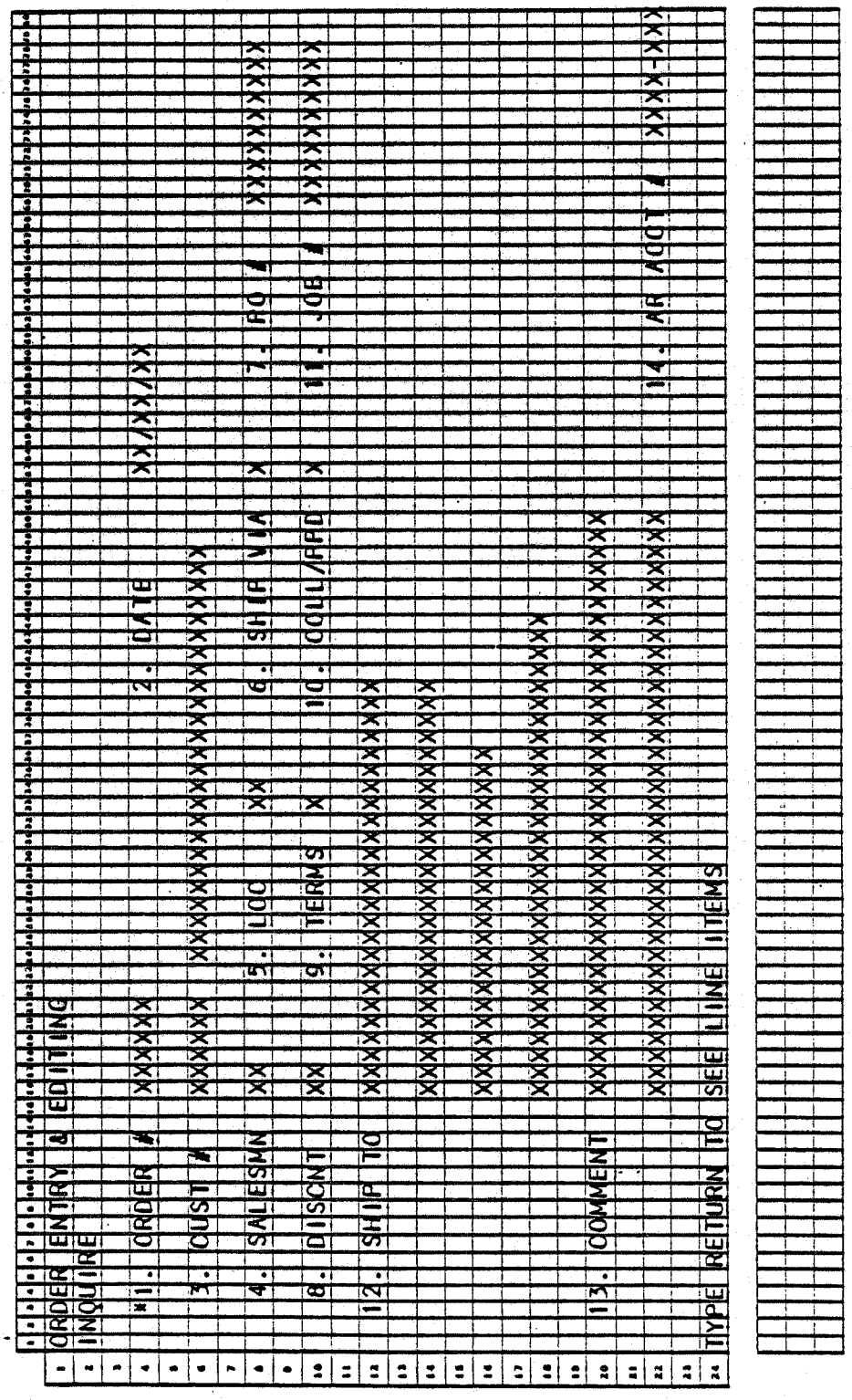
Remarks: (1) SUB-MENUS FOR SHIP-VIA, COLLECT/PPD, AND TERMS ARE DISPLAYED HERE.
(COLLECT/PPD WILL USE ONLY THE FIRST TWO LINES.)

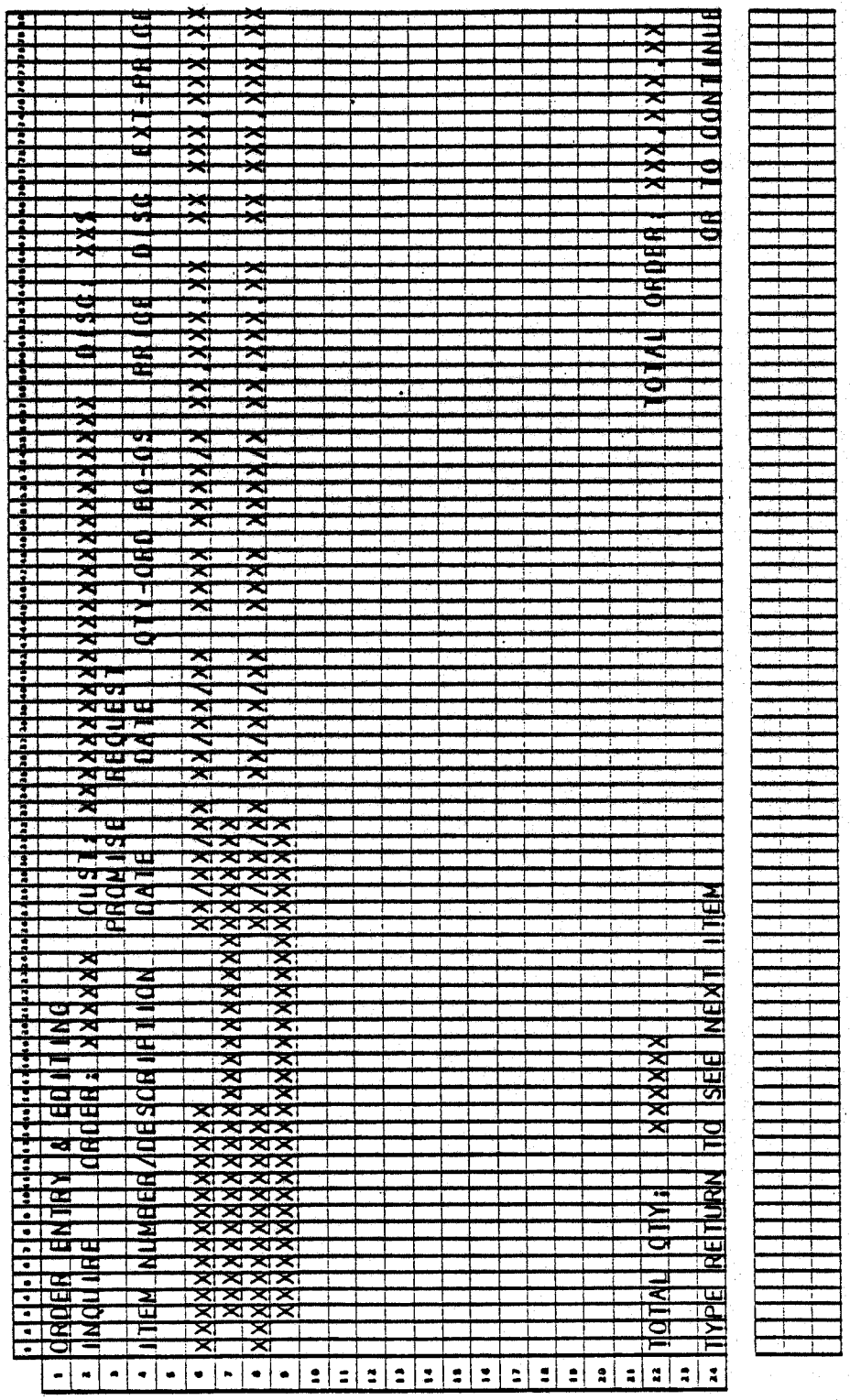
```
 1  ORDER ENTRY & EDITING
 2  ADD           ORDER: XXXXXX  CUST: XXXXXXXXXXXXXXXXXXXXXXXXX   DISC: XX%
 3                                       REQUEST
 4  ITEM NUMBER/DESCRIPTION                 DATE     QTY-ORD BO-OS    PRICE   DISC  EXT-PRICE
 5
 6  XXXXXXXXXXXXXXX                      XX/XX/XX      XXXX XXXX/X XX,XXX.XX XX XXX,XXX.XX
 7     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 8  XXXXXXXXXXXXXXX                      XX/XX/XX      XXXX XXXX/X XX,XXX.XX XX XXX,XXX.XX
 9     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
10  ???                                  XX/XX/XX      XXXX XXXX/X XX,XXX.XX XX XXX,XXX.XX
11     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
12                                                     ENTER ITEM COST: XXXXXX
13
14
15
16
17
18
19
20
21
22  RUNNING QTY: XXXXXX    REMAINING CREDIT: XXX,XXX.XX  RUNNING TOTAL: XXX,XXX.XX
23
24  ANY CHANGE ?  X

AVAILABLE QTY = XXXXX (NON B/O ITEM): 1-CANCEL, 2-ORDER IN STOCK, 3-OVERRIDE ? (1)
AVAILABLE QTY = XXXXX SELECT  1-CANCEL, 2-B/O BALANCE, 3-B/O ALL, 4-OVERRIDE ?
```

Remarks: (1) DISPLAYS IF THERE IS NOT ENOUGH AVAILABLE STOCK TO COVER AN ORDER FOR A STOCKED ITEM THAT IS NOT BACKORDERABLE. NBR (2) DISPLAYS IF THERE IS NOT ENOUGH AVAILABLE STOCK TO COVER AN ORDER FOR A STOCKED ITEM THAT CAN BE BACKORDERED.

(2) COLUMN TO RIGHT OF QTY-ORD AUTOMATICALLY DISPLAYS TO SHOW BACKORDERED OR OUT-OF STOCK AMOUNT. TAB KEY WILL BRING UP THE NEXT SEQUENTIAL ITEM WHEN CURSOR IS AT ITEM NUMBER (A GOOD USE OF THIS FEATURE IS TO BRING UP OPTIONS, REPRESENTING MODULAR BILLS THAT HAVE BEEN NUMBERED TO SEQUENTIALLY FOLLOW THEIR PARENT). THE SCREEN WILL SCROLL TO HANDLE MORE THAN A PAGE OF ITEMS.

Program: INQUIR

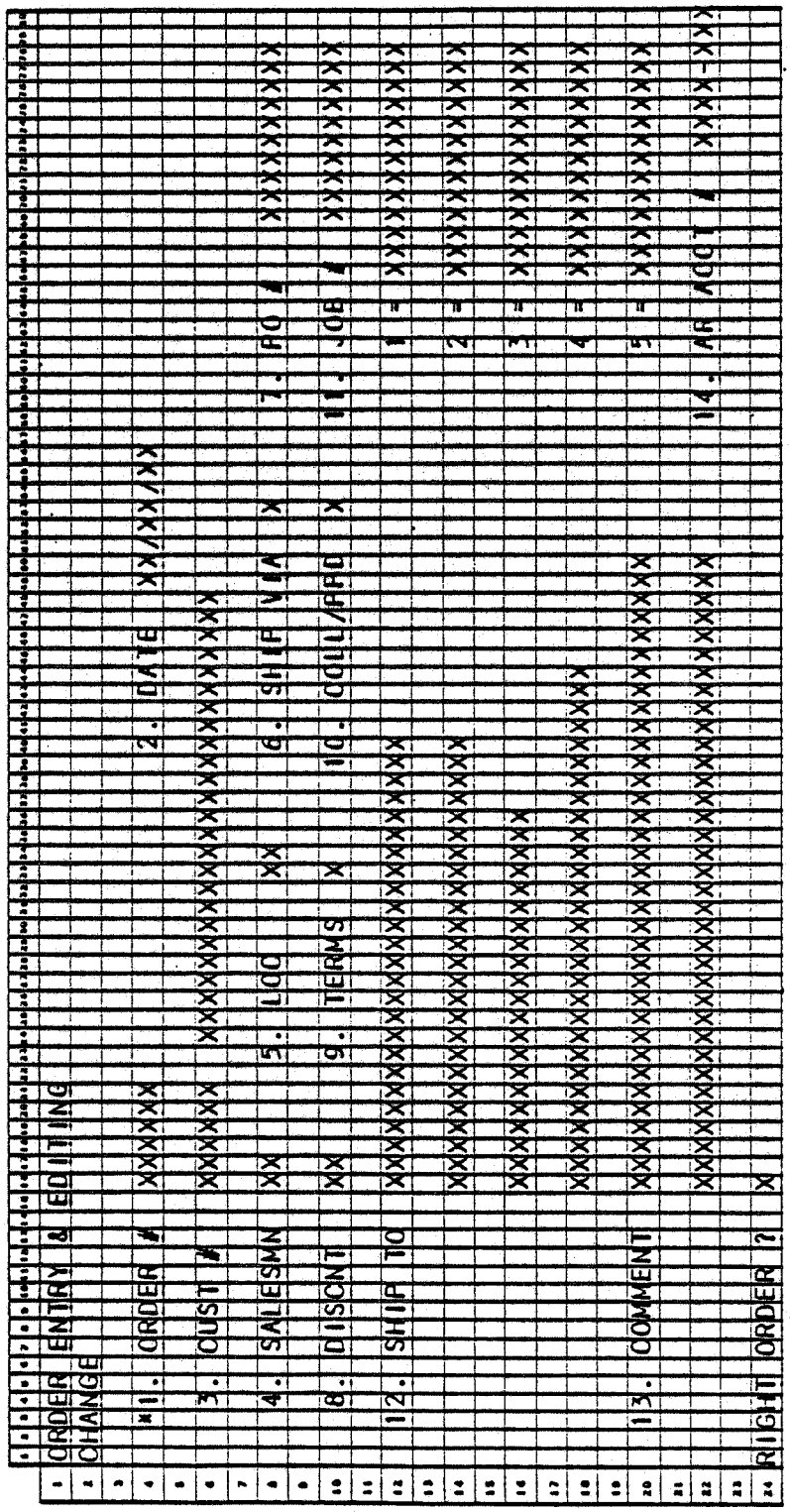




Program: INQUIR

```
ORDER ENTRY & EDITING
INQUIRE    ORDER: XXXXXX  CUST: XXXXXXXXXXXXXXXXXXXXXXXXX  DISC: XX%
                         PROMISE  REQUEST
ITEM NUMBER/DESCRIPTION   DATE     DATE   QTY-ORD BO-OS   PRICE  DISC  EXT-PRICE

XXXXXXXXXXXXXXX          XX/XX/XX XX/XX/XX  XXXX XXXX/X XX,XXX.XX  XX XXX,XXX.XX
  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXX          XX/XX/XX XX/XX/XX  XXXX XXXX/X XX,XXX.XX  XX XXX,XXX.XX
  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX












TOTAL QTY:   XXXXXX                                      TOTAL ORDER: XXX,XXX.XX

TYPE RETURN TO SEE NEXT ITEM                                      OR TO CONTINUE
```

Program: CHANGE:ORDCN



Remarks: * NO CHANGE TO ORDER NUMBER, CUSTOMER NUMBER, OR LOCATION.

Program: CHANGE:LINCN

```
ORDER ENTRY & EDITING
CHANGE        ORDER: XXXXXX  CUST: XXXXXXXXXXXXXXXXXXXXXXXXX  DISC: XX%



                         *1. ITEM #          XXXXXXXXXXXXXXX

                          2. DESCRIPTION     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

                          3. PROMISE DATE    XXXXXX

                          4. REQUEST DATE    XXXXXX

                          5. QTY ORDERED     XXXX

                         *6. QTY B/O OR O/S  XXXXB/O

                          7. PRICE           XXXXXXX

                          8. DISCOUNT        XX           EXTENDED PRICE:  XXXXXXXXXXX

                        **9. ITEM COST       XX,XXX.XX

RIGHT LINE ITEM ?  X
```

```
FIELD # TO CHANGE  XX
ITEM NOT IN ORDER - DO YOU WANT TO ENTER THIS ITEM ? X
```

Remarks: * NO CHANGE TO THESE FIELDS. LINE 9 DISPLAYS ONLY FOR ??? ITEMS (MISC. CATCH-ALL). IF NEW ITEM IS BEING ADDED, PRICE AND DISCOUNT WILL BE AUTOMATICALLY FOUND AND DISPLAYED AS PER PROCEDURE OUTLINED IN ORDER LINE ITEM-ADD DATA ENTRY SPECIFICATIONS.

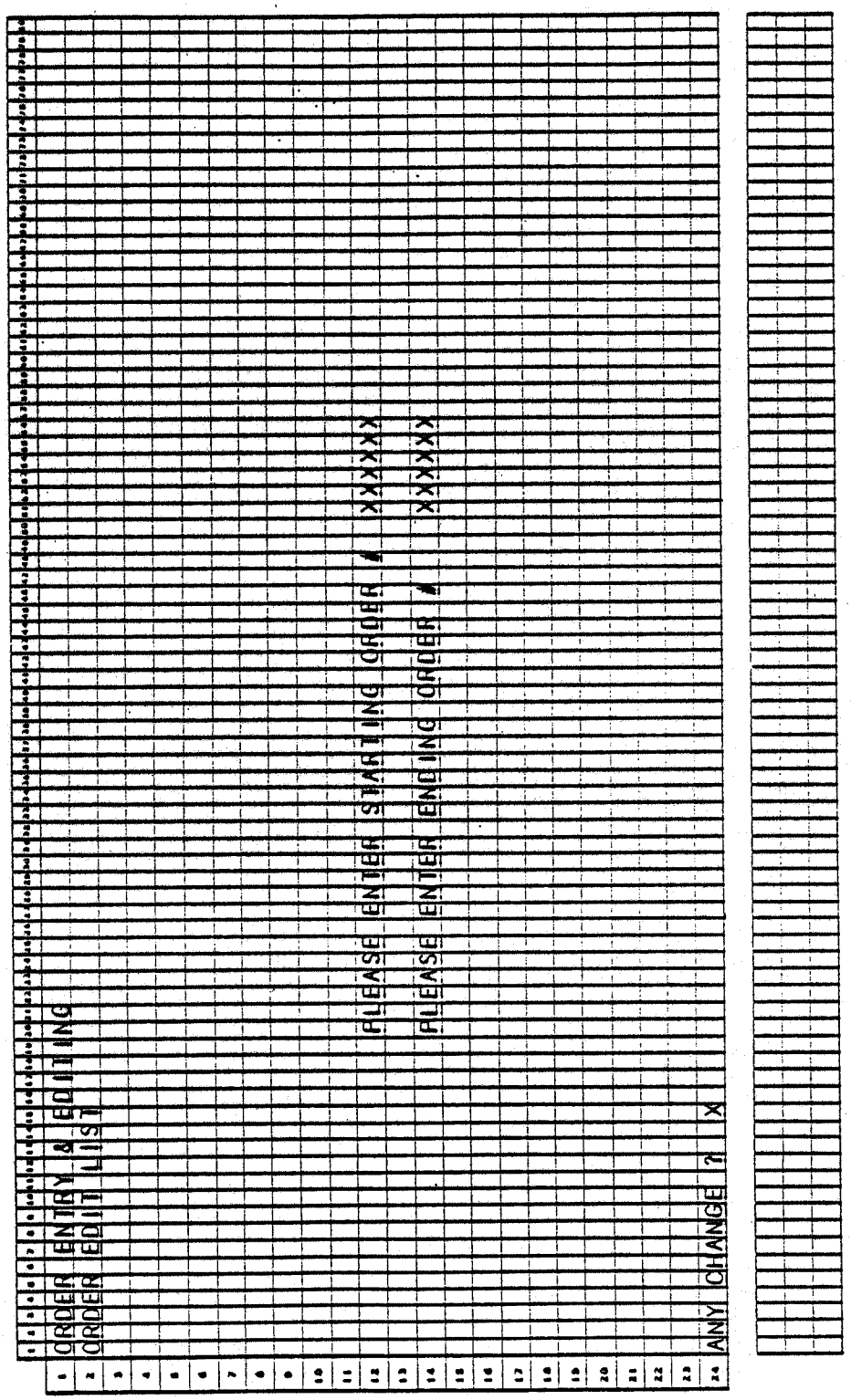
** APPEARS ONLY IF ITEM AVERAGE UNIT COST IN (1) IS NOT IN THE RECORD.

Program: CANCEL

```
ORDER ENTRY & EDITING
CANCEL

  *1. ORDER #   XXXXXX                      2. DATE  XX/XX/XX

   3. CUST #    XXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXX

   4. SALESMN   XX      5. LOC    XX        6. SHIP VIA  X       7. PO #    XXXXXXXXXX

   8. DISCNT    XX      9. TERMS  X        10. COLL/PPD  X      11. JOB #   XXXXXXXXXX

  12. SHIP TO   XXXXXXXXXXXXXXXXXXXXXXXXX

                XXXXXXXXXXXXXXXXXXXXXXXXX

                XXXXXXXXXXXXXXXXXXXXXX

                XXXXXXXXXXXXXXXXXXXXXXXXXXXXX

  13. COMMENT   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX    14. AR ACCT #  XXXX-XXX

OK TO CANCEL ?  X
```

```
ORDER CANCELLED                                              CR TO CONTINUE
```

Program: ORDEDT



ORDER ENTRY & EDITING APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Entry of orders - includes add, inquire, change, cancel and Order Edit List.

Input:	Files Updated:	Output:
KBD	ORDHDR	Order Edit List
ITMIDX	ORDLIN	
ITMMAS	ITMMAS	
CUSIDX		
CUSMAS		
PRDIDX (if BOMP installed)		
PRDSTR (if BOMP installed)		
SHIPTO		
SHPVIA		
ARTERM		

Enter Module From: CPMENU When Done Return To: CPMENU

Programs in Module: OEMNU, ORDADD, INQUIR, CHANGE, ORDEDT, CANCEL

Program Functions and Notes:

OEMNU

This will look like a modified transaction Maintenance module. There will be five main programs, two sorts, and an update counter program. The menu (write a subroutine called OEMNU) will be in the first main pagram. In addition, the first main program will handle add orders mode. The second program will contain the inquire orders mode, the third will contain change orders, the fourth, cancel orders, and the fifth will be the Edit List program.

ORDADD

In the parent program (OEMNU) open the necessary files in this order:

CUSIDX, ITMIDX, ORDHDR, ORDLIN, ITMMAS, SHIPTO, SHPVIA, ARTERM

Read the control record from the Item Master file. If the type system is three or four, open the PRDSTR file. Finally, open the CUSMAS file.

Call SCRN1, which is the subroutine to handle the first screen, that is, the order header.

1. Accept input per ADD HEADER screen. Normal mode for entry of customer order number is to press the RETURN key which will give the next sequential order number by first retrieving the last order number used from the Order Header Control record. Allow an order to be manually entered. In this case, do not update the last used order number from the control records.

If RETURN is used to get next sequential order number, update the last order number used in Control record of ORDHDR.

Search the Order Header file to be certain an order does not already exist with this order number.

Attempt to add the record. If not enough room, display "FILE FULL" message and exit to OEMNU.

Accept input from ADD HEADER screen as follows:

2. Default date to system date on RETURN.

3. No binary search of CUSIDX file, find customer, read Customer Master record, and if CUSCD is not equal to '***', display the customer name. Otherwise, do not allow entry of an order to this customer. Save the customer type, tax flag, credit limit, and discounts array in order to pass them to the next subroutine.

4. Salesman defaults to the salesman in this customer's record on RETURN.

5. Location defaults to location on customer record on RETURN.

6. Ship via and terms are one-digit codes corresponding to arbitrary tables of user's choice (these tables are maintained in separate maintenance modules, SHPMNT and ATMMNT respectively).

7. Default to blanks for purchase order number.

8. Default to zero on RETURN for discount.

9. Default to terms associated with this customer's record on RETURN.

10. Default to 1 for COLL/PPD on RETURN.

11. Default to blanks on RETURN for job number.

12. Default ship-to name and address to the bill-to address on RETURN. Allow user to enter four-digit numeric value here.

    Check ship-to number to see that is numeric. If so, search SHIPTO for this customer's ship-to number, and display the SHIPTO name and address from the Ship-to record.

13. Default to blanks on RETURN for comment lines.

14. If multiple A/R accounts was selected at entry time, default to whatever was set up.

Ask "FIELD # TO CHANGE" and accept changes.

When no more changes are indicated again write the ORDHDR record, and return to the calling program.

Call a subroutine to handle addition of line items as per ADD LINE ITEM screen. Do not limit the number of line items per order. If the screen gets full, scroll the screen up one line to make room for the next item entry.

Accept input from ADD LINE ITEM screen as follows:

1. Search the Item Master Index (ITMIDX) for item number and read the ITMMAS record. Display item description automatically, except for "???" item, in which case, allow description to be entered manually. Check the activity flag. If this item is non-active, do not allow it to be ordered.

   Permit user to override description by pressing CTRL/U (that is, press the CTRL and the U keys simultaneously) and then input item description.

2. Accept the request date; default is to the order date.

3. Accept the quantity to be ordered. On RETURN default to quantity ordered of one.

4. Spin the prices array to try and find a match between the Price codes and the Customer code which has been passed from the Screen one subroutine. If a match is found, display this price. If no match is found, use the first price in the array.

5. Spin the locations array for this item and try to match the location of the order which has been passed from the first screen, with an active location for this item. If no location match is found, do not allow this item to be ordered.

6. Stocked/Non-Stocked Items

   Stocked: (Item's Stocked Flag is "S".)

   Check quantity ordered against available quantity in inventory at that location (QTYONH-QTYCOM) and proceed as follows:

   A. If entire quantity is available, accept it and go on.

   B. If entire quantity is not available, check the Backorder code of the item.

      1. If Backorder code equals "0" (can be backordered) allow four options (cancel, backorder balance, backorder all, or override) and handle according to user's selection. If any part is backordered, display the backordered part immediately to the right of the order quantity. If cancel, wipe out the line. If override, ignore any backordere handling. If there is a B/O portion, set LSTATS equal to one ORDLIN record.

      2. If Backorder code equals "1" (cannot be backordered) allow three options (cancel, order available quantity, or override).

If the second option is selected, display the out-of-stock quantity immediately to the right of order quantity, also set LSTATS equal to 2 in ORDLIN record.

Do not check quantity available for "???" item.

Display price taken from inventory record. Permit user to override this price by pressing CTRL and the U keys simultaneously, and then input override price.

Spin the array which has been passed from the first screen of Product codes and discounts against the Product code of this item. If a match is found, use this discount as the line discount. If no match is found, spin the trade discount array from the ITMMAS Control record against the product category. If a match is found, use the corresponding discount as the line discount. If no match is found, use no discount. Display the discount, if any, and allow it to be accepted or overridden (CTRL/U). Calculate the extended price by first applying the line discount, then applying the order discount.

If item is "???" item, ask for and accept unit cost on the next line down on the screen. Accept it to two decimal places.

Ask if line is OK. If it is not, erase and start over. If it is OK, compute total running quantity and price and display on line 11. Also, compute remaining credit. If the remaining credit is exceeded by ordering this line, display a message to that effect.

Non-Stocked: (Item's Stocked Flag is "N")

If the item is non-stocked and the TYPSYS is greater than four, read the Control record of the PRDSTR file.

The Bill of Material Processor application must be installed to handle non-stocked items.

Search the Product Structure (PRDSTR) file for an occurrence of this item as a parent.

If it is not found, do not allow this item to be ordered.

If it is found, backup to the first occurrence of this item as a parent.

Call a subroutine named COMIT which will explode the bill and commit (allocate) the inventory at the first stocking level.

Set a counter to the first record and a bad bill flag of 0 before starting.

Blow down the bill in the manner described for MLTBOM (SEE BOMP documentation manual).

Ignore deleted and non-active structures.

For each new record that fits the criteria, check the component for stocked status. That is, STOKED is equal to "S".

If it is not, continue down the bill until you find a stocked item or run out of records.

If you run out of records, move back up one level without doing any processing.

Once a stocked level is found, check the controlled flag (CNTRLD = "C"). If it is not equal to "C", do not adjust any quantities allocated and move back up one level on the bill.

If it is controlled, explode the quantity ordered through the quantities per of the bill of this level. Round up for any fractional part greater than or equal to one-half. If the final quantity is less than or equal to one-half, allocate a quantity of one.

Spin the locations for this item to match the MFGLOC with an active location. (If a match is not made, an error condition exists.)

Increase the quantity allocated at this location and write the record back to the ITMMAS file.

Return to the calling program.

If an error condition exists, flip the error flag and reset the counter to the first record occurrence. Implement logic to de-allocate the inventory already allocated if any error condition is found.

If the next line will cause a screen overflow, roll the screen up one line thereby eliminating the top line displayed and allowing one more line to be entered.

Allow "END" to be typed in Item Number field to terminate line item input.

For each item, increment the LINSEQ.

After all items have been added rewrite ORDHDR record, after inserting the last used LINSEQ.

If a line is erased for some reason, allow RETURN to default to the previously entered item number.

INQUIR

Disply the order header screen as per the INQUIRE screen, and display an asterisk before the index number of 1 to indicate inquiry mode, i.e., no change allowed.

Accept order number and search the ORDHDR file for an occurrence of this header.

Display the header information as found. Display a message and accept RETURN to display the line items of this order.

Display the header information of the line item screen. Display individual lines as found from the Order Line Item file until you have run out of room on the screen (9 lines displayed) or you run out of lines in the Order Line Item file. If all lines fit on the screen, display the quantity and running dollar totals on line 11. If they do not, accept RETURN key to scroll all lines up one position and display the next line on the bottom (11th) line. Repeat display of lines in this manner until they are exhausted, at which time scroll once more to allow running totals to be displayed.

After totals are displayed accept return to go back to the header screen for more order numbers to be inquired upon.

CHANGE

(Also the only way to add line items to an existing order.)

Handle this module in a similar fashion to add mode of Order Entry. The major difference here will be how the line item screen is formatted, and in this module there will be cancellation logic. That is, we will not only be able to increase the quantity for inventory items, but we will also be able to decrease quantity allocated.

As in ADD mode, open all files necessary in the controlling program.

Chain to a subroutine to handle the order header in change mode.

Display the header as you would normally and accept an order number from the user.

If this order is found in the Header file, display all the information and ask if this is the correct order.

If it is, allow changes to any of the fields except order number, customer number, and location of order.

If discount is changed, remember to carry this to the line change program so that the order discount may be changed in each of the line items.

Allow normal ABORT logic to cancel changes.

If abort change has not been indicated, call the subroutine to handle change of line items.

Display the change line item screen.

Accept an item number in the Item Number field.

Locate this item in the Item Master file if it exists.

Find the item in the Order Line Item file if it exists.

If it is in the Line Item file, display it and ask if this is the right line item.

If it is, allow changes to any of the fields except item number and quantity backordered or out-of-stock.

Use identical logic to order entry line item and when changes are made to increase the quantity ordered or decrease the quantity ordered.

Adding a Line Item to an Existing Customer Order:

If you do not find this item in the Line Item file, or if the operator indicates that this is not the correct line and another line cannot be found, ask if the operator wishes to add this line to the order.

If yes is indicated, step through the fields in a manner identical to the logic in Order Entry Line Item fields.

Step through identical logic for product structure update if those files exist.

Rewrite the records that have been changed, write out any new records for the Line Item file, and return to the top of the program to accept another order number.

NOTE: Accept zero quantity in change mode as a means of cancelling an individaul line item from an order. In other words, if the quantity ordered on a line is changed to zero, further processing will ignore this line as though it were cancelled.

CANCEL

Open all the necessary files for normal access.

Display the order header screen as you normally would, but offset the first field with the word "CANCEL".

Accept an order number and locate the order in the Order Header file if it exists.

When found, display the order header and ask if this is the correct order.

If it is the correct order, delete it from the file. Find all occurrences of order lines with this order number in the Line Item file and delete them.

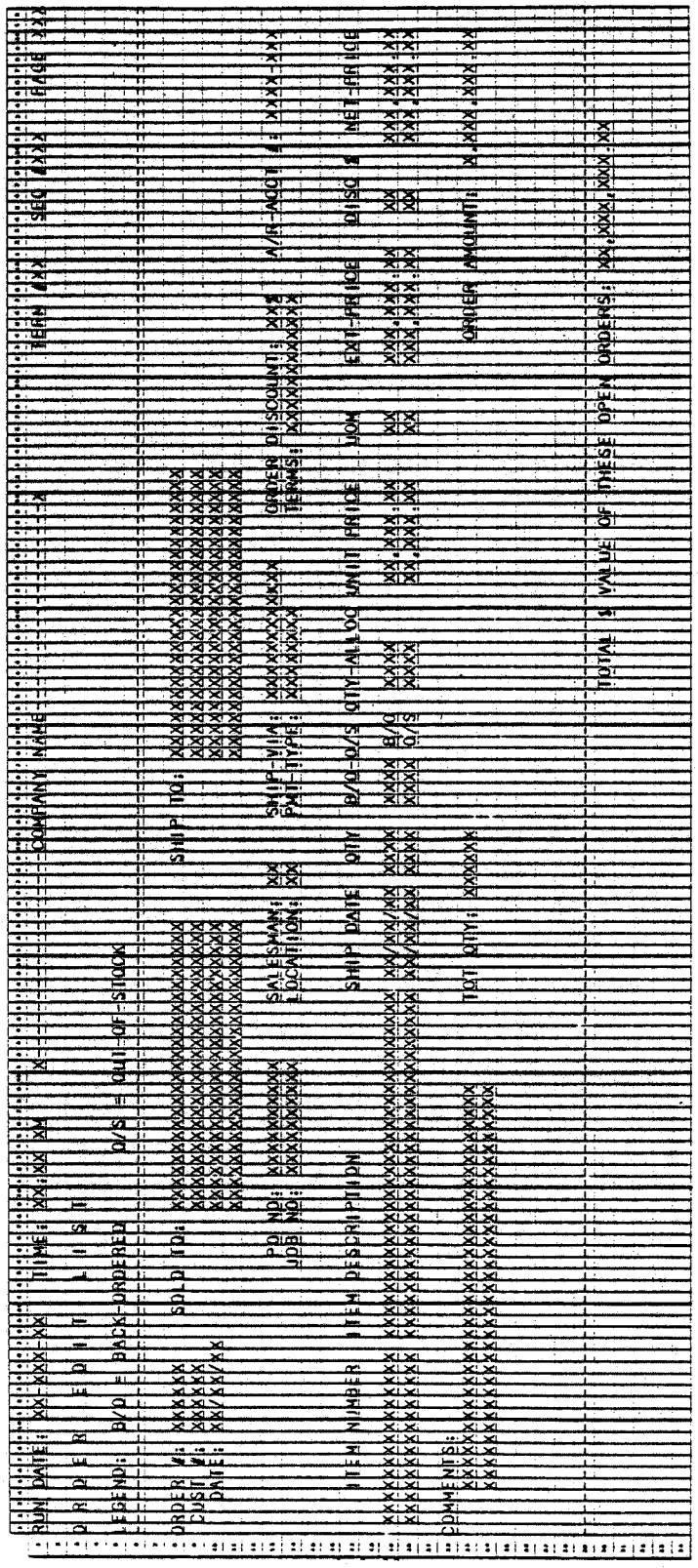
De-allocate all inventory that was allocated to these line items.

After no more line items are found, return to the top of the program for another order number to cancel.

ORDEDT

Write a standard print program as per the Report Format for Order Edit List.

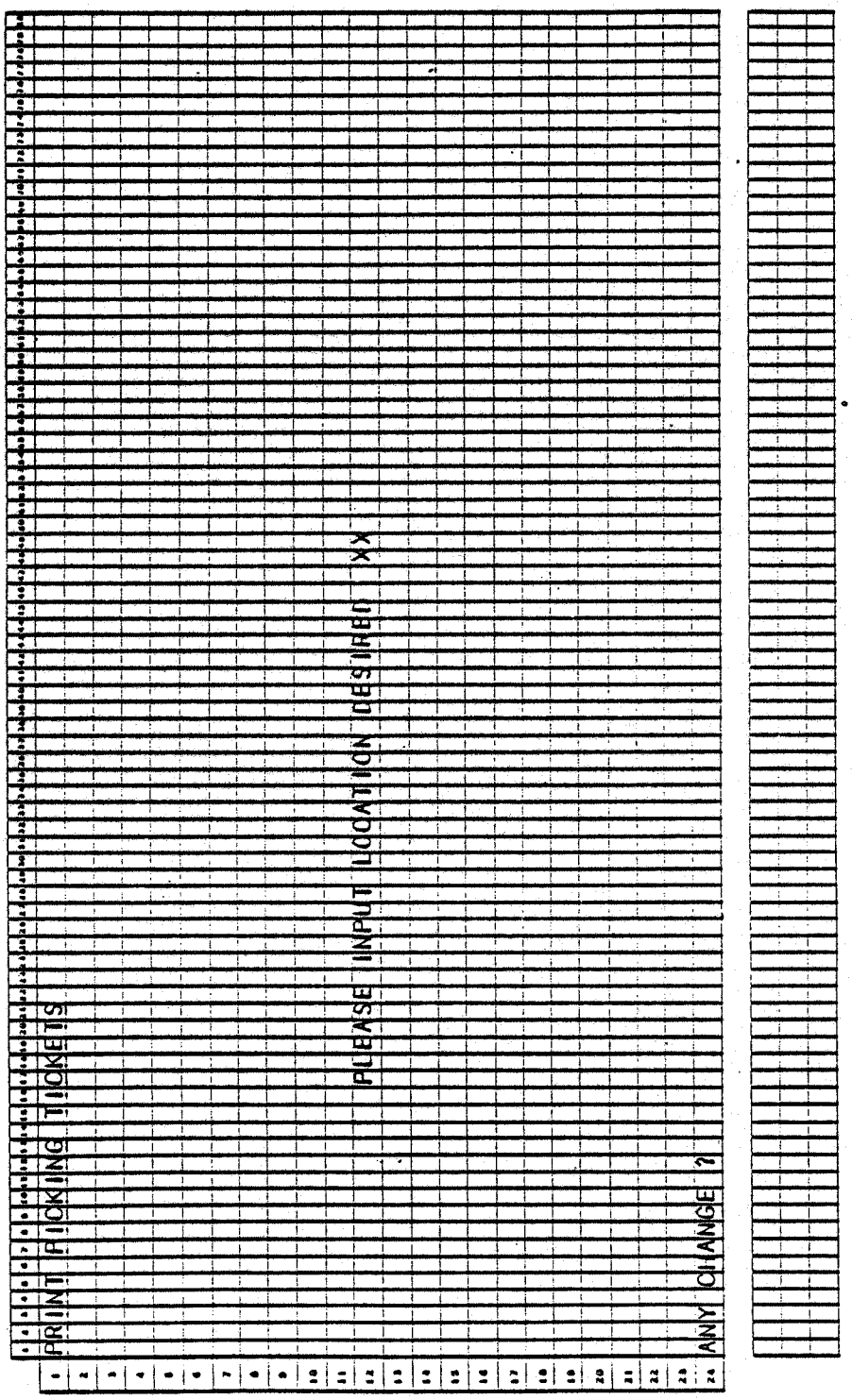
Program: ORDEDT (Order Edit List)



CUSTOMER ORDER PROCESSING PACKAGE
PRINT PICKING TICKETS APPLICATION
DIBOL JUN-84

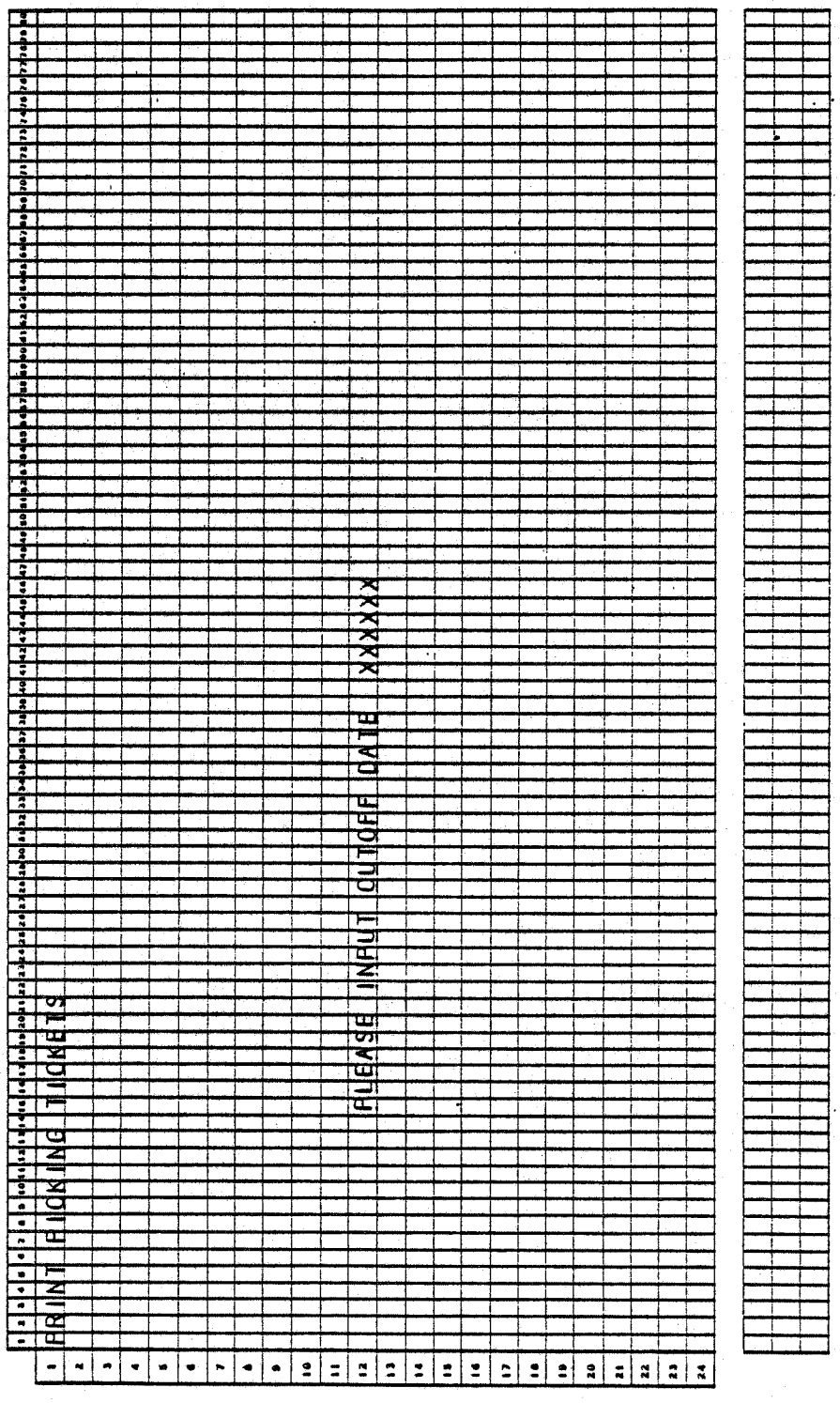
SCREEN FORMATS

Program: LINIDX



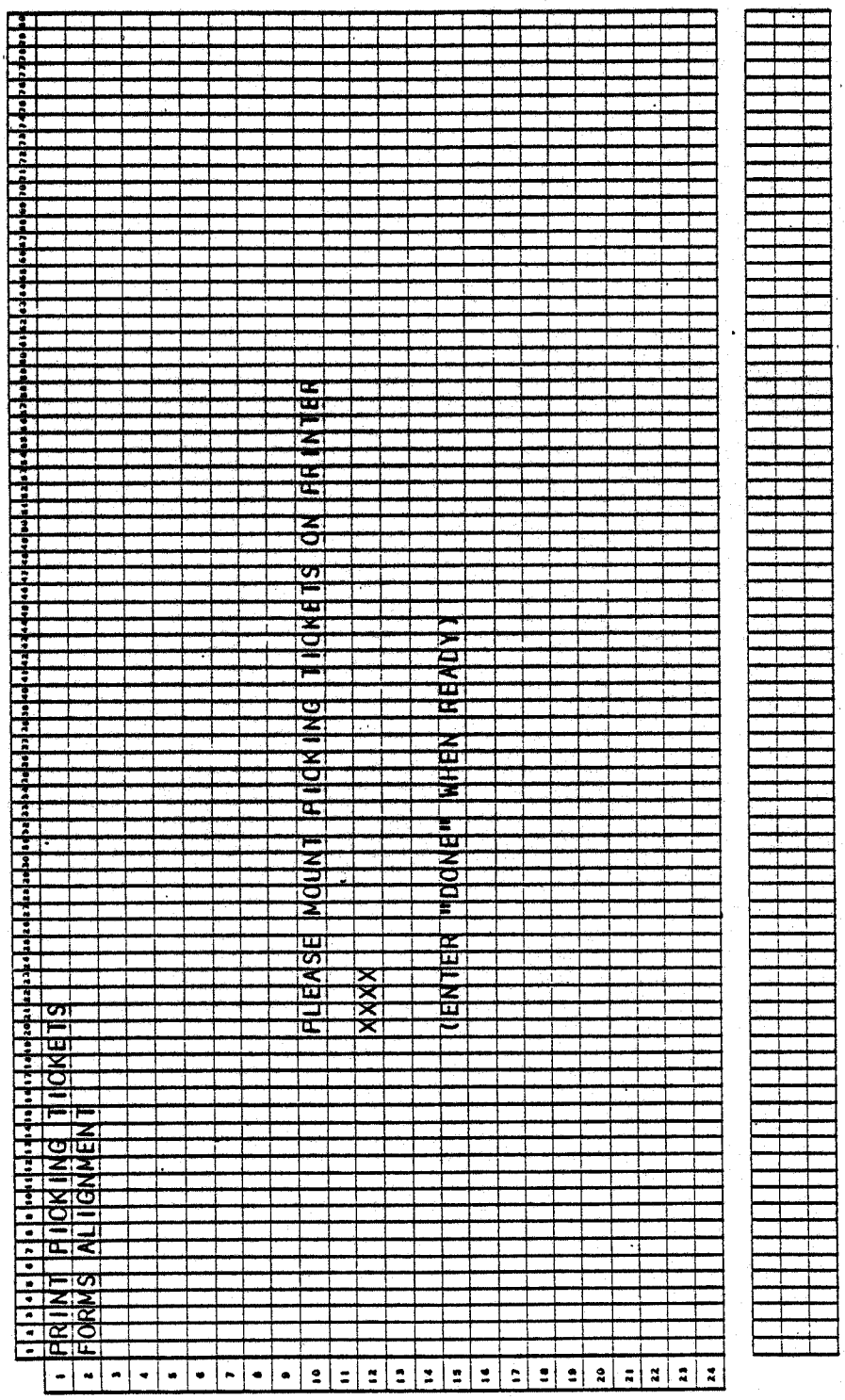
Remarks: ON BLANK INPUT DEFAULT TO I/M'S DEFAULT LOCATION.

Program: LINIDX

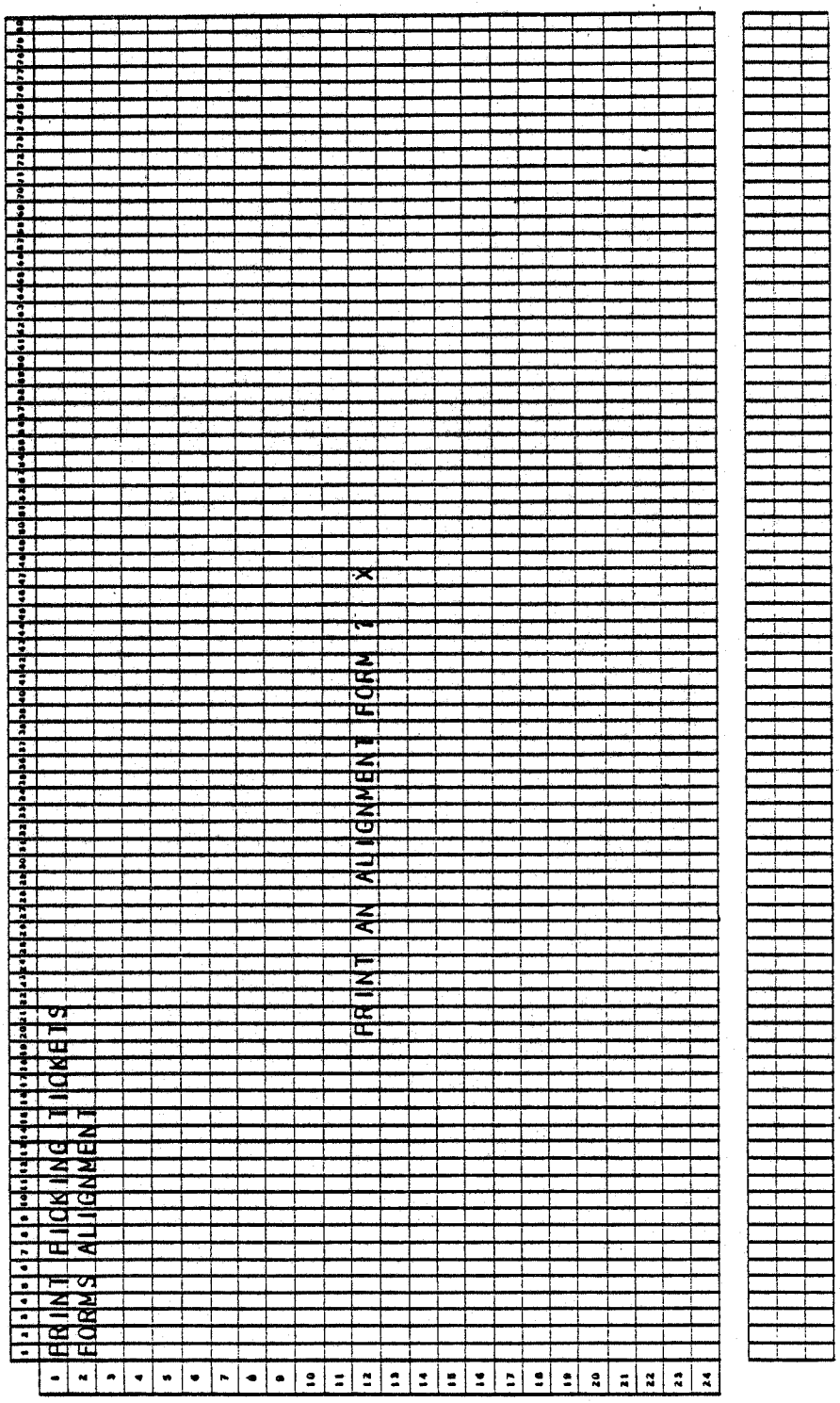


Remarks: DEFAULT IS TO SYSTEM DATE.

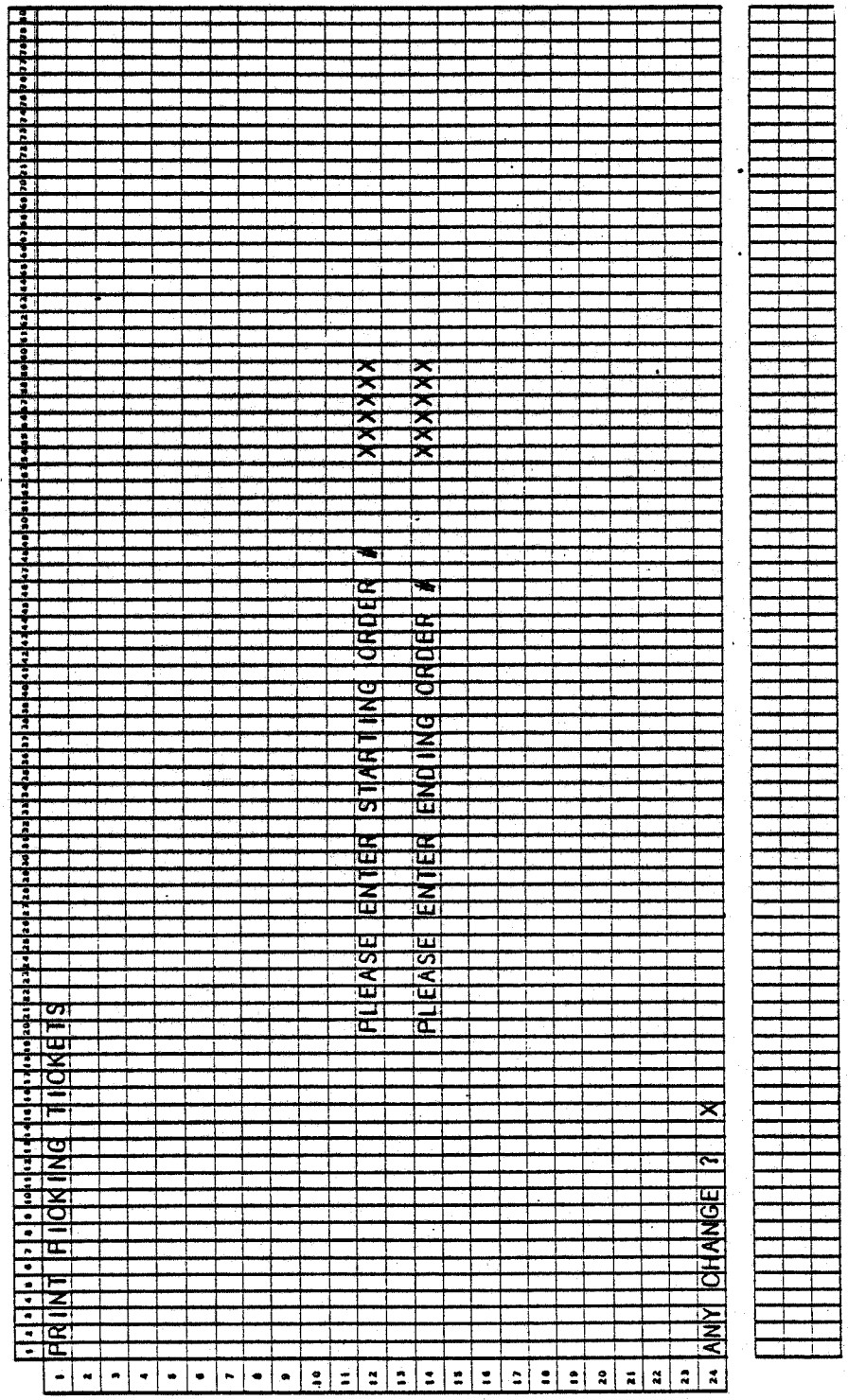
Program: ALNINV



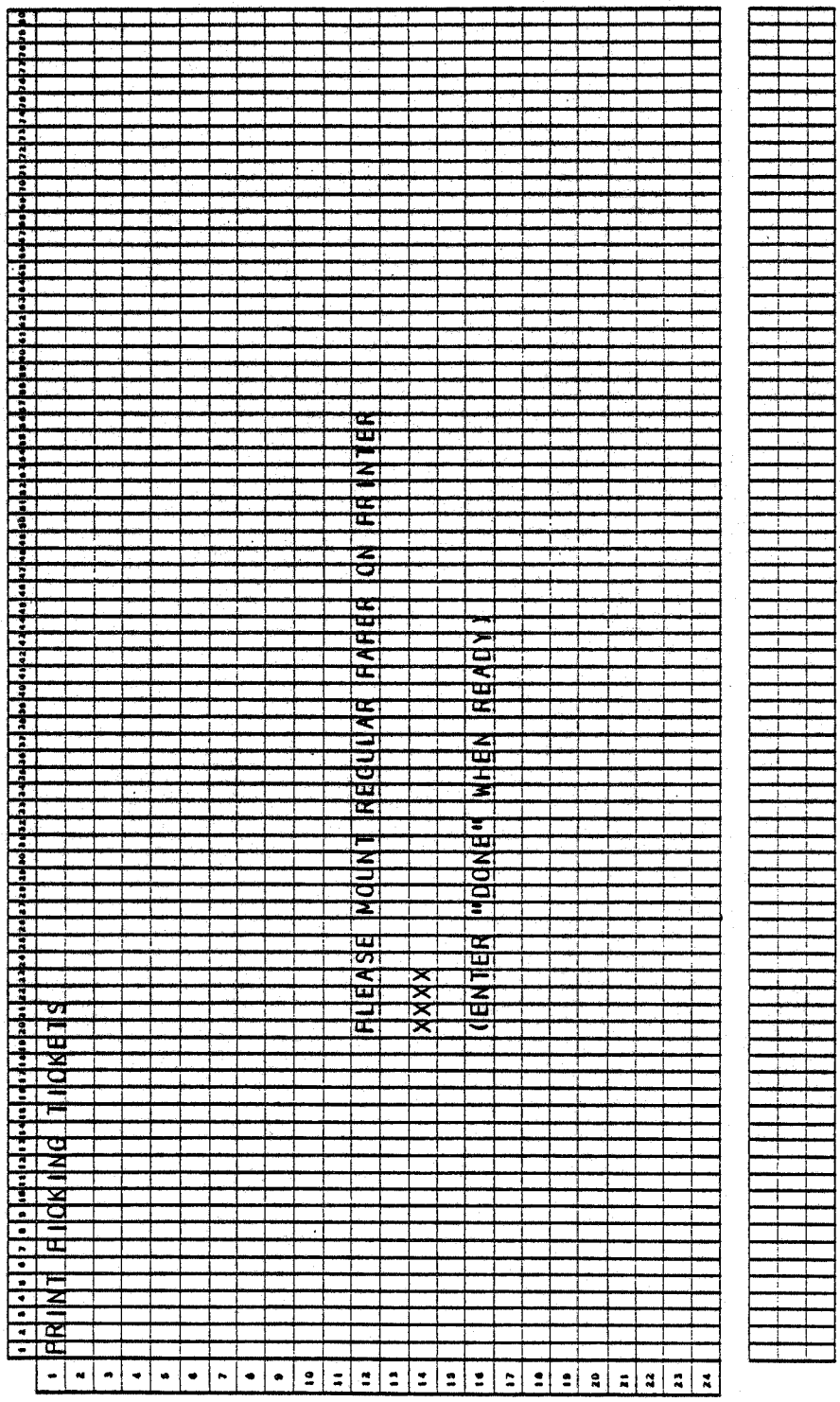
Program: ALNINV



Program: PIKTIK



Program: PIKTIK



PRINT PICKING TICKETS APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Prints Picking Tickets.

Input: ORDHDR
ORDLIN
CUSMAS
CUSIDX
ITMMAS

Files Updated: LINIDX

Output: Picking Tickets

Enter Module From: CPMENU

When Done Return To: CPMENU

Programs in Module: LINIDX, SRTLIX, ALNINV, PIKTIK

Program Functions and Notes:

The program flow for this function is:

LINIDX -- SRTLIX -- ALNINV -- PIKTIK

LINIDX

Display a screen asking for location desired. If blanks are input use the location located in the ITMMAS Control Record field DFLTLO.

Display a screen asking for cutoff date. If RETURN is pressed, default to the system date (today's date).

Sequentially read through ORDLIN file and for those line items which are from the location indicated and whose ship date (promise date if nonzero, else the request date) is less than the cutoff date, build the LINIDX file.

If no lines are found at that location, indicate such and return to the first display screen.

SRTLIX

Sort LINIDX by order number as the major key, and bin number (picking sequence) as the minor key.

ALNINV

Same program as used in invoice job stream in BILLS module.

PIKTIK

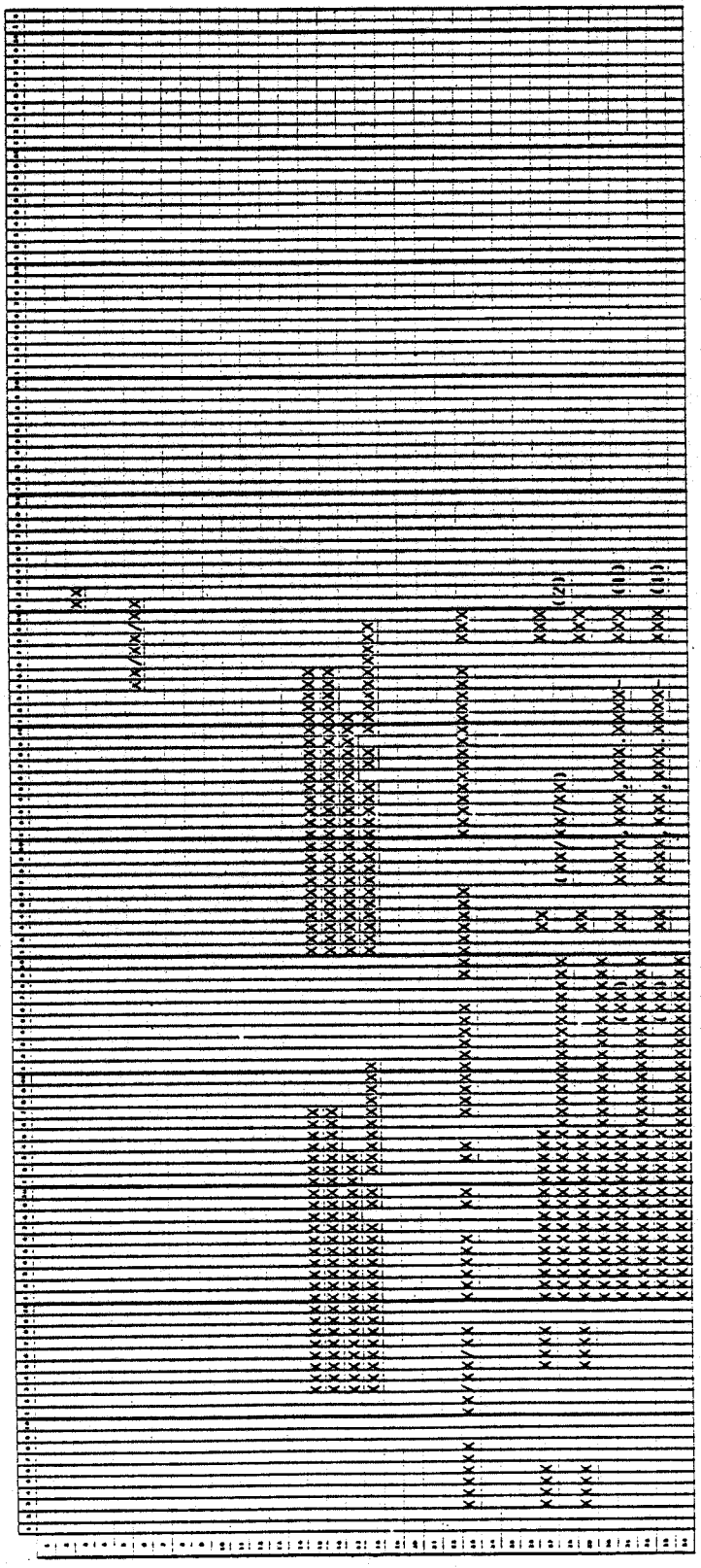
Print one picking ticket per order number. Pull the customer name and address from the customer's file. Blow down bill of materials and pick components for non-stocked items.

If the ship date (promise date if nonzero, otherwise the request date) is not equal to the order date, print the ship date in parentheses, next to the line item description.

Use the LINIDX file to access the ORDLIN file. Delete temporary LINIDX at end of printing.

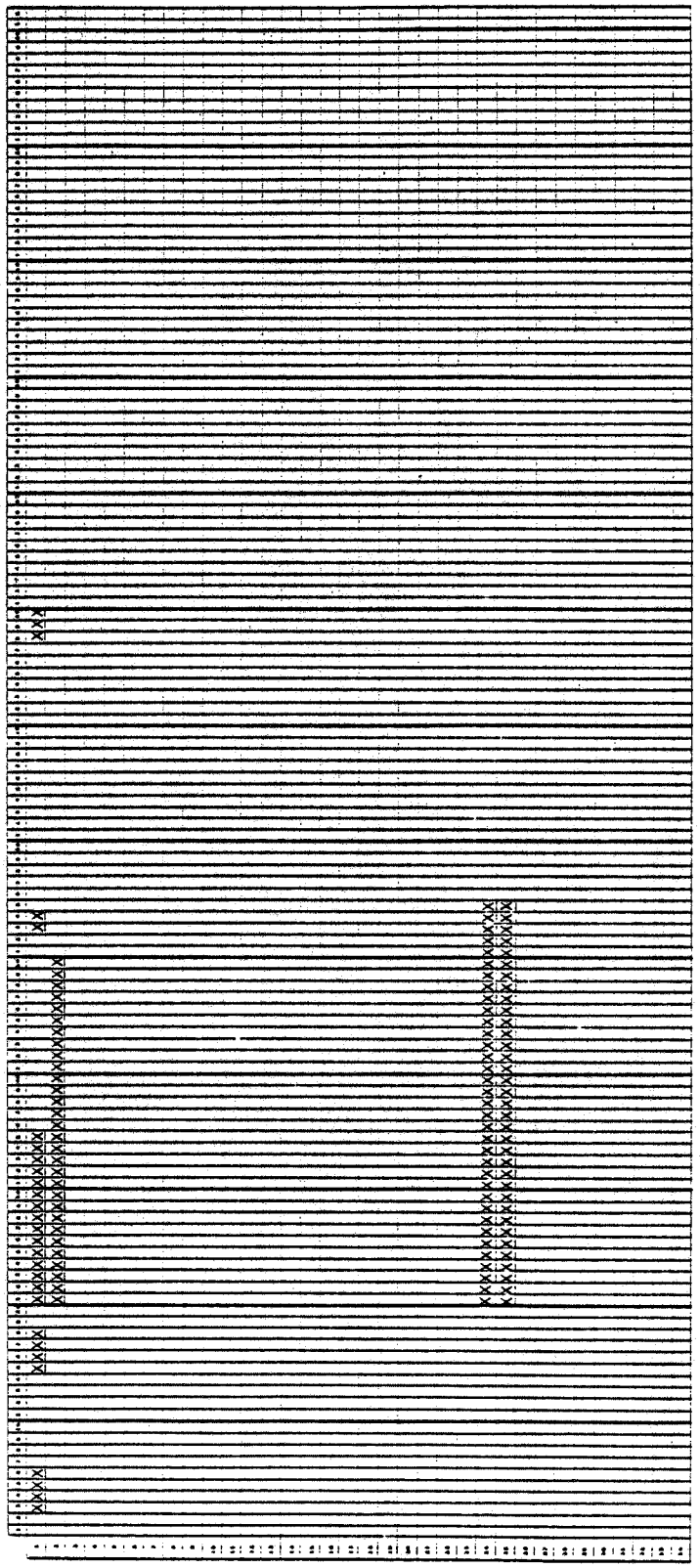
PRINT PICKING TICKETS APPLICATION
DIBOL JUN-84
REPORT FORMATS

Program: PIKTIK (Picking Ticket)



Remarks: (1) THE COMPONENTS OF NON-STOCKED ITEMS WILL PRINT BELOW THE PARENT. THE LEVEL IN THE BILL OF MATERIAL IS PRINTED IN PARENTHESES. THE QUANTITY TO PICK IS SHOWN FOR THE FIRST STOCKED LEVEL. (ONLY IF THE BILL OF MATERIAL PROCESSOR IS INSTALLED.) (2) THE SHIP DATE WILL BE PRINTED IN PARENTHESES FOR PARENT ITEMS WHEN THE SHIP DATE IS NOT EQUAL TO THE ORDER DATE.

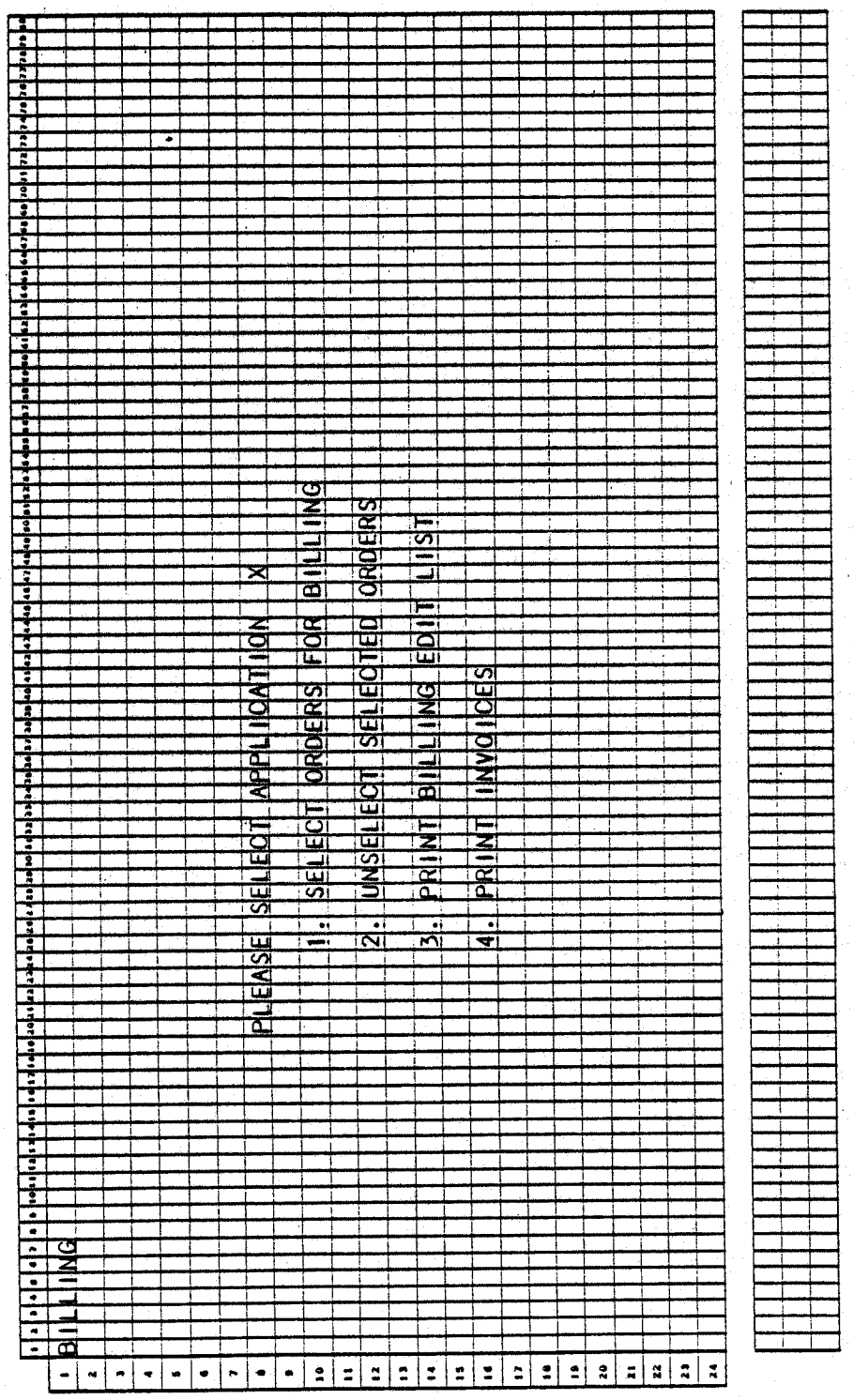
Program: PIKTIK



CUSTOMER ORDER PROCESSING PACKAGE
BILLING (PRINT INVOICES) APPLICATION
DIBOL JUN-84

SCREEN FORMATS

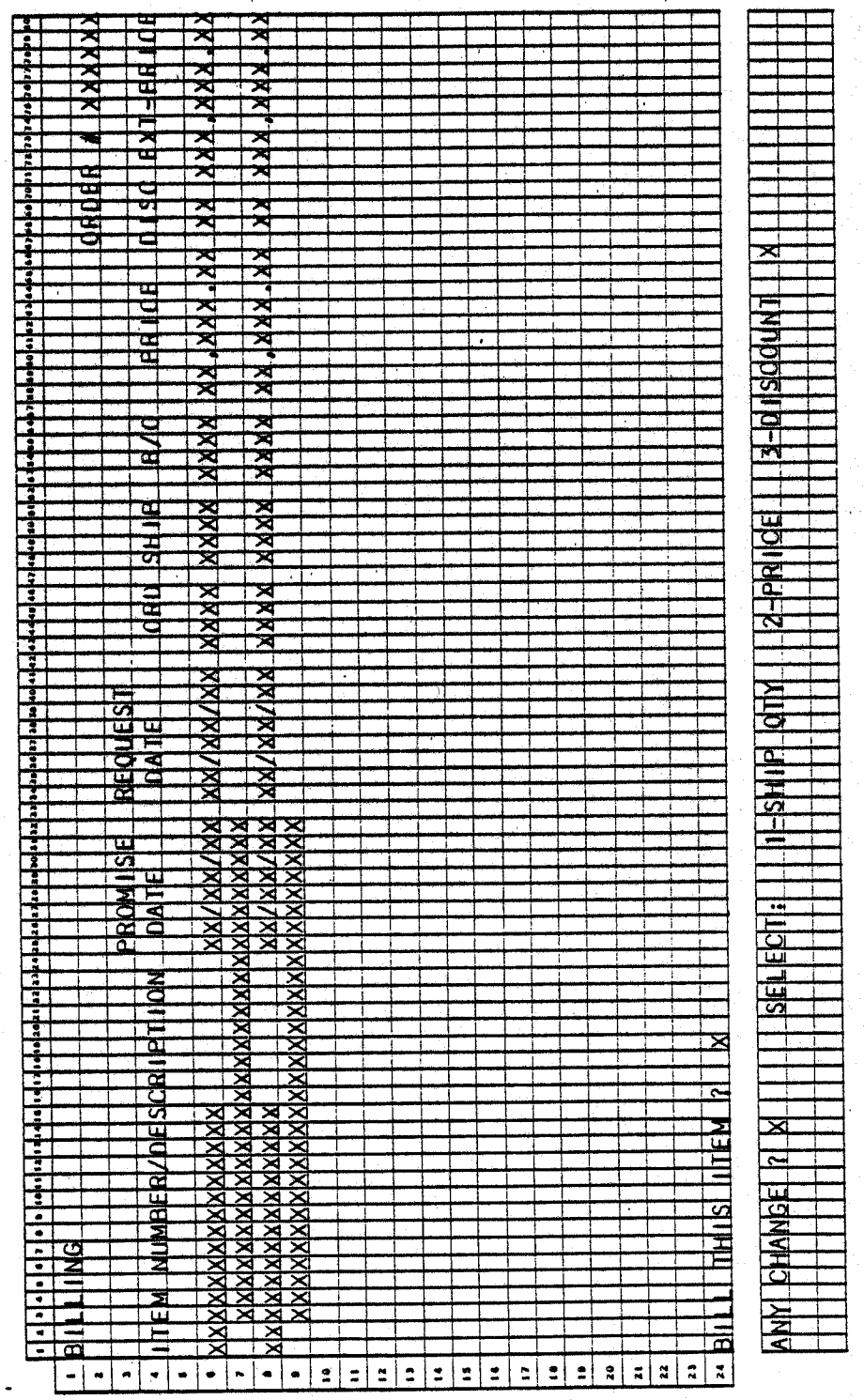
Program: BILLS:BLMNU



Program: BILLS:ORDBL

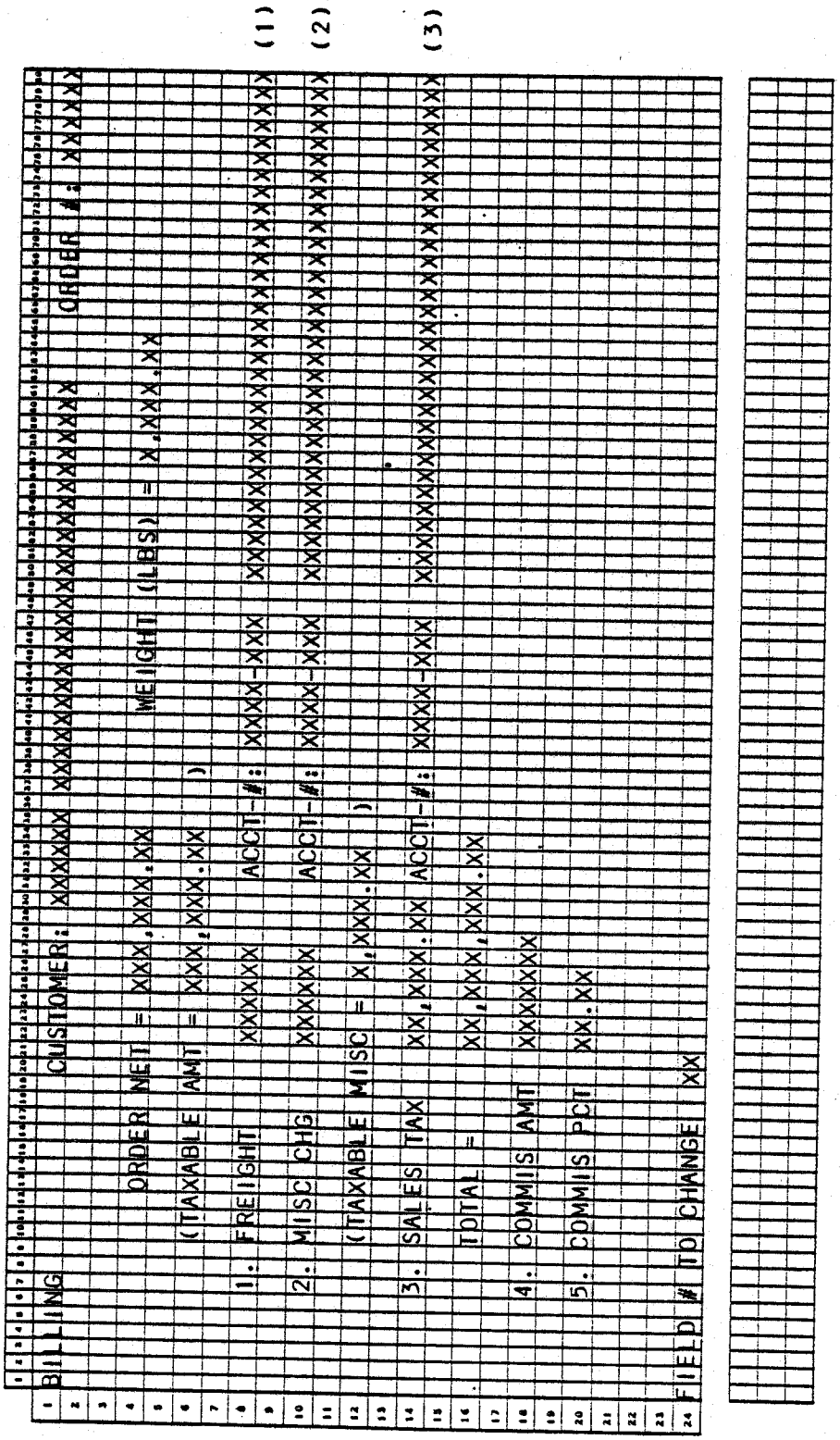
```
BILLING


   1. ORDER #   XXXXXX                  2. DATE  XXXXXX

   3. CUST #    XXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXX

   4. SALESMN   XX      5. LOC    XX    6. SHIP VIA  X      7. PO #   XXXXXXXXXX
                                                     XXXXXXXXXXXXXXX
   8. DISCNT    XX      9. TERMS X     10. COLL/PDD  X     11. JOB #  XXXXXXXXXX
                                 XXXXXXXXXXXXXXX
  12. SHIP-TO   XXXX
      13.  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      14.  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      15.  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      16.  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

   17. COMMENT  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX      18. AR ACCT #  XXXX-XXX

               XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


FIELD # TO CHANGE
```

Program: BILLS:LINBL



Remarks: "ANY CHANGE ?" DEFAULTS TO 1 ON RETURN. SCREEN WILL SCROLL TO HANDLE MORE THAN A PAGE OF ITEMS.

Program: BILLS



Remarks: SALES TAX DISPLAYS AUTOMATICALLY AS DOES ORDER NET, WEIGHT, TAXABLE AMT, TAXABLE FREIGHT AND TOTAL. CUSTOMER TYPE MATCHES AGAINST SALESMAN'S ARRAY TO DETERMINE THE COMMISSION PERCENT AUTOMATICALLY DISPLAYED. (1) DISPLAYED ONLY IF MULTIPLE MISCELLANEOUS ACCOUNTS. (2) DISPLAYED ONLY IF MULTIPLE FREIGHT ACCOUNTS. (3) DISPLAYED ONLY IF MULTIPLE SALES TAX ACCOUNTS.

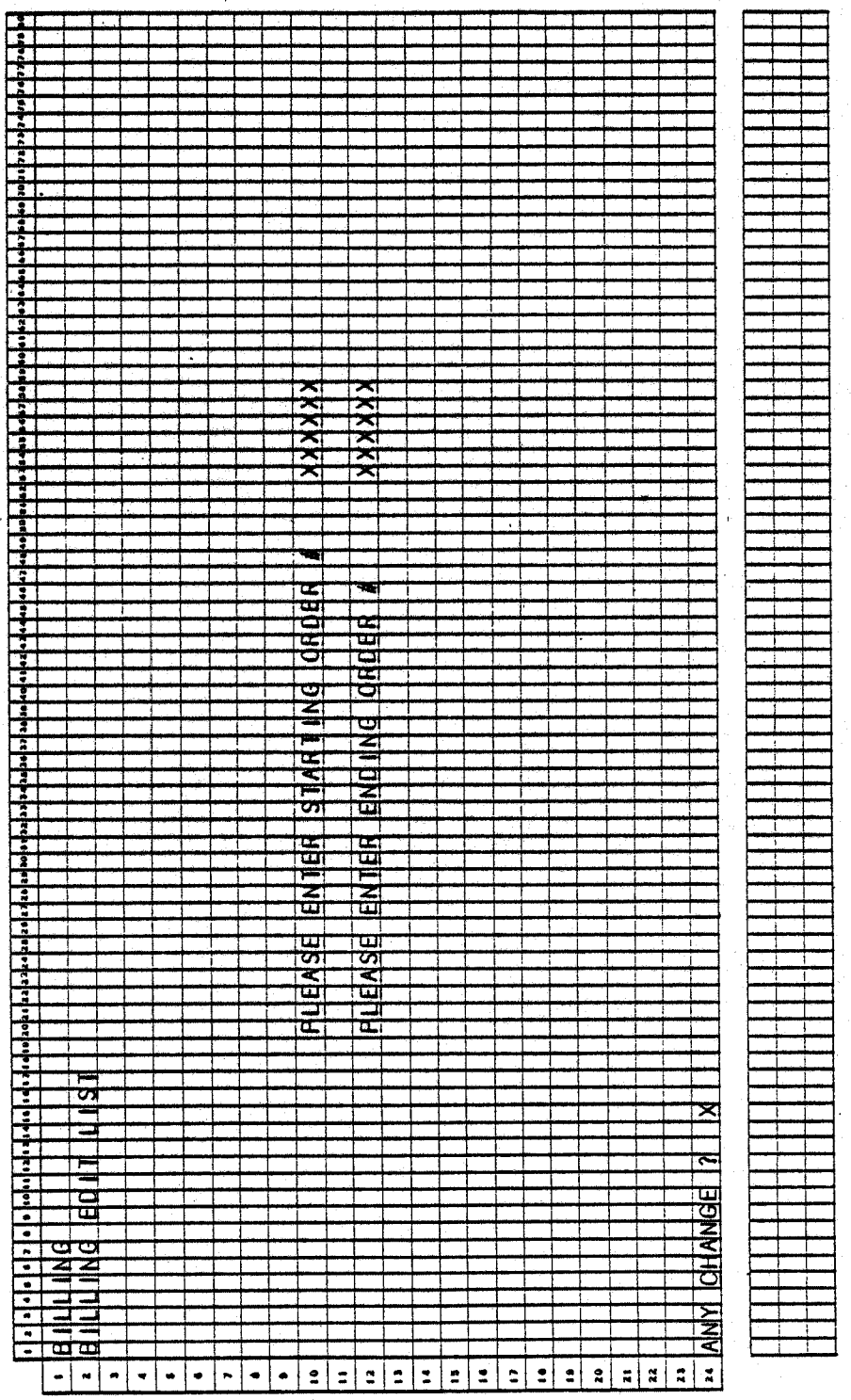
Program: UNBILL

```
BILLING

   1. ORDER #     XXXXXX                  2. DATE  XXXXXX

   3. CUST #      XXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXX

   4. SALESMN     XX     5. LOC   XX      6. SHIP VIA  X      7. PO #   XXXXXXXXXX
                                                       XXXXXXXXXXXXXXX
   8. DISCNT      XX     9. TERMS X      10. COLL/PDD  X     11. JOB #  XXXXXXXXXX
                                  XXXXXXXXXXXXXXX
  12. SHIP-TO     XXXX
      13.    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      14.    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      15.    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      16.    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

   17. COMMENT     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX      18. AR ACCT #  XXXX-XXX

               XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


RIGHT ORDER ?  X
```

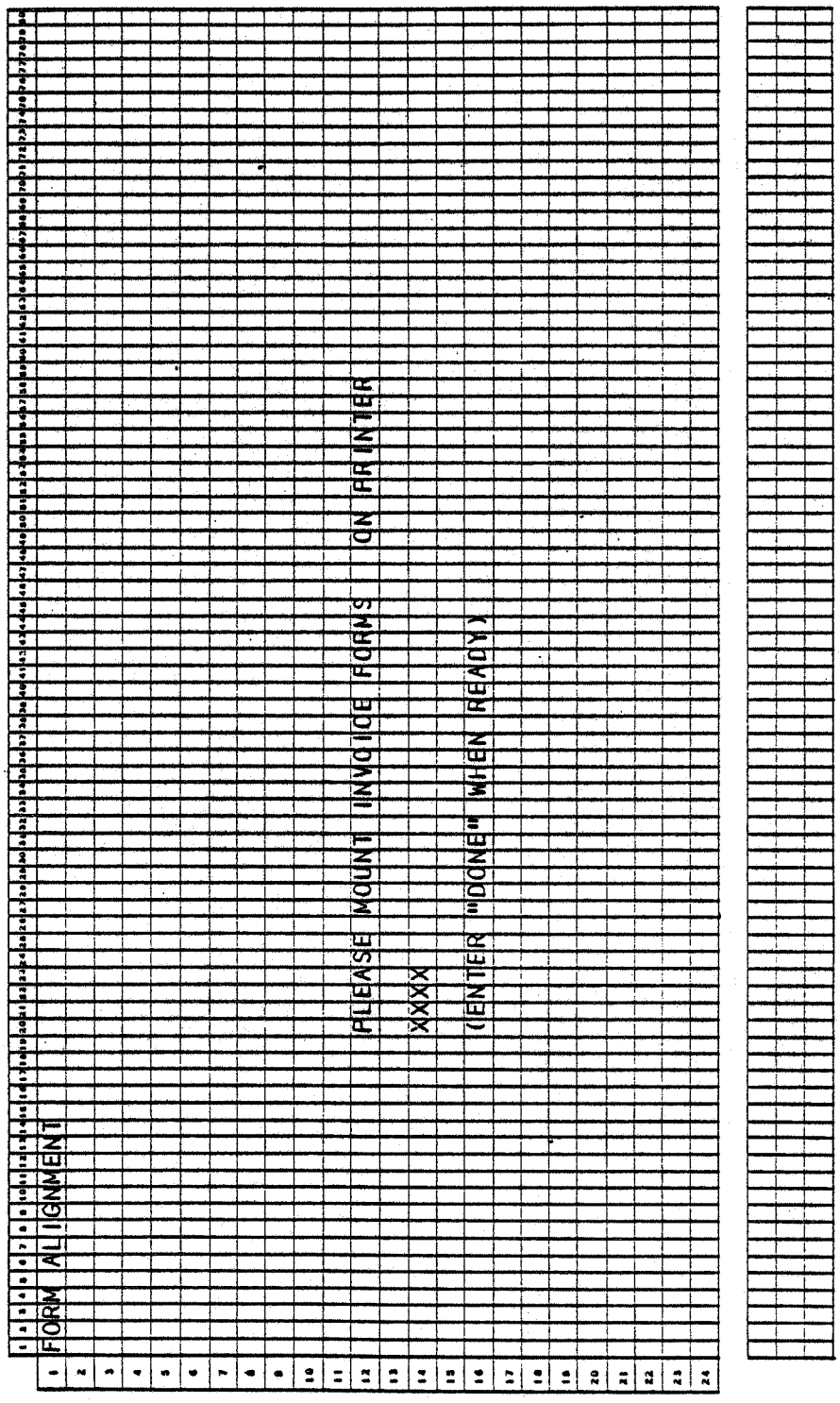
Program: UNBILL

```
BILLING
UNSELECT

  *1. ORDER #   XXXXXX                        2. DATE  XX/XX/XX

  *3. CUST #    XXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXX

   4. SALESMN   XX      *5. LOC     XX      6. SHIP VIA  X      7. PO #   XXXXXXXXXX

   8. DISCNT    XX       9. TERMS  X      10. COLL/PPD  X     11. JOB #  XXXXXXXXXX

  12. SHIP TO   XXXXXXXXXXXXXXXXXXXXXXXXX

                XXXXXXXXXXXXXXXXXXXXXXXXX

                XXXXXXXXXXXXXXXXXXXXX

                XXXXXXXXXXXXXXXXXXXXXXXXXXXXX

  13. COMMENT   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX      14. AR ACCT #  XXXX-XXX

RIGHT ORDER ?   X

ORDER UNSELECTED                                                   CR TO CONTINUE
```

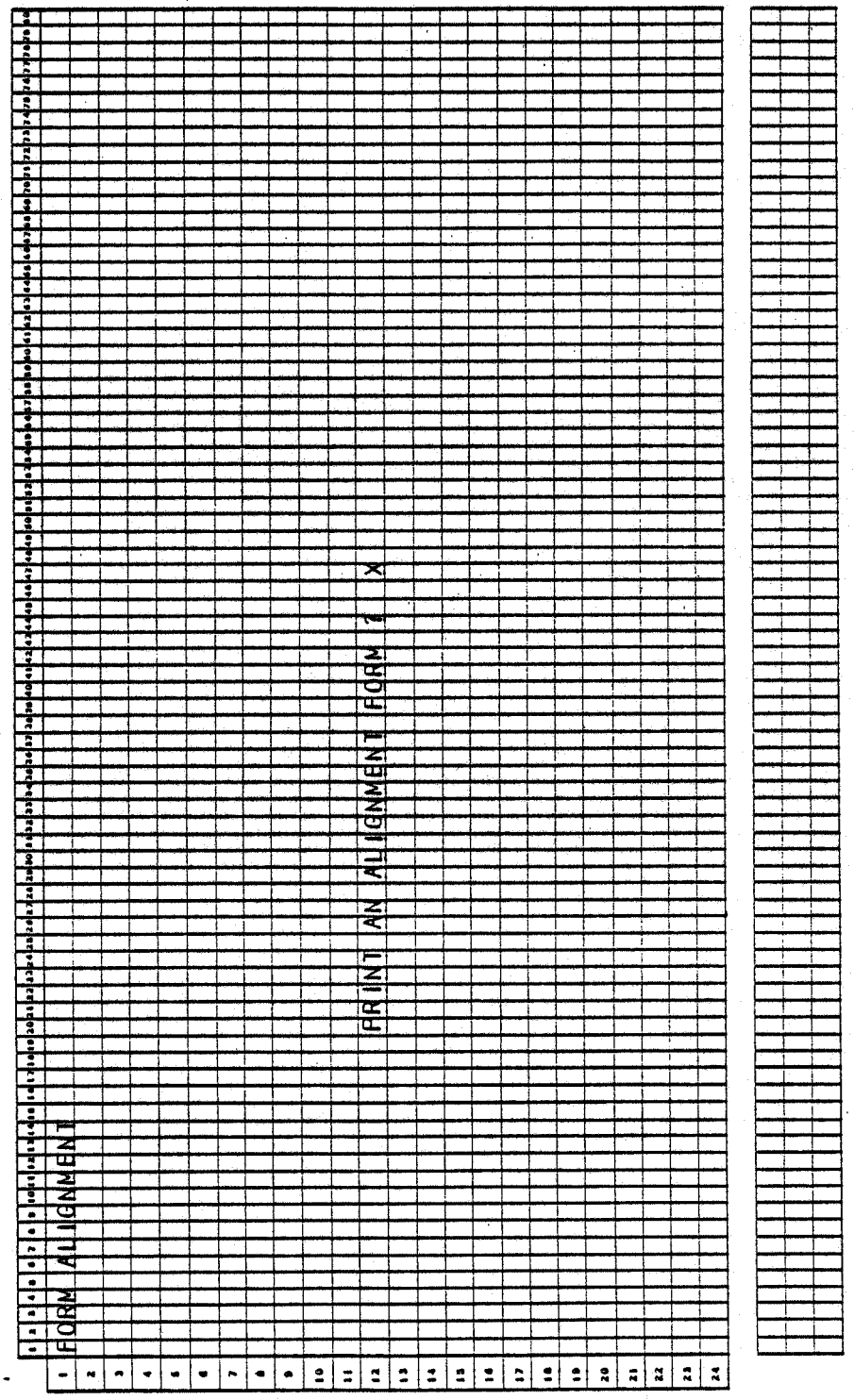
Program: BILEDT



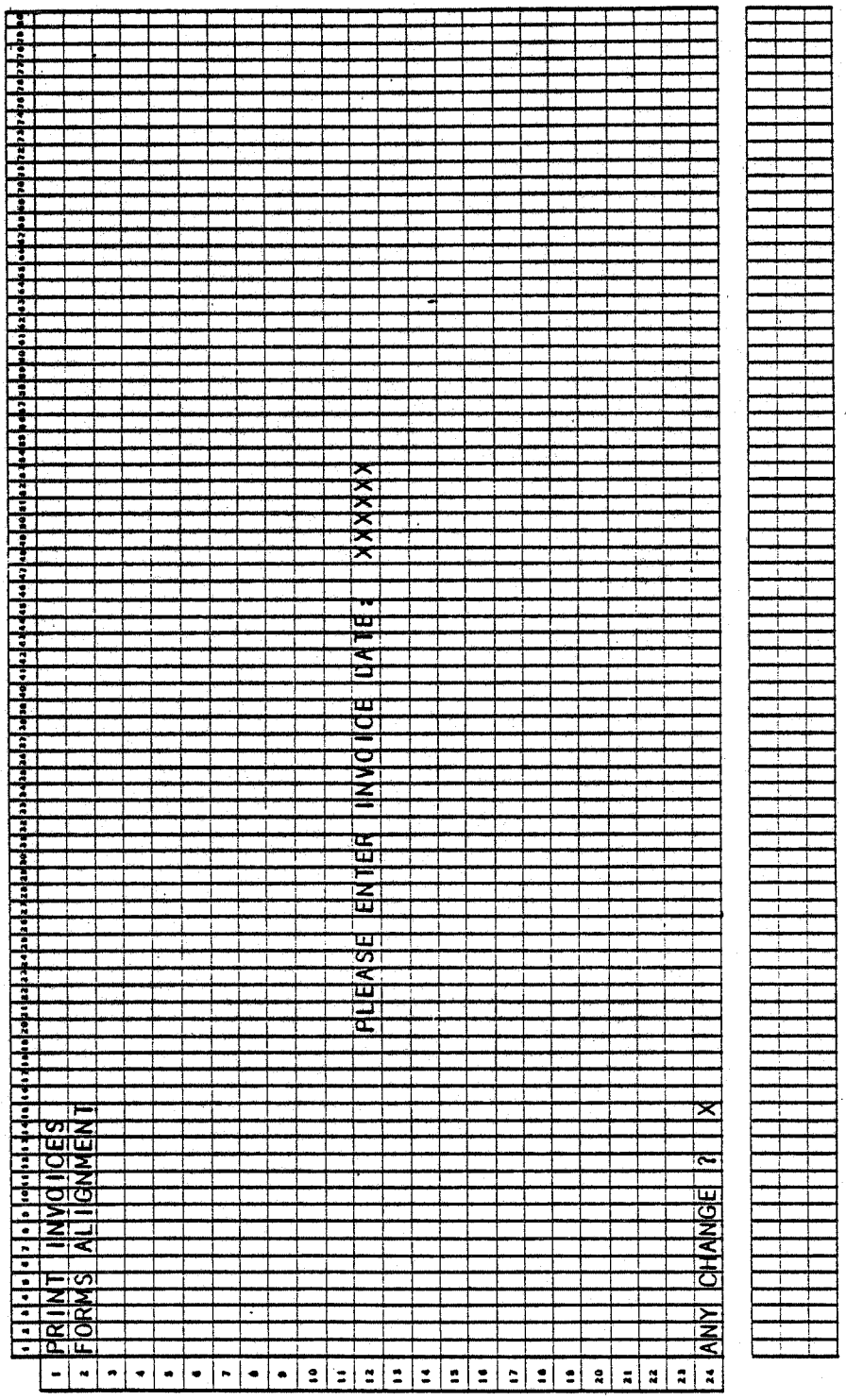
Program: BILLS:ALNINV



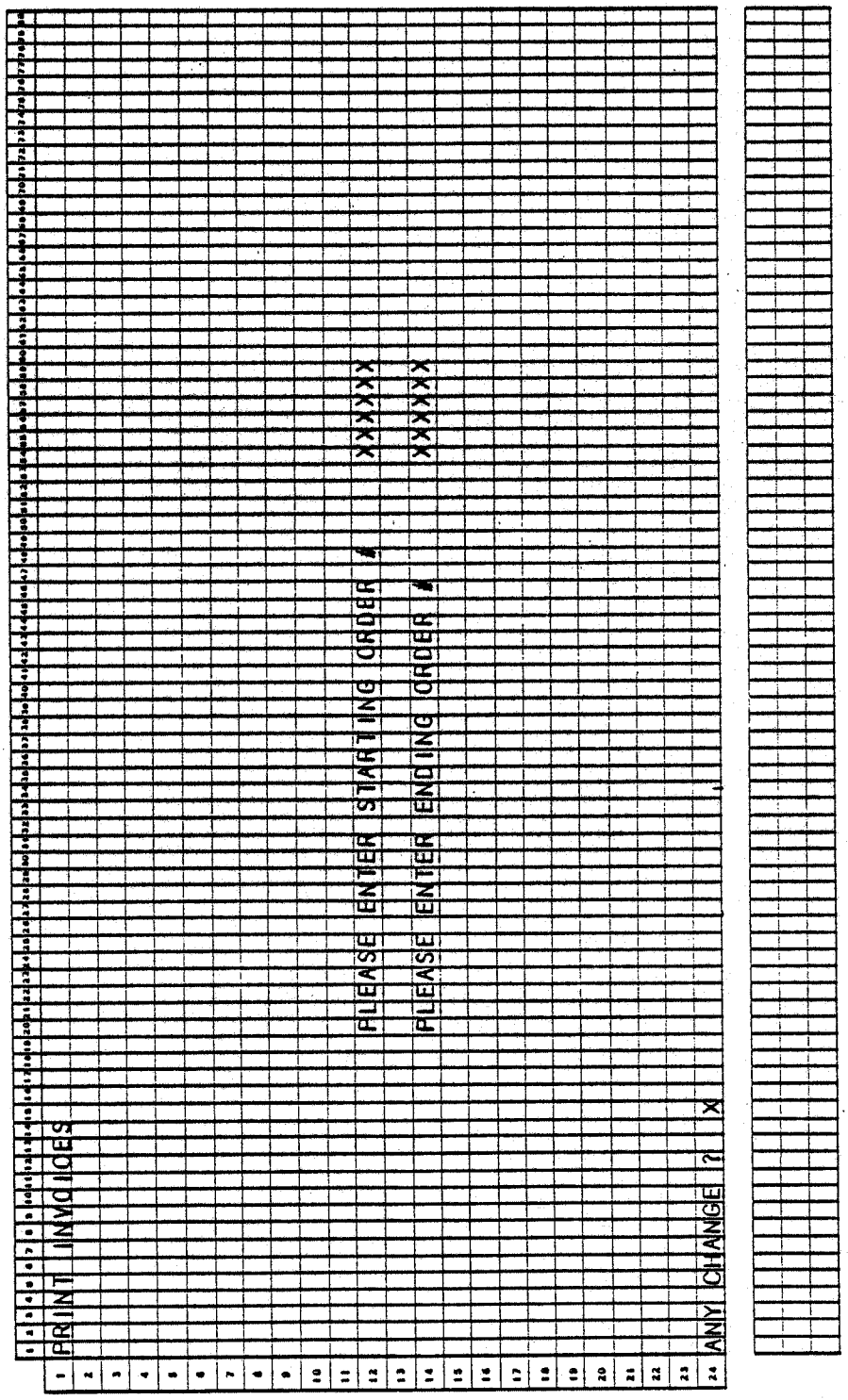
Program: BILLS:ALNINV



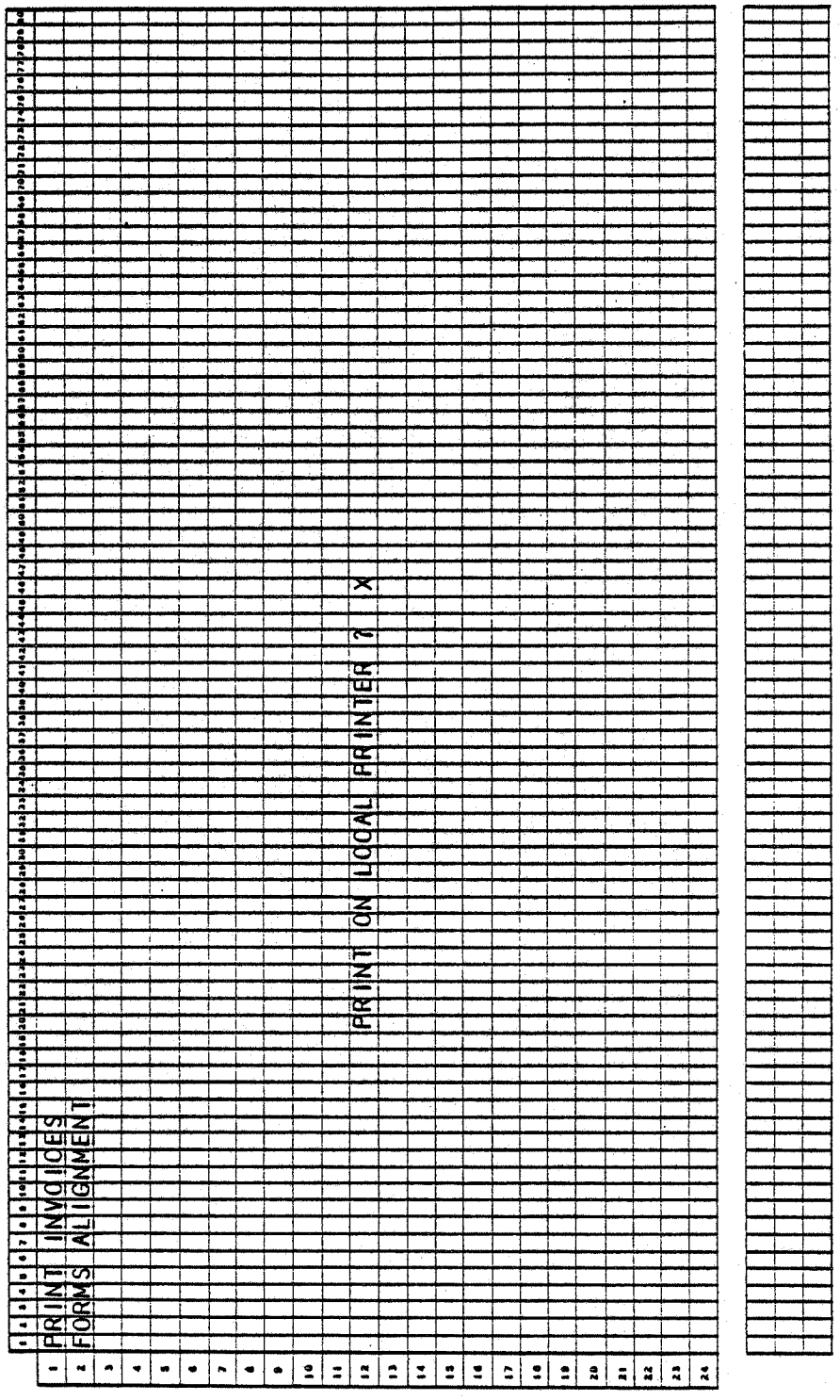
Program: INVOIC



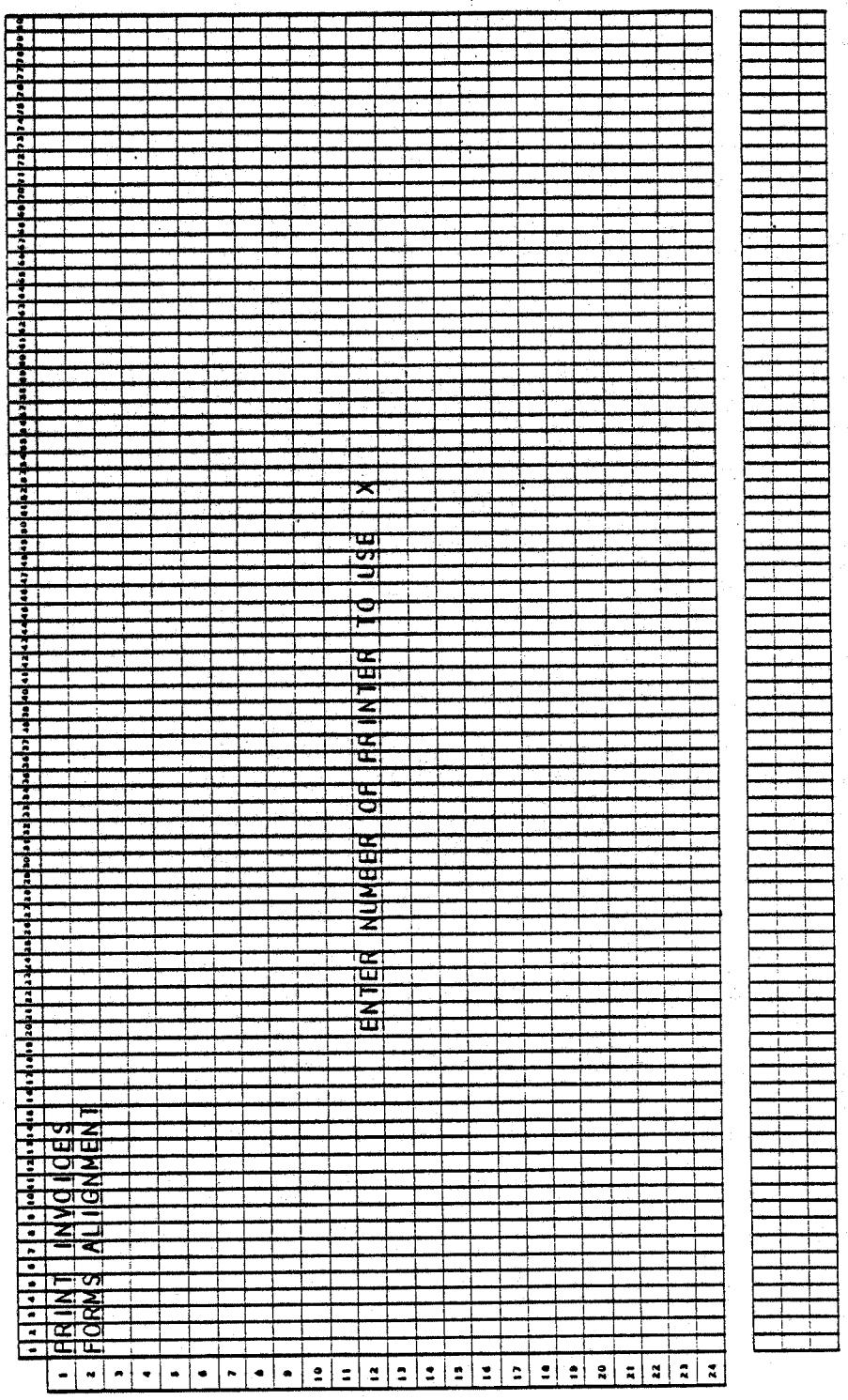
Program: INVOIC



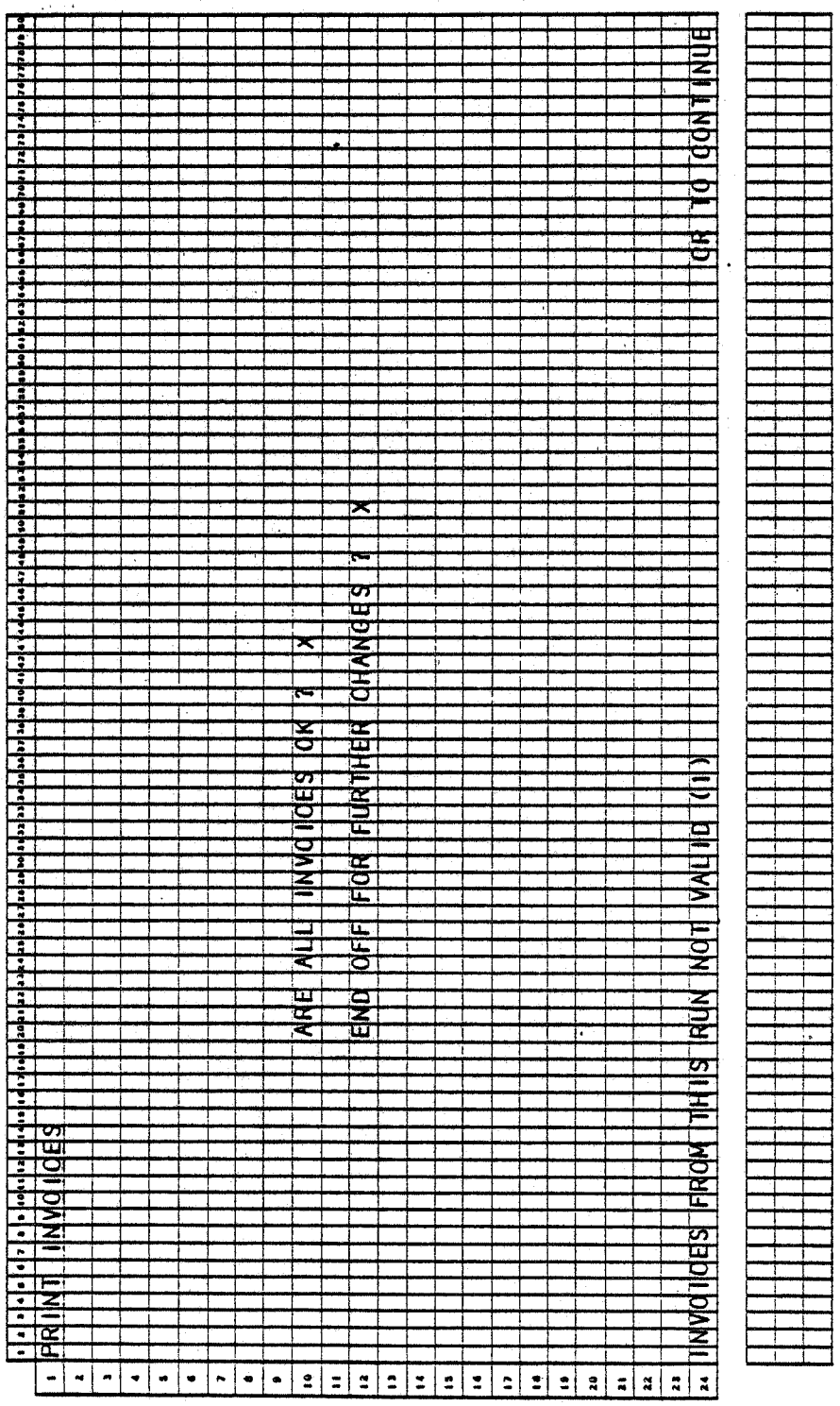
Program: INVOIC



Program: INVOIC

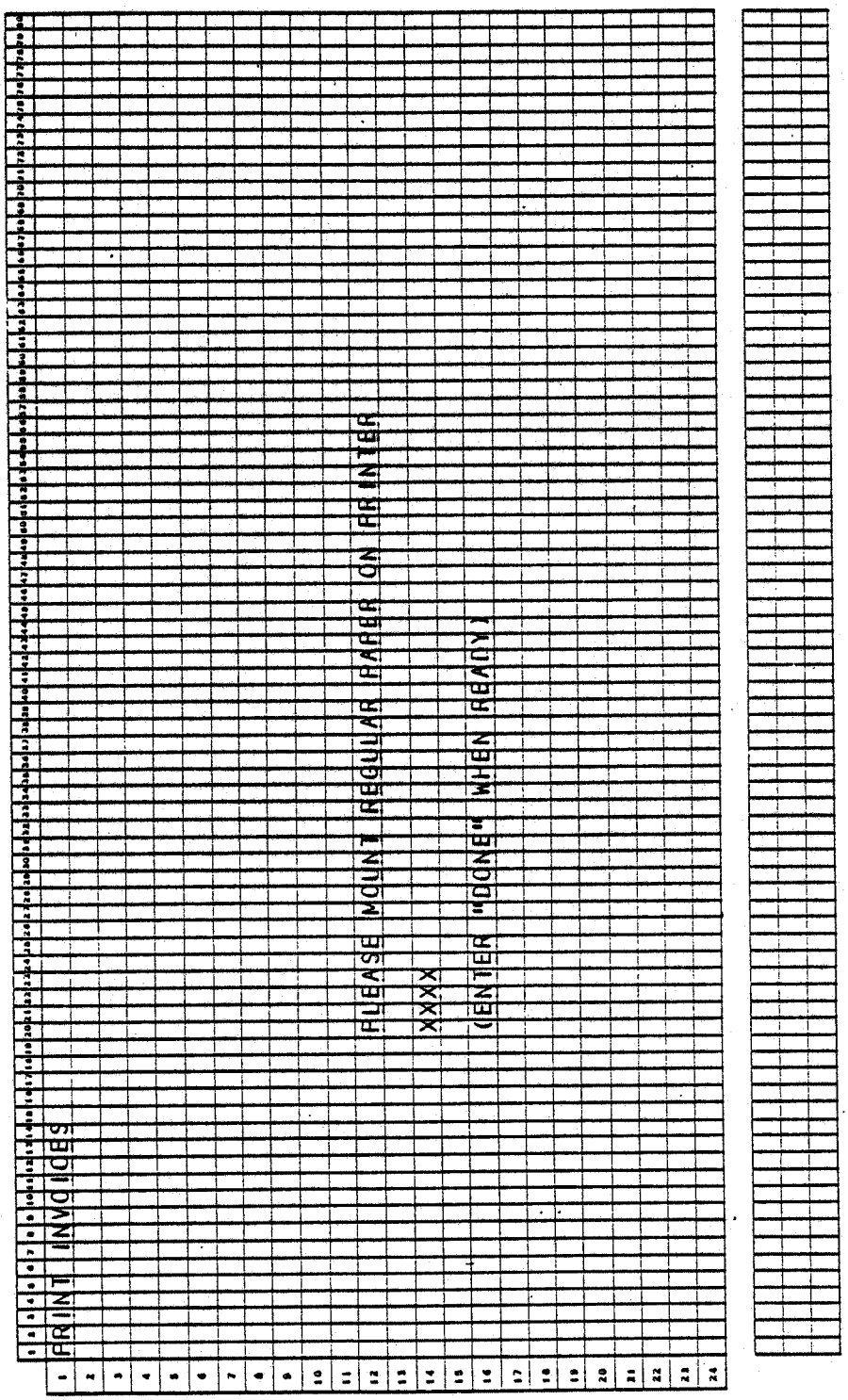


Program: INVOIC



Remarks: (1) IS DISPLAYED WHEN "Y" IS TYPED FOR "END OFF FOR FURTHER CHANGES ?".

Program: INVOIC



BILLING (PRINT INVOICES) APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Allows selection (and unselection) of customer orders for Invoicing, Billing Edit List, Invoice printing, Sales History Journal printing and update Sales History file.

Input:	File Updated:	Output:
KBD	ITMMAS	Billing Edit List
ORDHDR	SALESO	Invoices
ORDLIN	SLSHST	Sales History Journal
CUSMAS	SLHWRK	
CUSIDX	ORDLIN	
ITMIDX	ORDHDR	
SALMAN		
ITMMAS		
PRDSTR		
SLHWRK		
SHPVIA		
SHIPTO		
ARTCDE		
ARTERM		

Enter Module From: CPMENU When Done Return To: CPMENU

Programs in Module: BILLS, UNBILL, BILEDT, ALNINV, INVOIC, POSTAR, PSTINV, SLHJNL, SRTSLH, PSTSLH, CLRLIN, CLRHDR, UNPRBL

Program Functions and Notes:

Allow the following applications: Bill Selected Orders, Unselect Selected Orders, Print Billing Edit List, Print Invoices.

Handle select Unselect (UNBILL) application in main program (BILLS), but chain to separate programs for Edit List (BILEDT) and Invoices (INVOIC).

BILLS

Display Billing submenu (handle in a subroutine BLMNU) and accept application number.

Display screen (same as OE Header screen). Accept order number and find and display order header information. Allow RETURN to default to next sequential order number (or first) in ORDHDR file.

If the Comment fields of the ORDHDR record are blank, automatically insert the default comments.

Ask "BILL THIS ORDER ?". If "Y", allow changes to order header information using "FIELD # TO CHANGE". Do not allow changes to order number, customer number, or location.

Once no more changes are indicated, flag this order for billing and rewrite it to the ORDHDR file.

Ask "BILL ENTIRE ORDER ?".

If "Y", flag each line item of this order for billing and set the Quantity Shipped field equal to the (quantity ordered minus the quantity backordered). Then, go to the order total screen handling logic (explained below).

If the order discount has changed, step through the Order Line file and put the new order discount into each of the lines for this order.

If "N" (entire order not billed), display one line item for this order at a time.

For each item ask, "BILL THIS ITEM ?".

If "N", clear the line from the screen and display the next line in the same place. Clear the first item select flag as it may have been selected before.

If "Y", ask "ANY CHANGE ?".

If "N" is answered to "ANY CHANGE", flag the item for billing, rewrite out the line, and go on to the next line. (Before displaying the line, set the quantity-shipped amount equal to the quantity-order amount less the quantity backordered or out-of-stock amount.)

If "Y" is answered to "ANY CHANGE ?", display a selection menu on line 12 to allow changes to quantity-shipped, price and line discount only. Quantity-shipped may be set to any value, up to the quantity ordered. Do not check the inventory at this point for availability. If the quantity shipped is set to greater than the quantity ordered, it is automatically reset to the quantity ordered.

Automatically recalculate and redisplay the extended price. When all changes have been made, flag this line for billing and rewrite to the ORDLIN file.

Keep a running total of the extended prices, the order costs, and the weight of all selected items and the taxable amount. (Also, do this if entire order was selected for billing.)

Scroll the screen up one line at a time if more than nine lines are selected.

When there are no more line items and some items have been selected for this order, display the total information screen. Display the order net, weight, and taxable items. Accept a miscellaneous amount. Accept a freight amount, and display automatically the portion of the miscellaneous amount that is taxable. Calculate that ratio in the same ratio of dollar amount taxable to the net order total. Automatically calculate the sales tax according to the Ship-to's Tax code if valid, if not, then use the Bill-to's Tax code.

Using the salesman number as the relative record number, read the SALMAN file for this salesman's record. If an active record exists, spin the customer types array trying to match this customer's type with one of those codes. If a match is found, take that associated commission percentage and display it. (Note that commission format on the Salesman record is to one decimal place, this commission percentage is to two decimal places.) If no match is found, display the value from the first position of the array.

If no salesman found, accept input to the Commission Percentage field. If there is no input, accept input to Commission Dollar Amount field. Do not allow input to both of these fields. Store a commission percentage as a negative amount in the OCOMDU field; store an amount as a positive value.

Allow changes to all above amounts including tax, via "FIELD # TO CHANGE".

Automatically adjust order total as necessary.

When no more changes are indicated, flag the order header as selected for billing and rewrite it to the ORDHDR file.

UNBILL

Display order header screen and accept order number, find on ORDHDR file and display.

Ask "RIGHT ORDER ?". If "N", clear screen and get next order number. If "Y", turn off billing flag on the order, then turn off billing flag on all line items for this order.

Display message "ORDER UNSELECTED - CR TO CONTINUE" and continue.

BILEDT

Print Billing Edit List. Handle similarly to ORDEDT, one order per page. Ignore any out-of-stock quantity for a non-backordered item.

Print ship date (promised date or, if zero, request date) only if it is different from order date.

Print Invoices

The job stream for this goes as follows:

ALNINV -- INVOIC -- POSTAR -- PSTINV -- CLRLIN -- CLRHDR -- UNPRBL

ALNINV

Write an invoice alignment print program (this will also be used in the Picking Ticket and Credit Memo job streams, since the forms are almost identical). Print "X's" in all fields. Ask "PRINT ANOTHER ?". If "Y", do so. If "N", go to INVOIC.

INVOIC

Print invoices. Accept invoice date from the screen. Default to system date on RETURN. Ask "ANY CHANGE ?".

When no more changes accept starting and ending order numbers using STENO (note that not all orders selected for billing need be invoiced).

Print invoices for these orders, accessing the ORDHDR, ORDUN, CUSIDX and CUSMAS files for all necessary information. Flag each line item as invoiced if it appears on an invoice.

Flag order as having been invoiced, insert invoice number and invoice date and rewrite to ORDHDR file.

After "END" is entered into STENO, ask "ARE ALL INVOICES O.K. ?". If "N", allow more invoices to be printed. If "Y", go to next program.

POSTAR

Post invoiced order to A/R Sales Transaction file.

Read through ORDHDR file sequentially.

For an order that has been invoiced (check-flag), post the relevant amounts to the A/R Sales Transaction file and calculate the document due date based on the terms due days.

PSTINV

Post amounts to the Item Master file for each line item appearing on an invoice. Read ORDLIN sequentially. For each item flagged as having been invoiced, find the item in the Item Master file by searching the Item Master Index file.

Update the Quantity-on-Hand, Quantity-Allocated, Quantity Month-to-Date, Quantity Year-to-Date, Sales month-to-Date, Sales Year-to-Date, Cost Month-to Date and Cost Year-to-Date fields, and Usage Month-to-Date and Year-to-date fields.

SLHJNL

Sequentially read the Order Line file.

Create a Sales History Work record for each new line item record (LFLAG equals 2).

Also, use information from CUSMAS and ORDHDR files in the creation of Sales History Work record.

Print out a Sales Journal of all newly created Sales History Work records if PRNTSW equals 1.

Send a message to SRTSLH telling it to go PSTSLH.

Send a message to PSTSLH telling it to go to CLRLIN.

SRTSLH

Sort the SLHWRK file by:

- Customer Number
- Item Number
- Date (YY/MM/DD)

PSTSLH

Find out if the entrance point came from program SLHJNL or CSLHJL (via receive message). Sequentially read the Sales Work file (SLHWRK). Write each detail Sales History record into a work record, inserting the Sales Work records into the proper place. Write the three work records back into the Detail Sales History file. This program goes on to program CLRLIN (if invoices) or CLRCRH (if credit memos).

Protect SLHWRK earlier in stream.

CLRLIN

Clear line items of cancelled or aborted orders from ORDLIN file.

For each line item which has been invoiced (flag = 2) check to see if there is any ordered quantity that has not yet been shipped (invoiced). If no, remove the line item. If yes, set the quantity shipped to zero and adjust the Quantity-Ordered and Quantity-Backordered fields to show the current state of this item. Unflag the item and rewrite on ORDLIN file.

Insert as many dummy bracket records as necessary to keep the file at its original size. Ignore any out-of-stock quantities and clear these to zero.

CLRHDR

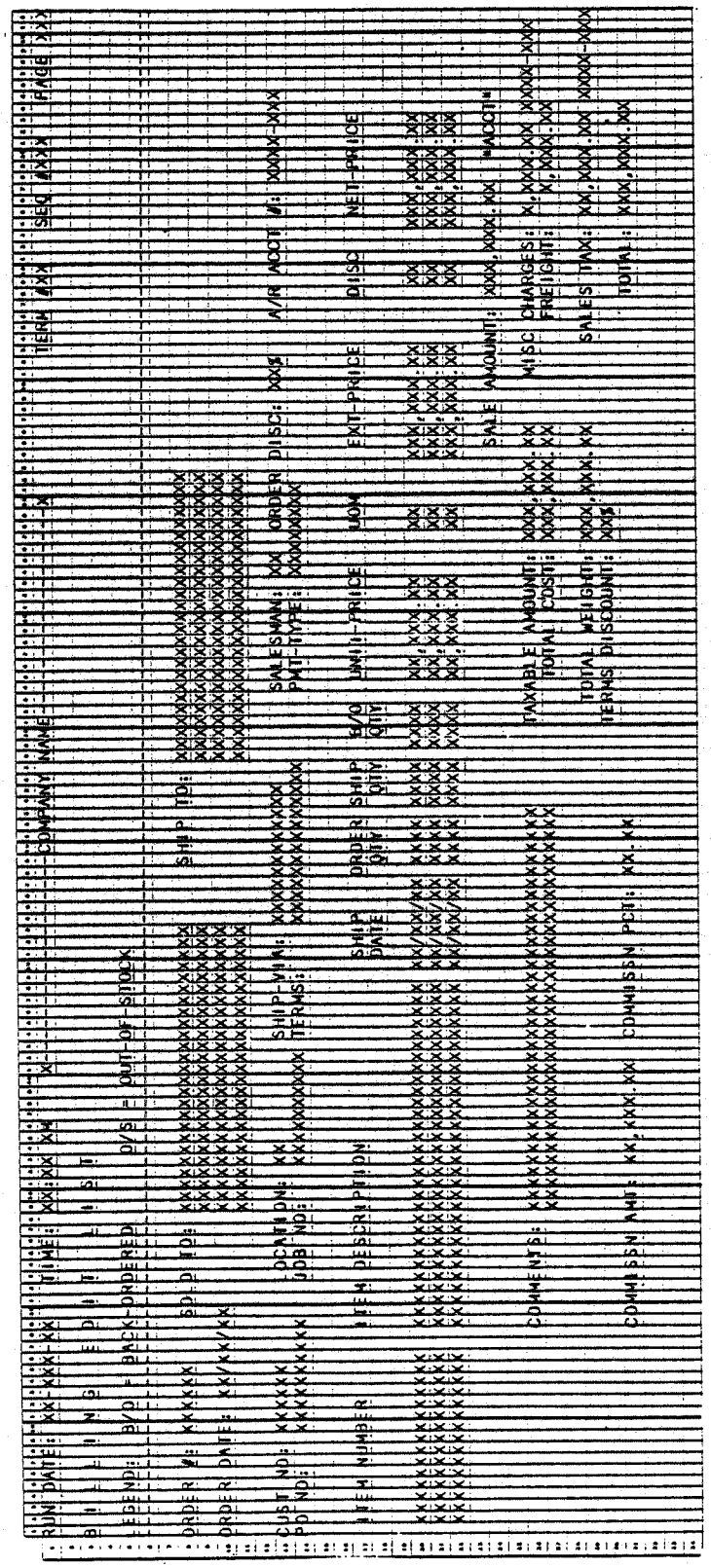
For each selected or invoiced order on the ORDHDR file, check ORDLIN file to see if any outstanding line items exist for this order. If no, clear the order from the file. If yes, keep the order on file, but unset the billing flag if the order has been invoiced.

UNPRBL

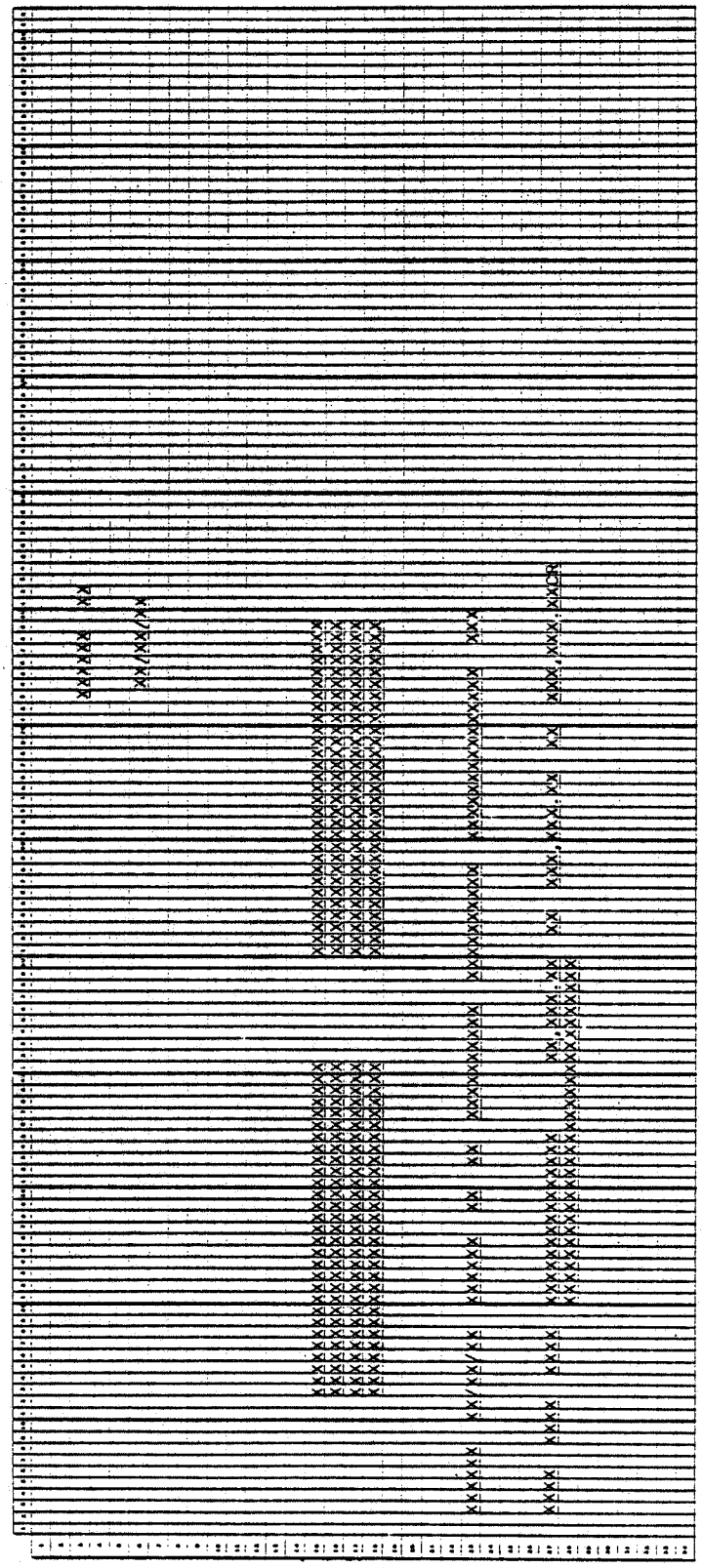
Unprotect ORDHDR, ORDLIN and decrement user count for ITMMAS, COPCTL, and CUSIDX.

BILLING (PRINT INVOICES) APPLICATION
DIBOL JUN-84
REPORT FORMATS

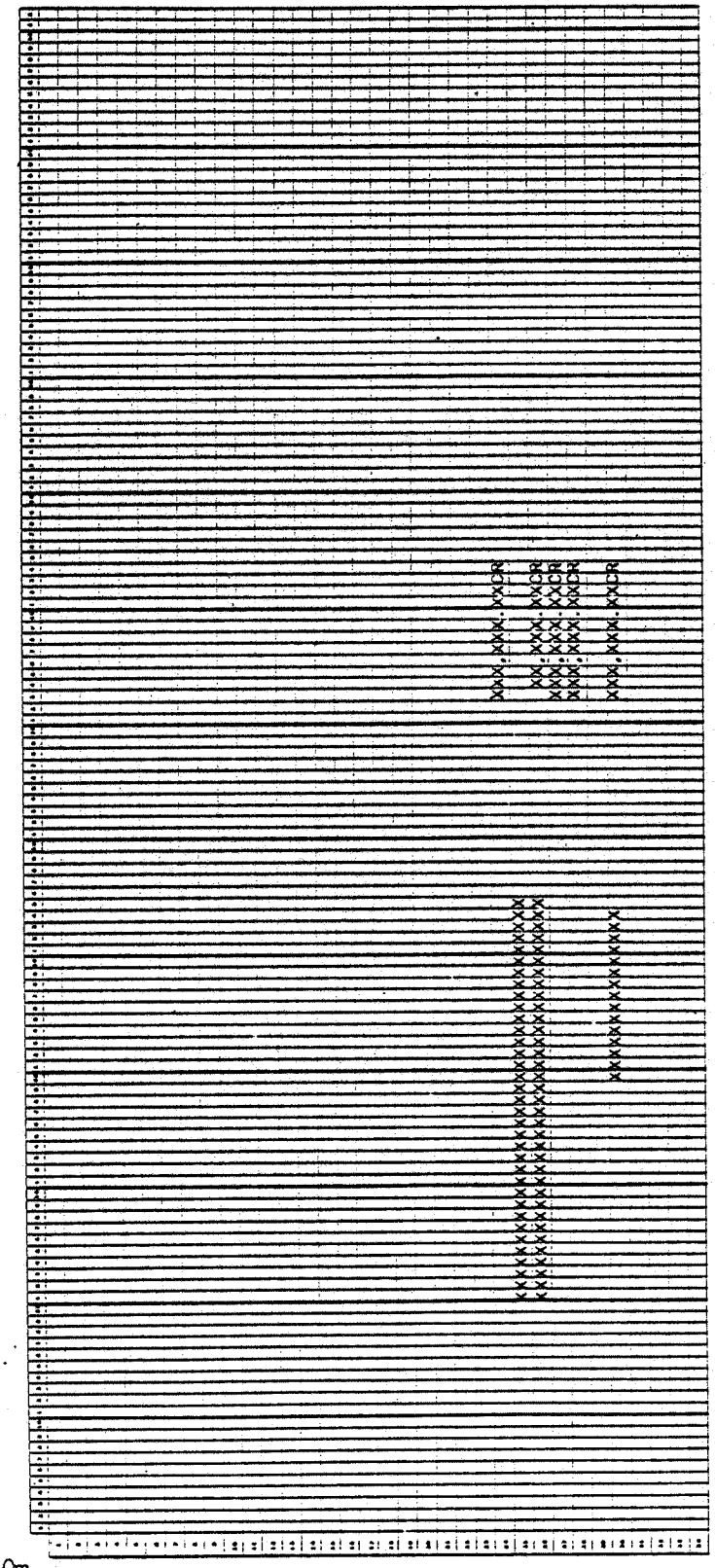
Program: BILEDT (Billing Edit List)



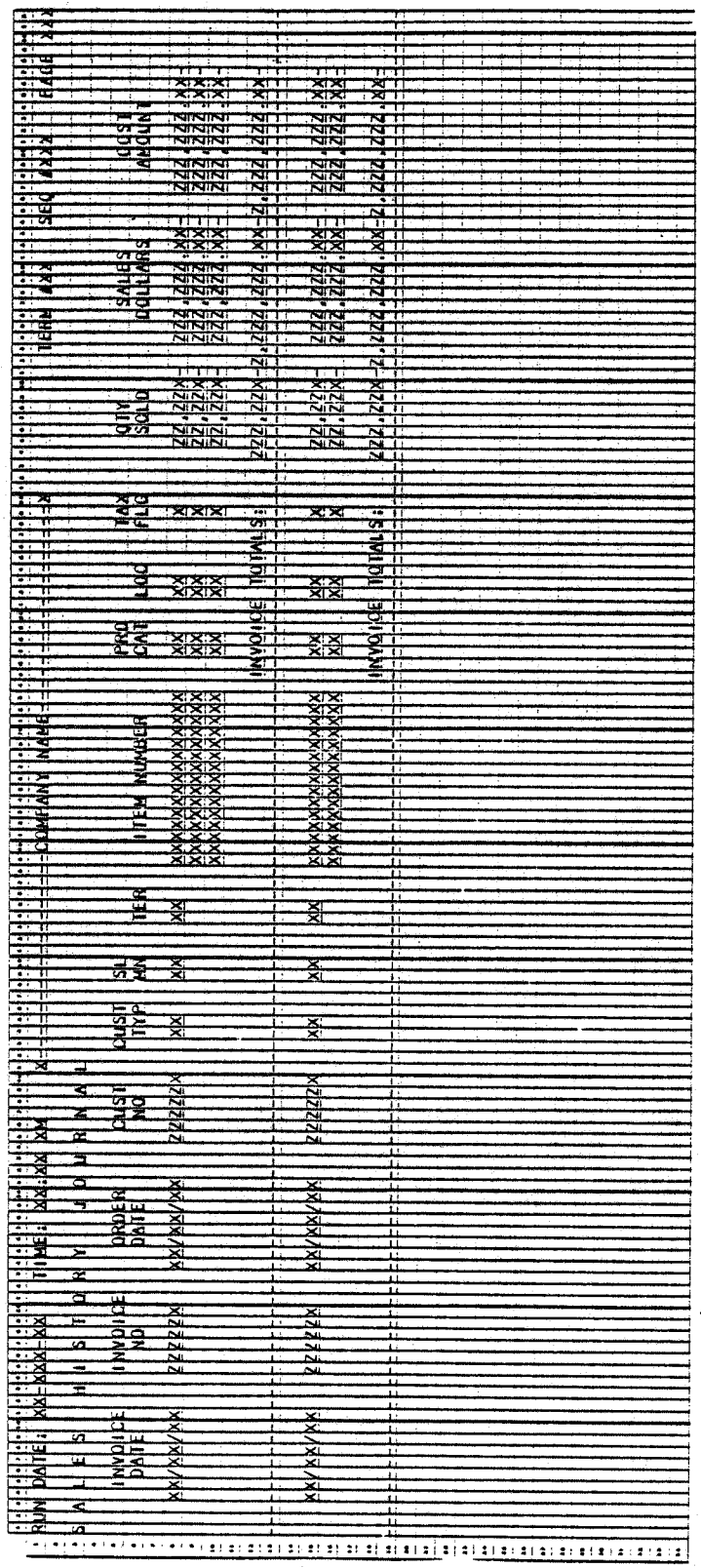
Program: INVOIC (Invoice)



Program: INVOIC (Invoice)



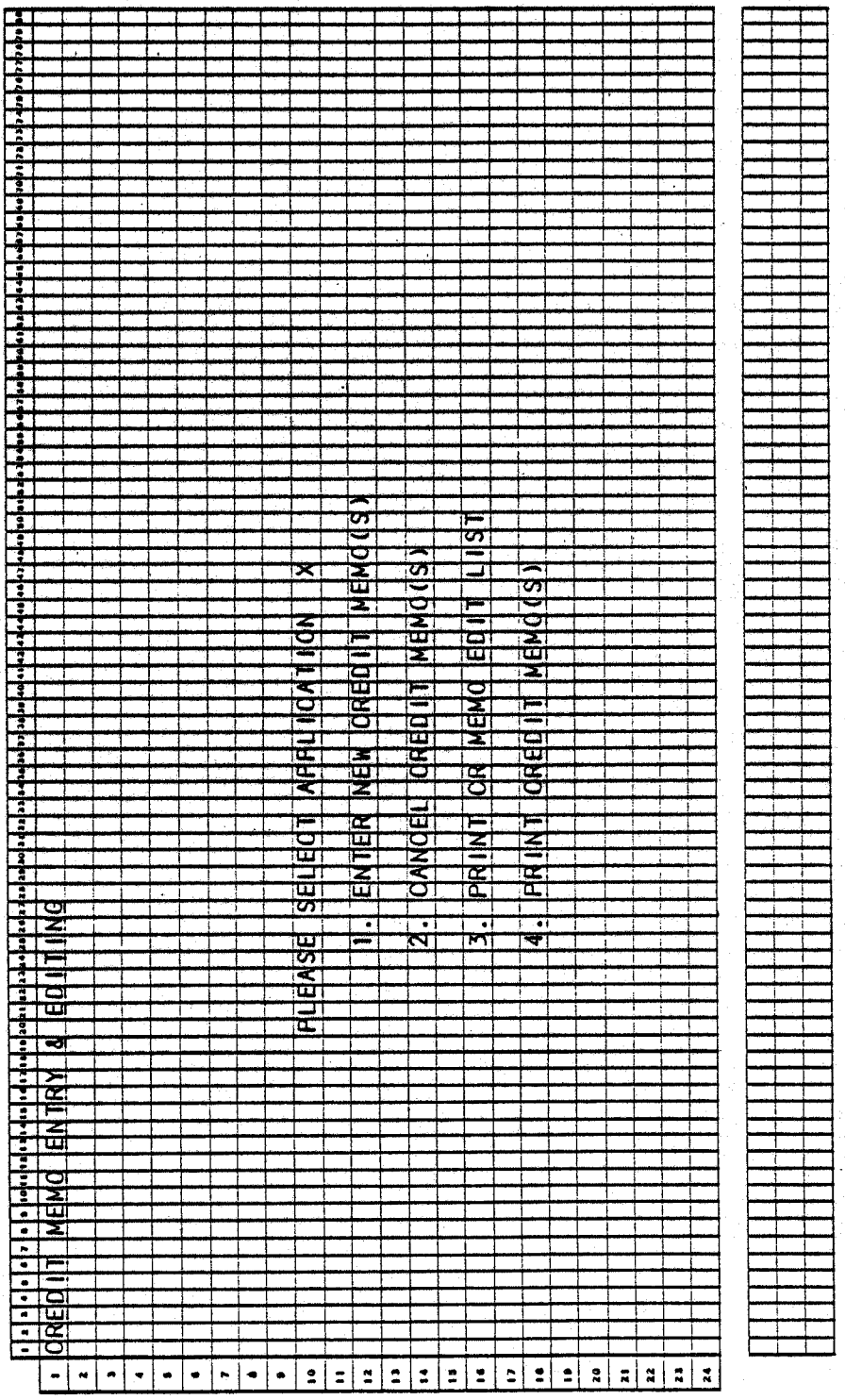
Program: SLHJNL (Sales History Journal)



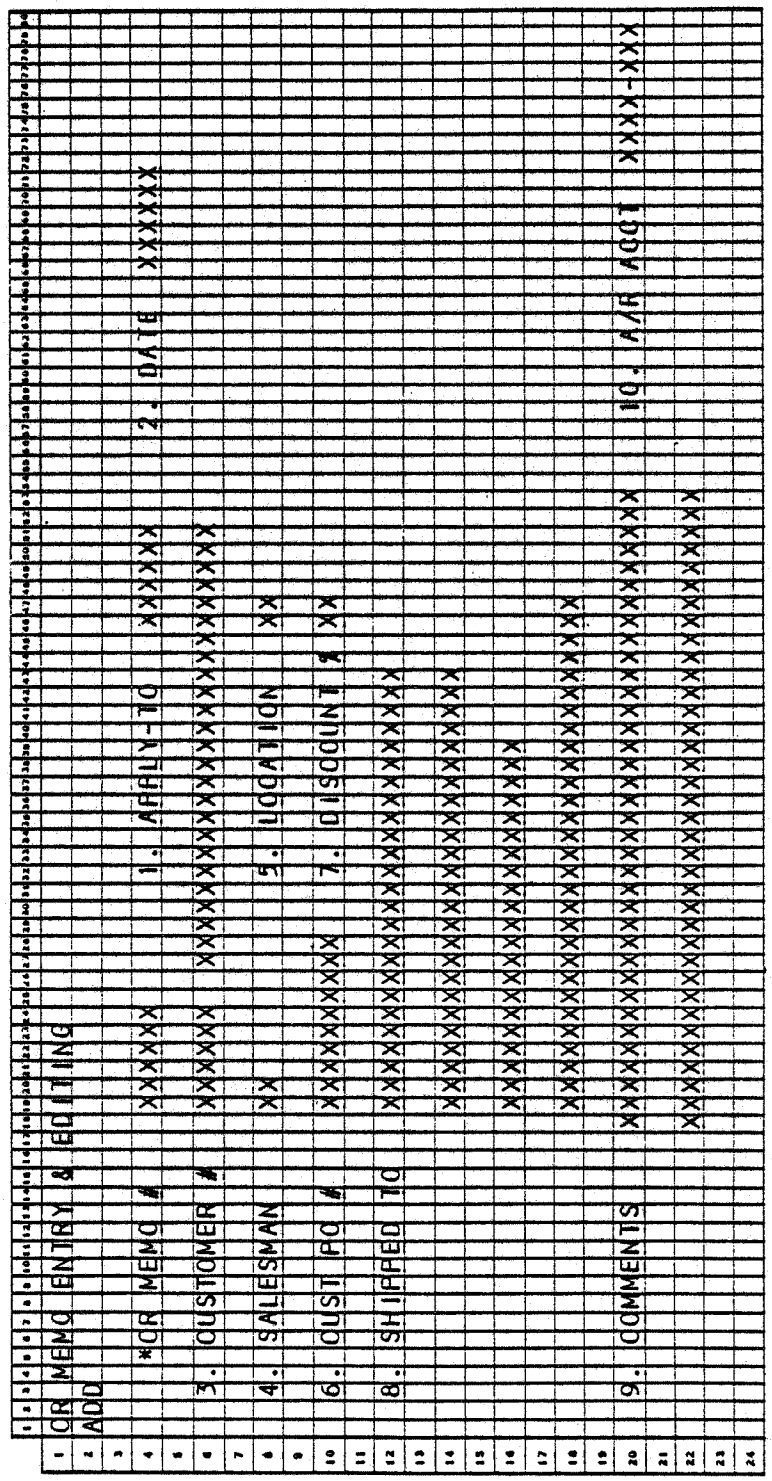
CUSTOMER ORDER PROCESSING PACKAGE
CREDIT MEMO ENTRY & EDITING APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: CRMENT:CRMNU



Program: CRMENT:CMSC1



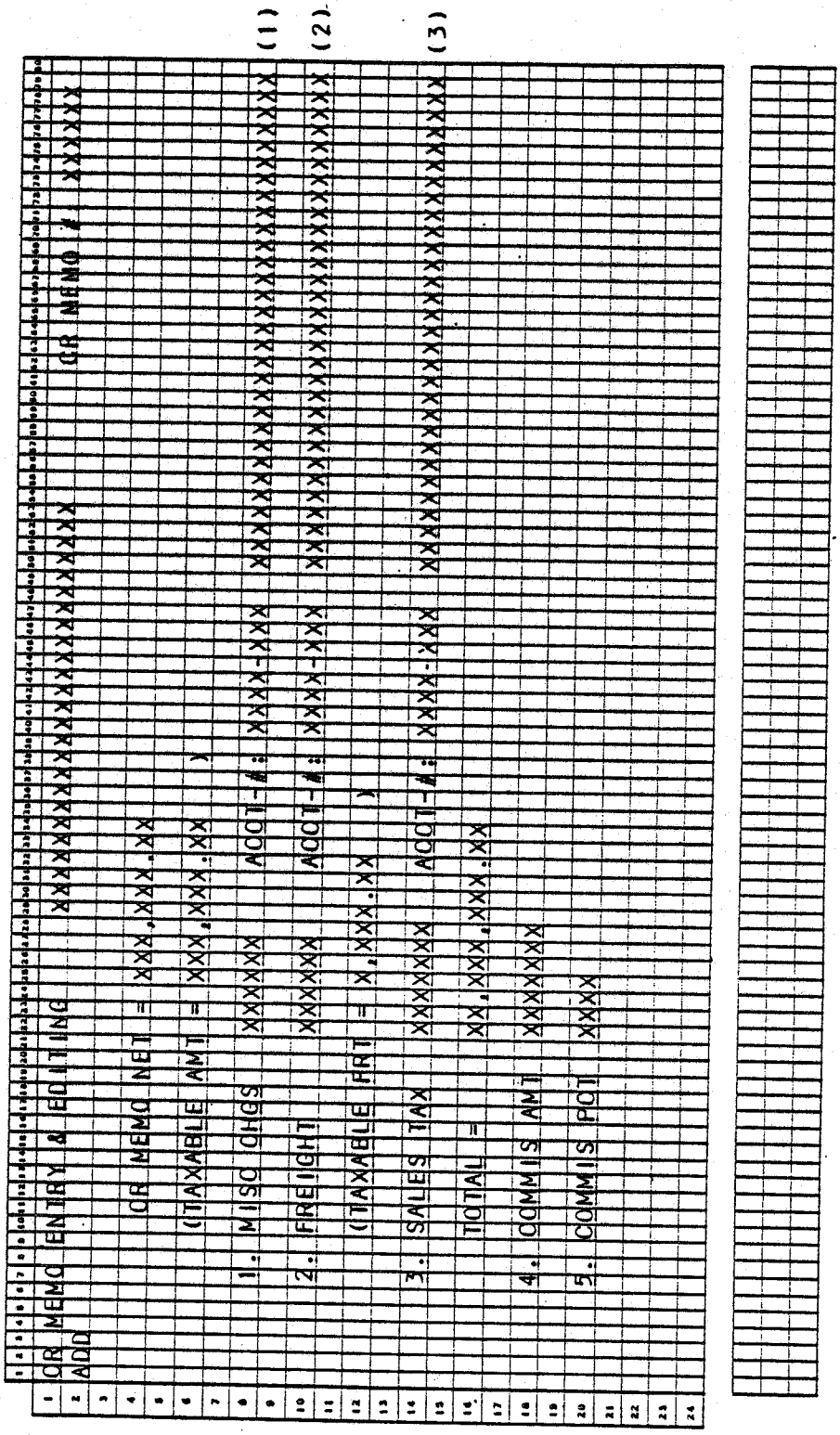
Remarks: FOR 8 ACCEPT A CUSTOMER NUMBER AND SEARCH FILE FOR ADDRESS. BLANK INPUT TO #8 DEFAULTS TO BILL-TO ADDRESS.

Program: CRMENT:CMSC2

```
 1 CR MEMO ENTRY & EDITING     XXXXXXXXXXXXXXXXXXXXXXXXX          DISCOUNT: XX%
 2 ADD
 3
 4 ITEM #          DESCRIPTION                    QTY    PRICE    DISC  EXT-PRICE
 5
 6 XXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXX  XXXXXXX   XX   XXX,XXX.XX
 7
 8 XXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXX  XXXXXXX   XX   XXX,XXX.XX
 9
10
11
12
13
14
15
16
17
18
19
20
21
22 RUNNING QTY: XXXXXX                               RUNNING TOTAL: XXX,XXX.XX
23
24 ADJUST QTY ON HAND ?  X
```

Remarks: DISPLAY SCROLLS IF MORE THAN 8 LINES.

Program: CRMENT:CMSC3



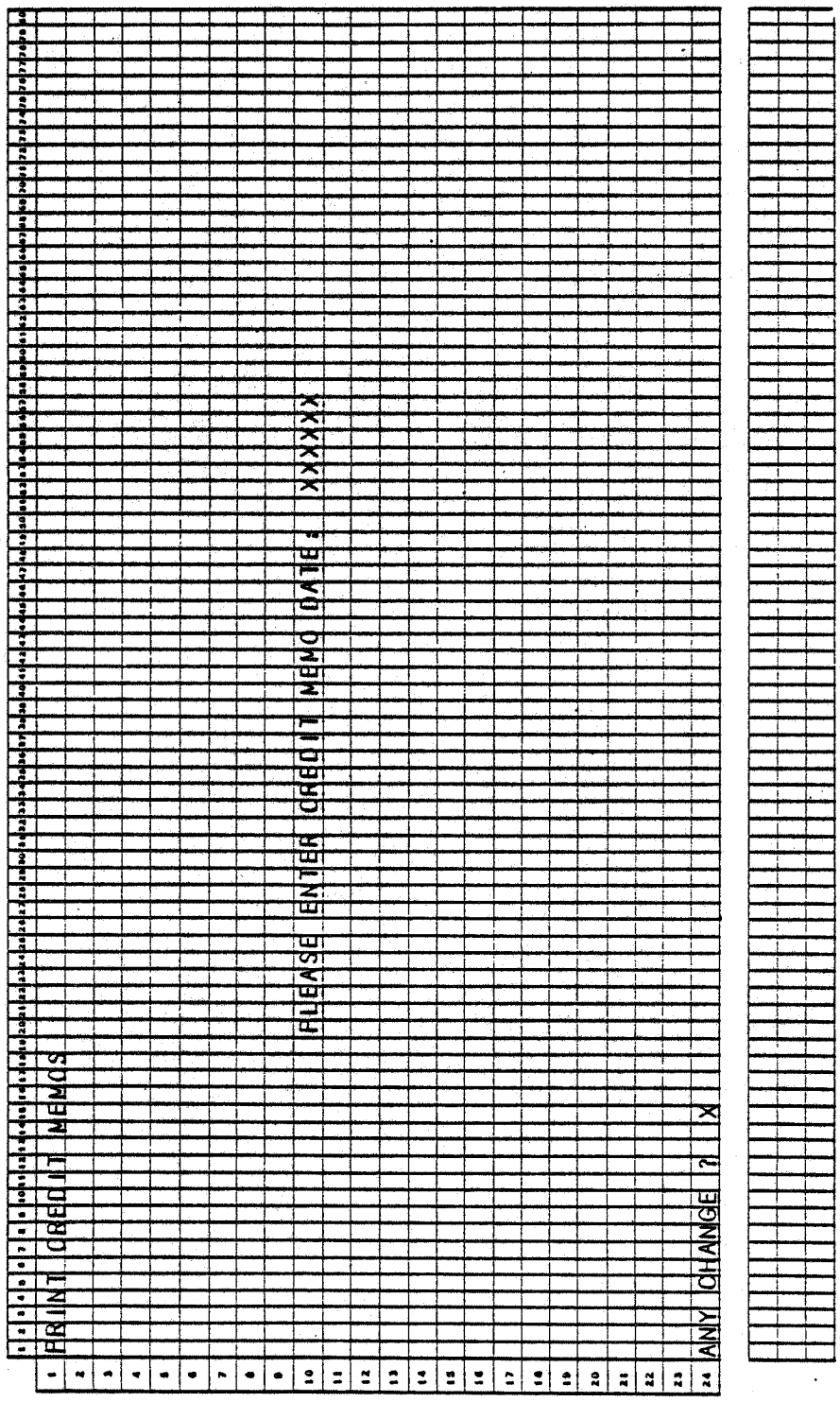
Remarks: (1) DISPLAYS ONLY IF MULTIPLE MISCELLANEOUS CHARGES ACCOUNTS.
(2) DISPLAYS ONLY IF MULTIPLE SALES TAX ACCOUNTS.
(3) DISPLAYS ONLY IF MULTIPLE FREIGHT ACCOUNTS.

Program: CRMCNC

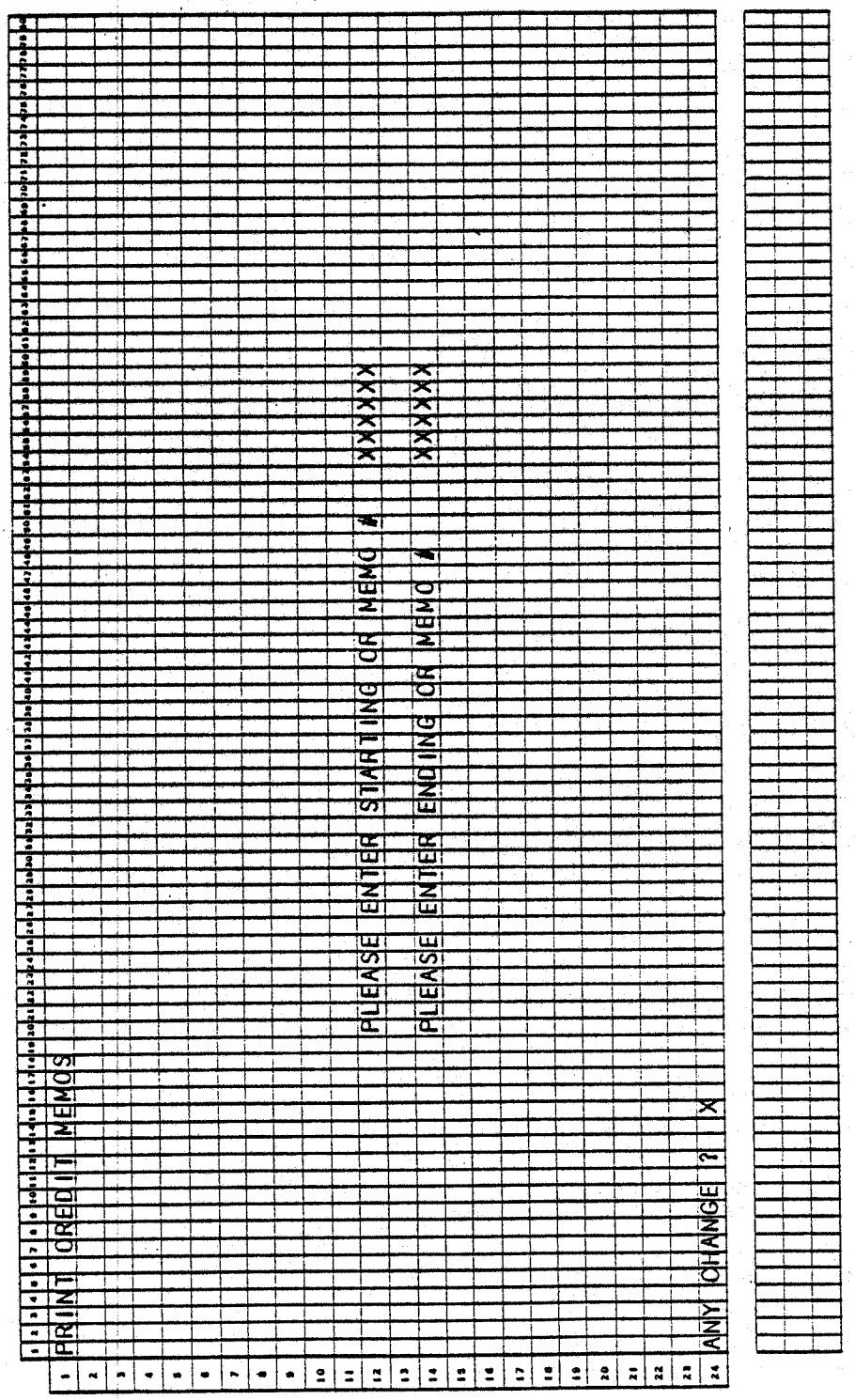
```
CR MEMO ENTRY & EDITING
CANCEL

                      *CR MEMO #    XXXXXX

                    1. APPLY-TO     XXXXXX

                    2. DATE         XX/XX/XX

                    3. CUSTOMER #   XXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXX

                    4. SALESMAN     XX

                    5. LOCATION     XX

                    6. CUST PO #    XXXXXXXXXX

                    7. DISCOUNT $   XX

                    8. COMMENTS     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

                                    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

OK TO CANCEL ?
```

```
CREDIT MEMO CANCELLED                                      CR TO CONTINUE
```

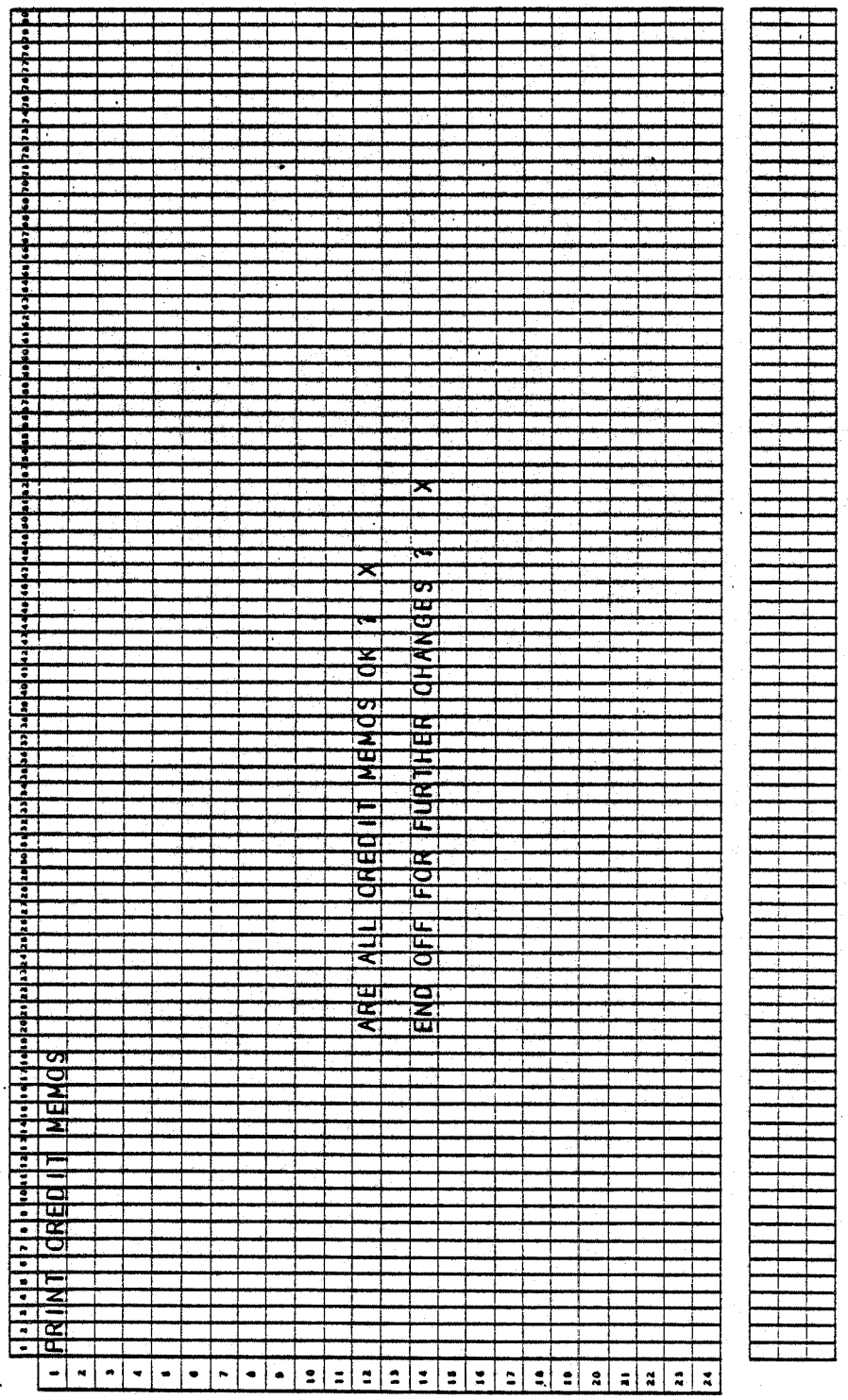
Program: PRTCRM



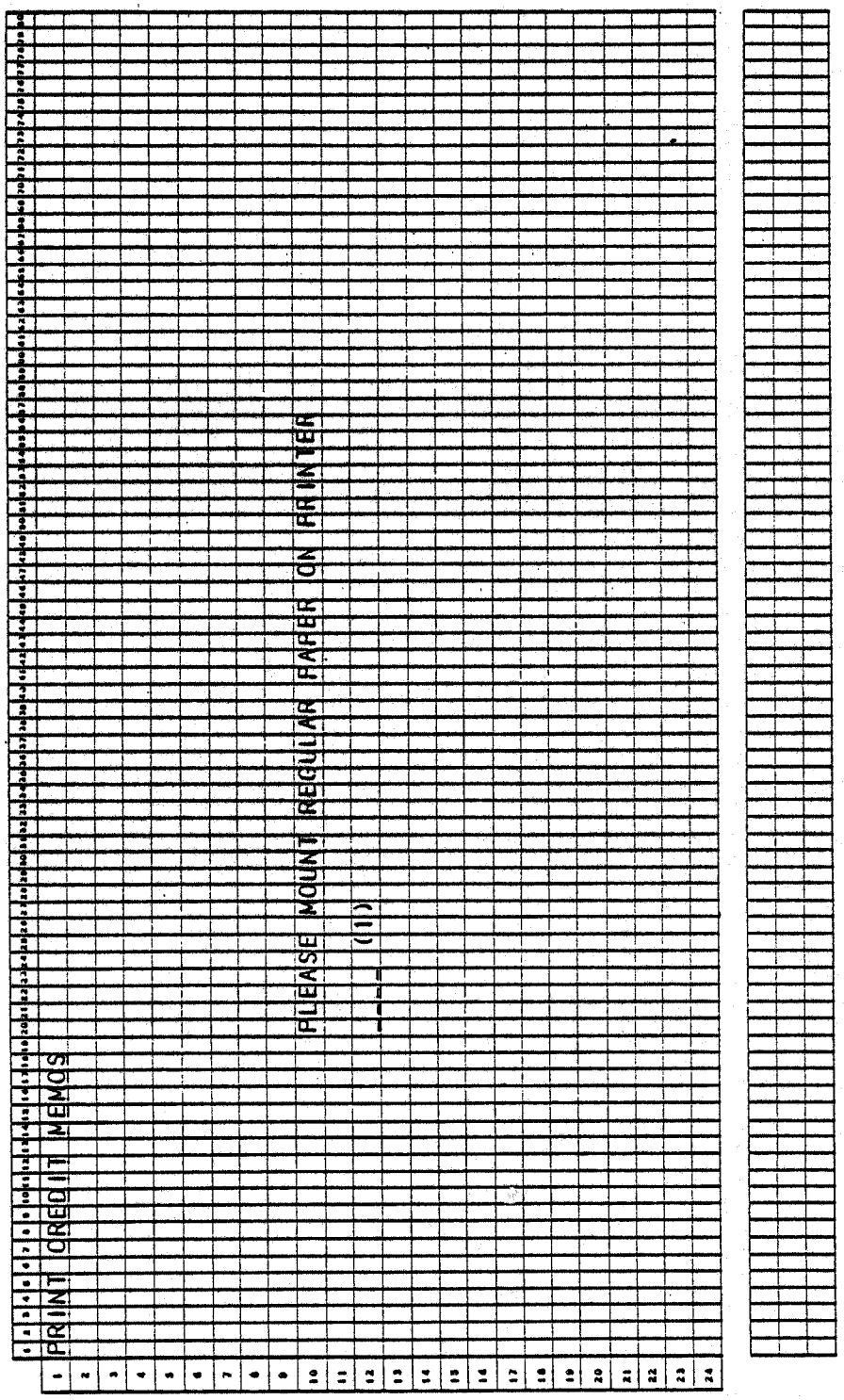
Program: PRTCRM



Program: PRTCRM



Program: PRTCRM



Remarks: (1) TYPE IN "DONE" WHEN DONE.

CREDIT MEMO ENTRY & EDITING APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Credit Memo Entry and Editing.

Input:	Files Updated:	Output:
KBD	CRMHDR	Credit Memo Edit List
CUSMAS	CRMLIN	Credit Memos
CRMLIN	ITMMAS	Sales History Journal-
CUSIDX	SLHWRK	Credit Memos
SLHWRK		
ITMMAS		
ITMIDX		
CRMHDR		
SHIPTO		
ARTCDE		

Enter Module From: CPMENU When Done Return To: CPMENU

Programs in Module: CRMENT, CRMCNC, CRMEDT, CRMINV, SRTCRH, SRTCRL, ALNINV, PRTCRM, CRMAR, PSTCRM, CSLHJL, SRTSLH, CLRCRH, CLRCRL, SALSRT

Program Functions and Notes:

This module will follow, in a very close manner, the set of programs for Order Entry & Editing. The main differences will be: this module will not access the Product Structure files, this module will not allow changes to be made to credit memos; if they are in error they must be deleted and then re-added; there will be no inquire mode, and the sort key on the Line Item file will be different.

Allow the following functions: Enter New Credit Memos, Cancel Credit Memos, Print Credit Memo Edit list, Print Credit Memos.

Add mode will be in the main program (CRMENT). The remaining three functions will each be in separate programs.

Write a separate subroutine to display the Credit Menu submenu (CRMNU). Credit Memo Print program. Otherwise, assume these files are not necessarily in sorted order.

CRMENT (ADD CREDIT MEMOS)

Open the files CUSIDX, ITMIDX, CRMHDR, CRMLIN, in that order. Open them in a normal fashion, but close them external to subroutine FILES.

Open the files CUSMAS, CUSIDX, CRMHDR, ORDHDR, in that order.

Display screen and accept apply-to number, and then reserve space on CRMHDR file for the record.

Reject a new entry if it already exists in the file.

Increment the record count and rewrite the Control record.

Increment and display the last-used credit memo number from the COPCTL file.

Accept credit memo date, default to system data on RETURN.

Accept customer number; find on CUSMAS file, and automatically display customer name. Save customer name and address in CRMHDR file.

Accept remaining information on this screen as in OE; allow changes via "FIELD # TO CHANGE". Allow changes to be made to anything except credit memo number, close the files CUSMAS, CUSIDX, CRMHDR, external of file subroutine, and return to parent program. If during processing an abort condition arises, set a flag to indicate such before returning to parent program. If an abort condition exists, close any open files, and return to the top of the program. When no more changes, go to Credit Memo Line Item screen and start accepting line items.

Line Item Entry:

Open the files ITMMAS, ITMIDX, CRMLIN, without changing user status.

Retrieve automatic discounting codes from the ITMMAS Control record in the same manner that OE does.

Accept item input data in the same manner as OE add mode, with the exclusion of Product Structure file inquiry, and also exclude any backorder logic.

Handle line item cancel, price extension, automatic discounting, and screen scrolling in the same manner as add mode of order entry.

After no change is indicated, ask the question "ADJUST QUANTITY ON HAND ?". If "Y" is answered to that question, change the quantity in the Line Item record to negative.

Otherwise, go immediately to next line item. Scroll the screen up one line at a time to make room for additional lines.

Accept "END" for Item Number to terminate line item entry and display third screen.

After "END" is input for item number, save the running total amount to pass back to the parent program, close the three files previously opened (not via FILES), and return to the parent program.

If the file is about to be exceeded in capacity, pass a flag indicating such back to the parent program.

If the file full indicator is such, close all files that were opened via subroutine FILES, and stop.

Handle the third screen of credit memo add in an analogous manner to program BILLS. Do not use Salesman file searching logic, however.

Automatically compute and display the tax, but allow it to be changed to any non-negative value. When no more changes are indicated, return to the parent program.

Rewrite the header record to CRMHDR with the total information, and return to the top of the ADD logic.

On CANCEL, EDIT LIST, or PRINT requests, chain to the appropriate program.

CRMCNC (CANCEL CREDIT MEMOS)

Allow cancellation only of entire credit memos. Open the files CRMHDR and CRMLIN in update mode, then re-open them on alternate channels, in input mode, without incrementing the user number in the device table a second time.

Display the header, then display the word 'CANCEL' before the credit memo number.

Accept credit memo number and search the input channel for a header record with this memo number.

If found, display the remainder of the header information and ask if this is the correct credit memo.

If it is, rewrite the header after replacing the customer name with "]]]CANCEL".

Find all credit memo lines and cancel them in a similar manner, putting the message in the description.

Display a message indicating the credit memo has been cancelled, clear the screen, and accept another credit memo to be input. Flag the header record and all Line Item records as cancelled.

Do a sequential search of CRMLIN file to find the first line item for current credit memo. Then read CRMLIN file sequentially to end of file, searching for all other line items with current credit memo number. (This must be done since CRMLIN file is not necessarily in sorted order at this point.)

Otherwise, proceed exactly as in the program CANCEL in the OE module.

CRMEDT (PRINT CREDIT MEMO EDIT LIST)

Write a print program to print an Edit List. Print only one credit memo per page.

Remember the CRMHDR and CRMLIN files are not necessarily in sort order. Also, the Quantity field may be negative, but print everything as positive.

Open the files CRMLIN, CRMHDR, CUSIDX, CUSMAS, for input mode via subroutine FILES.

Sequentially read through the file CRMHDR, and print the Edit List. Print the memo number for cancelled headers and a message to the effect that the memo has been cancelled.

Print as many lines as will fit on a page, and handle the header external of subroutine PRINT.

Interpret ADJUST INVENTORY answer as being YES if the quantity is negative, NO if it is a positive number.

Print Credit Memos

This job stream will have the following program flow:

CRMINV -- SRTCRH -- SRTCRL -- ALNINV -- PRTCRM -- CRMAR -- PSTCRM -- CLRCRH -- UNPRCM

The programs are all described below.

CRMINV

Protect CRMHDR and CRMLIN files and initiate sorts of these files, sending appropriate messages.

SRTCRH

Sort the CRMHDR file by credit memo number.

SRTCRL

Sort the CRMLIN file with credit memo number as major key and item number as minor key.

ALNINV

This is the same program that is in the INVOIC job stream in the BILLS module.

PRTCRM

Print Credit Memos. This program is essentially the same as INVOIC, except several fields do not appear, that do appear on invoices. Also, there is no select flag.

Open normally the CUSIDX and CUSMAS files.

Open without changing status the CRMHDR and CRMLIN files.

Accept from the terminal the credit memo date. Default to today's date on no entry.

Using STENO, accept starting and ending credit memo numbers to print.

Find the first non-aborted credit memo that satisfies the user input conditions.

Print the credit memo using the format defined.

For credit memos that take more than one form, print 'continued' in the total field, print the comments, and continue printing on the following form.

After exceeding the ending credit memo number, ask if all credit memos are okay. If they are not, ask "END OFF OR FURTHER CHANGES ?". If "Y" is input to the second question, unprotect all files and return to Customer Order Processing menu. If "N" answered to "END OFF ?" question, return to start of this program. If "Y" answered to "ALL CREDIT MEMOS OKAY ?" close CUSMAS and CUSIDX using FILES; close other files without using FILES.

In the remaining processing in this job stream, it is assumed that all Credit Memos on file have been printed, and all postings to other files assume this.

CRMAR

Post Credit Memo amounts to the A/R SALES file: SALESO.

Make sure to reverse the signs of all amount and quantity files to reflect credit.

PSTCRM

Update the ITMMAS file in a manner similar to program PSTINV in the invoicing job stream, with the following two exceptions: (1) do not post the inventory-on-hand amount for any record whose quantity is greater than or equal to zero in the CRMLIN file. However, remember to update month-to-date and year-to-date costs, sales dollars, and quantity sold amounts. (2) Ignore the product structure logic.

CSLHJL

Sequentially read the Credit Memo Line file.

Create a Sales History Work record for each new credit memo line item record.

Also, use information from CUSMAS, ITMMAS, and CRMHDR files in the creation of Sales History Work record.

Print out sales journal of all newly created Sales History Work records if PRNTSW equals 1.

Send a message to SRTSLH telling it to go to PSTSLH.

Send a message to PSTSLH telling it to go to CLRCRH.

SRTSLH

Sort the SLHWRK file by customer number, item number, year, and month/day.

Receive a message from SLHJNL or CSLHJL.

Sequentially read Sales Work file.

Write each detail Salas History record into a work record, inserting the Sales Work records into the proper place.

Write the work record back into the Detail Sales History file.

Go to program CLRLIN (if invoicing) or CLRCRH (if credit memos).

CLRCRH

Clear the CRMHDR file with bracket records.

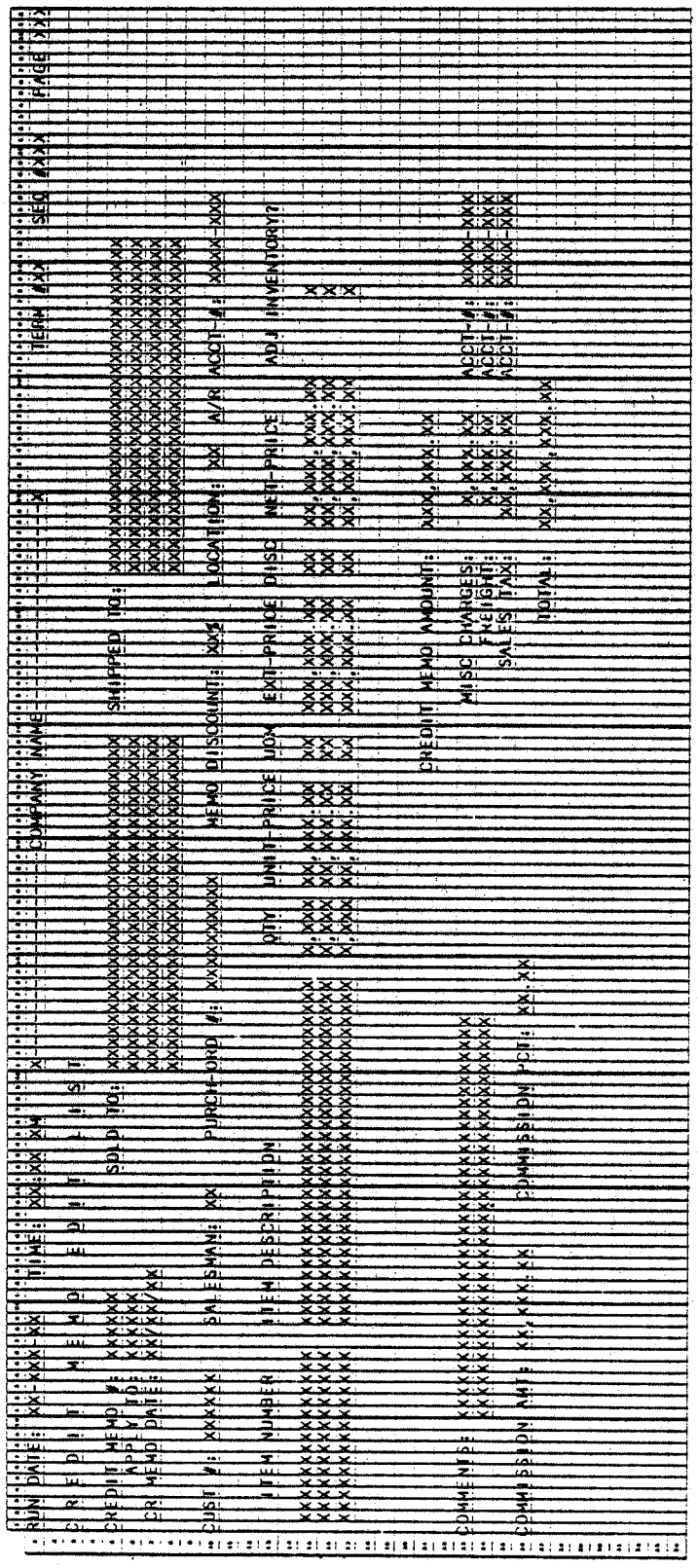
CLRCRL

Clear the CRMLIN file with bracket records.

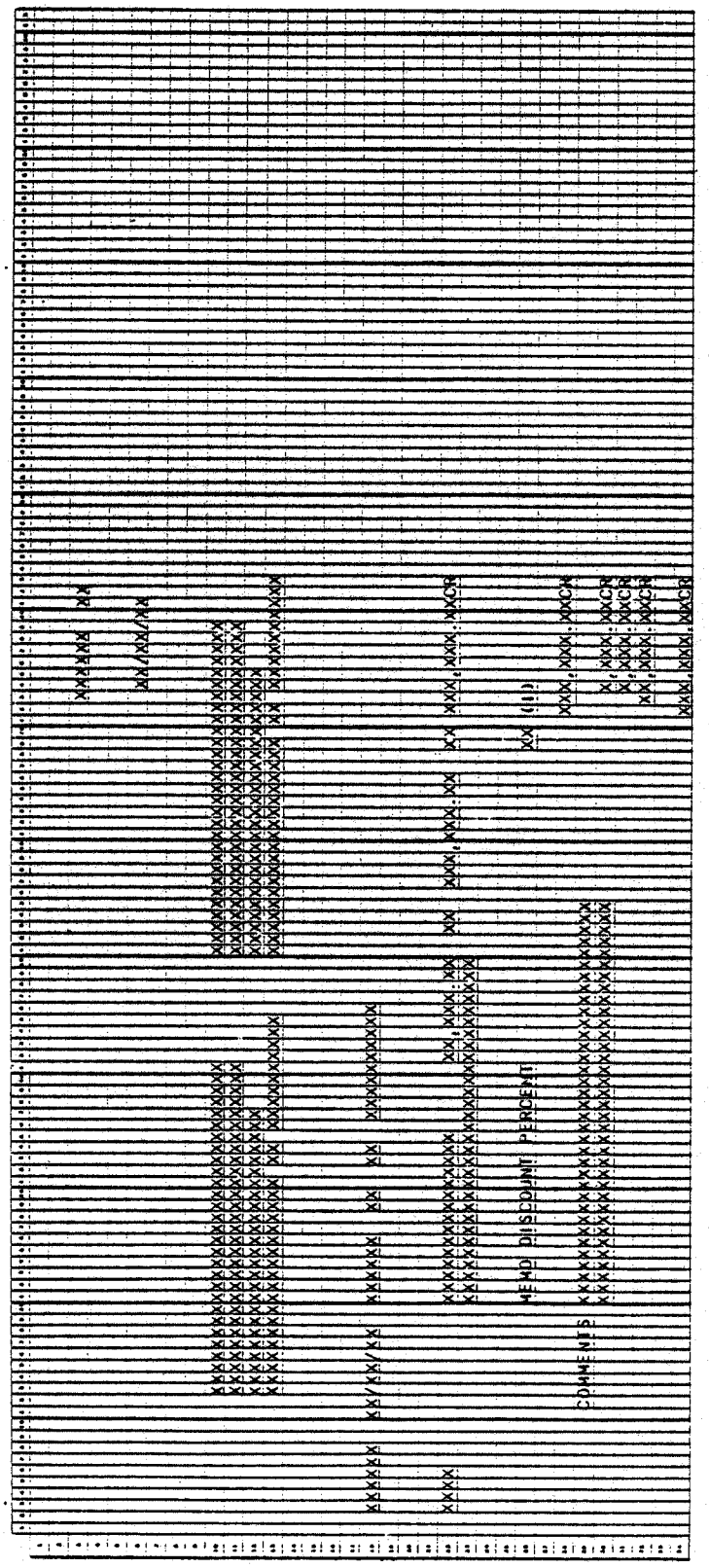
UNPRCM

Unprotect files used during posting. When done, chain back to CRMENT.

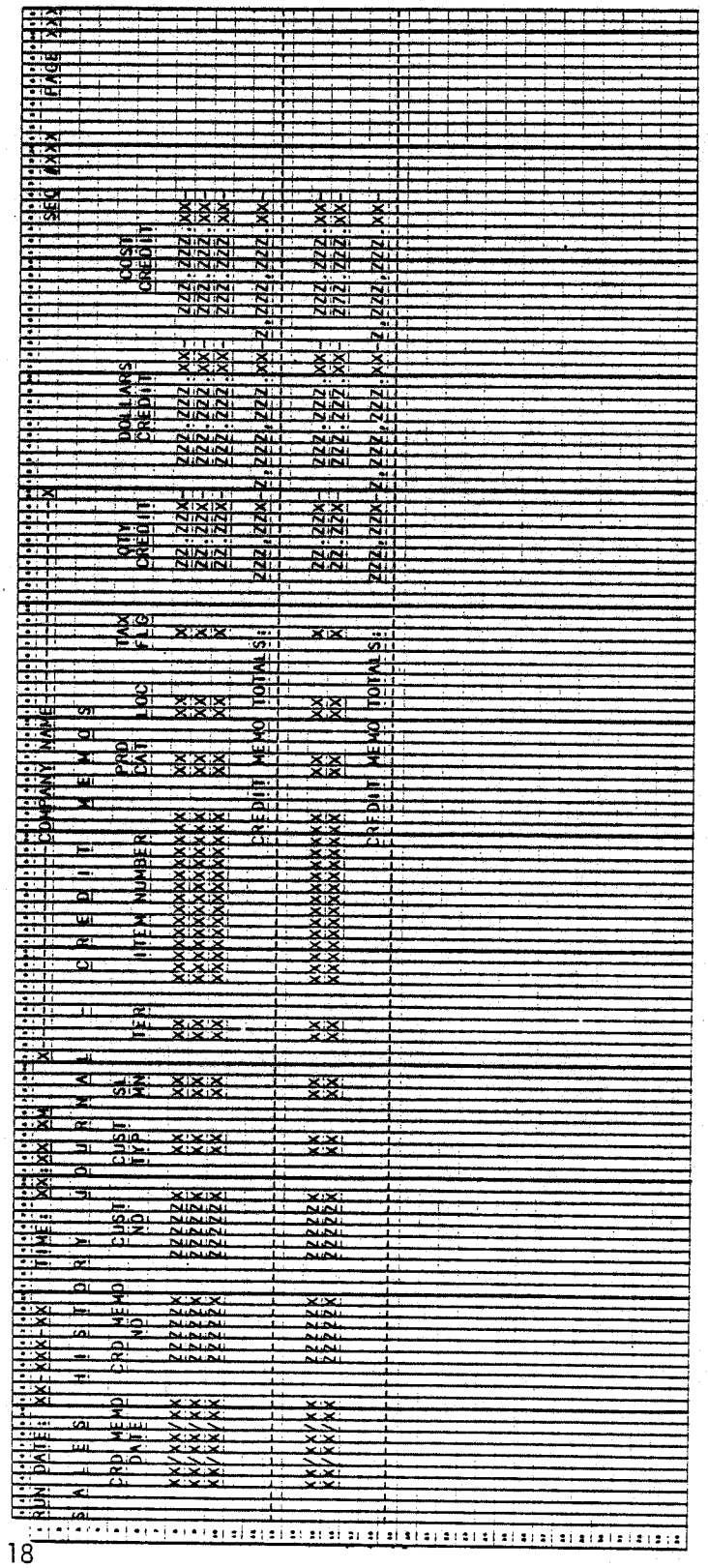
Program: CRMEDT (Credit Memo Edit List)



Program: PRTCRM (Credit Memo)



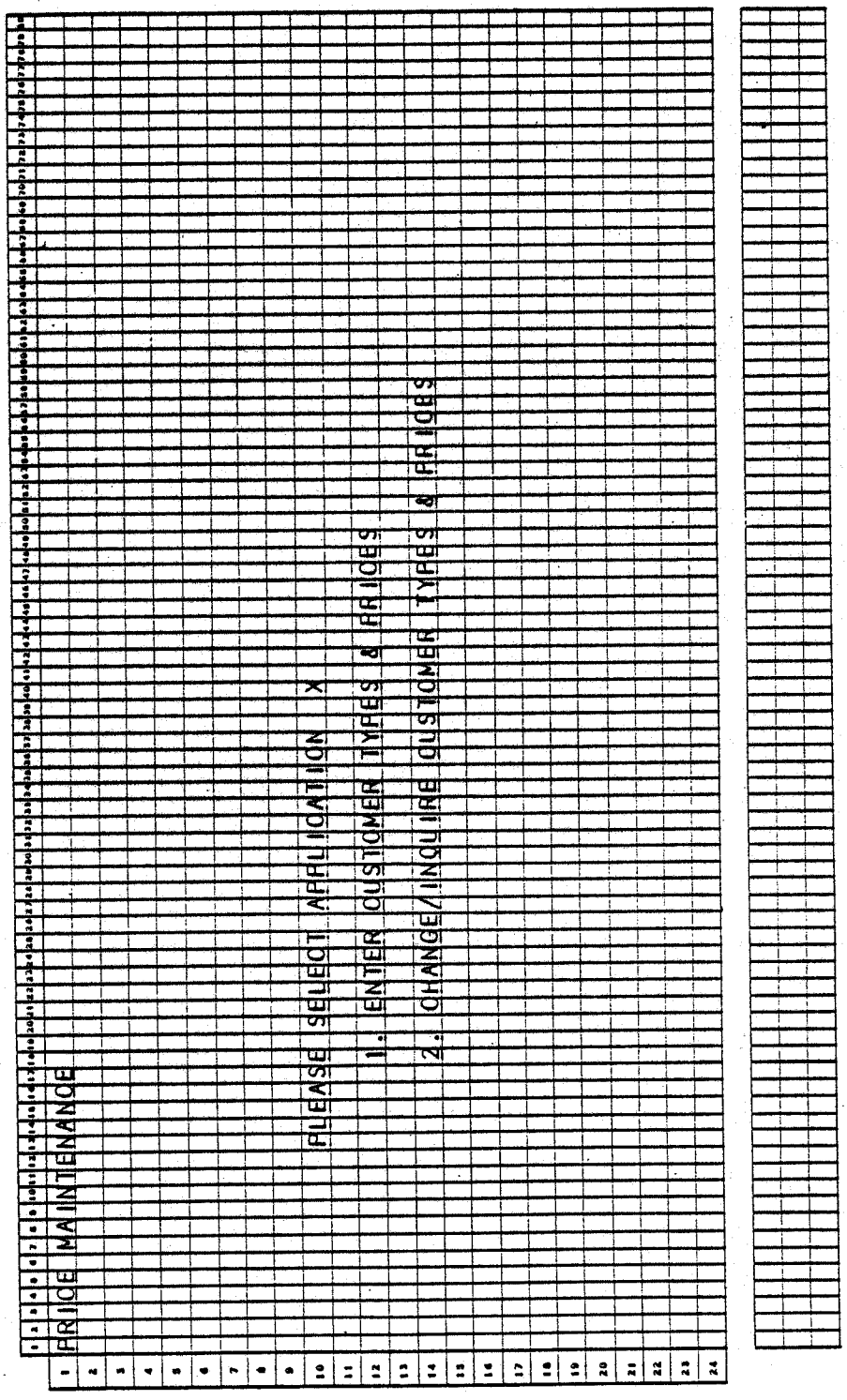
Program: CSLHJL (Sales History Journal - Credit Memos)



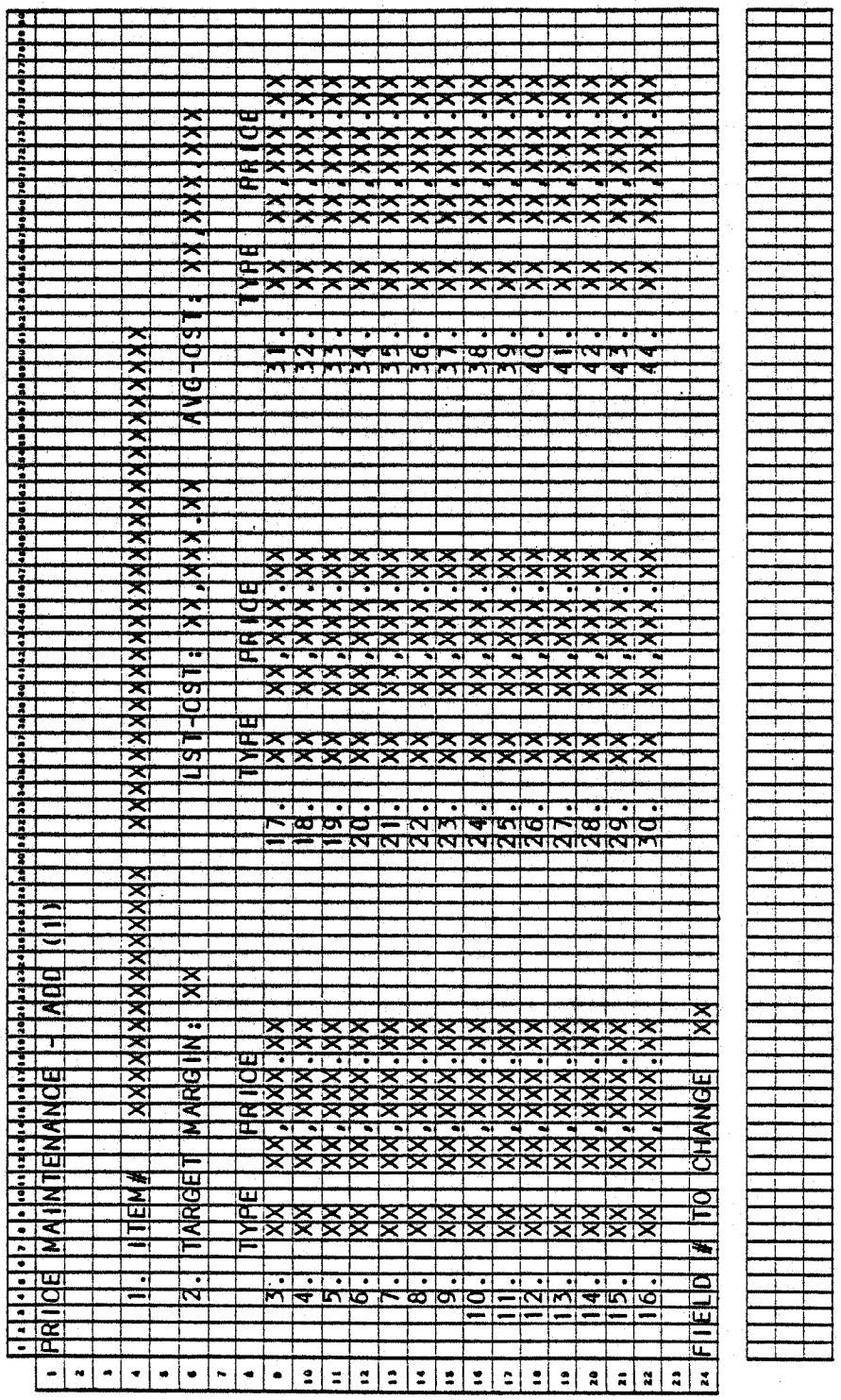
CUSTOMER ORDER PROCESSING PACKAGE
PRICE MAINTENANCE APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: PRCMNT



Program: PRCMNT



Remarks: PRICE ARRAY SIZE DEPENDS ON THE NUMBER OF PRICES PER ITEM GENERATED IN YOUR INVENTORY MANAGEMENT PACKAGE. (1) ADD OR CHANGE/INQUIRE WILL BE DISPLAYED, DEPENDING ON SELECTION FROM MENU.

PRICE MAINTENANCE APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Maintains multiple unit prices per item and associated Price codes that will be matched to like Customer Type codes in the Customer Master file when pricing activity occurs.

Input: ITMMAS
ITMIDX
KBD

Files Updated: ITMMAS

Output: None

Enter Module From: CPMENU

When Done Return To: CPMENU

Programs in Module: PRCMNT

Program Functions and Notes:

PRCMNT

Display the Price Maintenance submenu. After accepting the entered application (add or change/inquire) display as many line numbers as number of prices generated in the system (see ITMMAS Control record for layout).

Accept item number and if the item is found, display it and all prices.

In add mode, accept input of prices beginning with price 1 (field #3). Note that price 1 should not have a Price code associated with it.

Upon each entry of a price and code, (whether in add or change mode) examine the existing codes. Do not allow addition of a Price code that already exists (i.e, duplicate Price codes are not allowed).

Do not allow input of 0 price. Do not allow prices to be deleted except in change mode.

In change mode, allow deletion of any price (except the first price) by entering a RETURN in the Customer Type field. Then "pack" the remaining prices to the front of the array. Do not allow any non-active prices to exist in the middle of the prices in Price code arrays.

Allow a maximum of 42 unit prices per item.

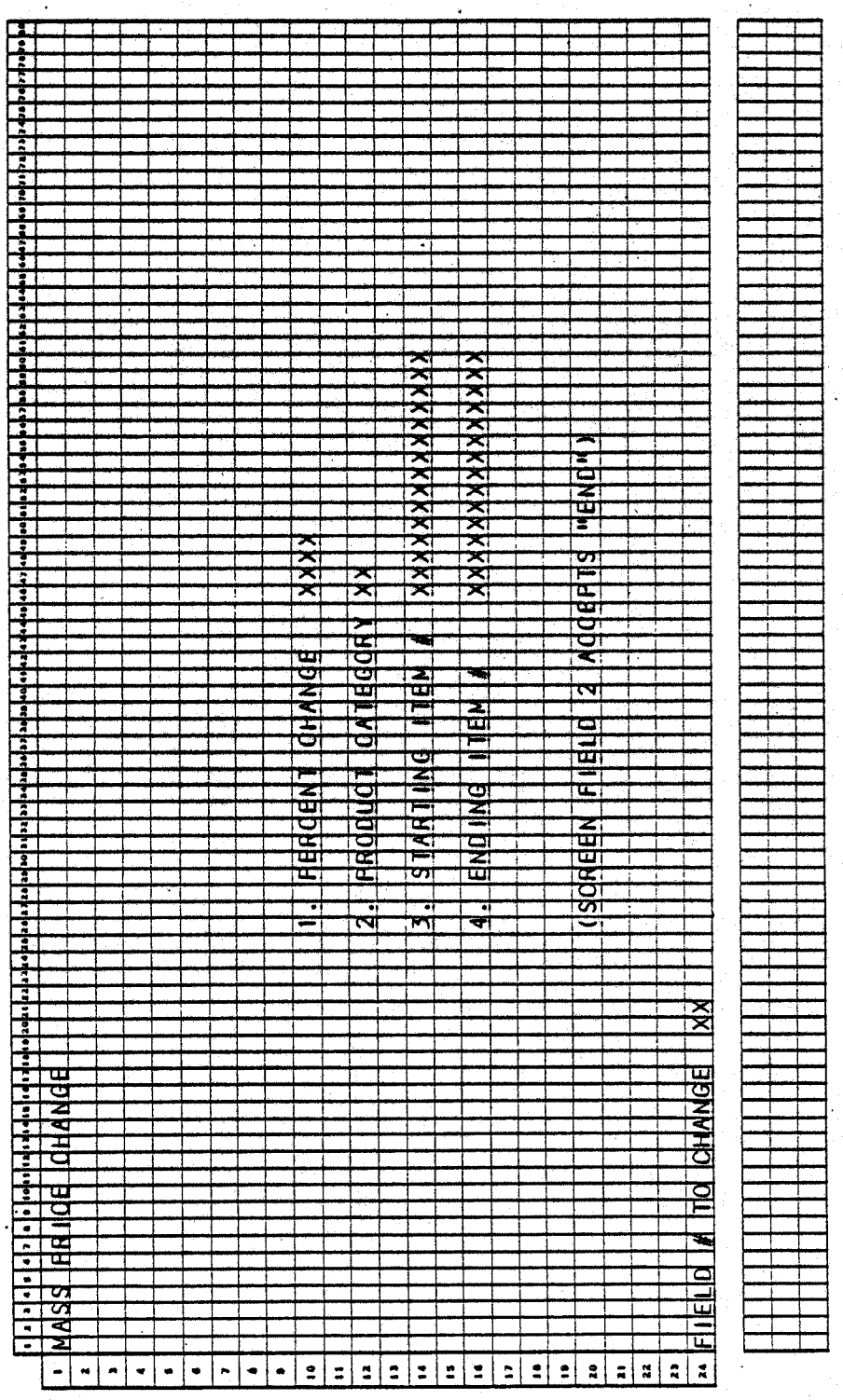
When no more changes are indicated, rewrite the record to the ITMMAS file.

This page intentionally left blank.

CUSTOMER ORDER PROCESSING PACKAGE
MASS PRICE CHANGE APPLICATION
DIBOL JUN-84

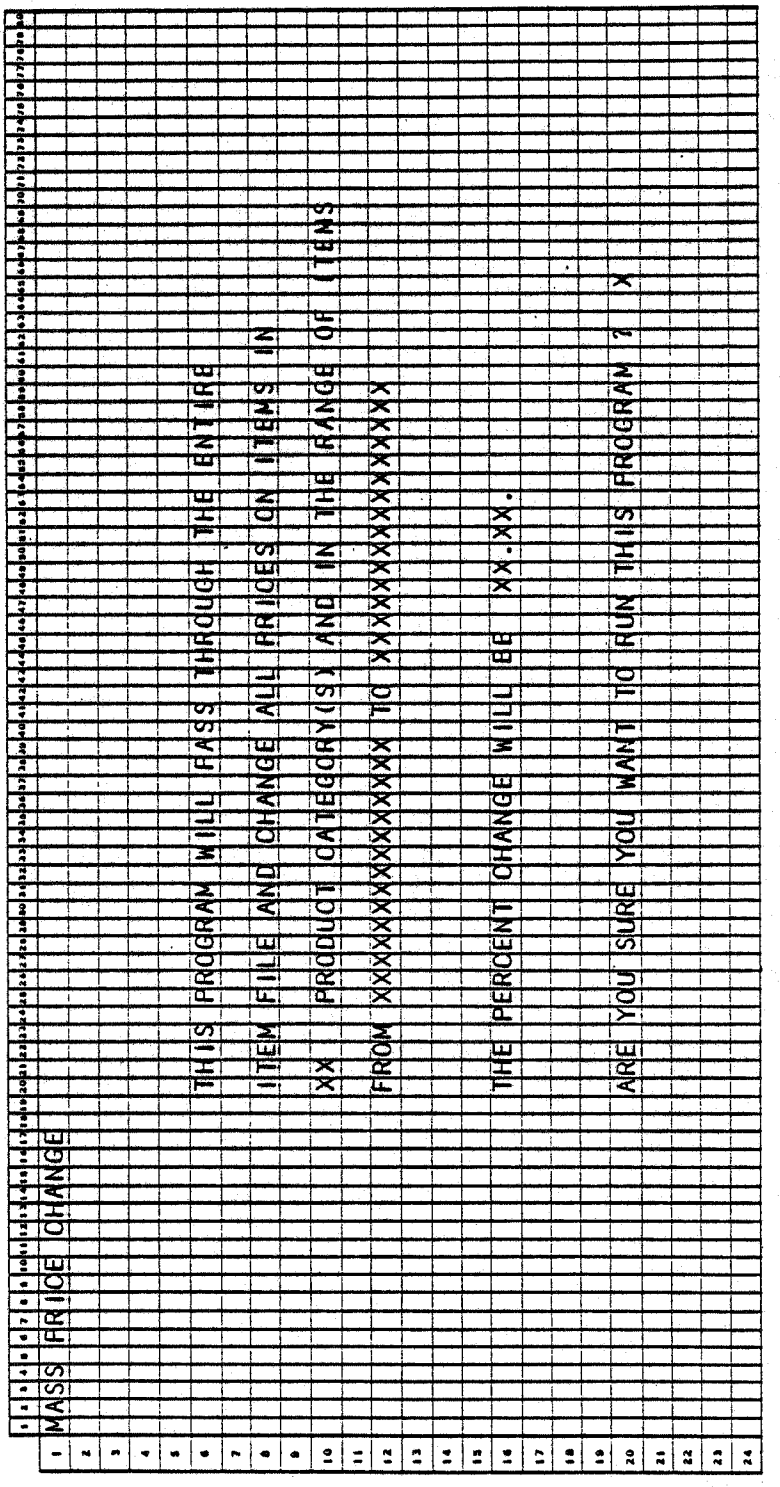
SCREEN FORMATS

Program: PRCCNG



Remarks: ITEMS 2 OR 3 DEFAULT TO "ALL" WHEN RETURN IS PRESSED. BEFORE PROCESSING, DISPLAY A WARNING SCREEN AND REQUIRE CONFIRMATION OF CHANGE.

Program: PRCCNG



Remarks: IMMEDIATELY RETURNS TO CPMENU ON "N" RESPONSE.

MASS PRICE CHANGE APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Mass price change of all unit prices per item, that match the user-entered select criteria, by the user-entered percent change.

Input: ITMMAS
ITMIDX
KBD

Files Updated: ITMMAS

Output: None

Enter Module From: CPMENU

When Done Return To: CPMENU

Programs in Module: PRCCNG

Program Functions and Notes:

PRCCNG

Display the Mass Price Change screen as per page one of the Screen Format.

Accept input for percent change, product category, starting and ending item number.

For percent change, allow the number to be positive or negative with two positions to the right of the decimal point (for instance, to increase a price by 1/2 of 1%, accept input 50 and redisplay it .50).

On blank input for product category, default to all items.

On blank input for starting item number, default to all items.

Display the "ARE YOU SURE ..." warning message (see page 2 of the Screen Format). If a negative answer is received, return to the Customer Order Processing menu.

If a range of items has been input, sequentially read the ITMMAS file until the initial item is found. Then, for all items fitting the designated criteria increase or decrease all prices for that item. Assume the file is sorted in order.

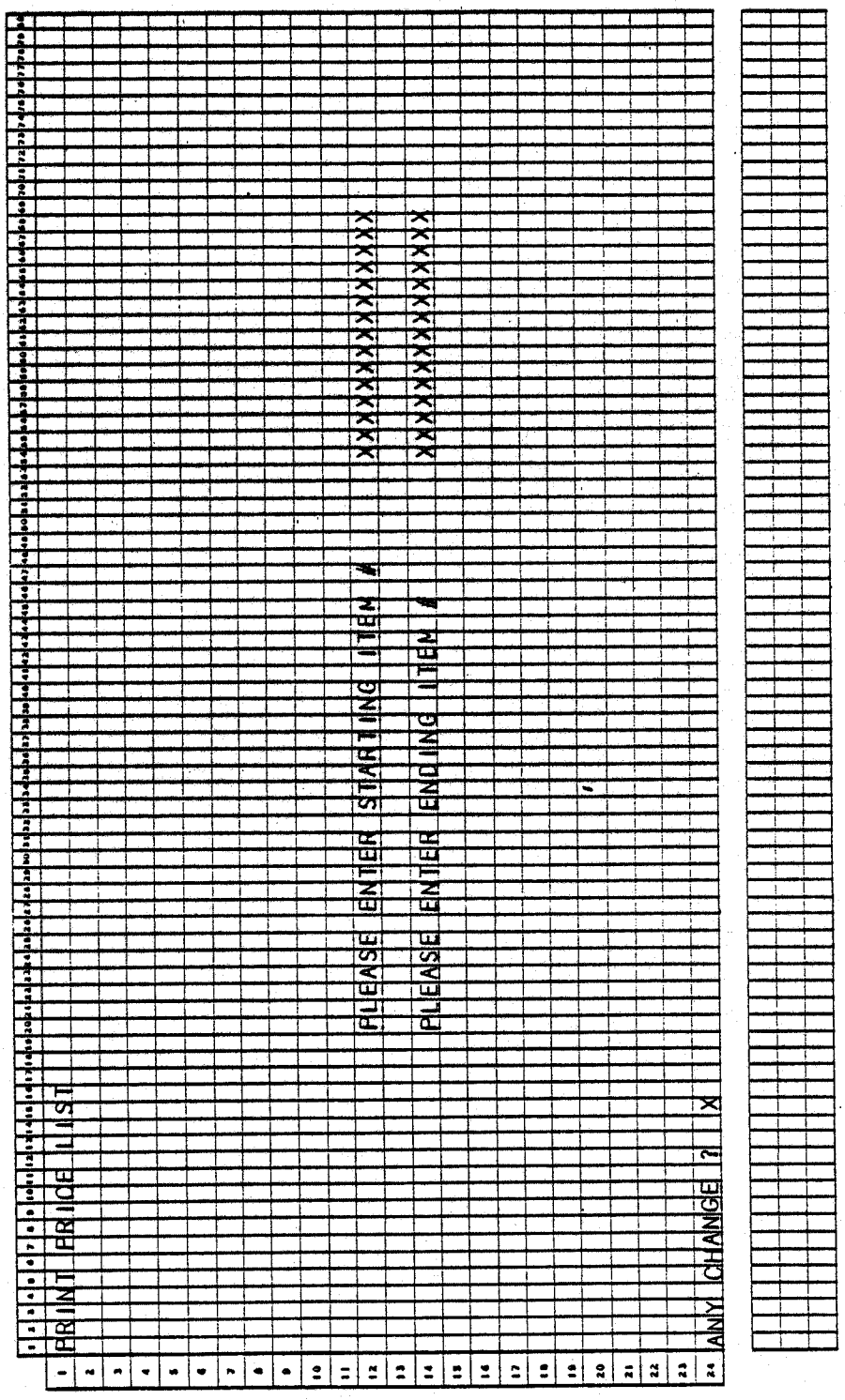
If a single item is selected, perform a binary search to find the item and update it individually.

This page intentionally left blank.

CUSTOMER ORDER PROCESSING PACKAGE
PRINT PRICE LIST APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: PRICES



PRINT PRICE LIST APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Prints Price List

Input: ITMMAS
ITMIDX

Files Updated: None

Output: Price List

Enter Module From: CPMENU

When Done Return To: CPMENU

Program in Module: PRICES

Program Functions and Notes:

PRICES

Write print program. (Note: To be printed on 8" paper--80 column.)

Accept starting and ending item number from keyboard.

Assume the Item Master Index (ITMIDX) is in order and sequentially read the file until you fall within the parameters of the report.

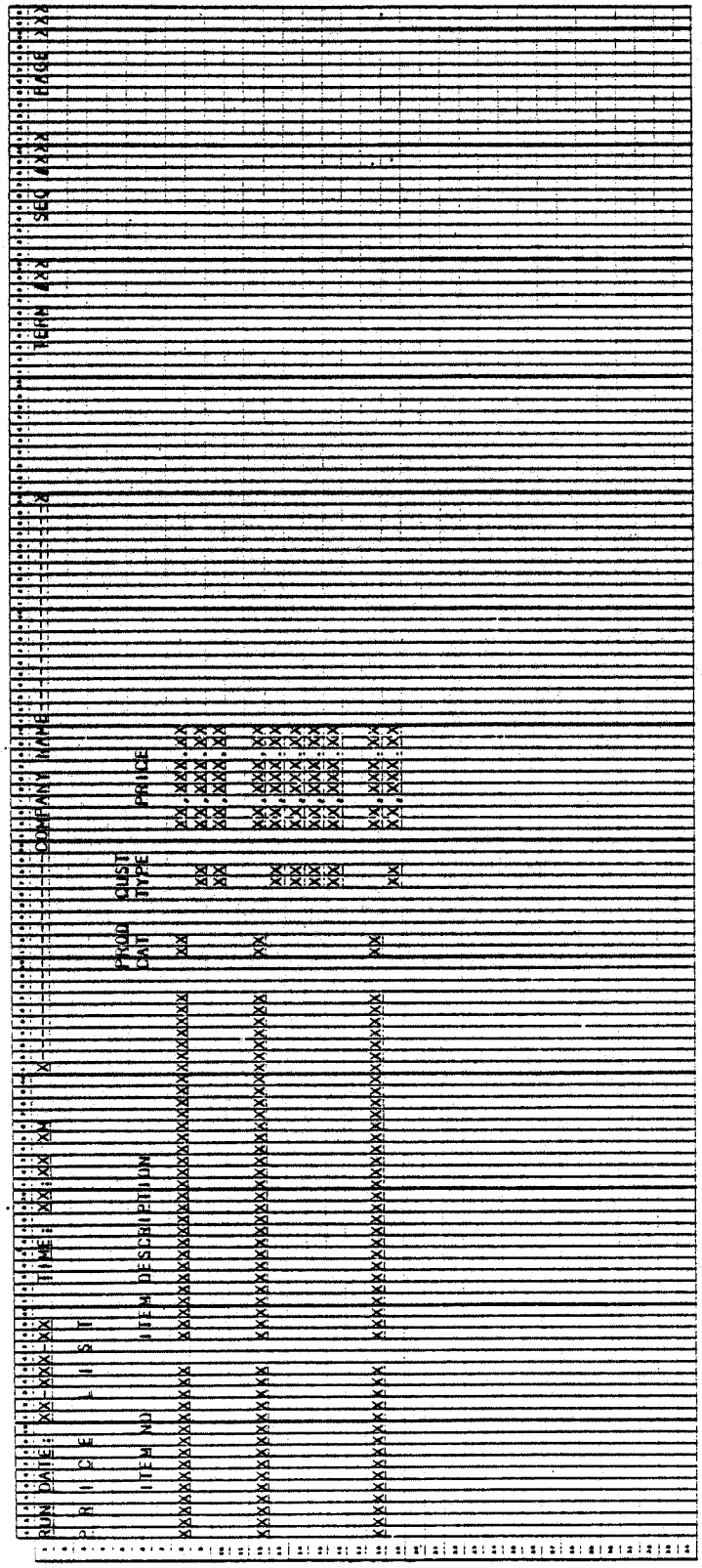
Print report noting that the first Price code in the price and Price code array should be blank.

Find the maximum number of allowable prices from the ITMMAS control record.

If a range was selected, return to accept another range when report is completed (if "ALL" was selected return directly to CPMENU).

PRINT PRICE LIST APPLICATION
DIBOL JUN-84
REPORT FORMATS

Program: PRICES (Price List)



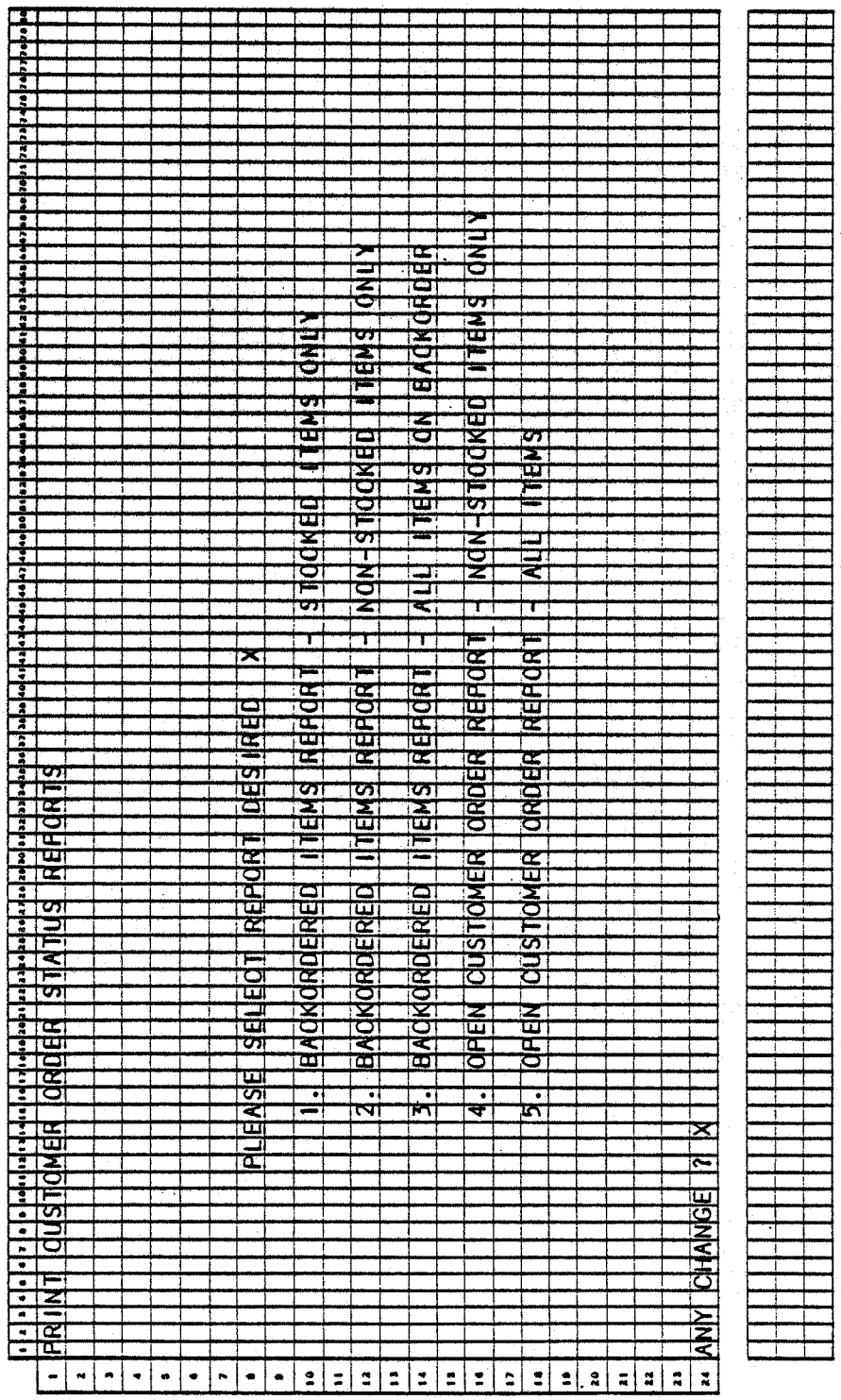
Remarks: FIRST PRICE OF MULTIPLE PRICES HAS NO CUSTOMER TYPE.

This page intentionally left blank.

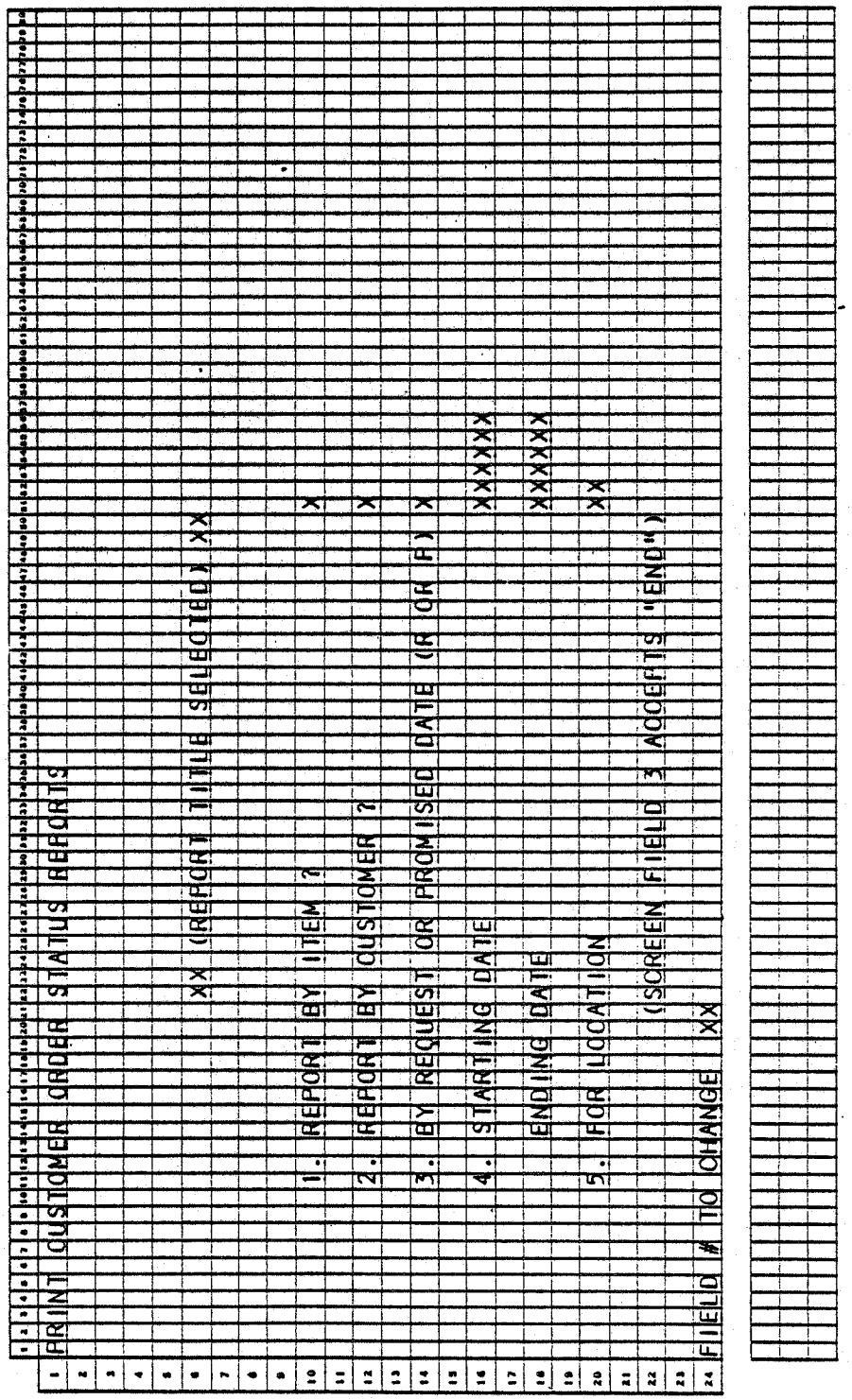
CUSTOMER ORDER PROCESSING PACKAGE
PRINT CUSTOMER ORDER STATUS REPORTS APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: BAKORD



Program: BAKORD



Remarks: RETURN DEFAULTS TO "ALL" FOR LOCATION AND STARTING DATE.

Program: BOITEM/BOCUST

```
 1 PRINT CUSTOMER ORDER STATUS REPORTS
 2 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX (1)
 3
 4
 5
 6
 7
 8
 9
10
11
12                               PLEASE ENTER STARTING ITEM #  XXXXXXXXXXXXXXX
13
14                               PLEASE ENTER ENDING ITEM #    XXXXXXXXXXXXXXX
15
16
17
18
19
20
21
22
23
24 ANY CHANGE ?  X
```

Remarks: (1) ONE OF THE FOLLOWING WILL DISPLAY ON LINE 2.
BACKORDER REPORT BY ITEM - STOCKED ITEMS ONLY
BACKORDER REPORT BY ITEM - NON-STOCKED ITEMS ONLY
BACKORDER REPORT BY ITEM - ALL ITEMS
OPEN ORDERS BY ITEM - NON-STOCKED ITEMS ONLY
OPEN ORDERS BY ITEM - ALL ITEMS

PRINT CUSTOMER ORDER STATUS REPORTS APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Prints Back Order Reports

Input:	Files Updated:	Output:
KBD	BAKORD	Customer Order Status Report by Item
ORDHDR	BOINDX	Customer Order Status Report by Customer
ORDLIN		

Enter Module From: CPMENU

When Done Return To: CPMENU

Programs in Module: BAKORD, SRTBIT, BOITEM, SRTBCU, BOCUST, UNPRBO

Program Functions and Notes:

The program flow for this module is:

BAKORD -- SRTBIT -- BOITEM -- SRTBCU -- BOCUST -- UNPRBO

Skip the second and third programs if no Back Order Report by Item, and the fourth and fifth programs if no Back Order Report by Customer.

BAKORD

Ask which report is desired and set flags.

For that report, determine parameters within which to report. Default to all dates by carriage return at starting date, default to all locations by carriage return at location.

Build the temporary BAKORD file and its index BOINDX using the ORDHDR and ORDLIN files.

If "by item" report is selected, change to SRTBIT. If not, chain to SRTBCU.

SRTBIT

Sort the BOINBX file by item number, location, and date, in that order of importance.

Note that this date might be request date or promised date, although which it is, does not matter for this sort.

BOITEM

Print Open Order Report by Item.

Before printing accept range of items from terminal or "ALL".

If a range was selected, return to request another range before proceeding.

If end input for starting item number, proceed to SRTBCU or UNPRBO, as appropriate.

SRTBCU

Sort BOINDX file by customer number, item number, then order (or schedule) date in that order of importance.

BOCUST

Print Open Order Report by Customer.

Before printing, accept starting and ending customer numbers, or "ALL".

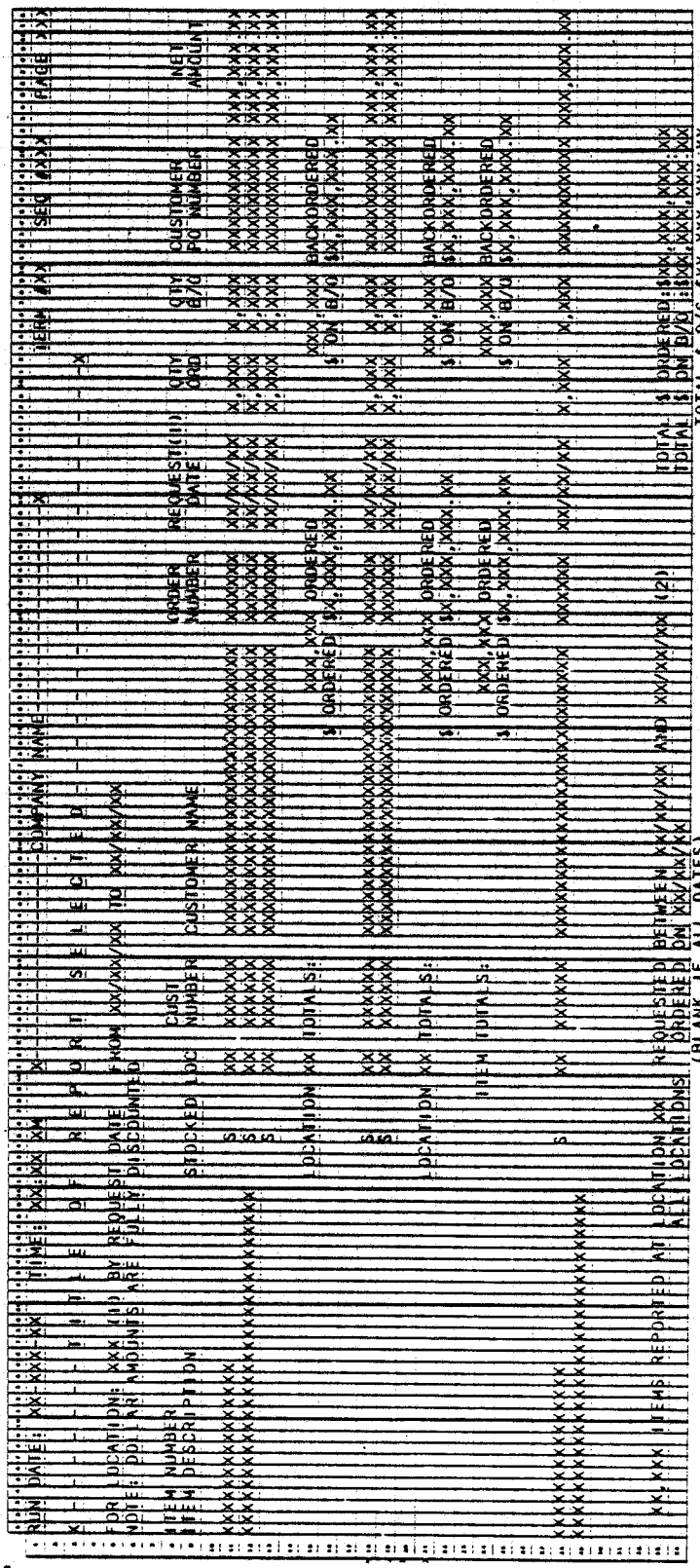
If a range is printed, return to starting and ending input when report is finished.

UNPRBO

Unprotect BAKORD and BOINDX files, and delete them.

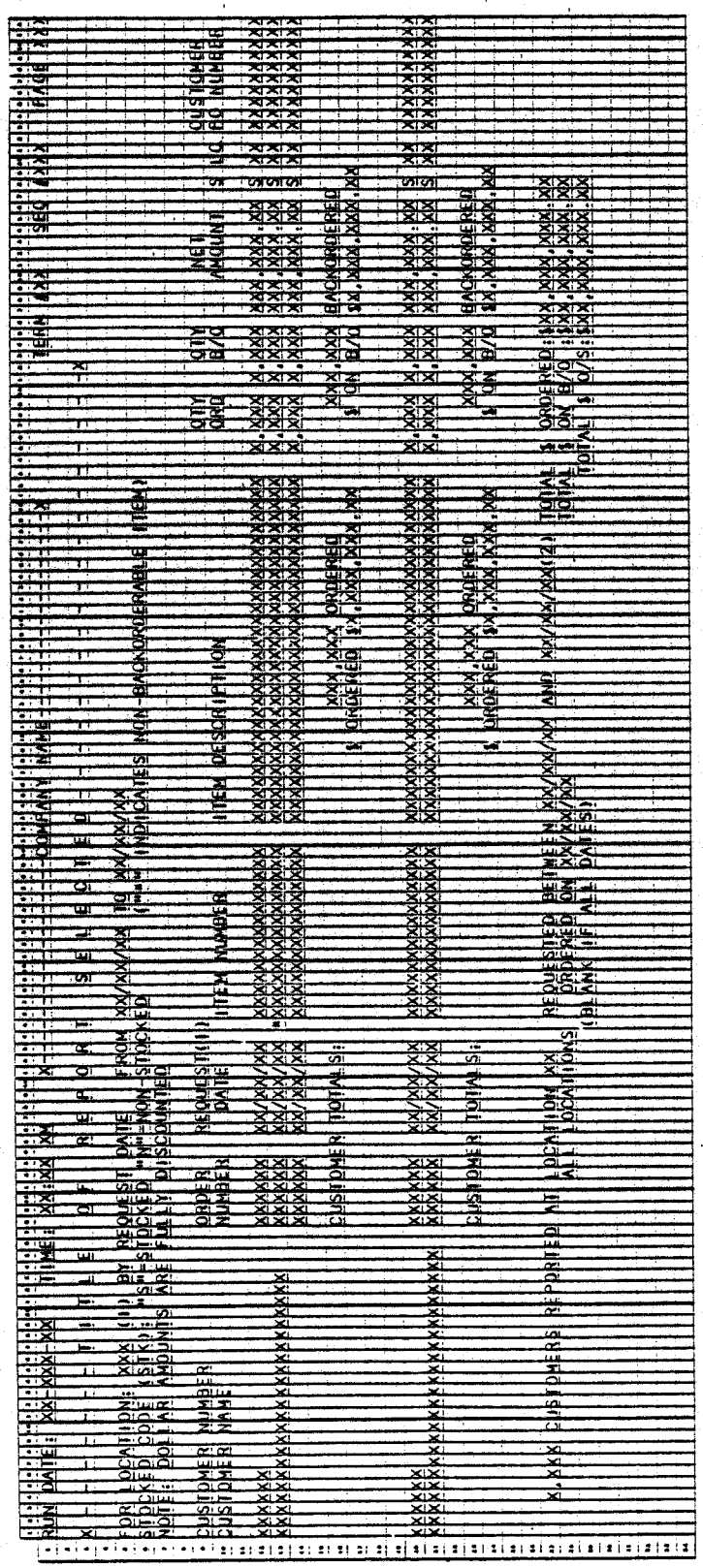
REPORT FORMATS

Program: BOITEM (Customer Order Status Report By Item)



Remarks: SORTED BY ITEM NUMBER, THEN BY LOCATION, THEN BY DATE. SUMMARY LINES DO NOT PRINT IF THERE IS ONLY 1 DETAIL LINE. IF PROMISED DATE WAS SELECTED AND IS ZERO FOR AN ITEM, THE REQUEST DATE WILL BE SUBSTITUTED.
(1) WILL PRINT EITHER "REQUEST" OR "PROMISE" DEPENDING ON SELECTION.
(2) WILL PRINT EITHER "REQUESTED" OR "PROMISED" DEPENDING ON SELECTION.

Program: BOCUST (Customer Order Status Report By Customer)



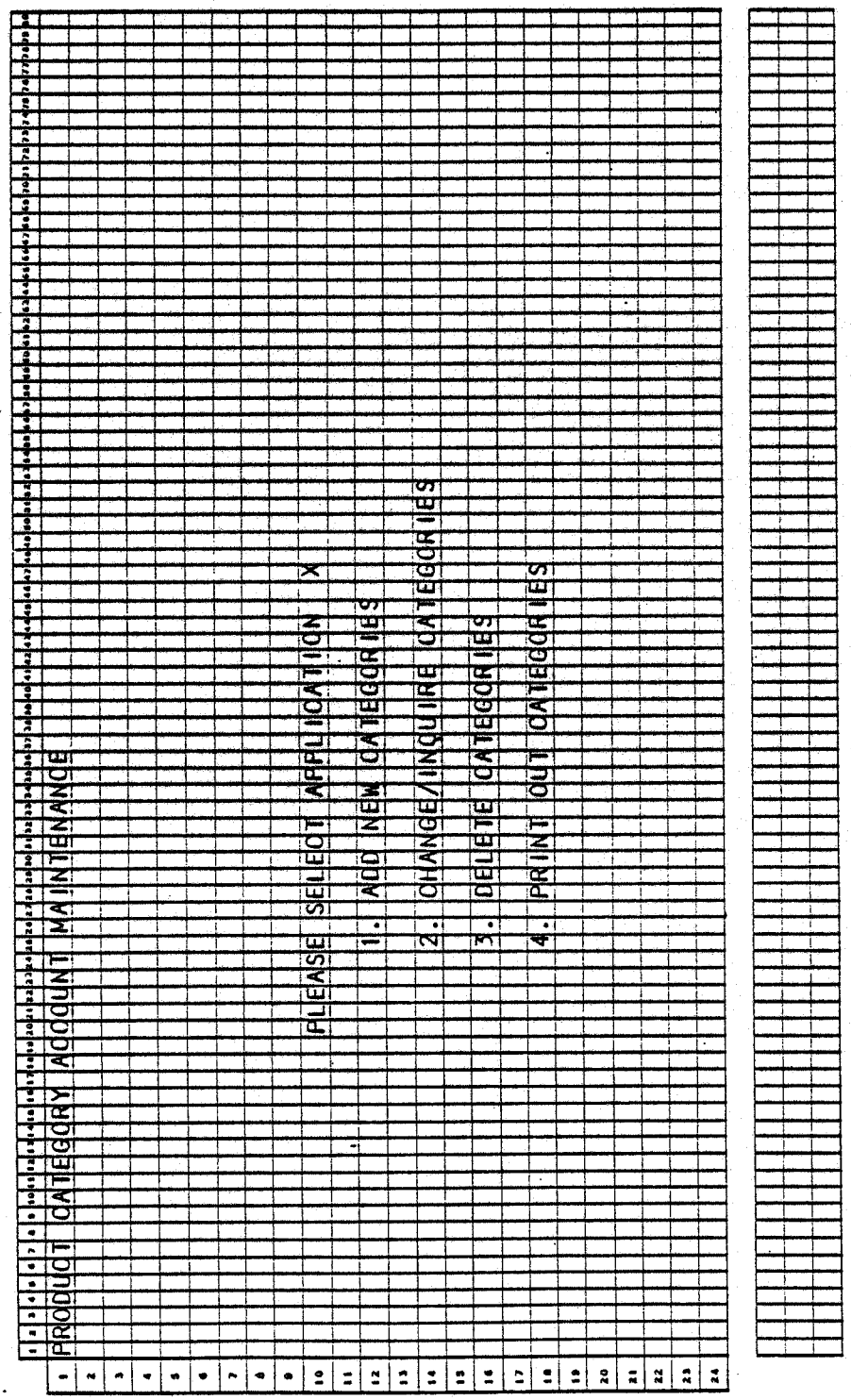
Remarks: SORTED BY CUSTOMER NUMBER, THEN BY ITEM NUMBER, THEN BY REQUEST (OR PROMISED) DATE. SUMMARY LINE DOES NOT PRINT IF THERE IS ONLY 1 DETAIL LINE. IF REPORTING BY PROMISED DATE WAS SELECTED AND PROMISED DATE IS ZERO, THE REQUEST DATE WILL BE SUBSTITUTED.
(1) WILL PRINT EITHER "REQUEST" OR "PROMISE" DEPENDING ON PROGRAM SELECTED.
(2) WILL PRINT EITHER "REQUESTED" OR "PROMISED" DEPENDING ON PROGRAM SELECTED.

This page intentionally left blank.

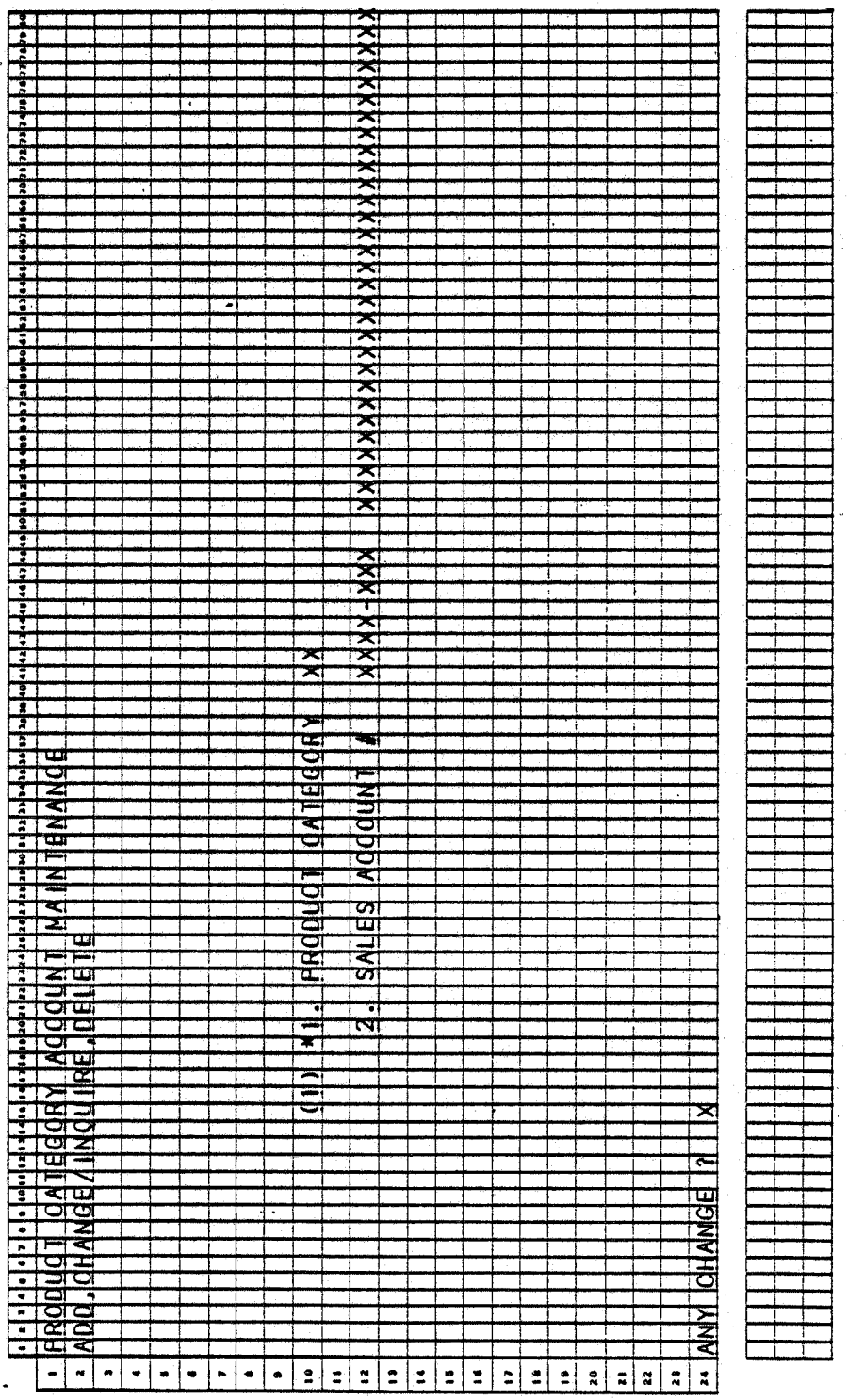
CUSTOMER ORDER PROCESSING PACKAGE
PRODUCT CATEGORY ACCOUNT MAINTENANCE APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: PDAMNT

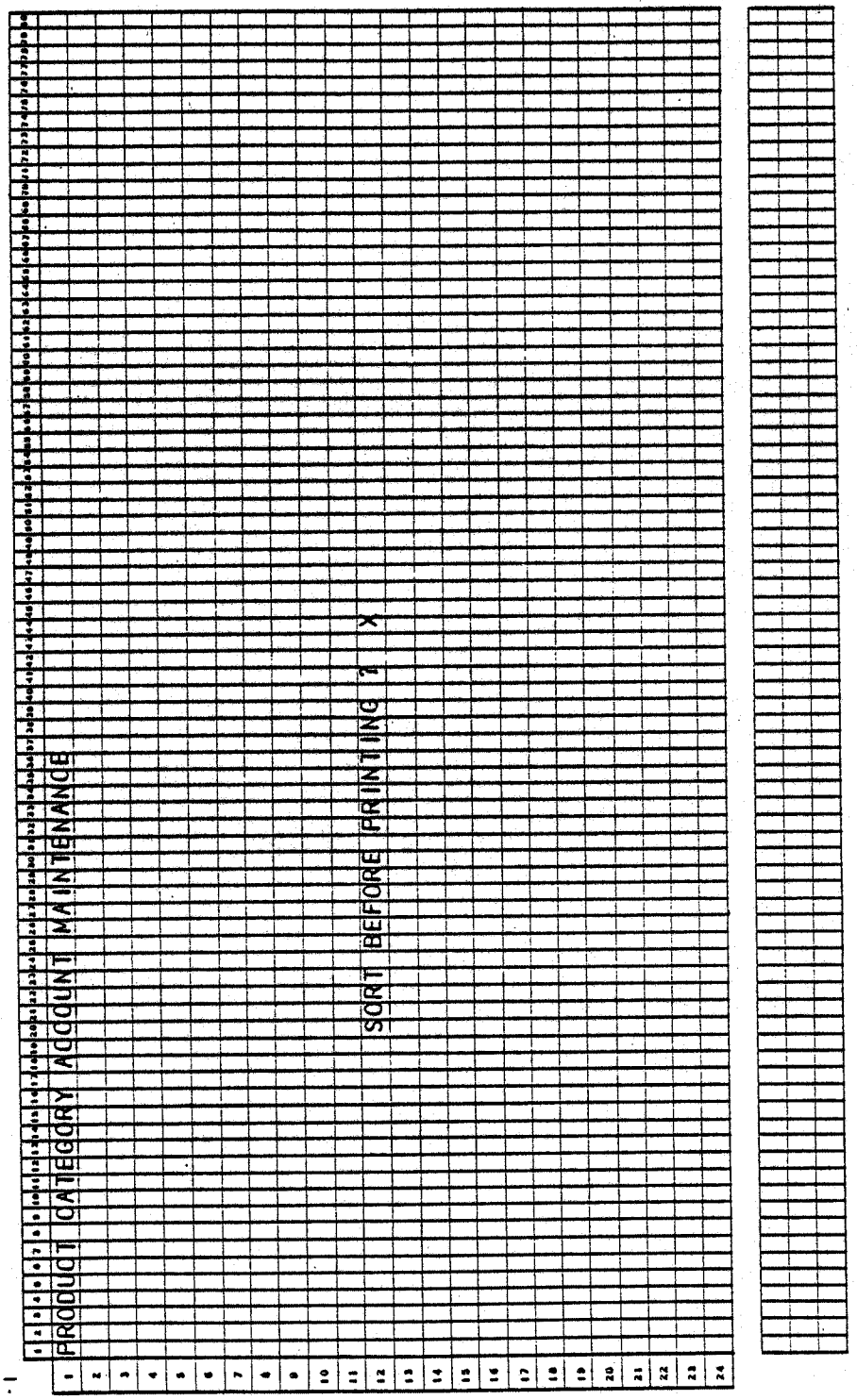


Program: PDAMNT

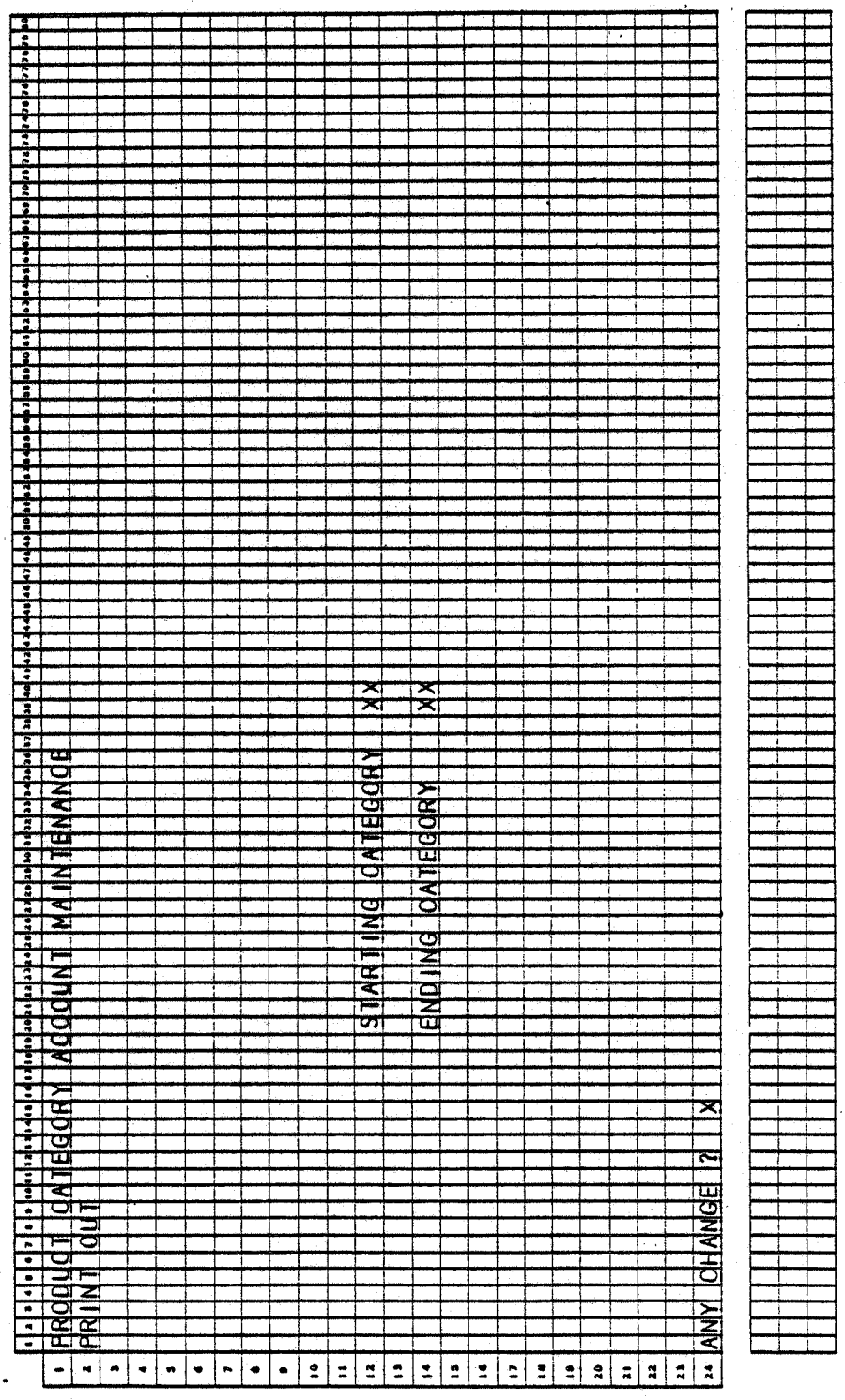


Remarks: (1) ASTERISK WILL BE OMITTED IN "ADD" MODE.

Program: PDAMNT



Program: PDALST



PRODUCT CATEGORY ACCOUNT MAINTENANCE APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Maintains the relationship between product categories in the Item Master records and G/L sales account numbers. Allows multiple distribution of sales.

Input: KBD
PRDACT

Files Updated: PRDACT

Output: Product Category Account File Print-Out

Enter Module From: CPMENU

When Done Return To: CPMENU

Programs in Module: PDAMNT, PDALST

Program Functions and Notes:

PDAMNT

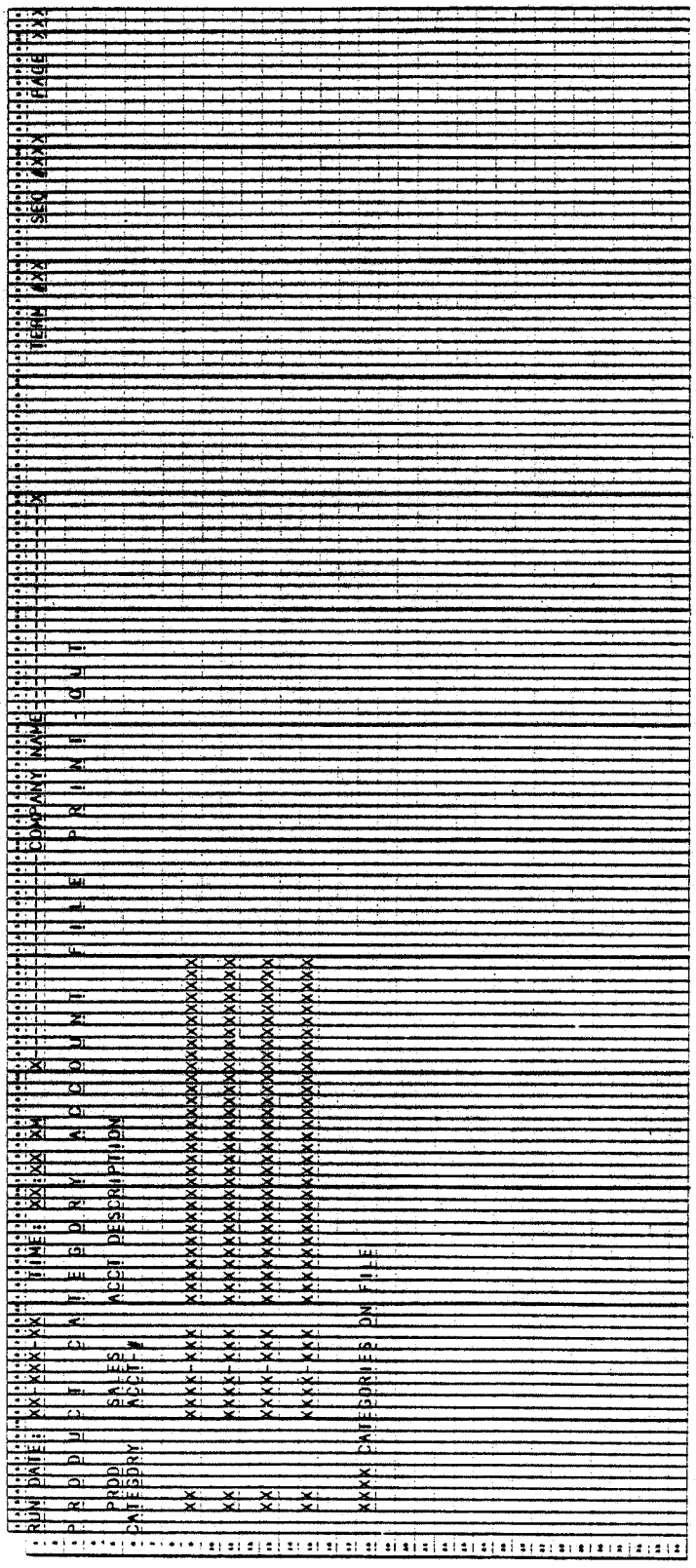
Allow only one account number per product category. There is no validation done on the product category itself. But, check for duplicates in the PRDACT file and do not allow.

Validate the account number from the ARACCT file and display the account description. (The only reason the account description is kept on file is because the record would be too short for a control record otherwise).

PDALST

List the records in the PRDACT per range selected per Report Format.

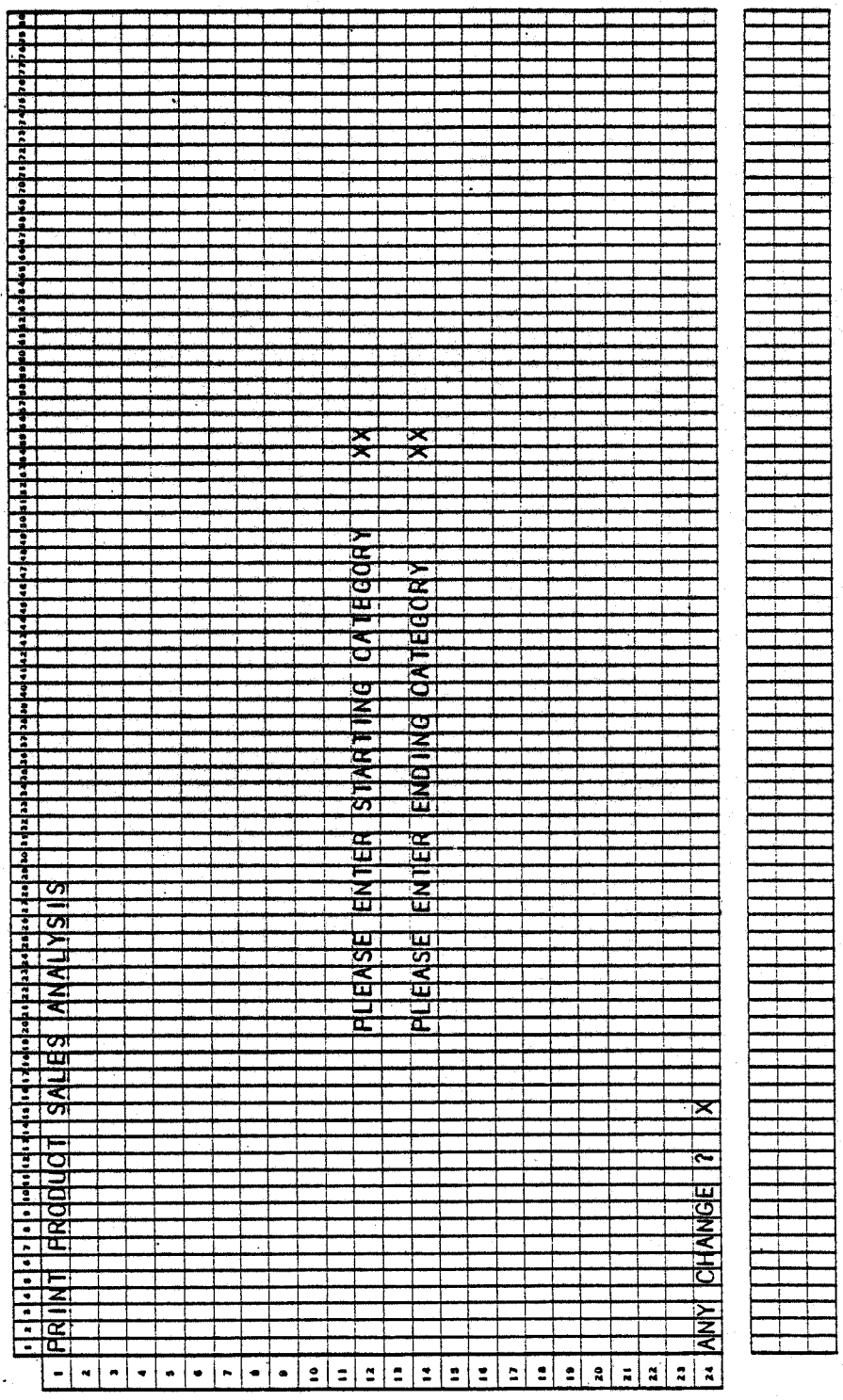
Program: PDALST (Product Category Account File Print-Out)



CUSTOMER ORDER PROCESSING PACKAGE
PRINT PRODUCT SALES ANALYSIS APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: ANALYS



PRINT PRODUCT SALES ANALYSIS APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Prints Sales Analysis Report

Input: ITMIDX
ITMMAS

Files Updated: SAPIDX

Output: Sales Analysis by Product Category

Enter Module From: CPMENU

When Done Return To: CPMENU

Programs in Module: ANALYS, STSAPC, SAPCAT

Program Functions and Notes:

ANALYS

Request entry of starting and ending product category and create SAPIDX of all items in these categories.

STSAPC

Sort the SAPIDX file on item number within category.

SAPCAT

For each category 1) Read from beginning to end of category. Accumulate monthly and yearly sales and cost of sales. Compute gross profit. 2) Read through category again and print data indicated for each item. Percentages are sales and profit of that item as compared to the sales and profit totals of the product category. 3) Print category totals, accumulate category sales, cost of sales and gross profit into grand total accumulators. Clear out category totals. Repeat 1-3 above for all categories.

The "Summary" is a recapitulation of the category totals monthly and yearly figures.

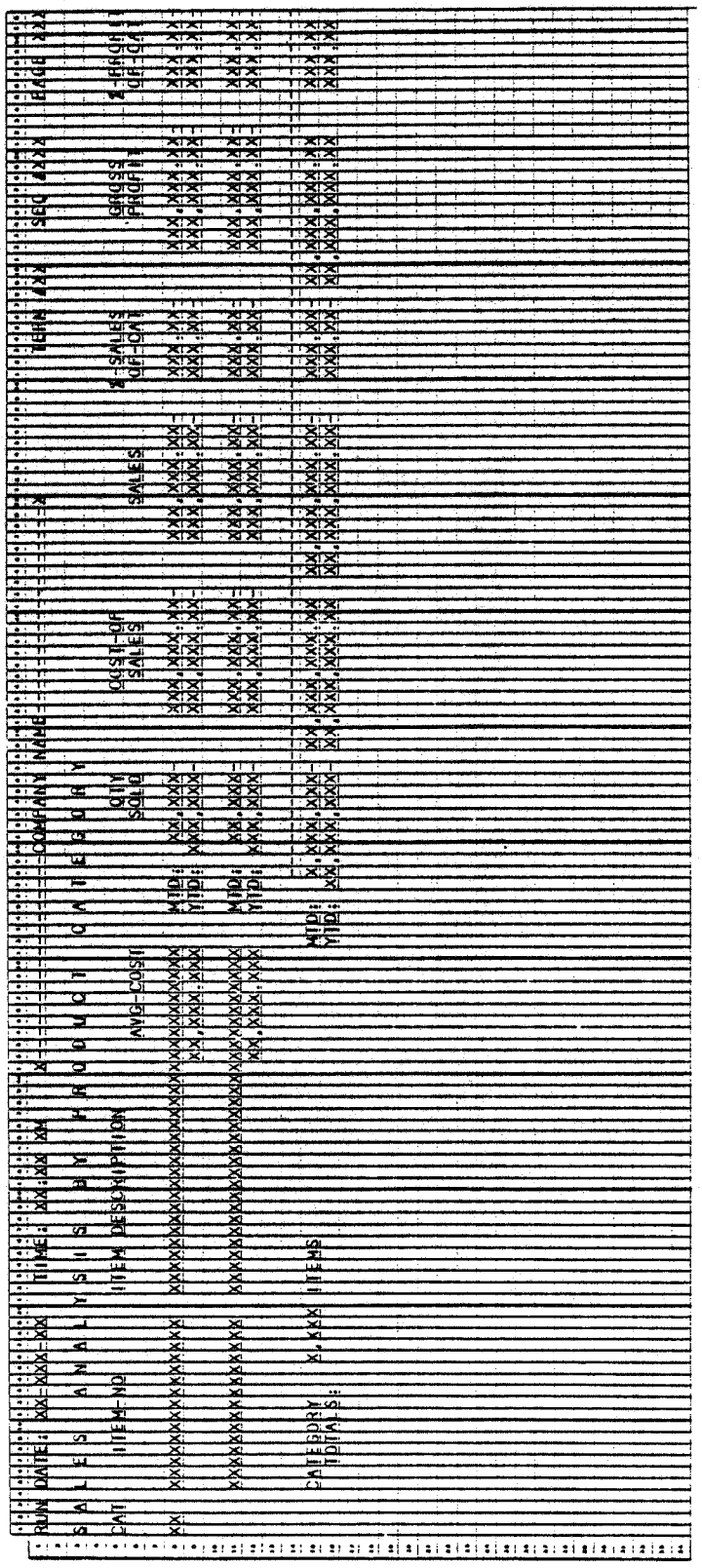
Read through the index file again, level breaking on each new category. Then calculate percentages category totals to grand totals. Print category figures. Include a count of items in the category. Print grand totals. Category averages are the grand totals divided by number of categories.

Delete SAPIDX file.

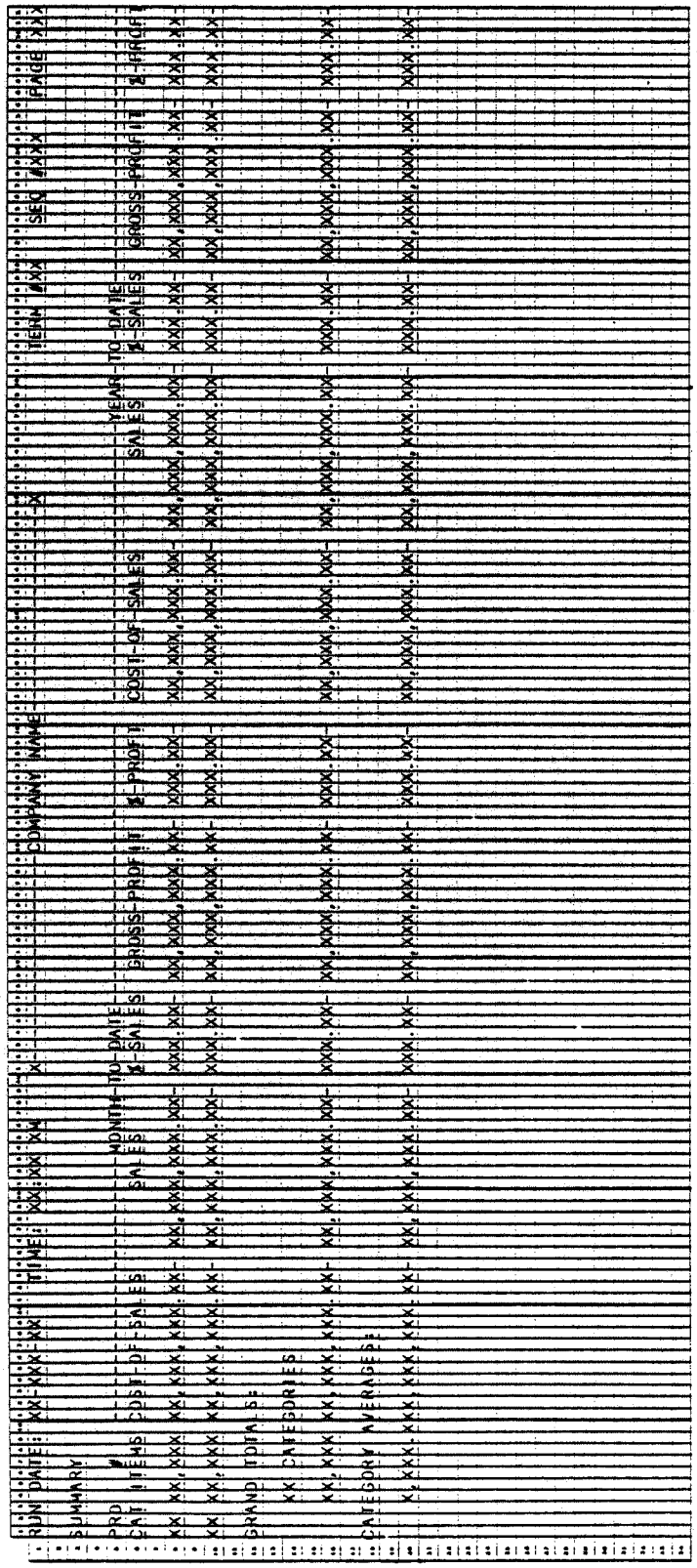
NOTE:

Leave SAPIDX file protected for exclusive use, from time of creation until it is deleted, at start of program. Proceed only if the permanent Master File Index (ITMIDX) is not protected elsewhere on the system.

Program: SAPCAT (Sales Analysis By Product Category)



Program: SAPCAT (Summary)

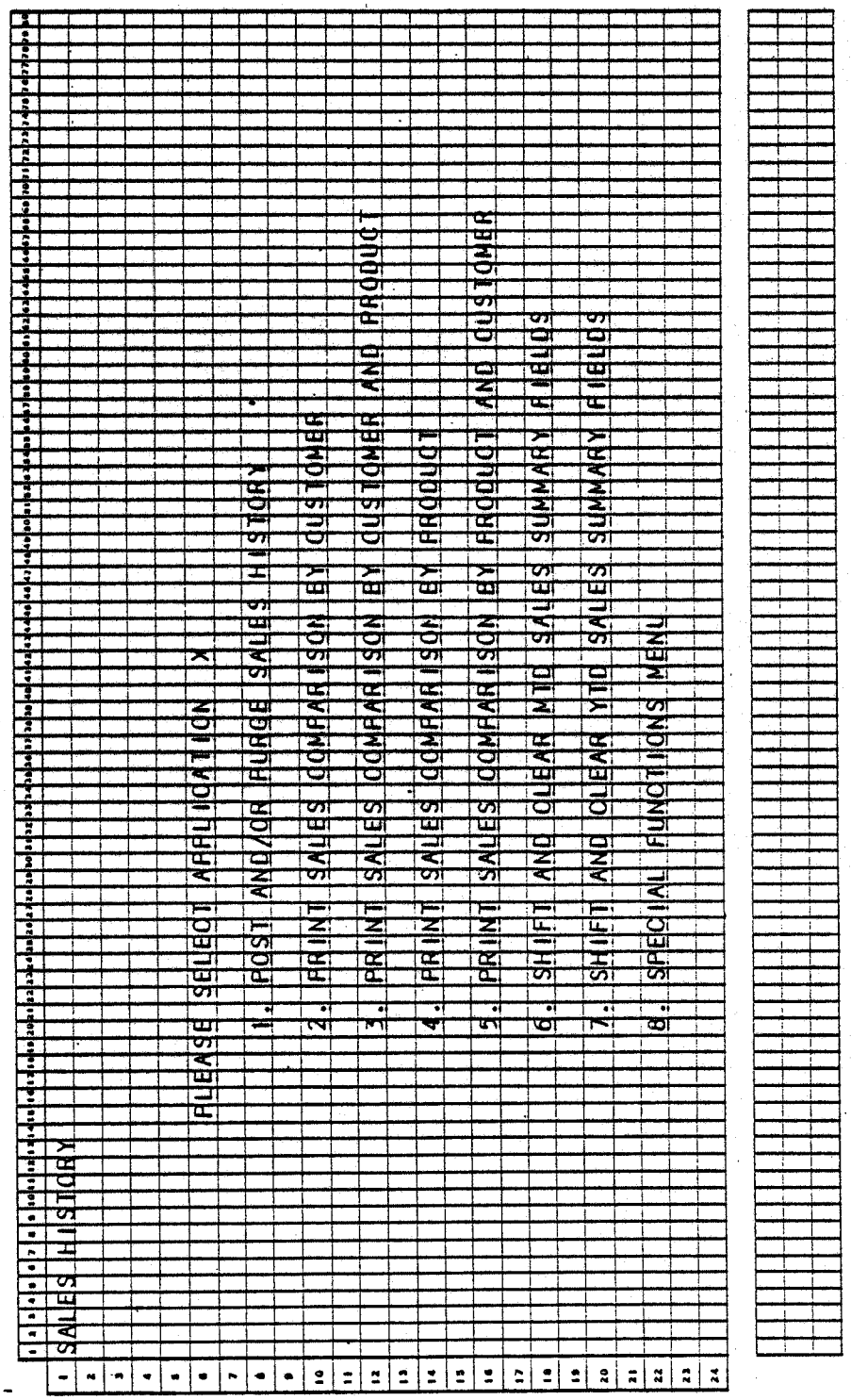


This page intentionally left blank.

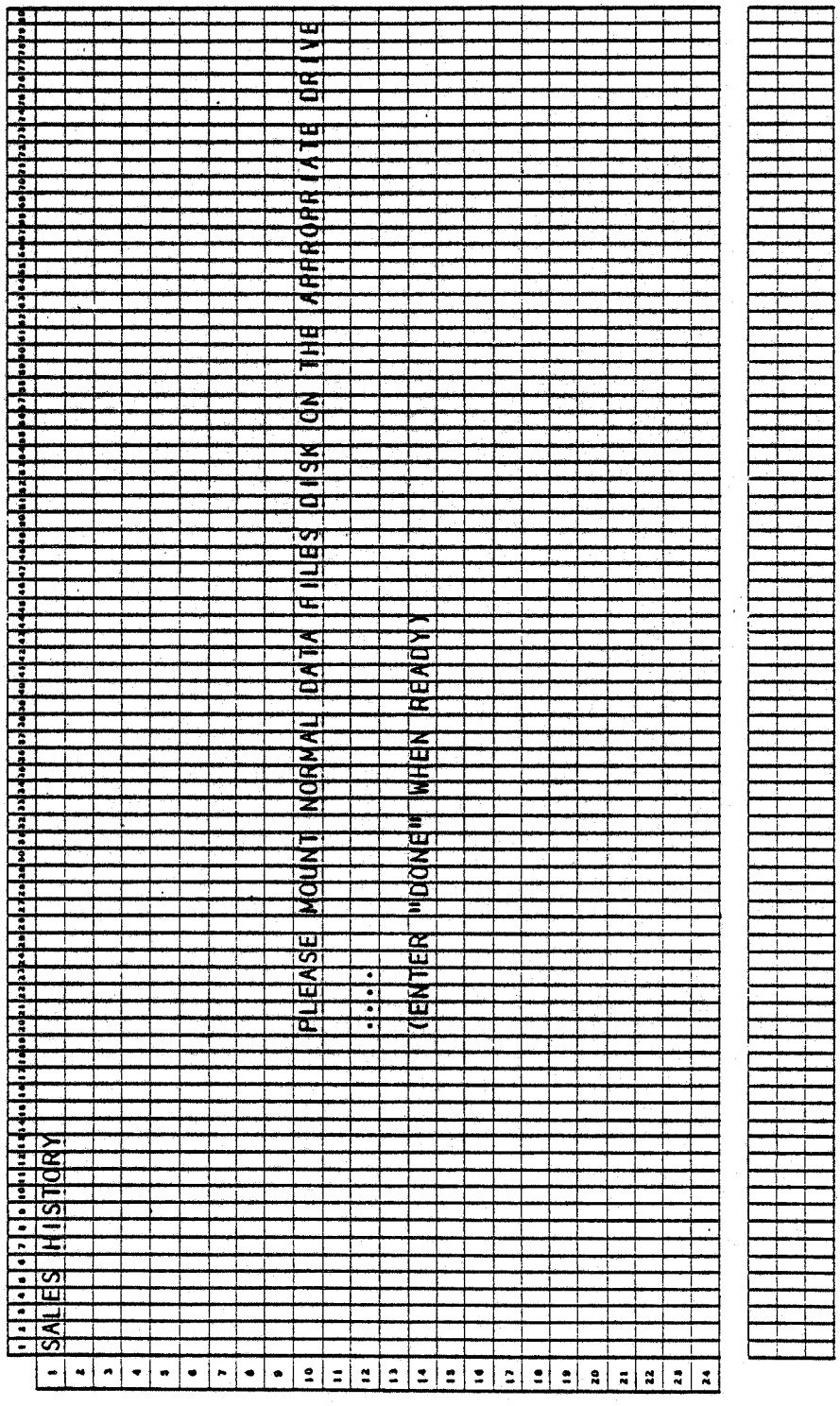
CUSTOMER ORDER PROCESSING PACKAGE
SALES HISTORY MENU APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: SSMENU



Program: SSMENU



SALES HISTORY MENU APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Sales History menu.

Input: Files Updated: Output: None

Enter Module From: CPMENU When Done Return To: CPMENU

Programs in Module: SSMENU

Program Functions and Notes:

SSMENU

Display screen as per the Screen Format.

Accept selection and go to appropriate program.

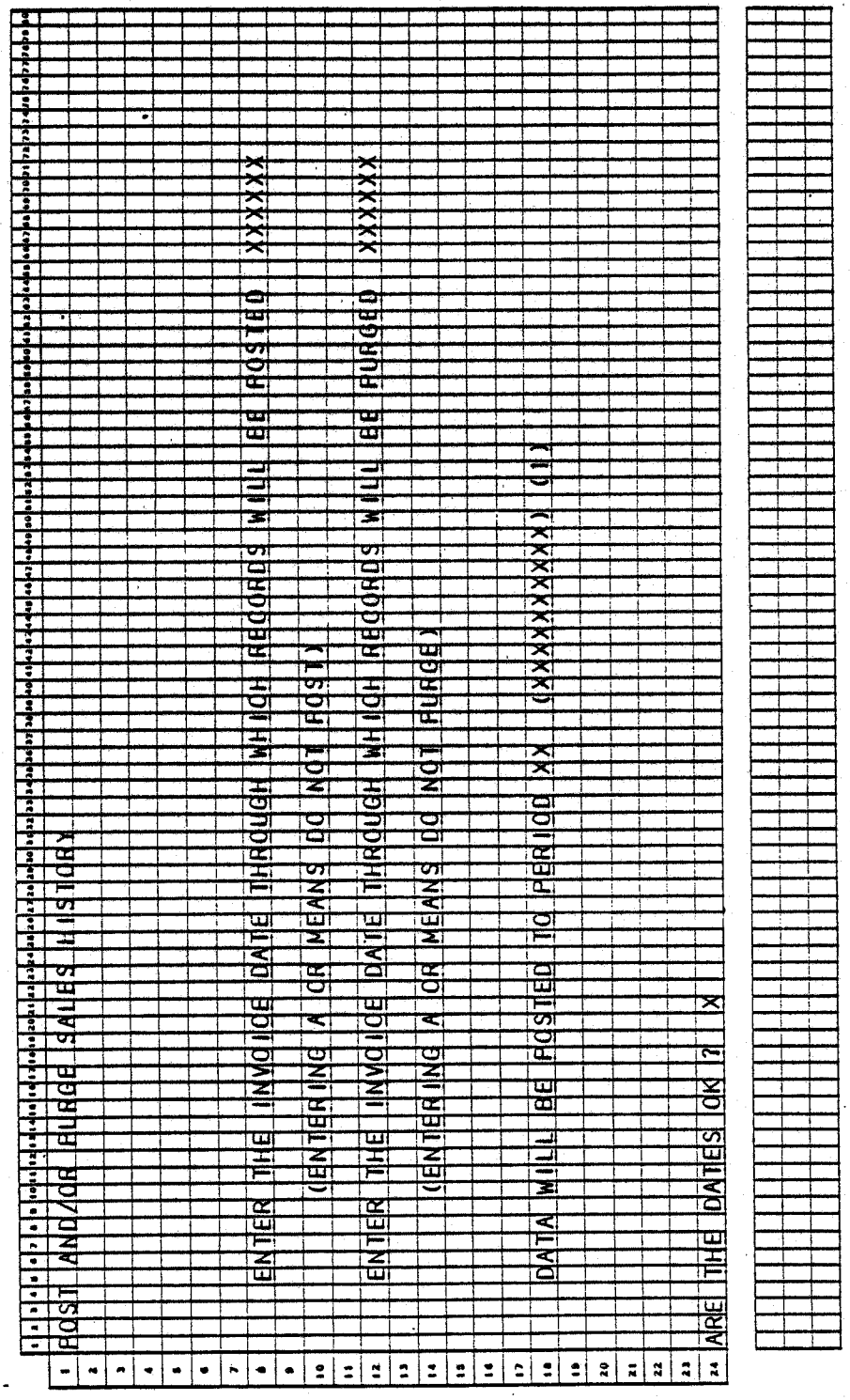
If "END" entered for selection, return to Customer Order Processing menu.

This page intentionally left blank.

CUSTOMER ORDER PROCESSING PACKAGE
POST AND/OR PURGE SALES HISTORY APPLICATION
DIBOL JUN-84

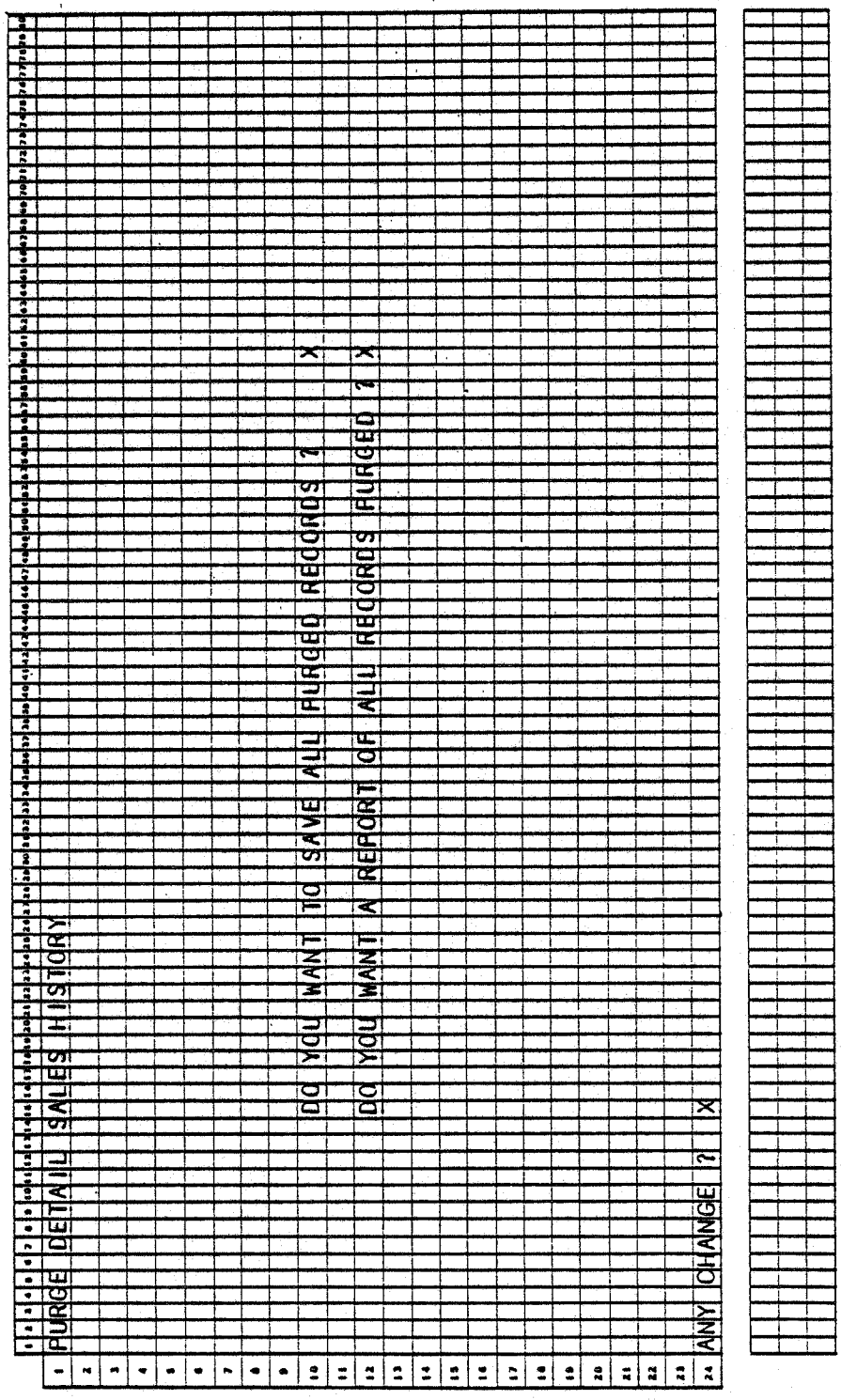
SCREEN FORMATS

Program: HSTSEL

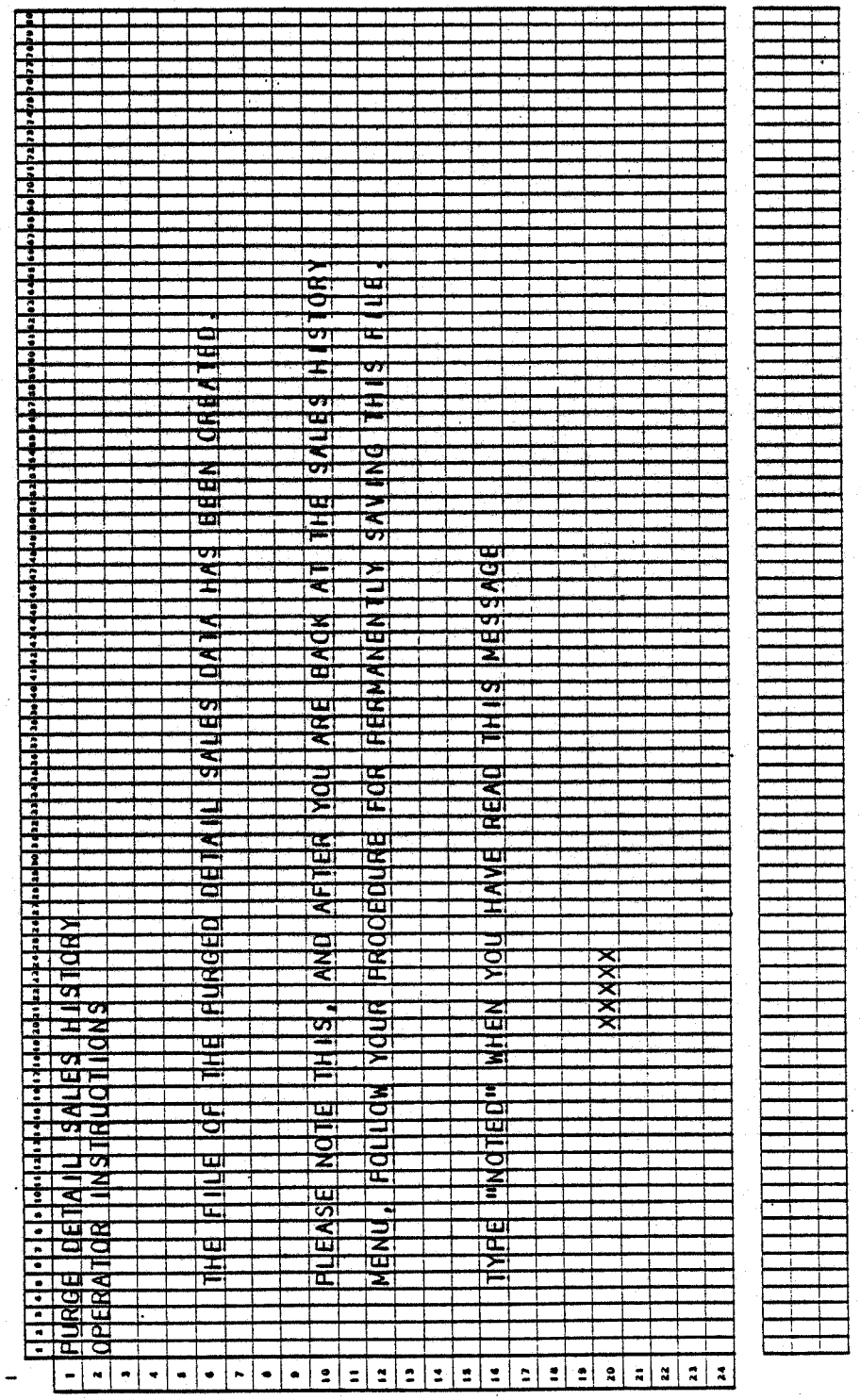


Remarks: (1) THIS MESSAGE WILL BE PRINTED ONLY IF THERE IS A POSTING DATE ENTERED.

Program: PRGSLH



Program: PURMSG



POST AND/OR PURGE SALES HISTORYS APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Post and/or purge Detail Sales History.

Input:	Files Updated:	Output:
SLSHST	SLSHST	Purged Detail Sales
SLSSUM	SLSSUM	History Records
SLSIDX	SLSIDX	
CUSMAS	SSVDSH	
CUSIDX		
ITMMAS		
ITMIDX		

Enter Module From: SSMENU When Done Return To: SSMENU

Programs in Module: HSTSEL, SSUPD, SRTSIX, SSCNT, SSIUPD, PRGSLH, PURMSG, UNPSLH

Program Functions and Notes:

HSTSEL

Display screen as per the Screen Format.

Accept a date through which records will be posted from the Detail Sales History file to the Sales Summary file and its accompanying index.

Accept a date through which records will be purged from the Detail Sales History file.

If the posting date entered is not equal to 0, read through the entire SLSHST file and set the post/purge flag POSTED to 1, if the invoice date is less than the posting date entered.

Send the selected purging date to purge program PRGSLH, if the date is not equal to 0.

If the selected posting date is not equal to 0, transfer to posting program SSUPD.

Otherwise, transfer to purge program PRGSLH.

SSUPD

Read sequentially through the Detail Sales History file.

If POSTED equals 1 (record selected for posting), and if a record with the same customer number or item number does not exist in the Sales Summary file, add a new record to the Sales Summary file.

If POSTED equals 1 and if a record with the same customer number or item number does exist in the Sales Summary file, update the fields in that record.

After the record has been posted to the Sales Summary file, update POSTED to 2 (record posted) and also update the Sales Index.

Send SSCNT as the name of the program to stop to after SRTSIX.

Go to SRTSIX.

SRTSIX

Sort the Sales Index by customer number and item number.

When done, go to SSCNT.

SSCNT

Update sorted record counter in SLSSUM file.

Unprotect SLSSUM file but leave Sales Index protected for updating.

Go to SSIUPD.

SSIUPD

Read sequentially through the Sales Index.

For each record in the Sales Index file, update the non-key fields directly from the CUSMAS file.

For each record in the Sales Index file, update the non-key fields directly from the ITMMAS file.

(This assumes that sales reports are printed out with current and consistent fields for selection; i.e., changing fields in the ITMMAS or CUSMAS files will not distort Sales History Reports.)

Go to PRGSLH.

PRGSLH

Receive a message from HSTSEL containing the purge date.

If no message received (operator entered 0 as purge date in HSTSEL), return to SSMENU.

Ask the operator if he wants to save detailed records and if he wants a purged record report.

Read sequentially through the Sales History file.

If a record has been selected for purging (POSTED equals 2) and the invoice date is less than the operator-selected purge date, do not write the record back into the Sales History file.

Otherwise, put the record into a 40-record long work file and write the work file back into the Sales History file when it becomes full or when there are no unread records left in the Sales History file.

If a report is requested, print each record in the Detail Sales History file that is being purged.

If the operator answers yes to save detailed records, write each record being purged out into a special file, SSVDSH, which saves purged records.

If records are being printed and the printer is busy and the operator will not wait, stop to UNPSLH.

Otherwise, go to SSMENU.

PURMSG

Display operator instructions regarding saving purged Detail Sales History records, as per the Screen Format.

After operator acknowledges instructions, go to SSMENU.

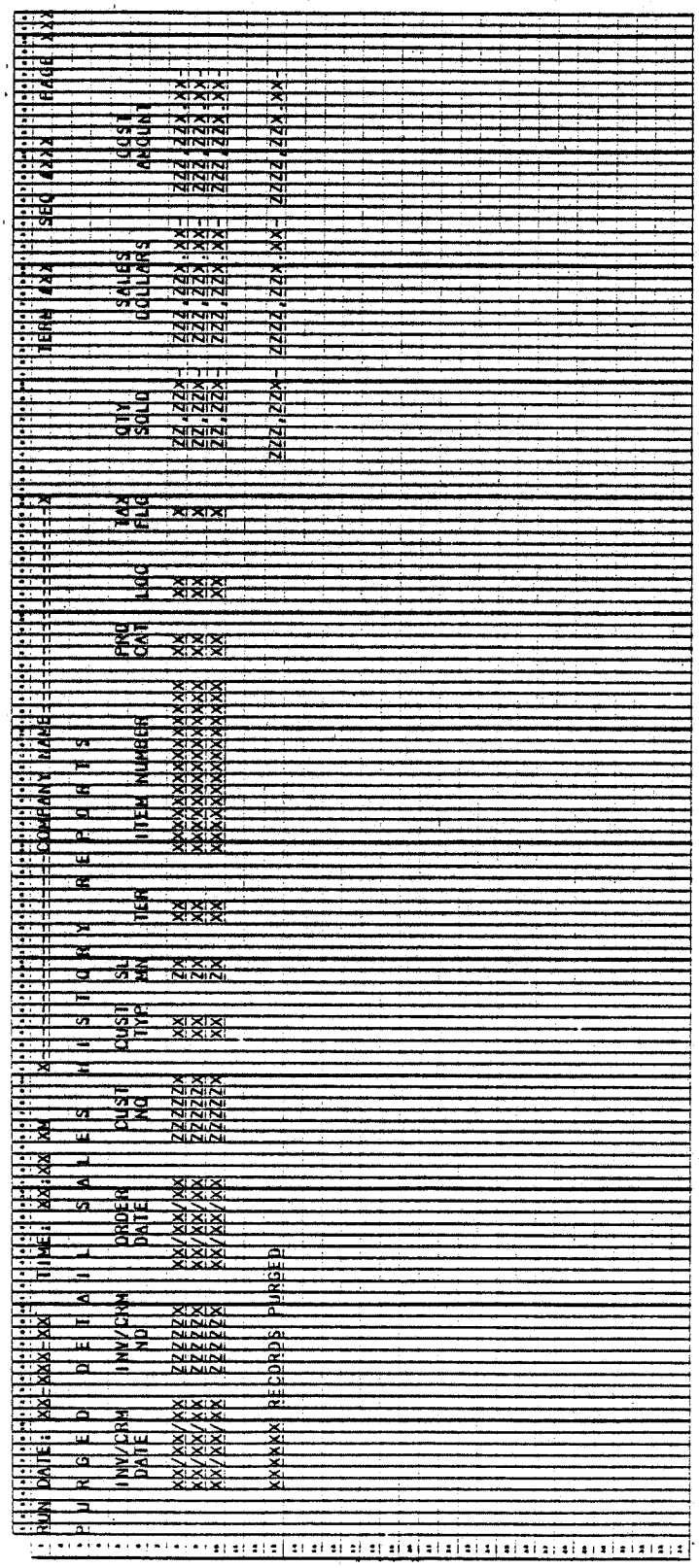
UNPSLH

Unprotect the Detail Sales History file.

Go to SSMENU.

POST AND/OR PURGE SALES HISTORYS APPLICATION
DIBOL JUN-84
REPORT FORMATS

Program: PRGSLH (Purged Detail Sales History Reports)

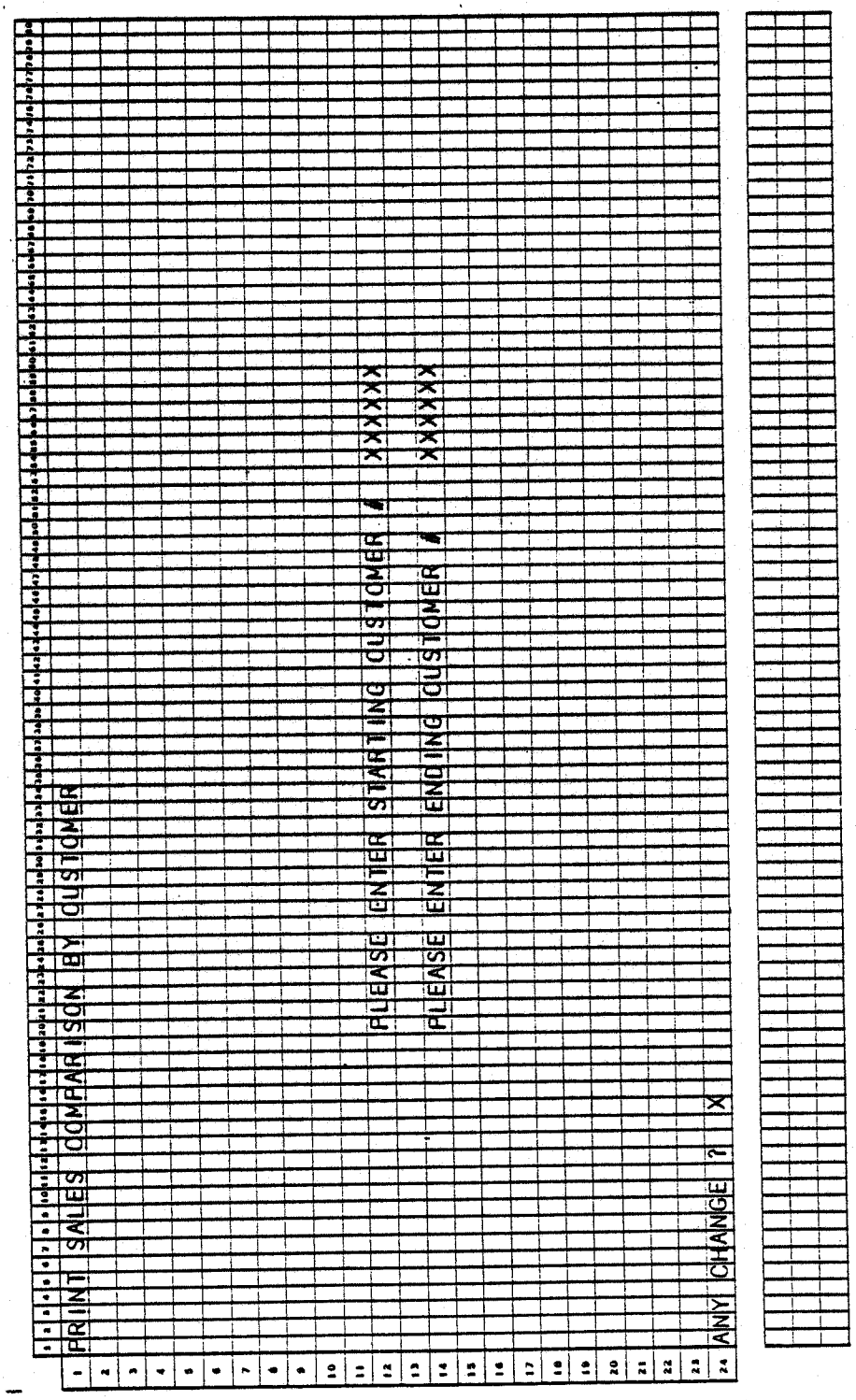


This page intentionally left blank.

CUSTOMER ORDER PROCESSING PACKAGE
PRINT SALES COMPARISON BY CUSTOMER APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: CUSSLS



PRINT SALES COMPARISON BY CUSTOMER APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Print Sales Summary Report by Customer.

Input: SLSIDX
SLSSUM
CUSIDX
CUSMAS

Files Updated:

Output: Sales Comparison by Customer Report

Enter Module From: SSMENU

When Done Return To: SSMENU

Programs in Module: CUSSLS

Program Functions and Notes:

CUSSLS

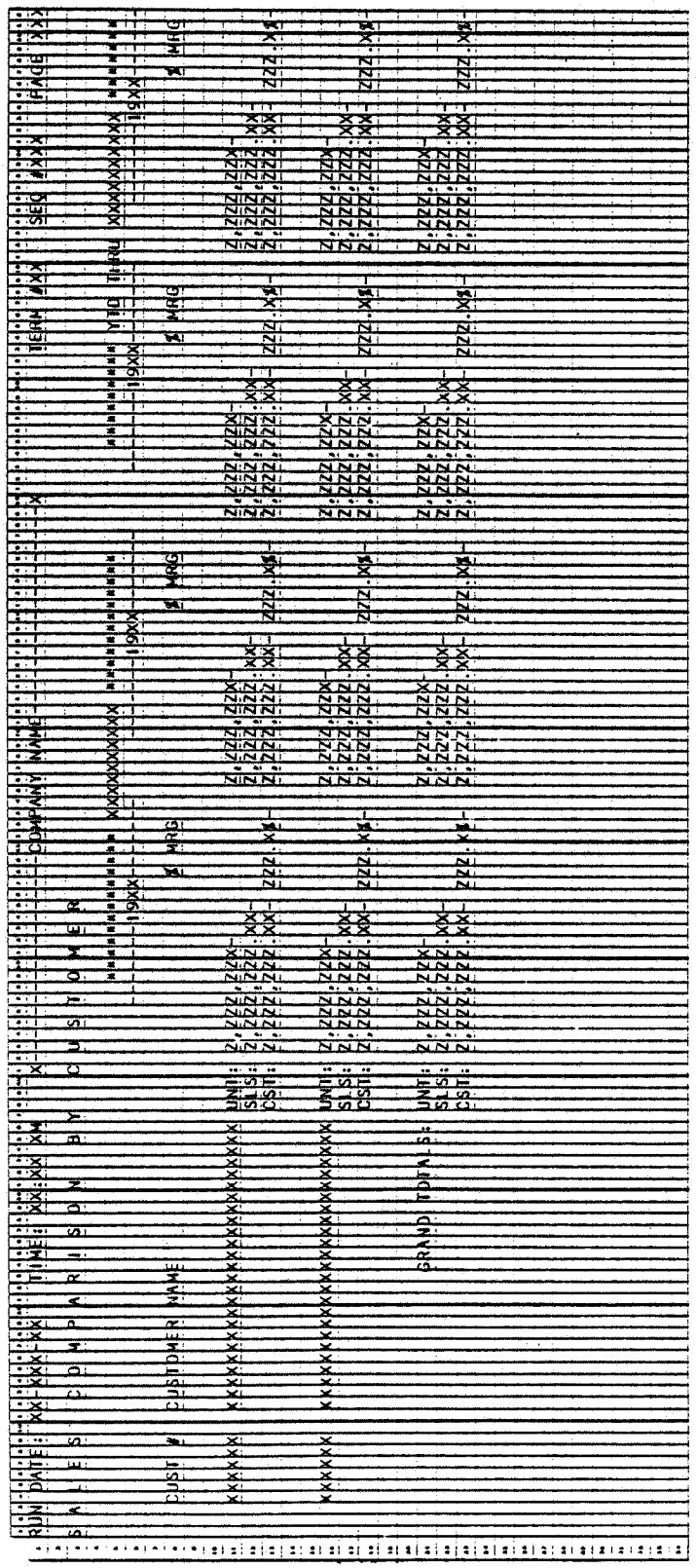
Accept operator input for starting and ending customers.

Print out report on all customers within selected range as per Report Format.

Return to SSMENU.

PRINT SALES COMPARISON BY CUSTOMER APPLICATION
DIBOL JUN-84
REPORT FORMATS

Program: CUSSLS (Sales Comparison By Customer)

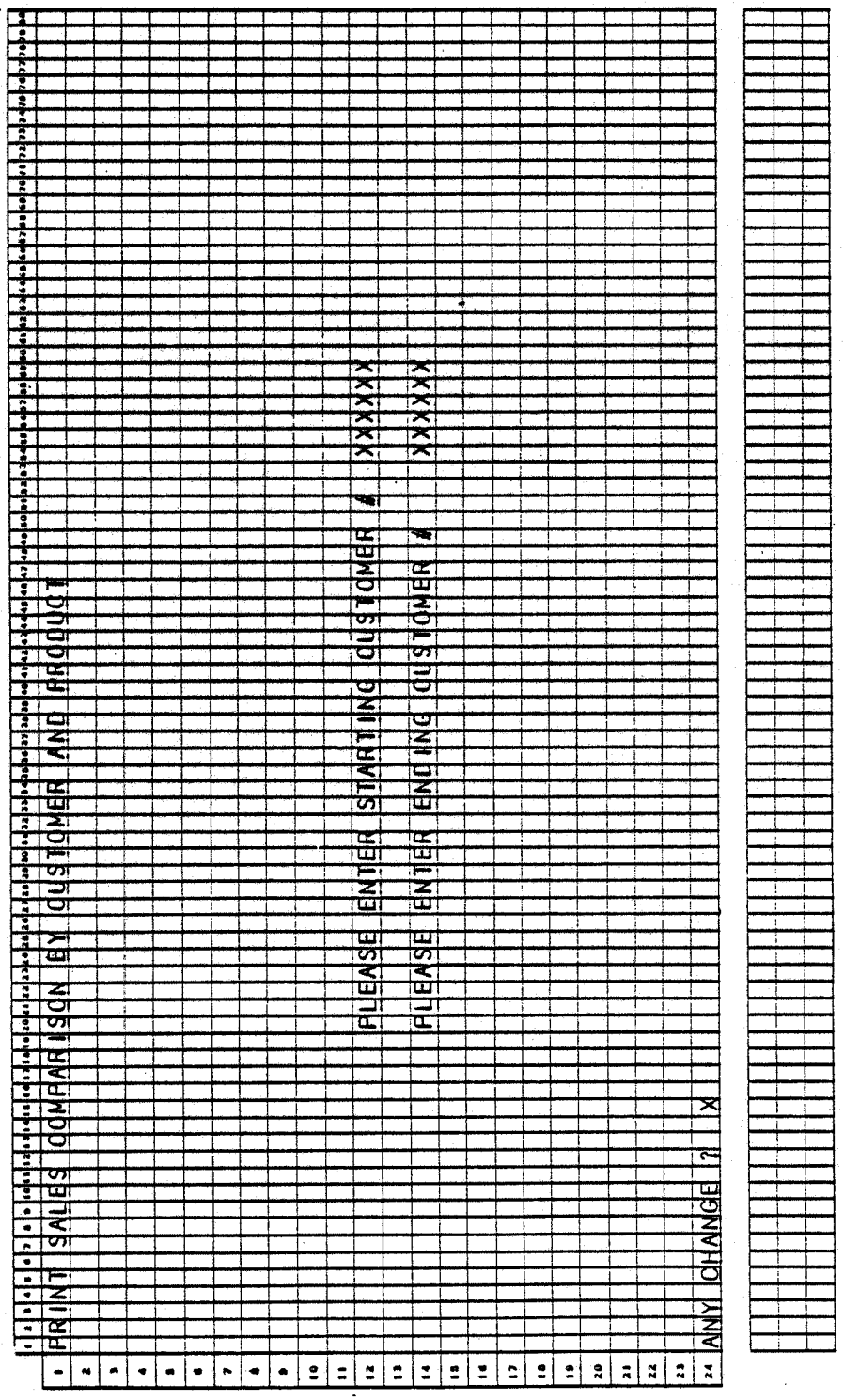


This page intentionally left blank.

CUSTOMER ORDER PROCESSING PACKAGE
PRINT SALES COMPARISON BY CUSTOMER AND PRODUCT APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: CPRSLS



PRINT SALES COMPARISON BY CUSTOMER AND PRODUCT APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Print Sales History Report by Customer/by Product.

Input: SLSIDX
ITMIDX
ITMMAS
CUSIDX
CUSMAS

Files Updated:

Output: Sales Comparison by Customer and Product

Enter Module From: SSMENU

When Done Return To: SSMENU

Programs in Module: CPRSLS

Program Functions and Notes:

CPRSLS

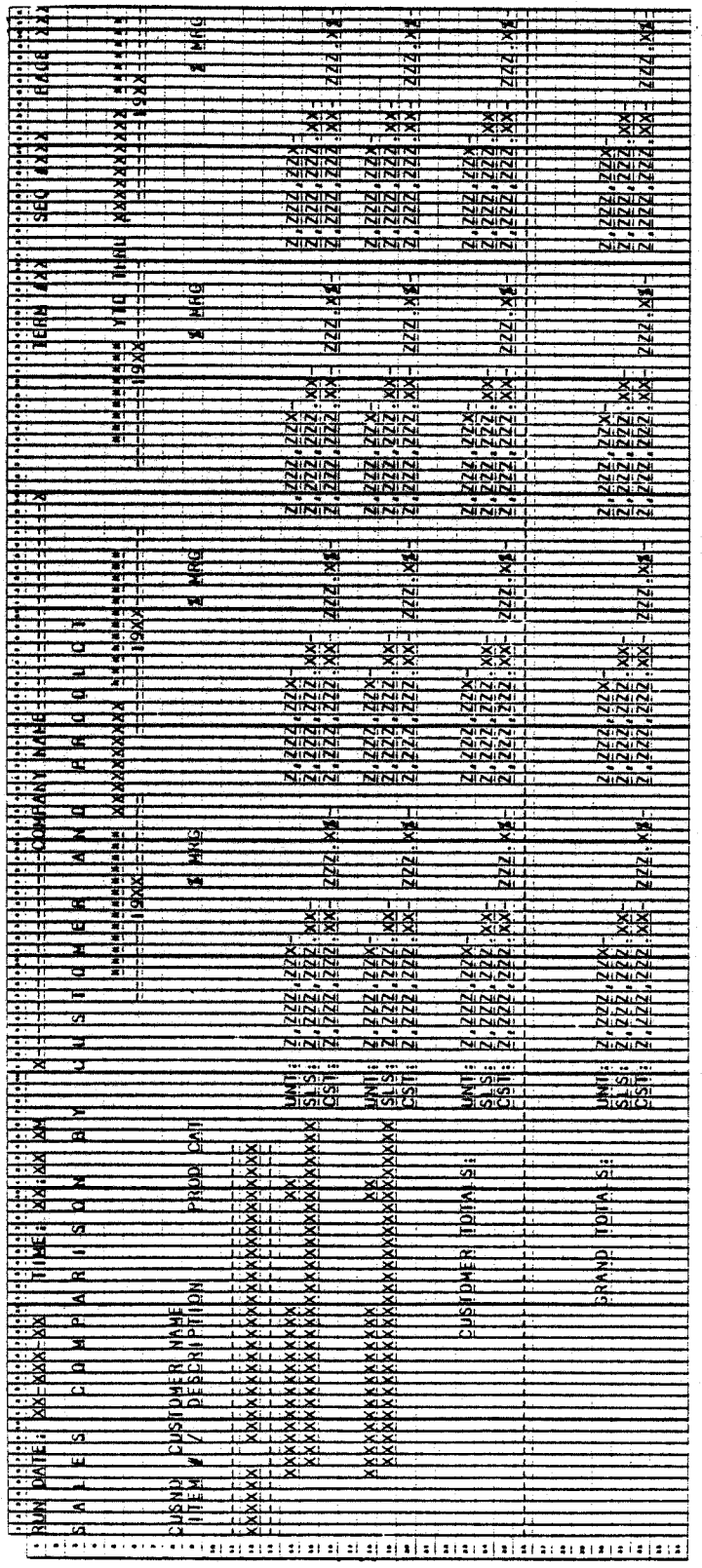
Accept operator input for starting and ending customers.

Print out report on all customers within selected range as per Report Format.

Return to SSMENU.

PRINT SALES COMPARISON BY CUSTOMER AND PRODUCT APPLICATION
DIBOL JUN-84
REPORT FORMATS

Program: CPRSLS (Sales Comparison By Customer And Product)

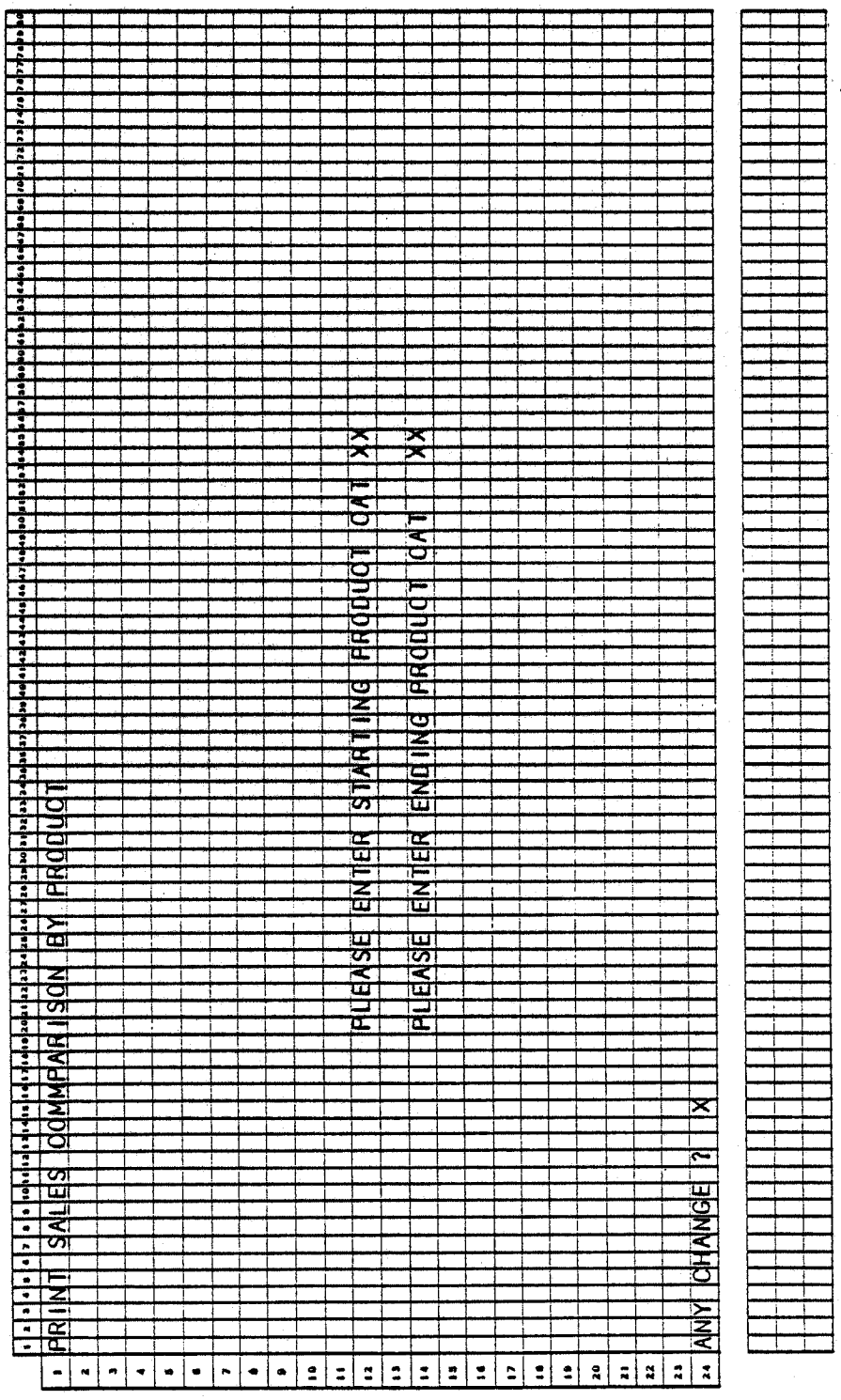


This page intentionally left blank.

CUSTOMER ORDER PROCESSING PACKAGE
PRINT SALES COMPARISON BY PRODUCT APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: PRDSEL



PRINT SALES COMPARISON BY PRODUCT APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Selects SLSIDX records for sales comparison by product. Sorts SLHWRK file by product category by item number. Prints out report.

Input: SLSIDX
SLHWRK
ITMMAS
ITMIDX
SLSSUM

Files Updated: SLHWRK

Output: Sales History by Product Category by Product

Enter Module From: SSMENU

When Done Return To: SSMENU

Programs in Module: PRDSEL, SRTPIX, PRDSLS

Program Functions and Notes:

PRDSEL

Accept operator input for starting and ending product categories.

Write all SLSIDX records, within product category range, out to work file SLHWRK.

Send a message to SORT program SRTPIX, telling it to go to PRDSLS.

SRTPIX

Sort the Sales History Work file by product category and item number.

PRDSLS

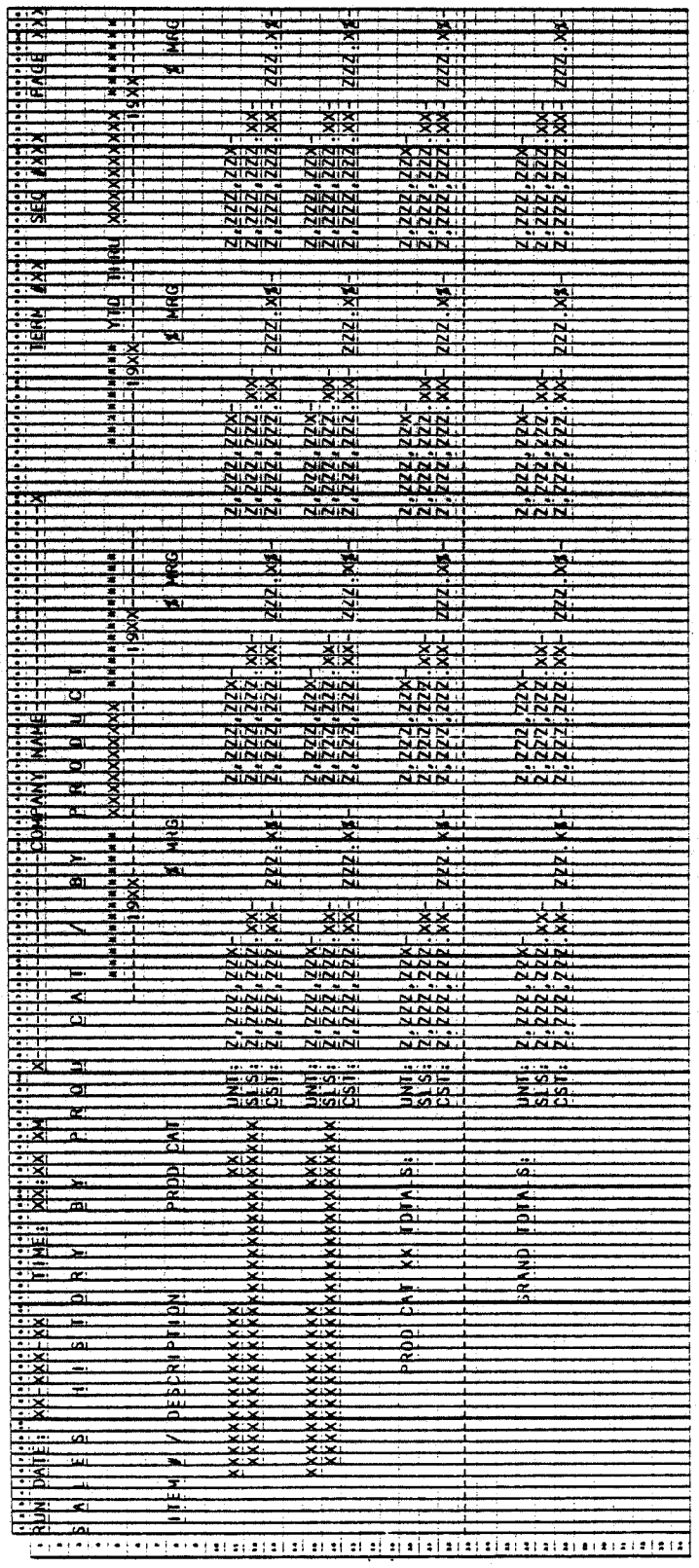
Sequentially read the SLHWRK file.

Retrieve information from the ITMMAS file.

Print out report on sales comparison by product as per the Report Format.

PRINT SALES COMPARISON BY PRODUCT APPLICATION
DIBOL JUN-84
REPORT FORMATS

Program: PRDSLS (Sales History By Prod Cat/By Product)



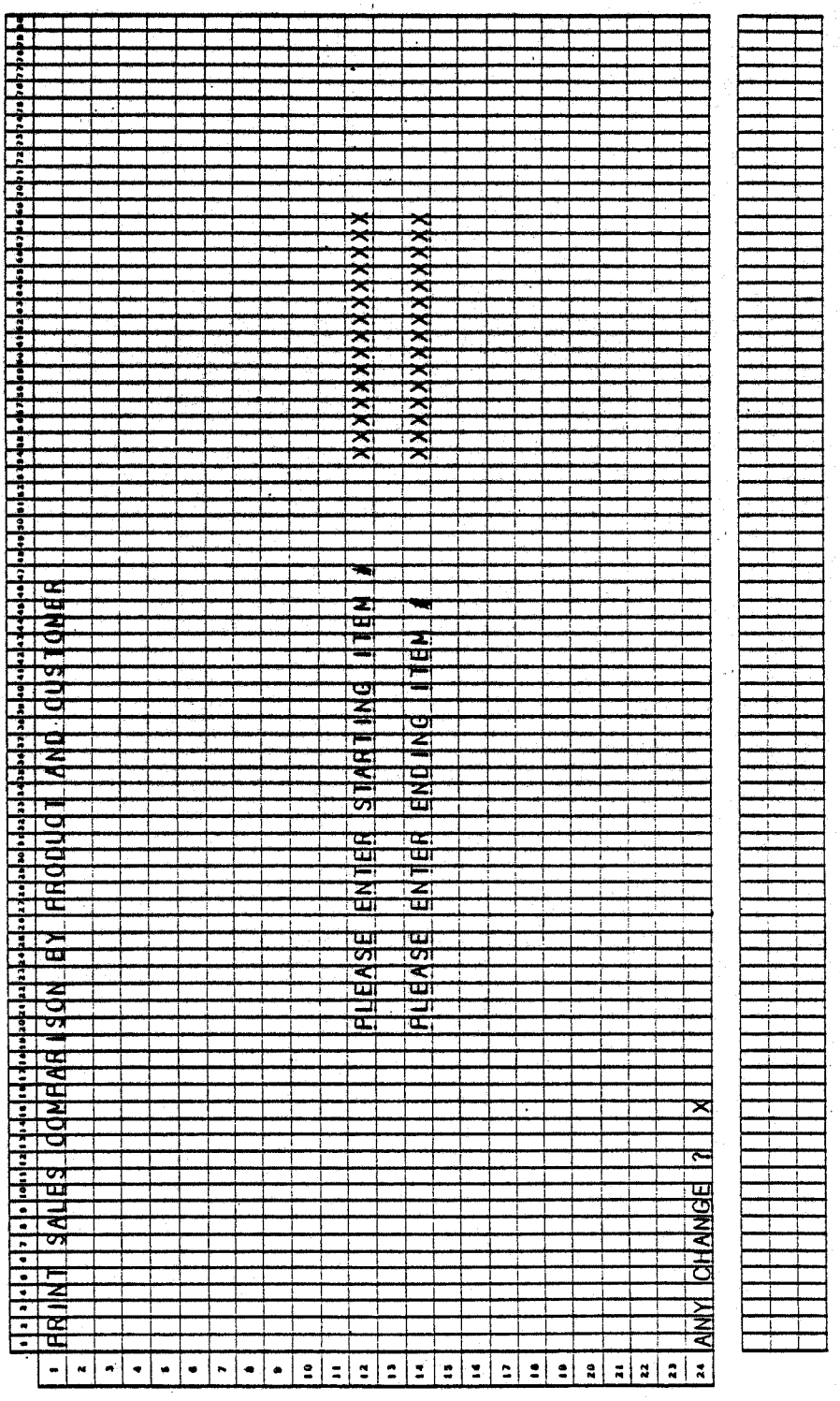
Remarks: $ MRG = PERCENT MARGIN = (SLS - CST) ÷ SLS x 100.

This page intentionally left blank.

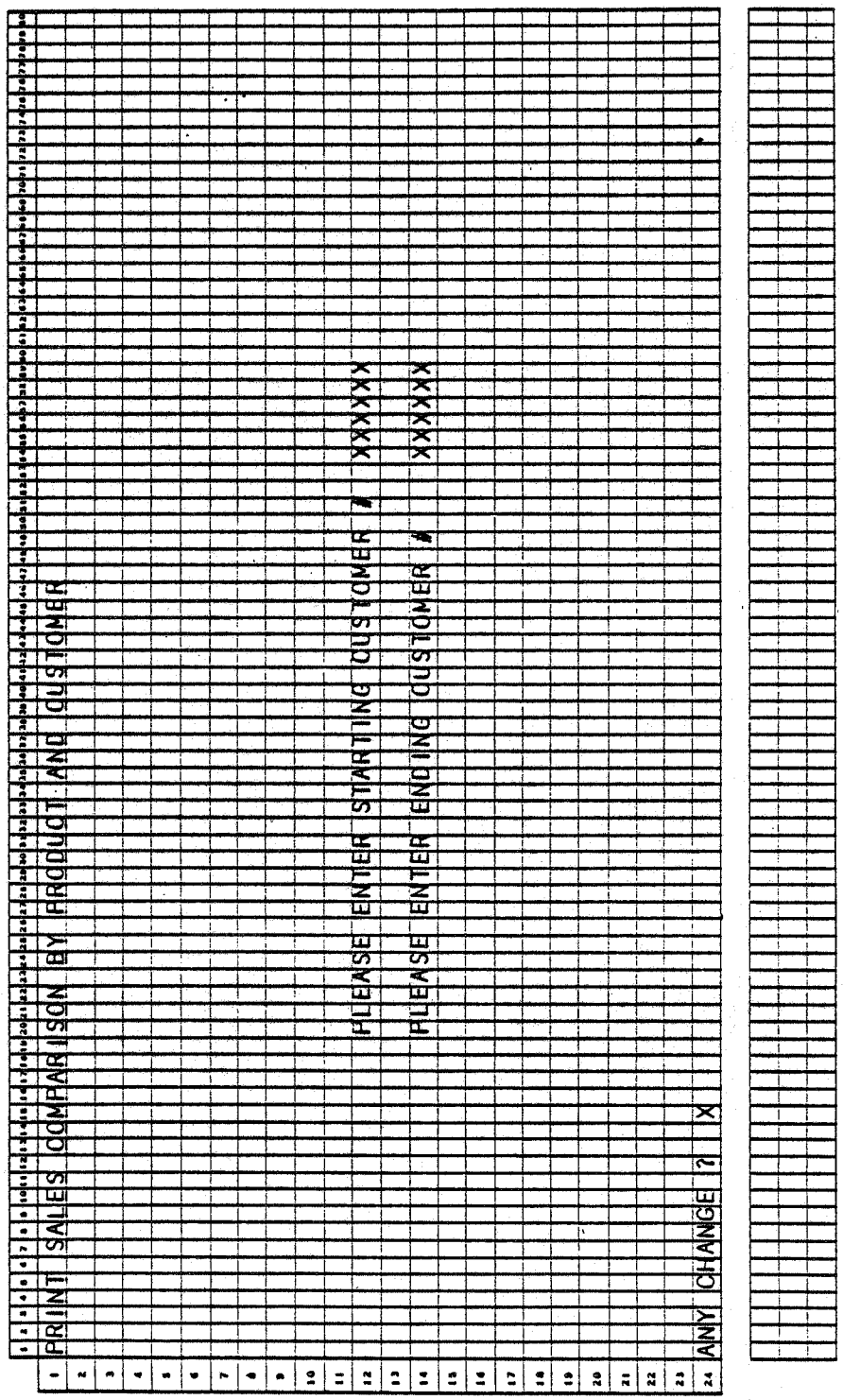
CUSTOMER ORDER PROCESSING PACKAGE
PRINT SALES COMPARISON BY PRODUCT AND CUSTOMER APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: PRCSEL



Program: PRCSEL



PRINT SALES COMPARISON BY PRODUCT AND CUSTOMER APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Selection of SLSIDX records for sales comparison by customer within product. Sort SLHWRK file by item number within customer number and then print the report.

Input:	Files Updated:	Output:
SLSIDX	SLHWRK	Sales Comparison by Product and Customer Report
SLHWRK		
ITMMAS		
ITMIDX		
CUSMAS		
CUSIDX		
SLSSUM		

Enter Module From: SSMENU

When Done Return To: SSMENU

Programs in Module: PRCSEL, SRTPRC, PRCSLS

Program Functions and Notes:

PRCSEL

Accept operator input for starting and ending item number and starting and ending customer number.

Write all SLSIDX records within both ranges out to work file SLHWRK.

Send a message to sort program SRTPRC telling it to go to PRCSLS.

SRTPRC

Sort the SLHWRK file by item number and customer number.

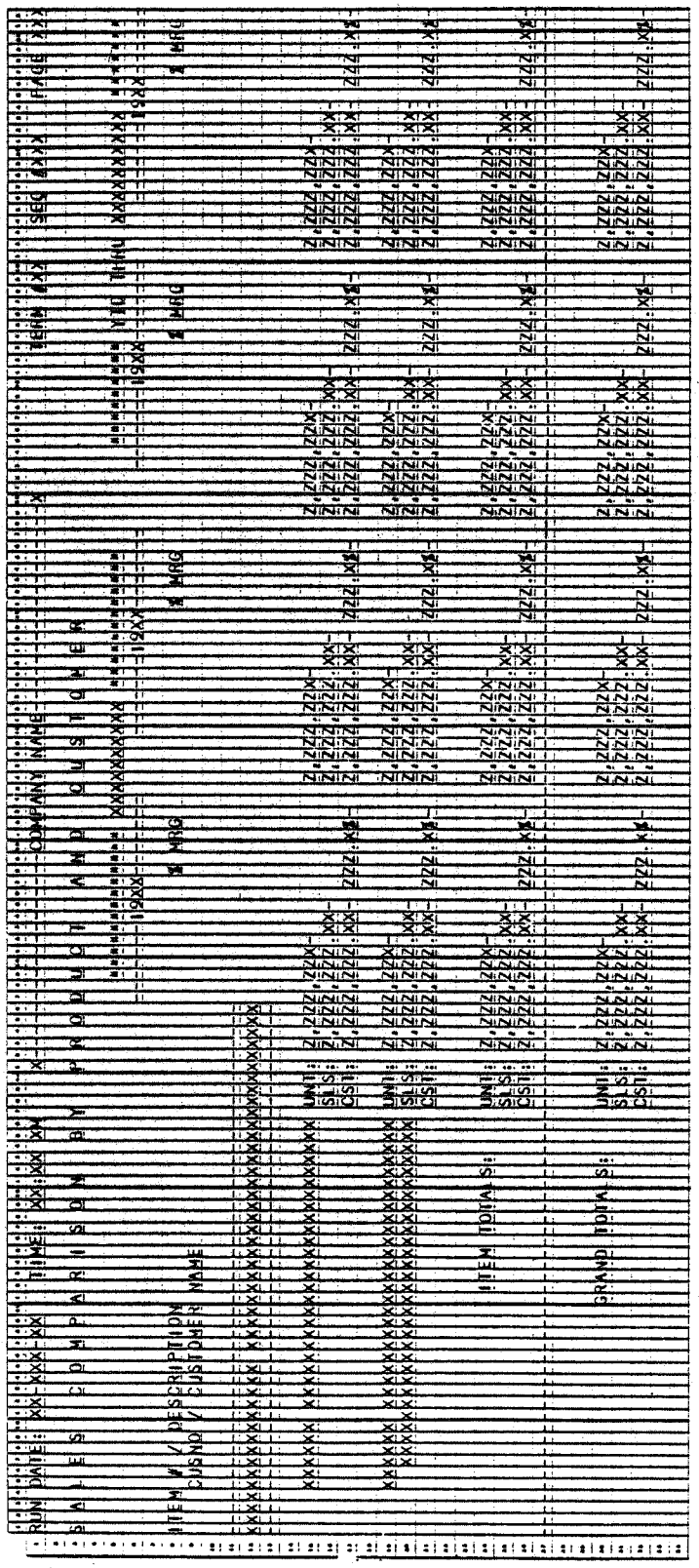
PRCSLS

Sequenctially read the SLHWRK file.

Retrieve information from the ITMMAS and CUSMAS files.

Print out report on sales comparison by customer within product.

Program: PRCSLS (Sales Comparison By Product And Customer)

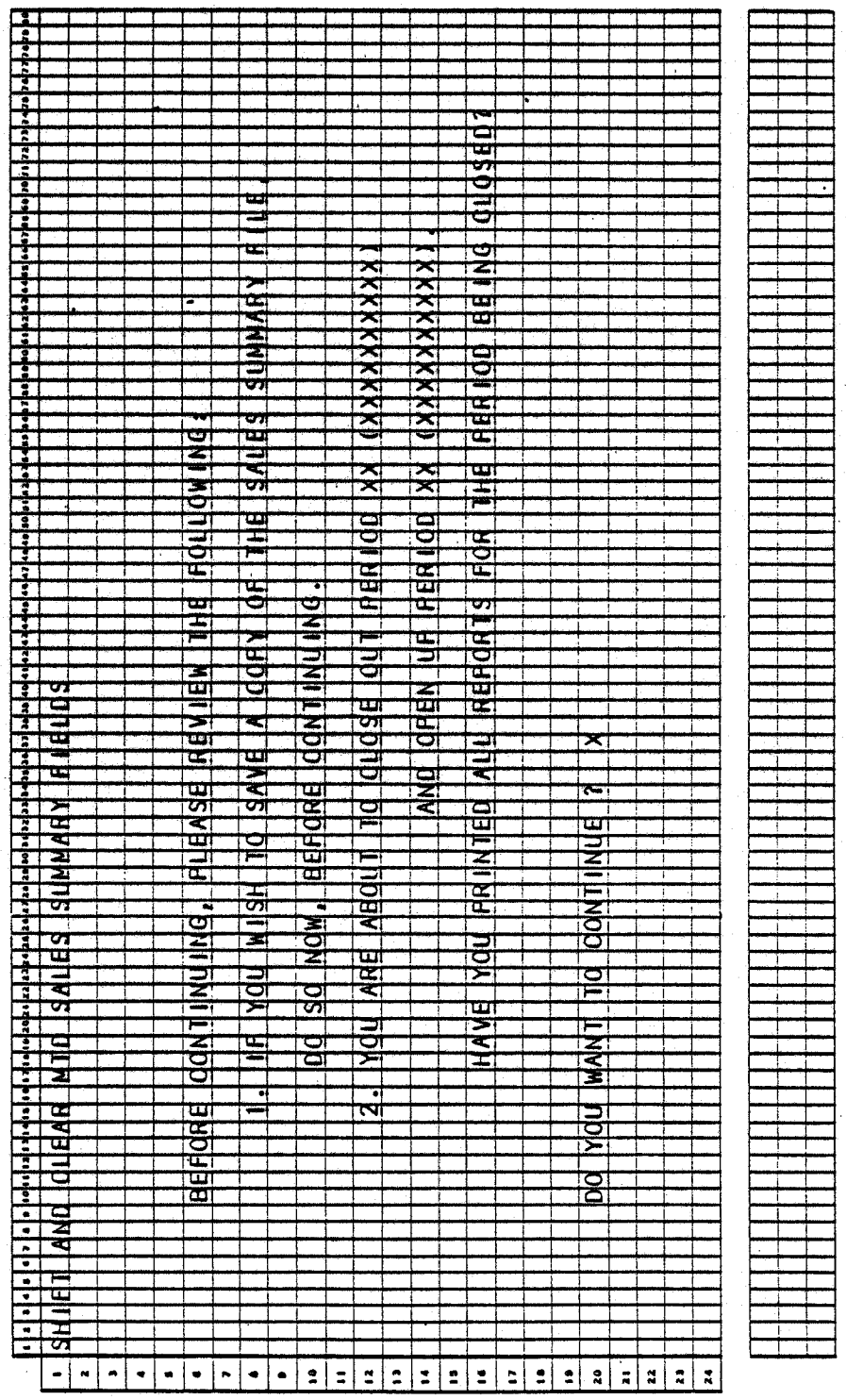


Remarks: % MRG = PERCENT MARGIN = (SLS - CST) ÷ (SLS) x 100.

CUSTOMER ORDER PROCESSING PACKAGE
SHIFT AND CLEAR MTD SALES SUMMARY FIELDS APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: CLRMSS



SHIFT AND CLEAR MTD SALES SUMMARY FIELDS APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Clear SLSSUM file Month-to-Date brackets for beginning of a new month.

Input: SLSSUM Files Updated: SLSSUM Output: None

Enter Module From: SSMENU When Done Return To: SSMENU

Programs in Module: CLRMSS

Program Functions and Notes:

CLRMSS

Display operator instructions as per the Screen Format.

If operator agrees to continue, update Units, Sales, and Costs fields in the following way:

Add last period sales, units, cost to year-to-date sales, units, cost.

This period sales, units, cost initialized.

Reorganizes data in each Sales Summary File record:

(1) Recalculate the Year-to-Date fields (ARRAYS: SSSYTD, SSUYTD, SSCYTD) by adding the last period values to the year-to-date values for 2 periods ago, and store the result in the last period year-to-date.

(2) Put this period (C/M) values (SSSALES, SSUNIT, SSCUST) into the last period cost (SSLSTS, SSLSTU, SSLSTC).

(3) Initialize the new "this period" (C/M) values to zero.

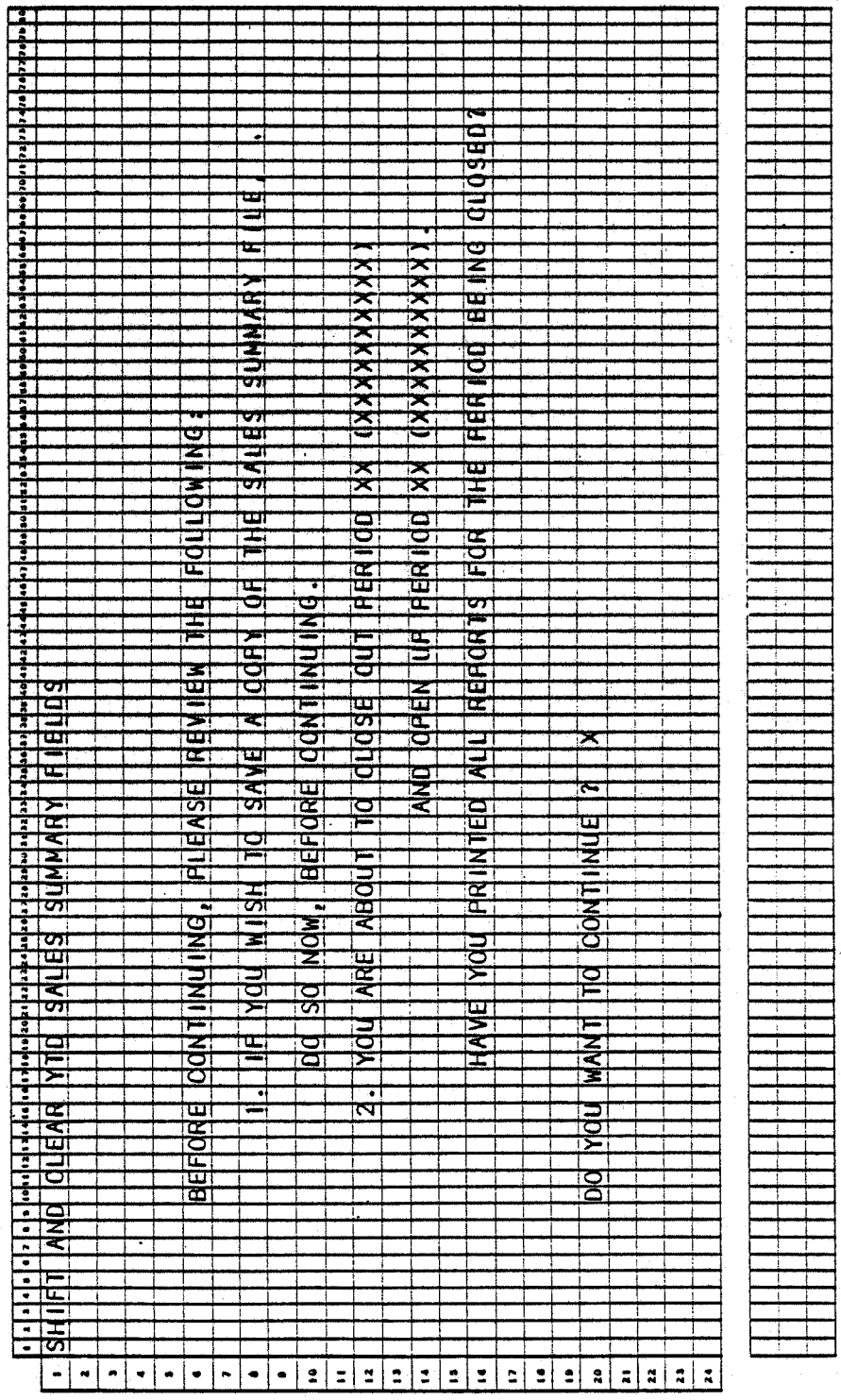
When all records are complete, increment the period number.

Return to SSMENU.

CUSTOMER ORDER PROCESSING PACKAGE
SHIFT AND CLEAR YTD SALES SUMMARY FIELDS APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: CLRYSS



SHIFT AND CLEAR YTD SALES SUMMARY FIELDS APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Reset SLSSUM file for beginning of a new year and mark records with no history for deletion. Remove SLSSUM records marked for deletion and update SLSIDX.

Input:	Files Updated:	Output:
SLSIDX SLSSUM	SLSIDX SLSSUM	None

Enter Module From: SSMENU

When Done Return To: SSMENU

Programs in Module: CLRYSS, ORGSLS

Program Functions and Notes:

CLRYSS

Read sequentially through the SLSSUM file.

Update Year-to-Date Sales fields for periods 11 and 12 and initialize Year-to-Date Sales fields, and reset period and year.

Mark SLSSUM records with no year-to-date sales for deletion.

ORGSLS

Remove records marked for deletion in SLSSUM file by writing non-marked records over them.

Keep track of deleted records and their relative position (pointer in SLSIDX file) in the SLSSUM file.

Remove records from SLSIDX file, which pointed to those records deleted in SLSSUM file, by writing records which point to non-deleted SLSSUM records over them.

Resetting for a New Year

A Year-End Shift replaces the Month-End Shift for the last month of the year. The logic:

1. Performs all the functions of the Month-End Shift.

2. Sets up fields for the new year

   . Year = Year + 1
   . Period # = 1
   . Sets up for new year-to-date numbers

3. Checks to see if the record ought to be deleted (i.e., if all cost, units and sales fields = 0, then should be deleted).

Purge Any Deleted Records

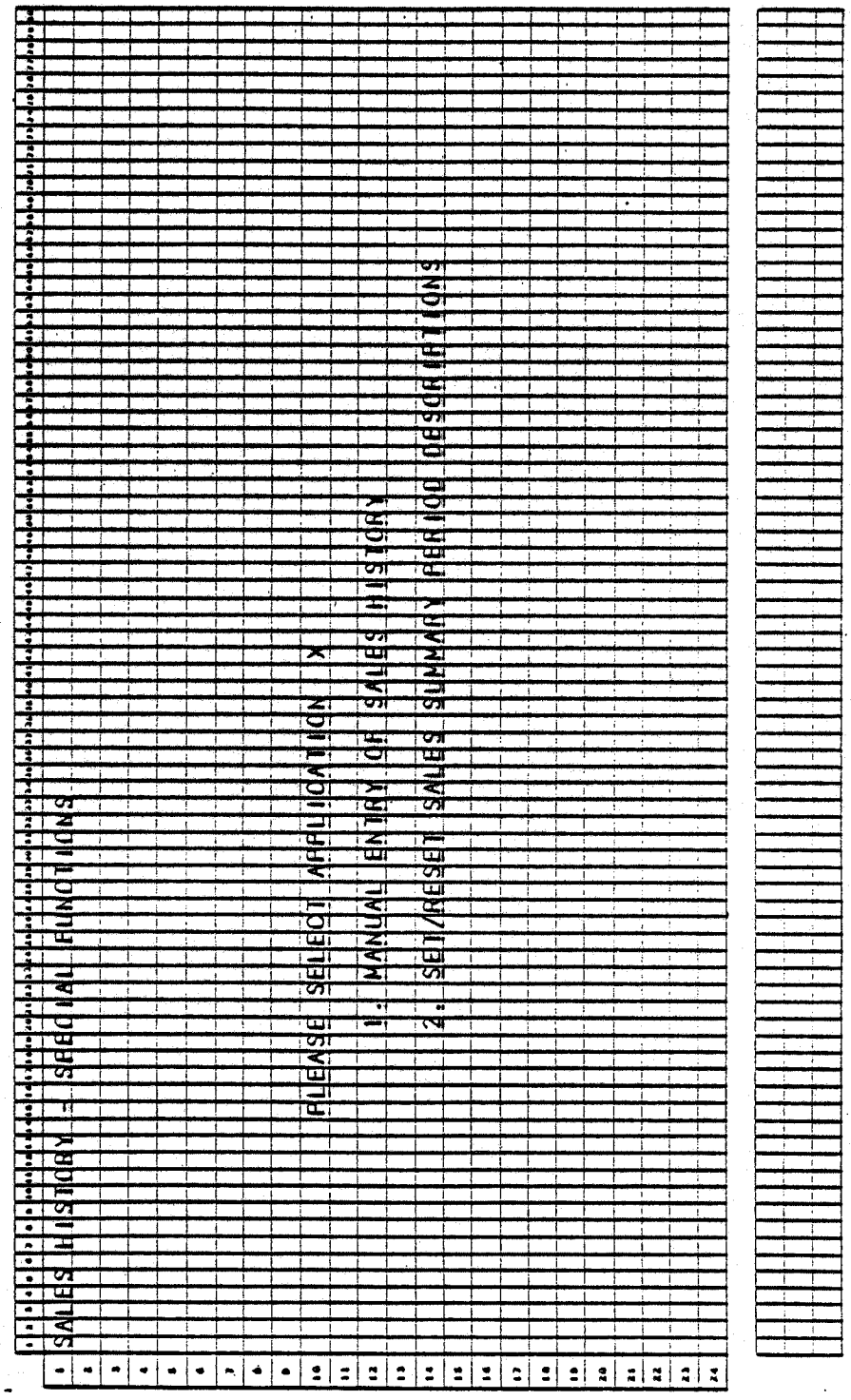
Compress all deleted records out of the Summary file. Then, correct the relative record pointer in the index by the number of records that had been deleted (in the SLSSUM file previous to this record).

This page intentionally left blank.

CUSTOMER ORDER PROCESSING PACKAGE
SALES HISTORY SPECIAL FUNCTIONS APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: SSSFMN



SALES HISTORY SPECIAL FUNCTIONS APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Special Sales History Functions submenu for program selection.

Input: KBD Files Updated: Output: None

Enter Module From: SSMENU When Done Return To: SSMENU

Programs in Module: SSSFMN

Program Functions and Notes:

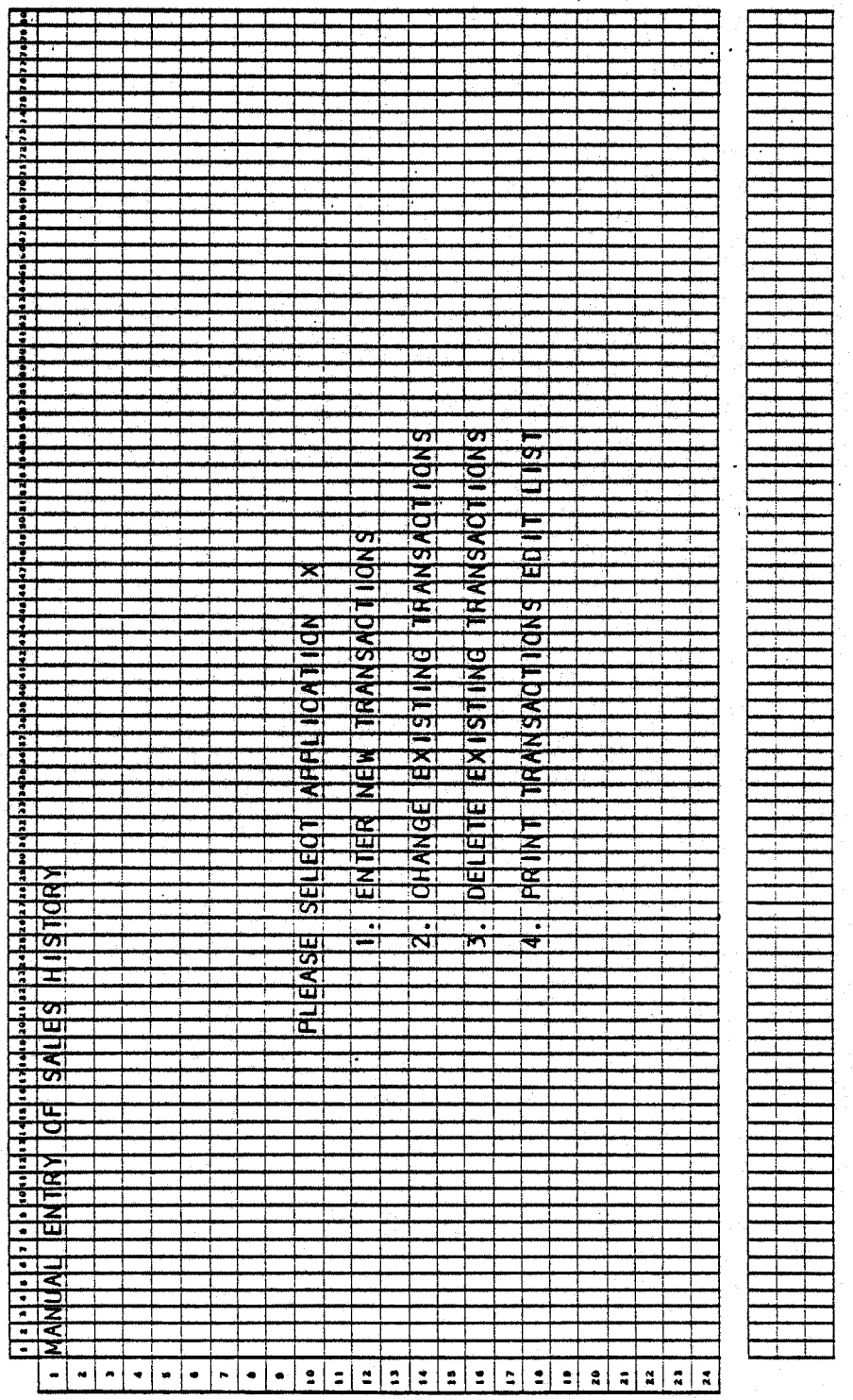
SSSFMN

Display Sales History Special Functions screen. Accept selection and chain to appropriate program. If the END key is pressed, return to SSMENU.

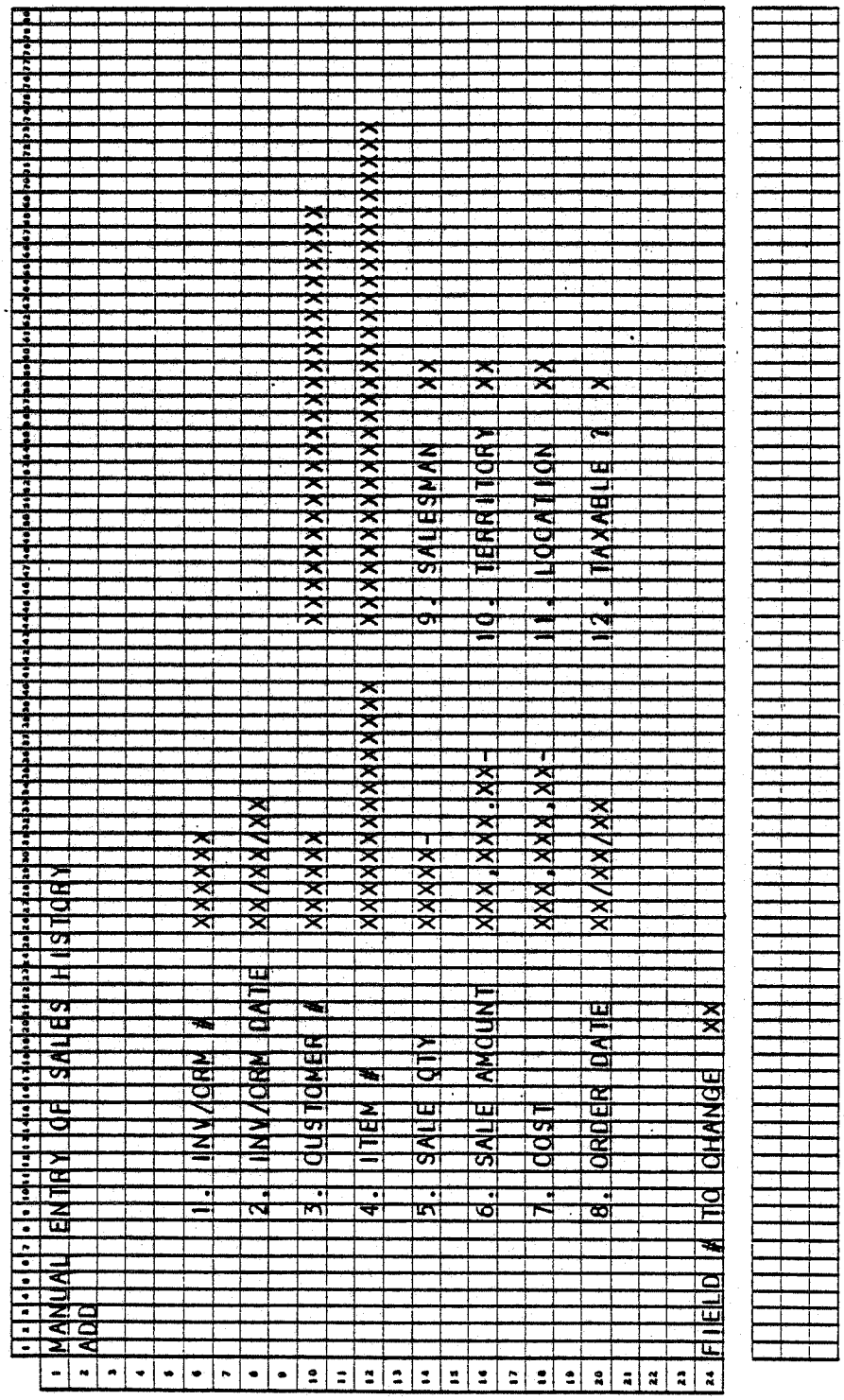
CUSTOMER ORDER PROCESSING PACKAGE
MANUAL ENTRY OF SALES HISTORY FILE APPLICATION
DIBOL JUN-84

SCREEN FORMATS

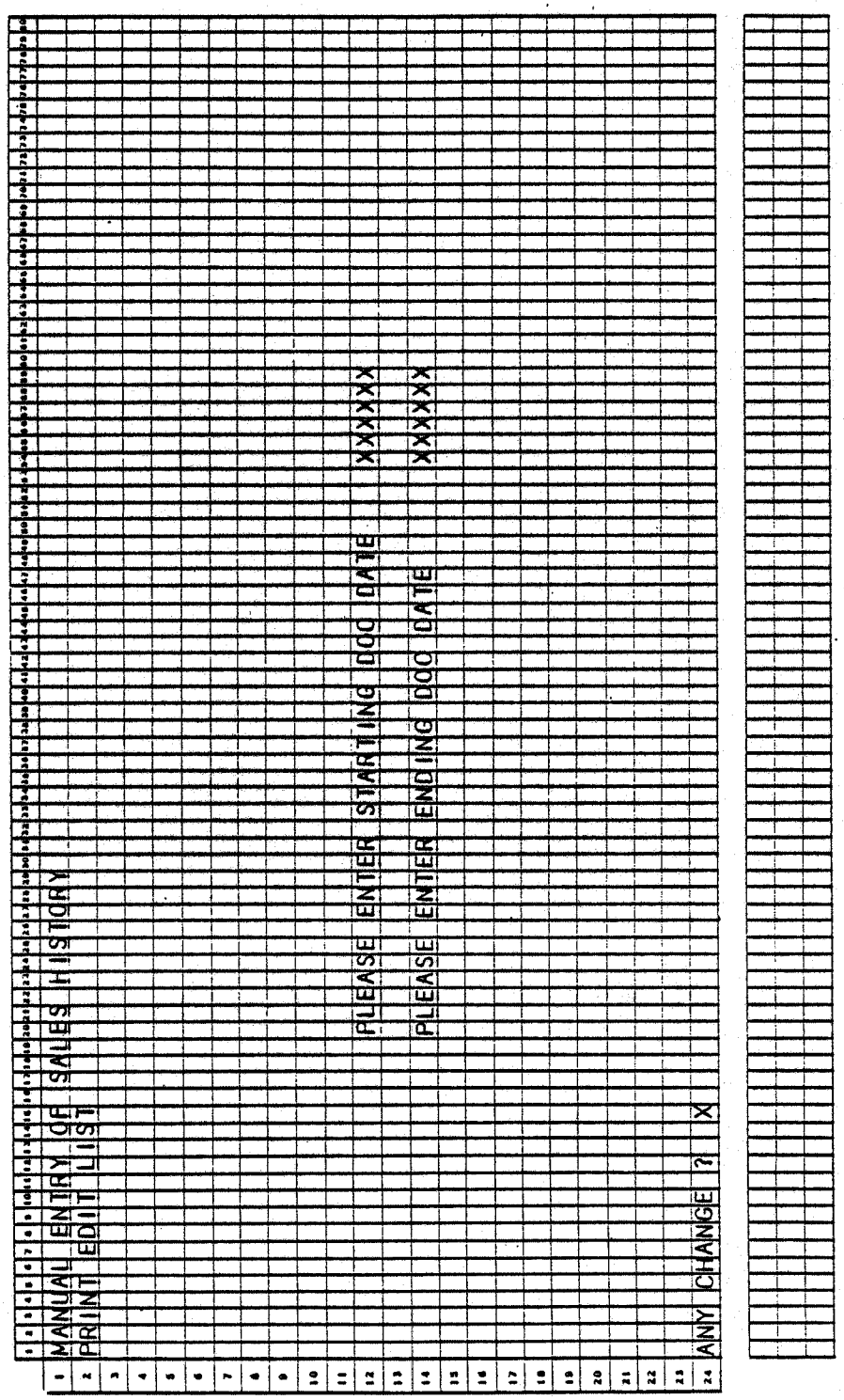
Program: BLDSLH



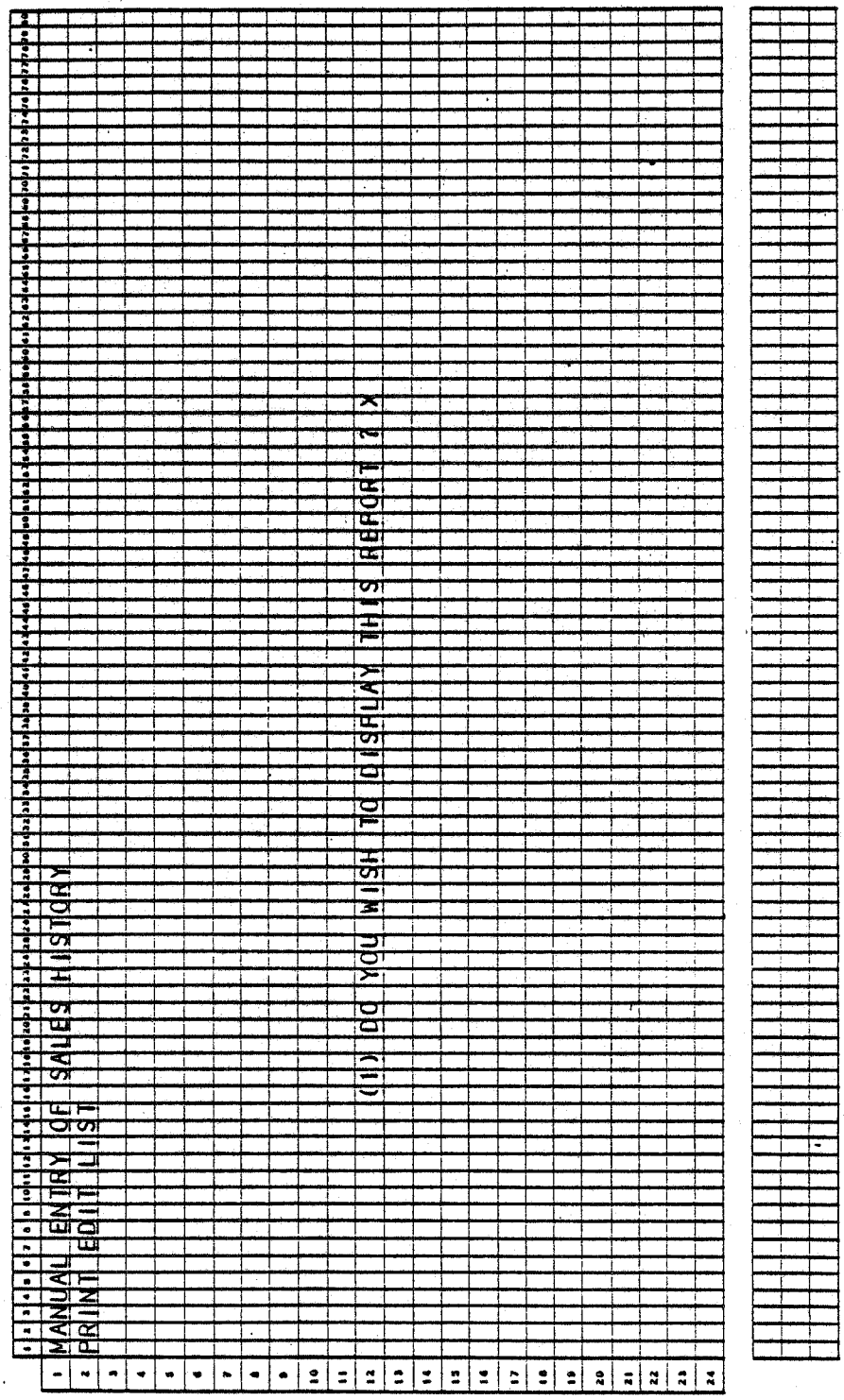
Program: BLDSLH



Program: SLHEDT



Program: SLHEDT



Remarks: (1) FOR VT100 TERMINALS ONLY.

MANUAL ENTRY OF SALES HISTORY FILE APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Manual entry of Detail Sales History. Sort SLHST file by customer number, item number, and date (YY/MM/DD).

Input:	Files Updated:	Ouput:
CUSMAS CUSIDX ITMMAS ITMIDX SLSHST	SLSHST	Detail Sales History Edit list

Enter Module From: SSSFMN When Done Return To: SSSFMN

Programs in Module: BLDSLH, SRTBLD, SLHEDT

Program Functions and Notes:

BLDSLH

Display screen as per Screen Format.

Enter ADD, CHANGE, DELETE, or PRINT mode at operator request.

If in ADD mode:

Display screen as per the Screen Format.

Enter standard SLSHST information; i.e., invoice number, invoice date, customer number, product number, sale quantity, sale amount, cost.

Retrieve information from CUSMAS and ITMMAS fields for use in SLSHST file.

Write record out to SLSHST file.

If in CHANGE mode:

Enter invoice number and customer number.

Search SLSHST file for first record with same invoice number and customer number.

Display transaction as per the Screen Format.

Display next transaction if operator responds no to "RIGHT TRX ?".

Allow changes to transaction if operator responds yes to "RIGHT TRX ?" and "ANY CHANGE ?".

If in DELETE mode:

Follow same procedure as in CHANGE mode, except after yes response to "RIGHT TRX ?" zero out invoice date, quantity, sale amount, and cost in SLSSUM file.

If in PRINT mode:

Send SLHEDT as program name to go to after sort to SRTBLD.

Otherwise:

Send UNPSLH as program name to go to after sort to SRTBLD.

SRTBLD

Sort SLSHST file by customer number, item number, year, month and day.

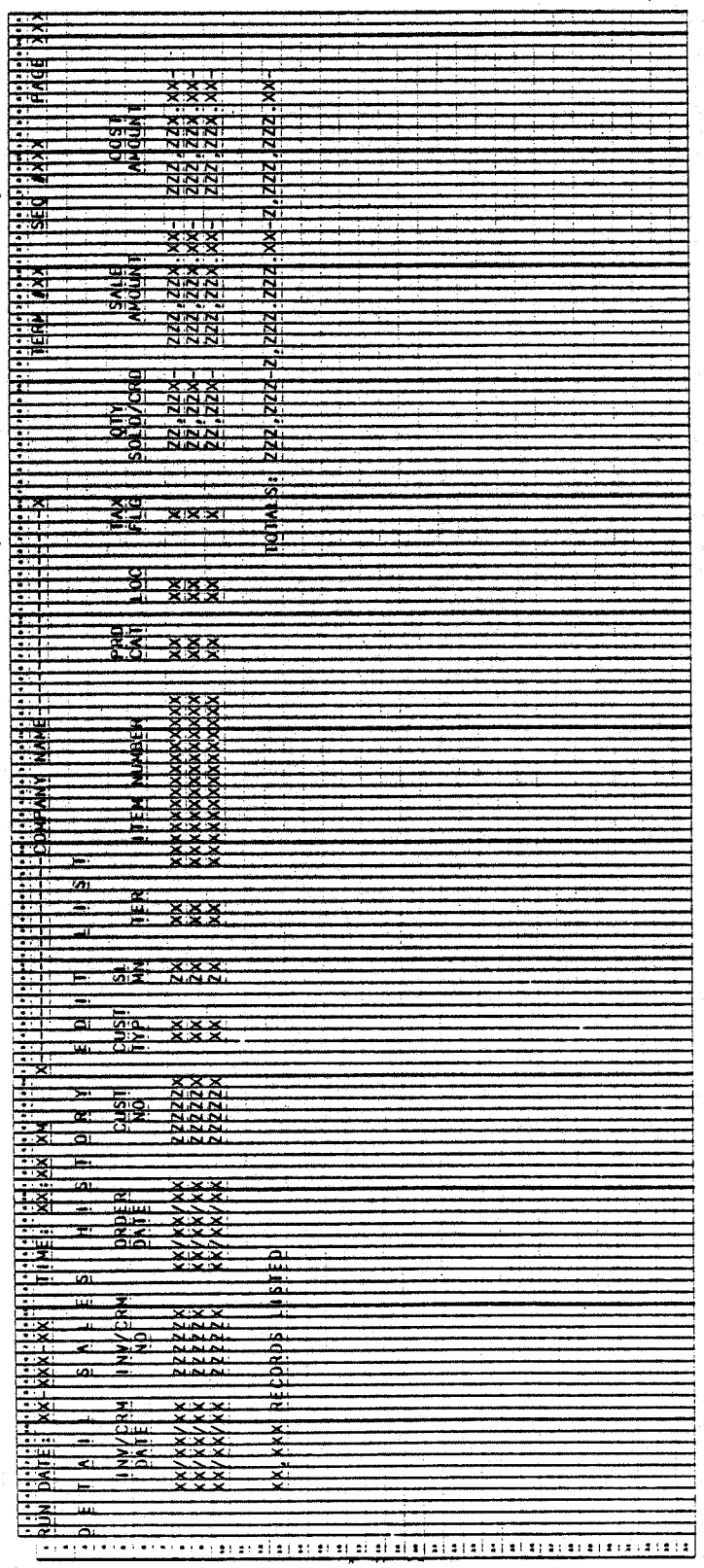
Go to UNPSLH or SLHEDT depending on message sent by BLDSLH.

SLHEDT

Request starting and ending invoice date from operator.

Print out all Detail Sales History records falling within that range.

Program: SLHEDT (Detail Sales History Edit List)

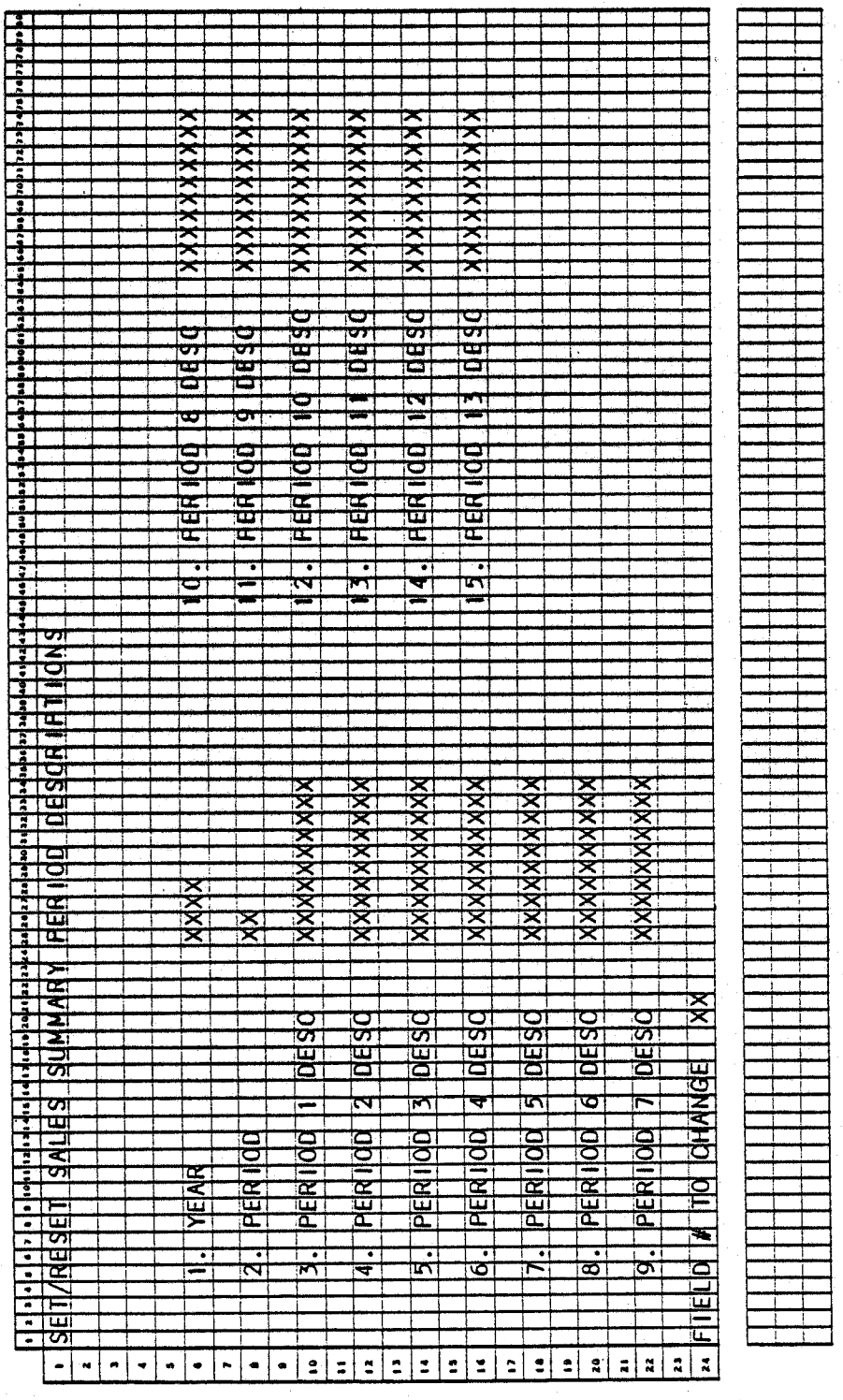


This page intentionally left blank.

CUSTOMER ORDER PROCESSING PACKAGE
SET/RESET SALES SUMMARY PERIOD DESCRIPTIONS APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: SSSET



SET/RESET SALES SUMMARY PERIOD DESCRIPTIONS APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Reset SLSSUM period descriptions.

Input: SLSSUM Files Updated: SLSSUM Output: None

Enter Module From: SSSFMN When Done Return To: SSSFMN

Programs in Module: SSSET

Program Functions and Notes:

SSSET

Read SLSSUM control record.

Display screen as per the Screen Format.

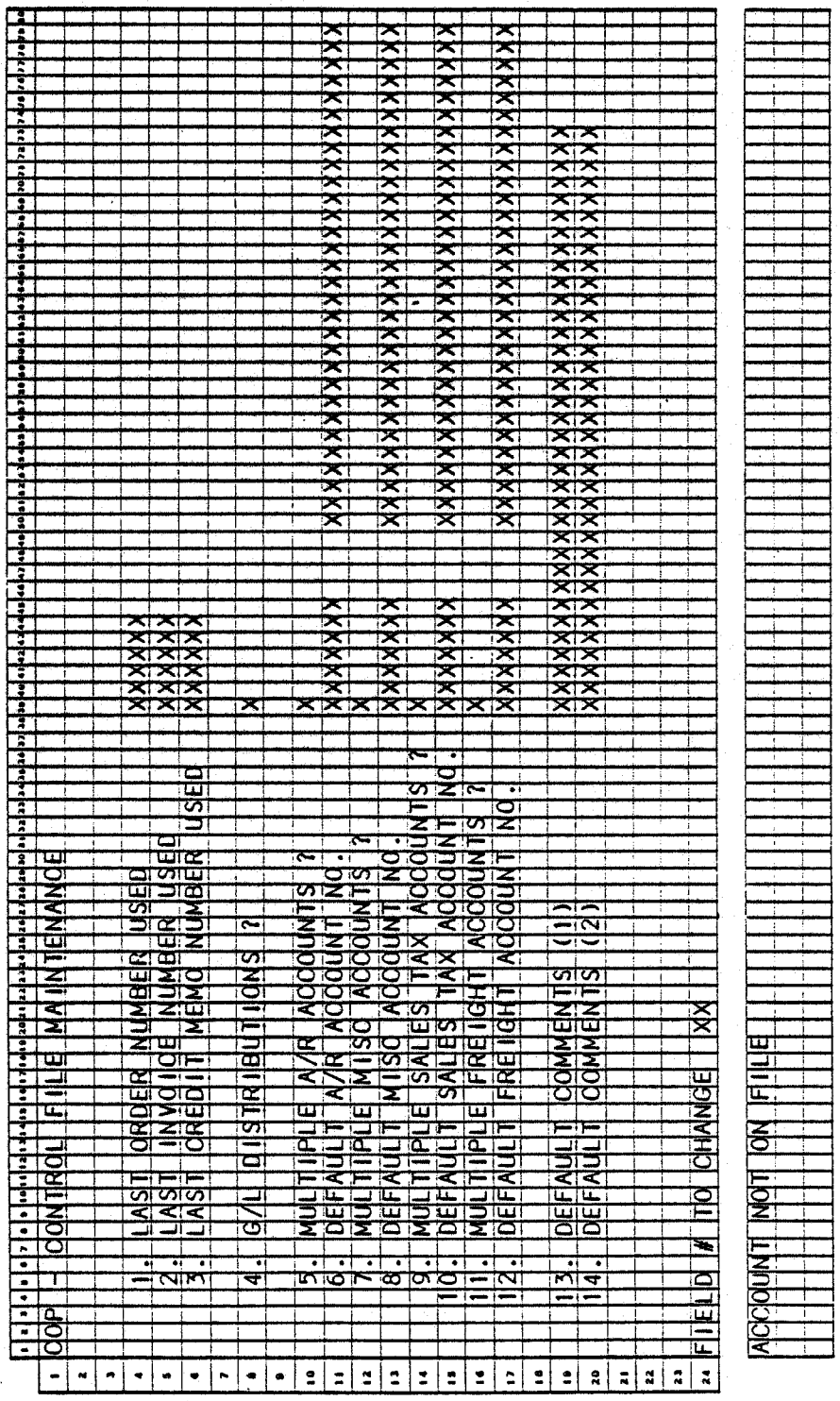
Allow user to add or change, one at a time, each description for the 13 periods used in the SLSSUM file.

User can also add or change the current period or year.

CUSTOMER ORDER PROCESSING PACKAGE
COP CONTROL FILE MAINTENANCE APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: CCTMNT



PROGRAM SPECIFICATIONS

Function: Maintains the COP Control file which contains the default values used within the COP package.

Input: KBD
COPCTL
ARACCT

Files Updated: COPCTL

Output: None

Enter Module From: SPCFUN

When Done Return To: SPCFUN

Programs in Module: CCTMNT

Program Functions and Notes:

CCTMNT

Open COPCTL (U), ARACCT (1) files. Read the Control record of ARACCT. Read the COPCTL record. If the COPCTL record is brackets, assume add mode, otherwise, assume change mode.

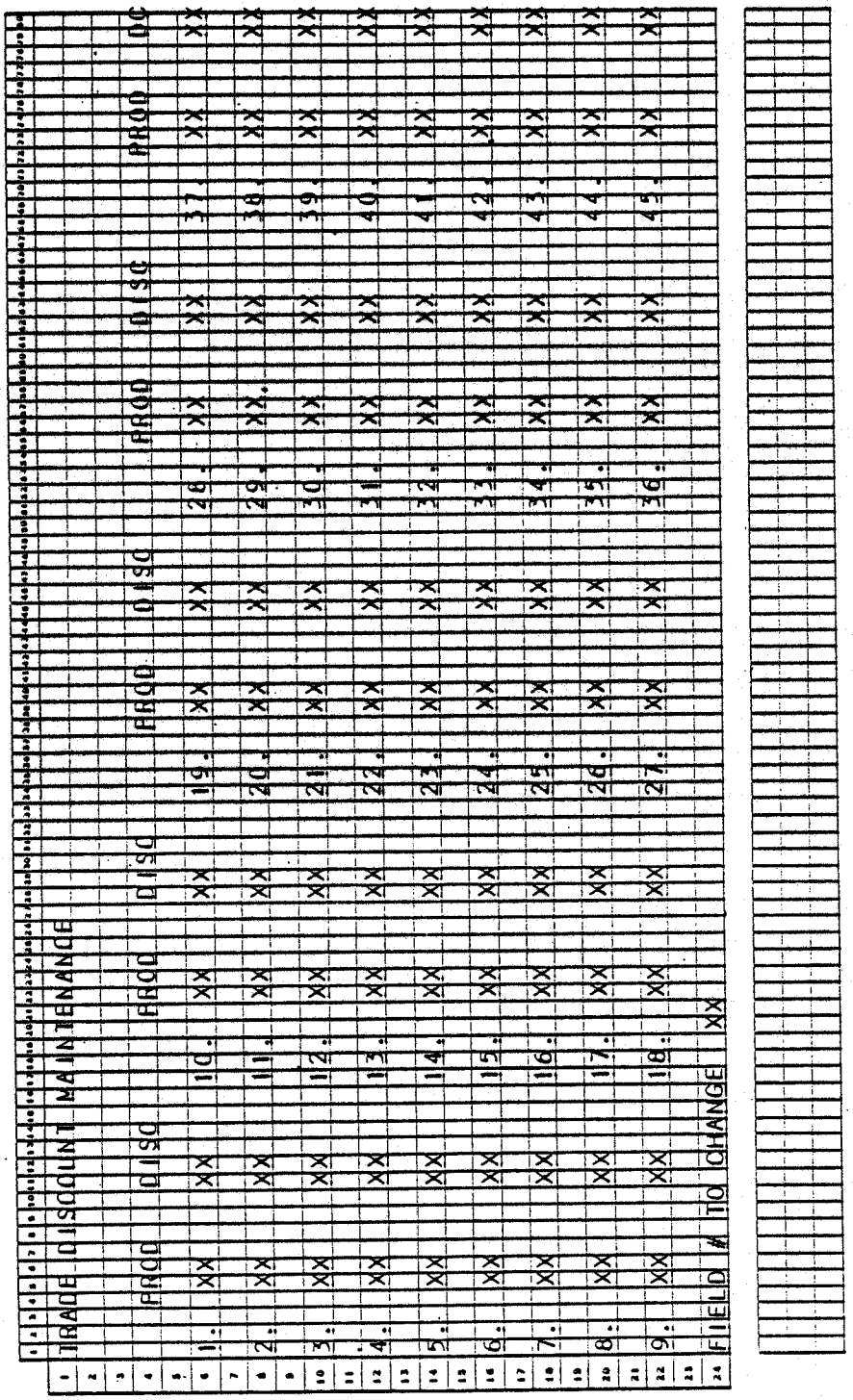
Display screen. If add mode, accept entry of each field. Check ARACCT file for valid accounts. If not found, reenter.

Accept changes. When done, write out the COPCTL record, close files and exit.

CUSTOMER ORDER PROCESSING PACKAGE
TRADE DISCOUNT MAINTENANCE APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: TRADED



TRADE DISCOUNT MAINTENANCE APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Trade (line item) discount array maintenance.

Input: ITMMAS Files Updated: ITMMAS Output: None

Enter Module From: SPCFUN When Done Return To: SPCFUN

Programs in Module: TRADED

Program Functions and Notes:

TRADED

Display all existing product categories and corresponding discounts from the ITMMAS control record.

Accept additions, changes, or deletions to the array.

If the operator selects a number for which there is no category, and that number is not the next available slot, direct him to that slot, regardless of what number he has input.

Allow an existing category to be deleted by blank for product category. After a code is deleted, "SQUEEZE" the array so that all active categories occupy the first positions. For instance, if there are three product categories in the array, they must occupy DPRDCD(1), DPRDCD(2), and DPRDCD(3).

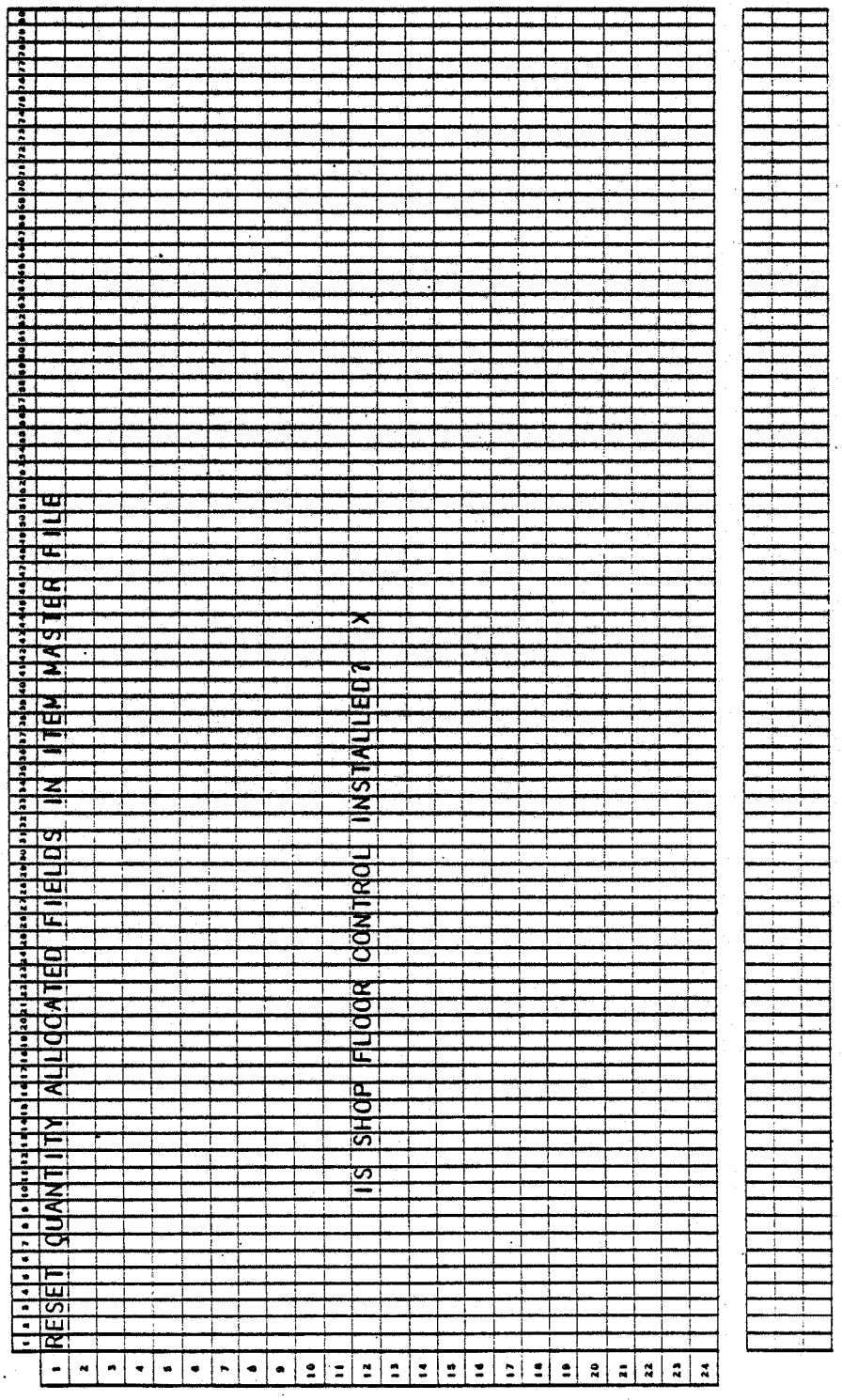
If a new product category is added, or an existing category is changed, check to be sure this category does not already exist in the array. Do not allow duplicate categories to be entered.

Allow the program to begin running only if exclusive use of the ITMIDX and ITMMAS file is obtainable.

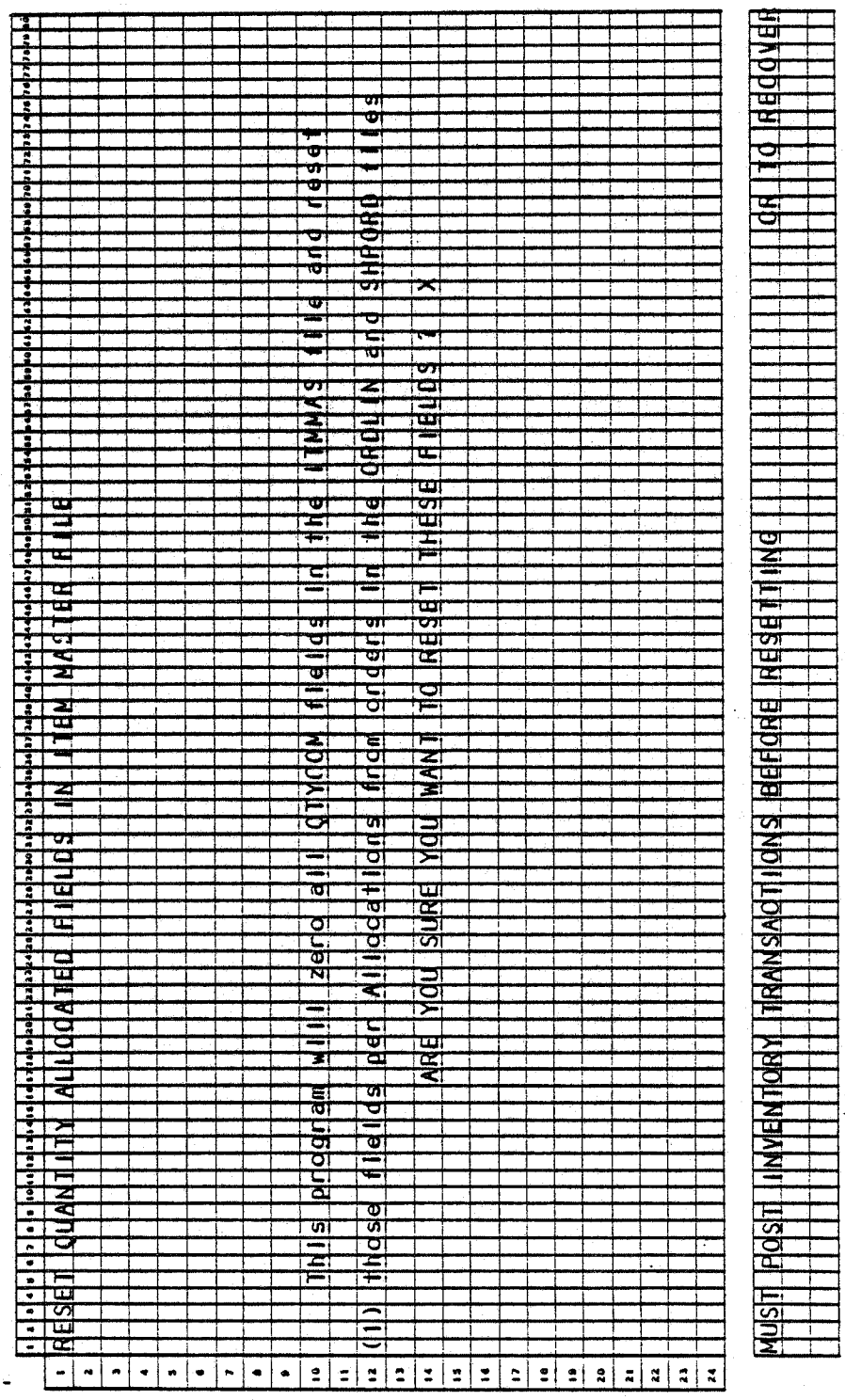
CUSTOMER ORDER PROCESSING PACKAGE
RESET QUANTITY ALLOCATED FIELDS ON ITEM MASTER FILE APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: RSTCOM



Program: RSTCOM



Remarks: (1) IF SFC IS NOT INSTALLED, "SHPORD" WILL BE OMITTED.

RESET QUANTITY ALLOCATED FIELDS ON ITEM MASTER FILE APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Resets the Quantity Allocated field for all locations of all items in the Master file.

Input: SHPORD
SHPIDX
ORDLIN
ITMMAS
ITMIDX
INVTRX
PRDSTR

Files Updated: ITMMAS

Output: (optional) Error List

Enter Module From: SPCFUN

When Done Return To: SPCFUN

Programs in Module: RSTCOM (COMSB is a subroutine of RSTCOM)

Program Functions and Notes:

RSTCOM

Receives message from applicable menu (may be entered through Customer Order Processing SPCFUN or Shop Floor Control SFMENU) giving the name of the program to chain to when done.

Asks if Customer Order Processing is installed (or Shop Floor Control, if entered from SPCFUN).

Opens the above files as follows: open and protect SHPORD and SHPIDX. If Customer Order Processing is installed, open the ORDLIN file. Open the Inventory Management files (ITMIDX, ITMMAS, INVTRX). Read the INVTRX control record. If any unposted activity, display message and exit. If no records are in the INVTRX file and Bill of Material Processor is installed (noted in the ITMMAS control record) open the PRDSTR file.

Sequentially read through the ITMMAS file, setting all QTYCOM fields for all locations to 0 and writing back to the file.

If COP is installed, read through the entire ORDLIN file sequentially, updating the QTYCOM field for the order location for each line item. No updating is done for miscellaneous items ("???") or non-controlled items.

If BOMP is installed (TYPSYS = 6), the non-stocked items are handled separately. If an item is non-stocked, the subroutine COMSB is called to explode the bill of materials of the non-stocked item. The correct quantities are allocated at the first stocked level on every explosion chain.

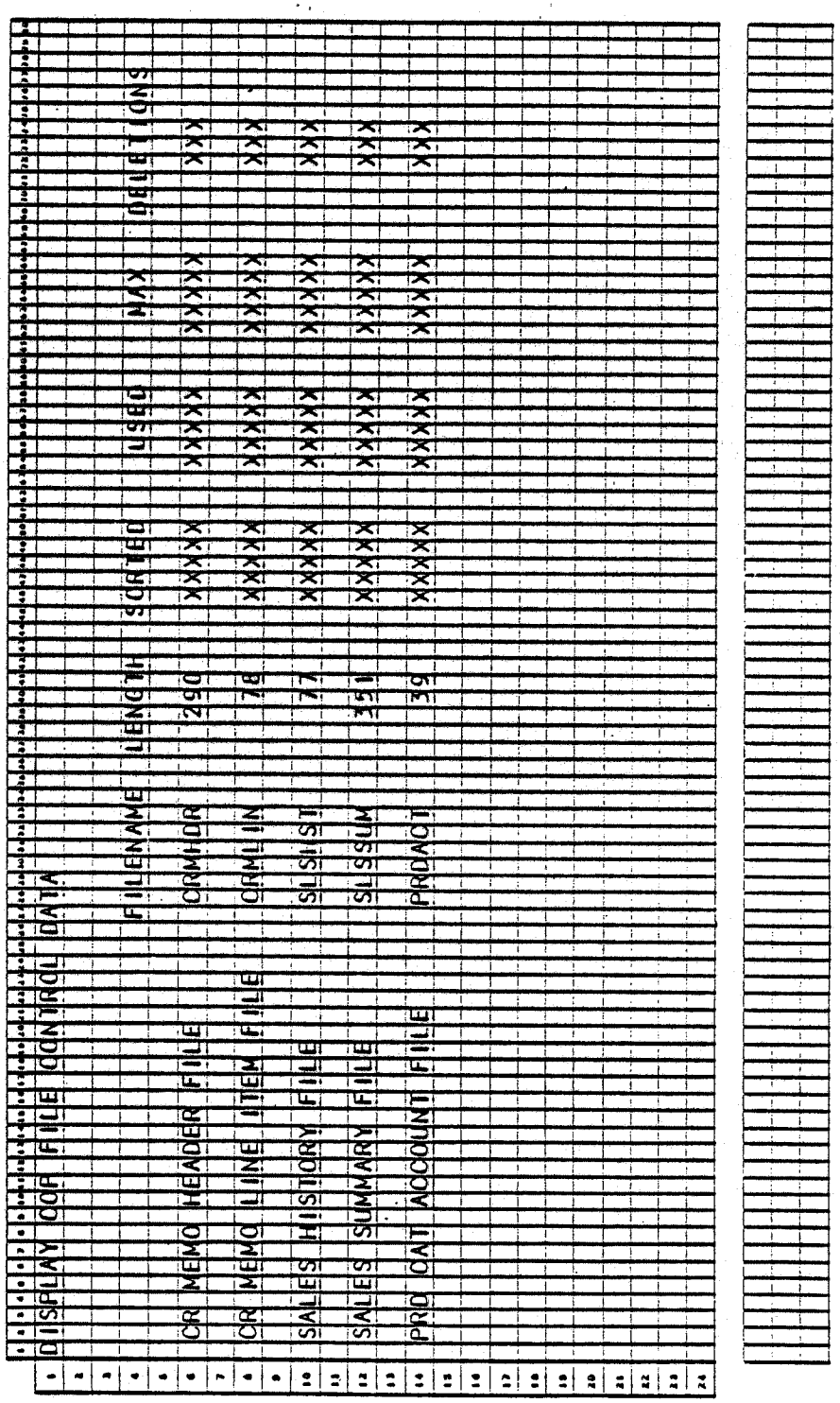
Any errors encountered during bill explosion are printed to the line printer.

If Shop Floor Control interfaces to I/M (IMFLAG = 1), read through the Shop Order file (using the Shop Index) and for any allocated, released or completed order, locate material transactions that are marked as allocated. Search for a match in the ITMMAS file and, if it is controlled, update the QTYCOM field for the shop order location by the shop component's remaining quantity allocated amount. Do not process substitutes and non-controlled items. When done, chain to program passed to PRGNAM field, or default to SPCFUN in Customer Order Processing.

CUSTOMER ORDER PROCESSING PACKAGE
DISPLAY COP FILE CONTROL DATA APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: CPFILS



DISPLAY COP FILE CONTROL DATA APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Displays the file control data from the control records of all of the main files in the COP package.

Input: ORDHDR
ORDLIN
CRMHDR
CRMLIN
SLSHST
SLSSUM
PRDACT
SHIPTO
SHPVIA

Files Updated: None

Output: None

Enter Module From: SPCFUN

When Done Return To: SPCFUN

Programs in Module: CPFILS

Program Functions and Notes:

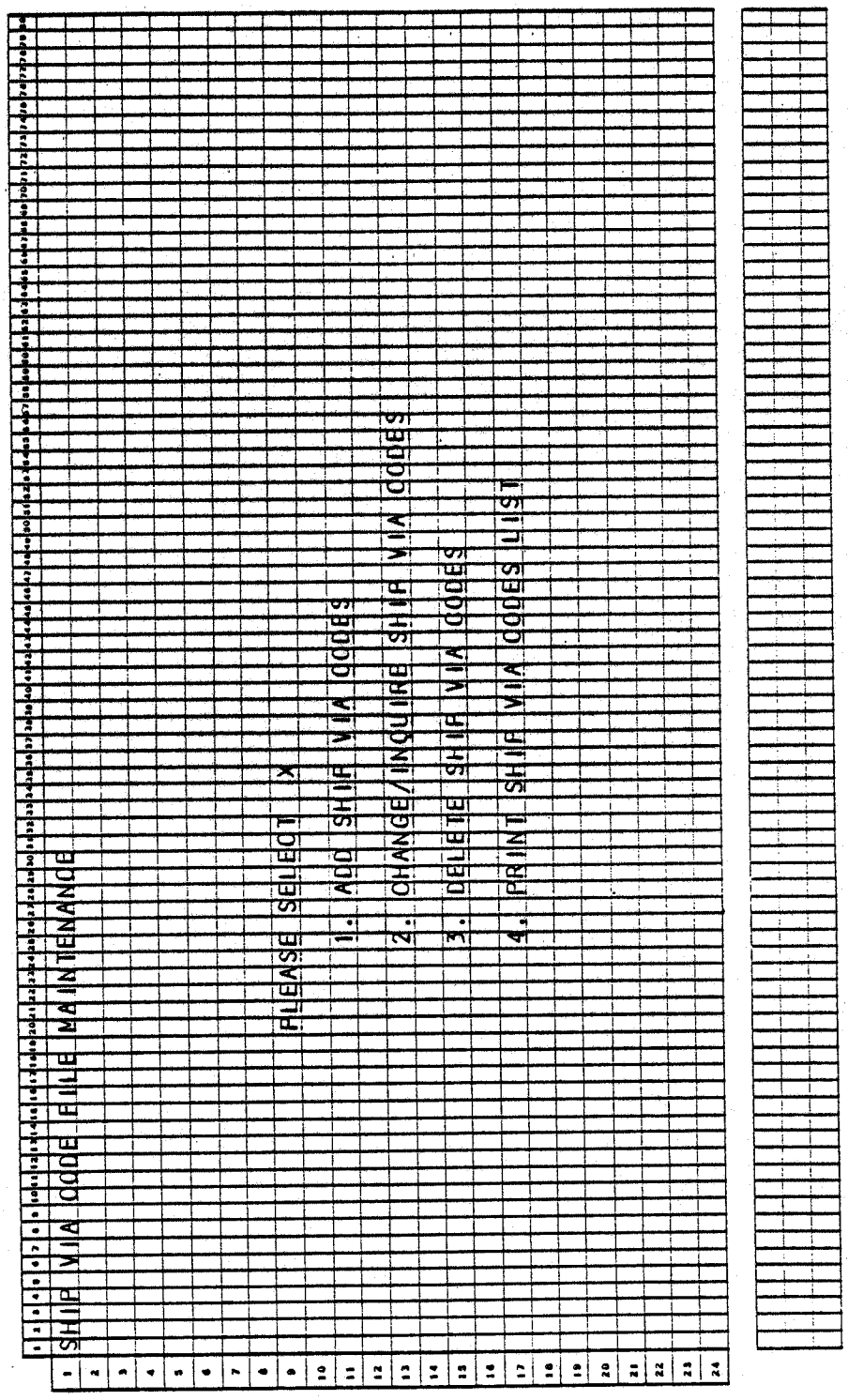
CPFILS

This program does not update the use count on any files. It reads the control records of all of the files shown above, and displays the ORGCNT, RECCNT, MAXCNT and DELCNT fields in an array on the screen.

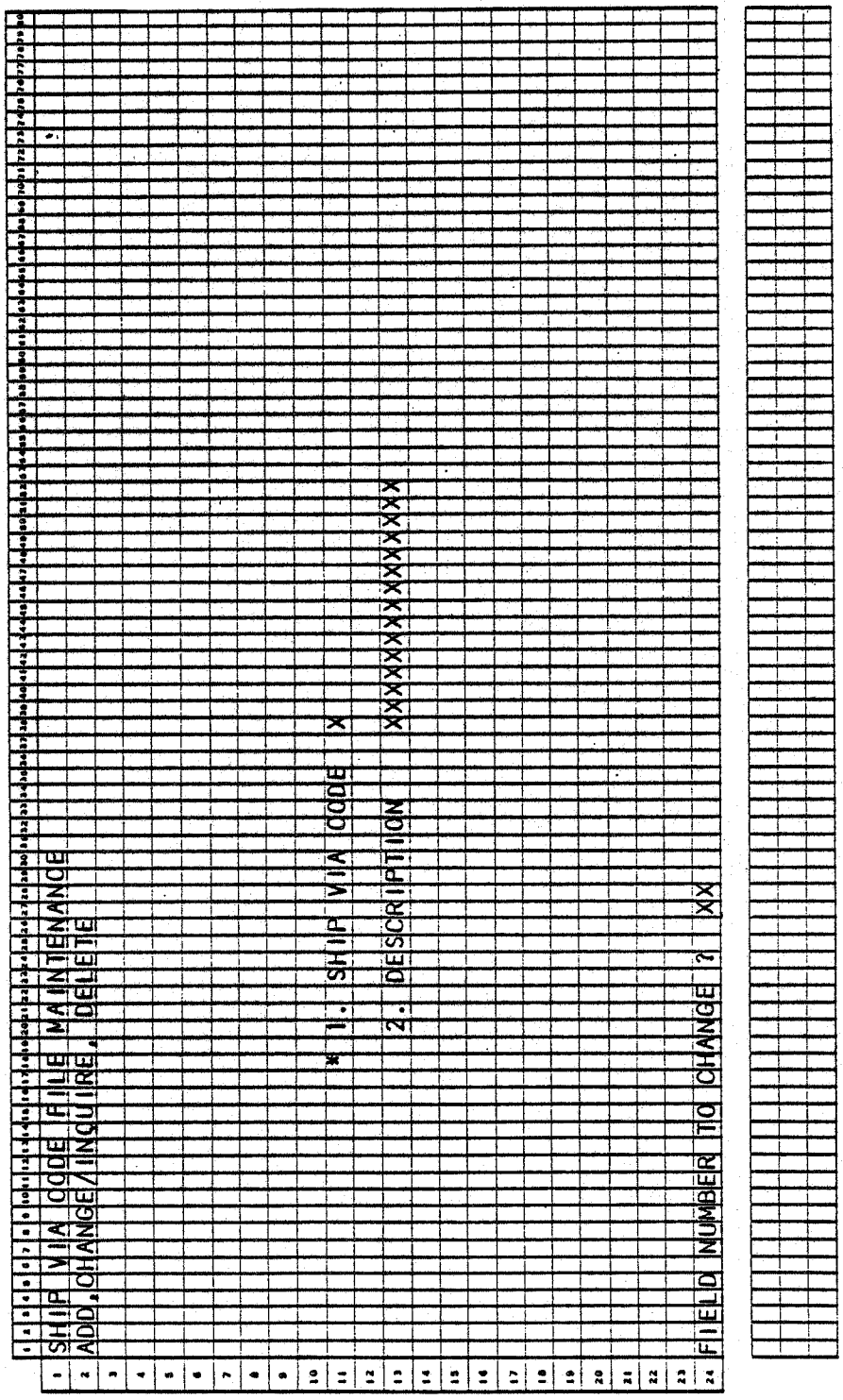
CUSTOMER ORDER PROCESSING PACKAGE
SHIP-VIA FILE MAINTENANCE APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: SHVMNT



Program: SHVMNT



Remarks: * ONLY DISPLAYS IN THE CHANGE/INQUIRE AND DELETE MODES

SHIP-VIA FILE MAINTENANCE APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: This module allows the addition, change, deletion and listing of ship-via information for customers.

Input: SHPVIA Files Updated: SHPVIA Output: Ship-via Code List

Enter Module From: SPCFUN When Done Return To: SPCFUN

Programs in Module: SHPMNT, SHPPRT, ORGSHP, SRTSHP, SHPCNT

Program Functions and Notes:

SHVMNT

This is a standard maintenance module menu.

SHVPRT

This is a completely standard print-out program of the Ship-via codes file.

ORGSHV

This is a completely Standard Master file reorganization program, which physically purges logically selected records from the SHPVIA. There is no index for the Ship-via file.

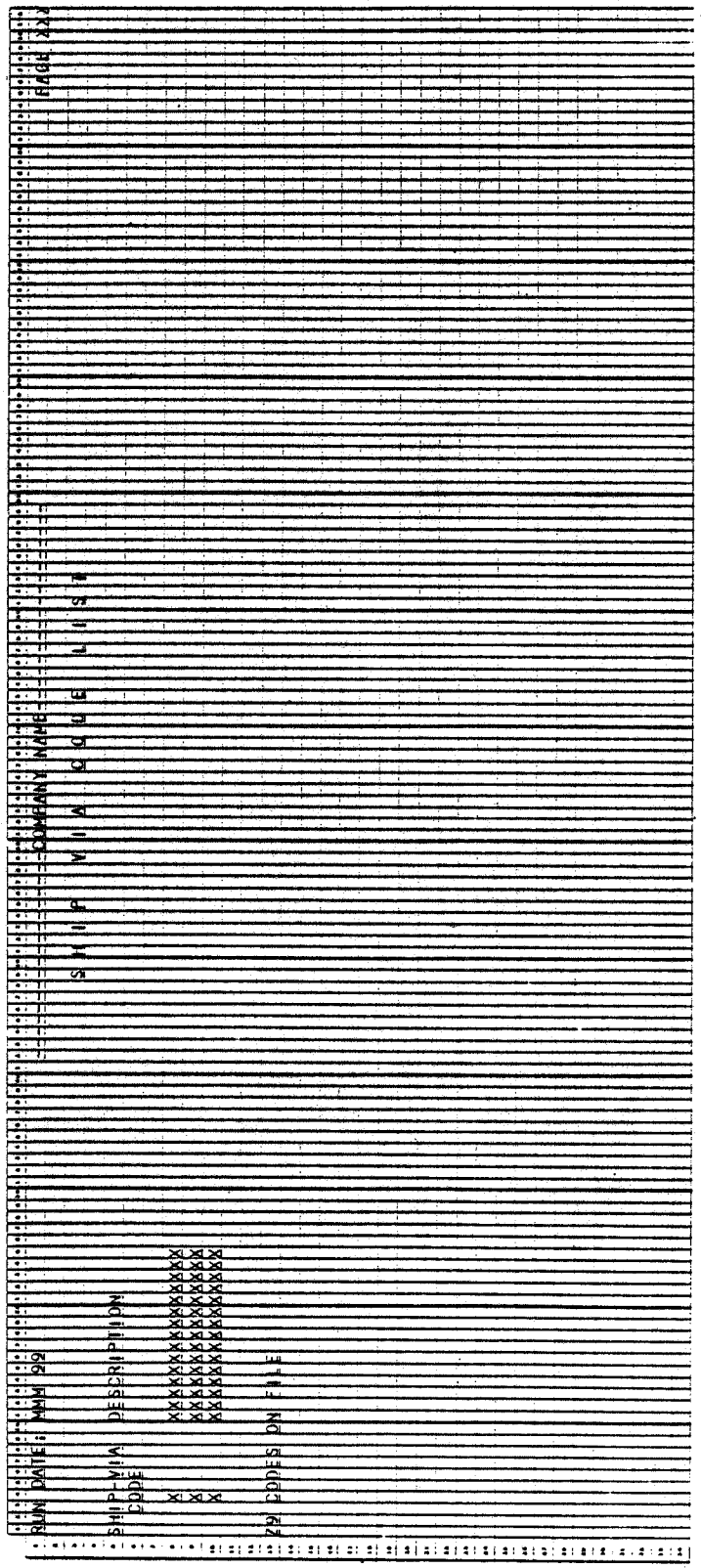
SRTSHV

This is a standard MCBA Sort done on the SHPVIA file, with ascending alpha keys being the Ship-via code.

SHVCNT

This is the standard program to update the Control Record of SHPVIA file after the sort of Ship-via file.

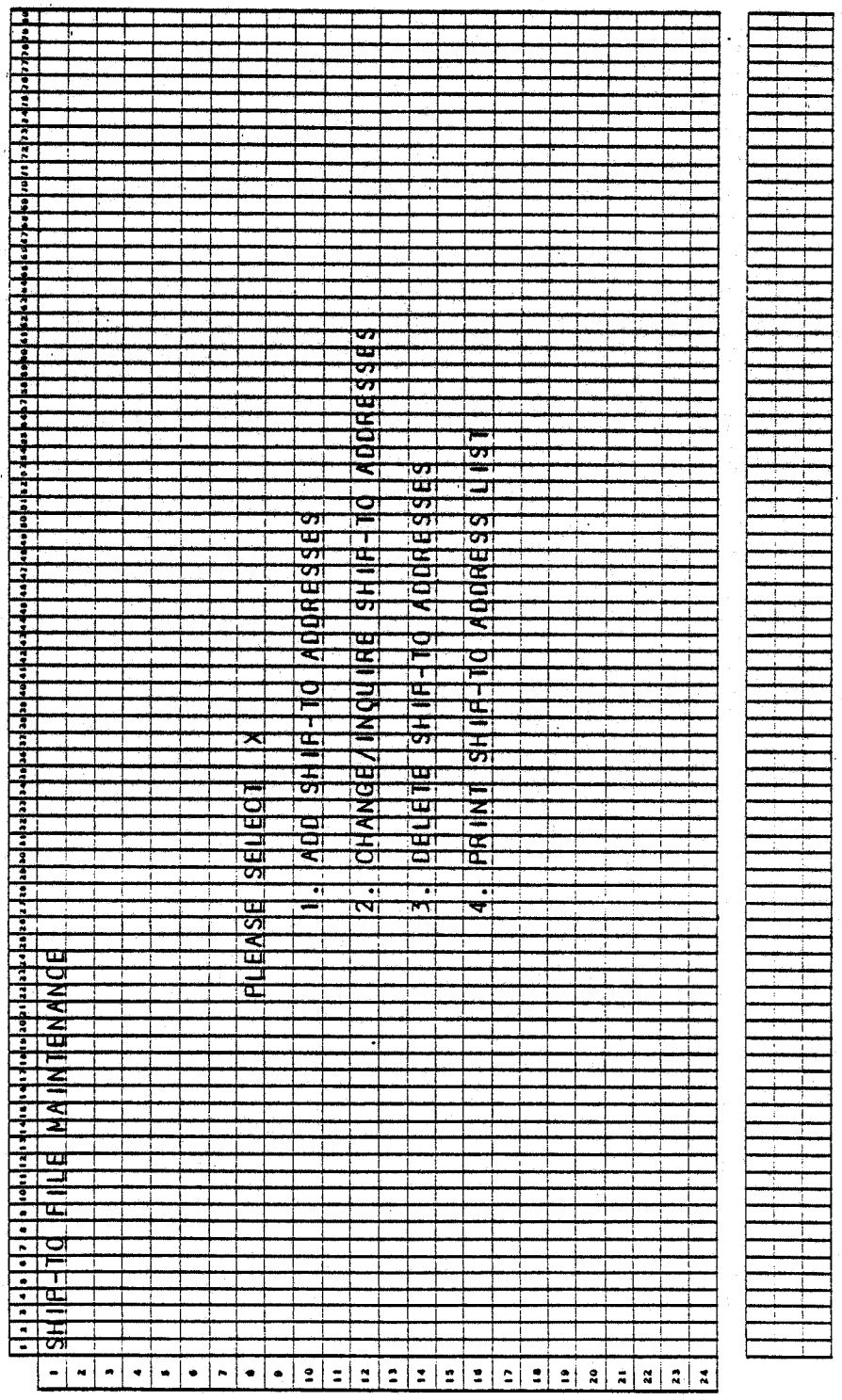
Program: SHVPRT (Ship-Via Code List)



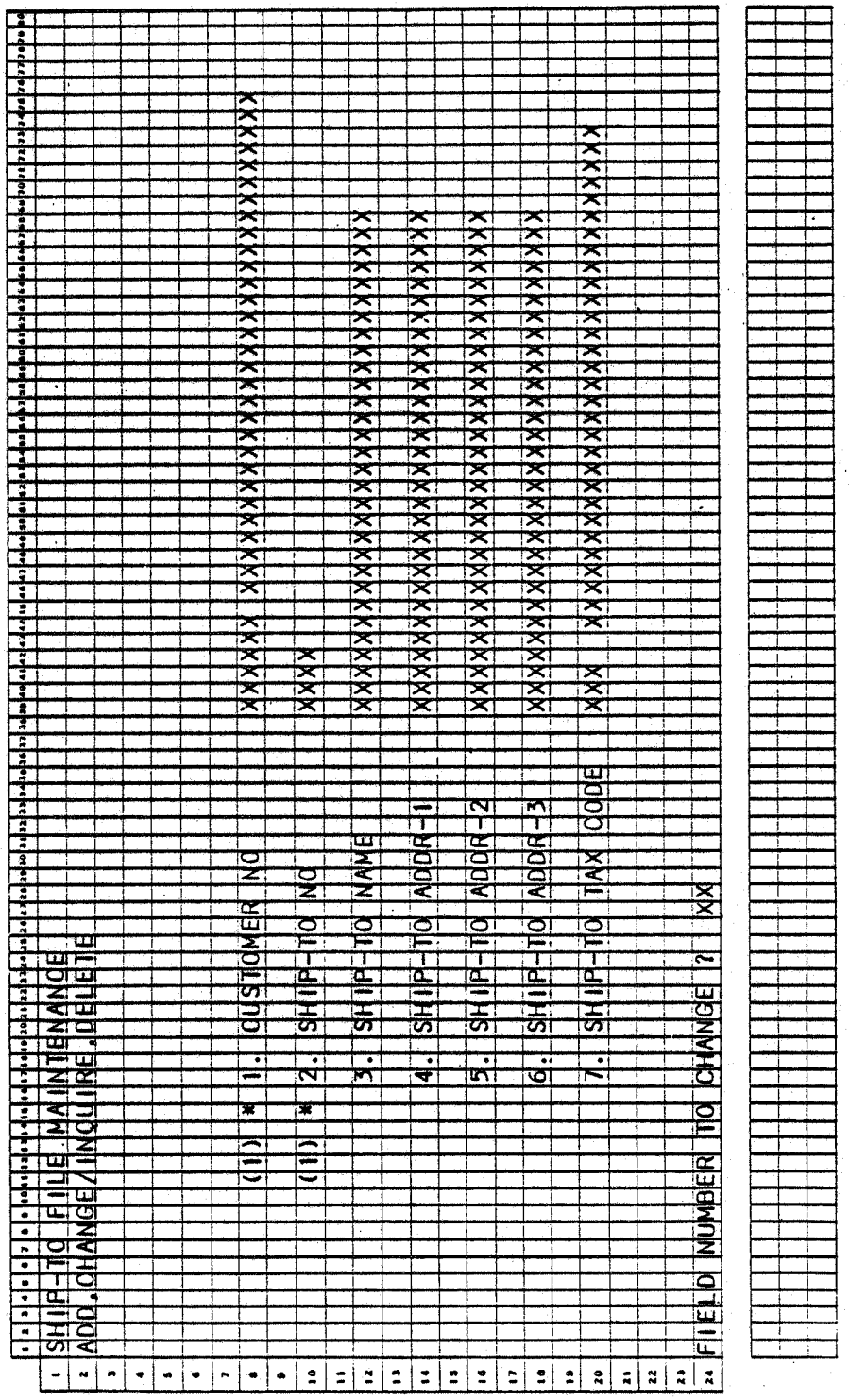
CUSTOMER ORDER PROCESSING PACKAGE
SHIP-TO FILE MAINTENANCE APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: SHTMNT

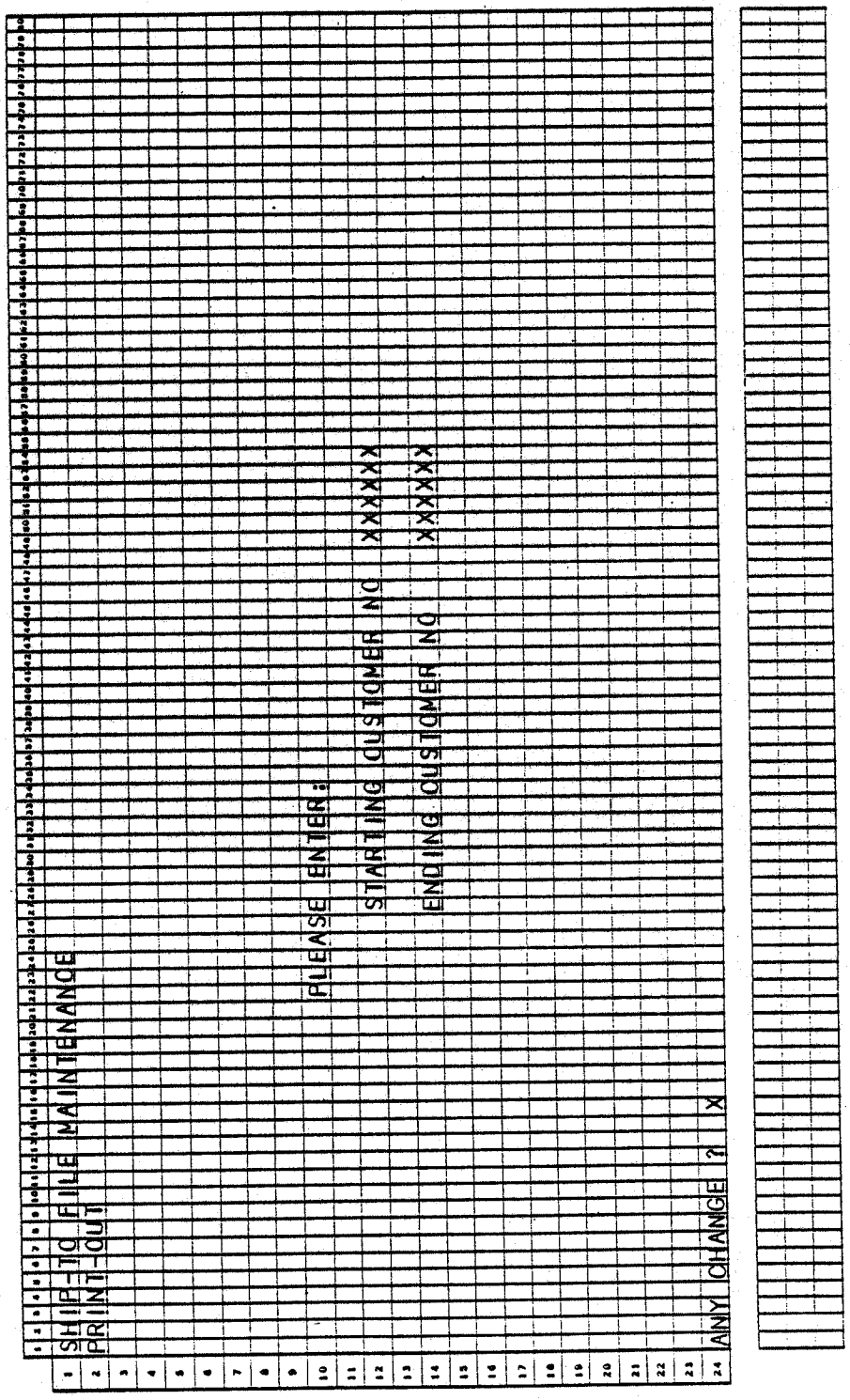


Program: SHTMNT



Remarks: (1) "*" ONLY DISPLAYS IN CHANGE/INQUIRY AND DELETE MODES.

Program: SHTPRT



SHIP-TO FILE MAINTENANCE APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: This module allows the addition, change, deletion and listing of shipping information for customers.

Input: SHIPTO
CUSMAS
CUSIDX
ARTCDE

Files Updated: SHIPTO

Output: Ship-To Address Print-Out

Enter Module From: SPCFUN

When Done Return To: SPCFUN

Programs in Module:

Program Functions and Notes: SHTMNT, SHTPRT, ORGSHT, SRTSHT, SHTCNT

SHTMNT

This is a standard maintenance module menu.

SHTPRT

This is a completely standard print-out program of the Ship-to addresses file.

ORGSHT

This is a completely Standard Master file reorganization program, which physically purges logically selected records from the SHIPTO. There is no index for the Ship-to file.

SRTSHT

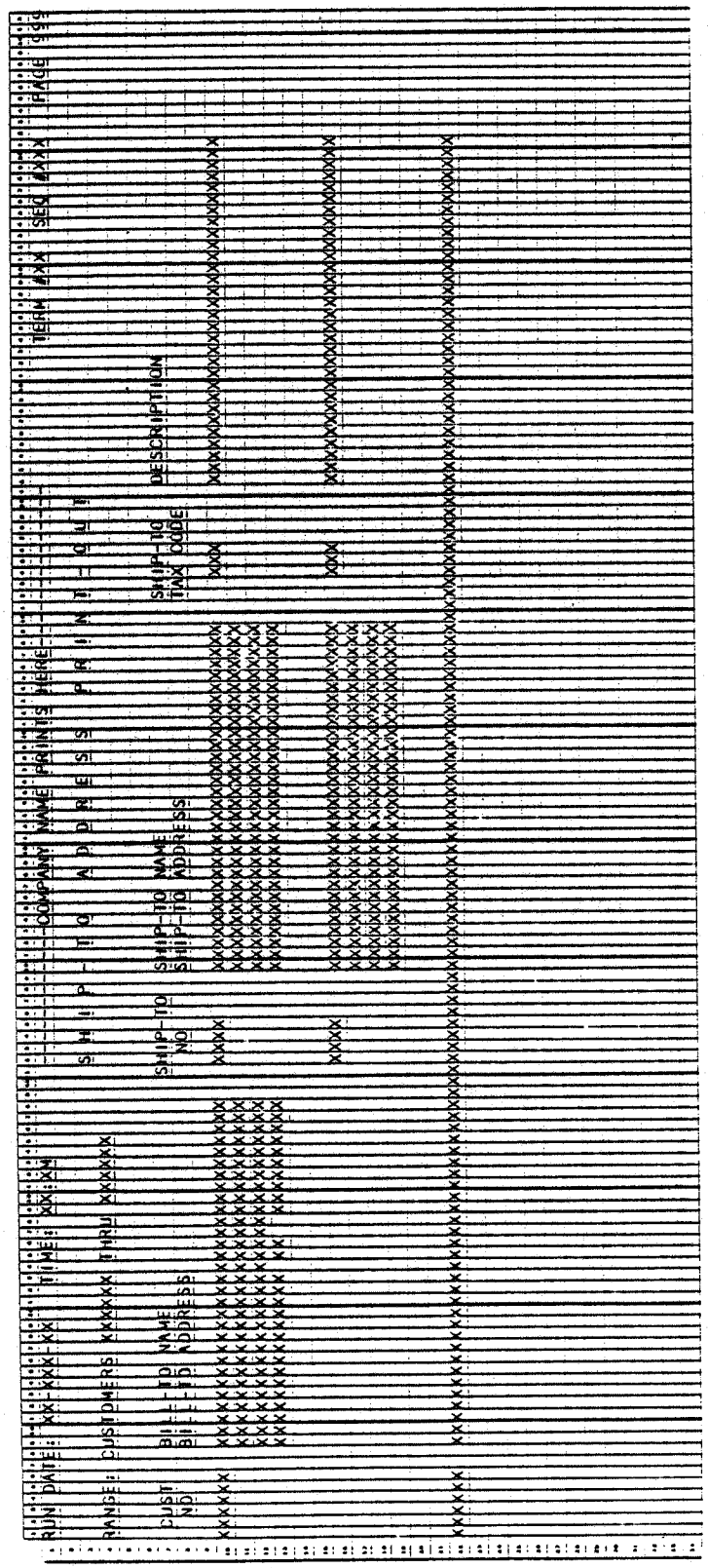
This is a standard MCBA Sort done on the SHIPTO file, with ascending alpha keys being the Customer Number and Ship-to number.

SHTCNT

This is the standard program to update the Control Record of SHIPTO file after the sort of Ship-to file.

SHIP-TO FILE MAINTENANCE APPLICATION
DIBOL JUN-84
REPORT FORMATS

Program: SHTPRT (Ship-To Address Print-Out)

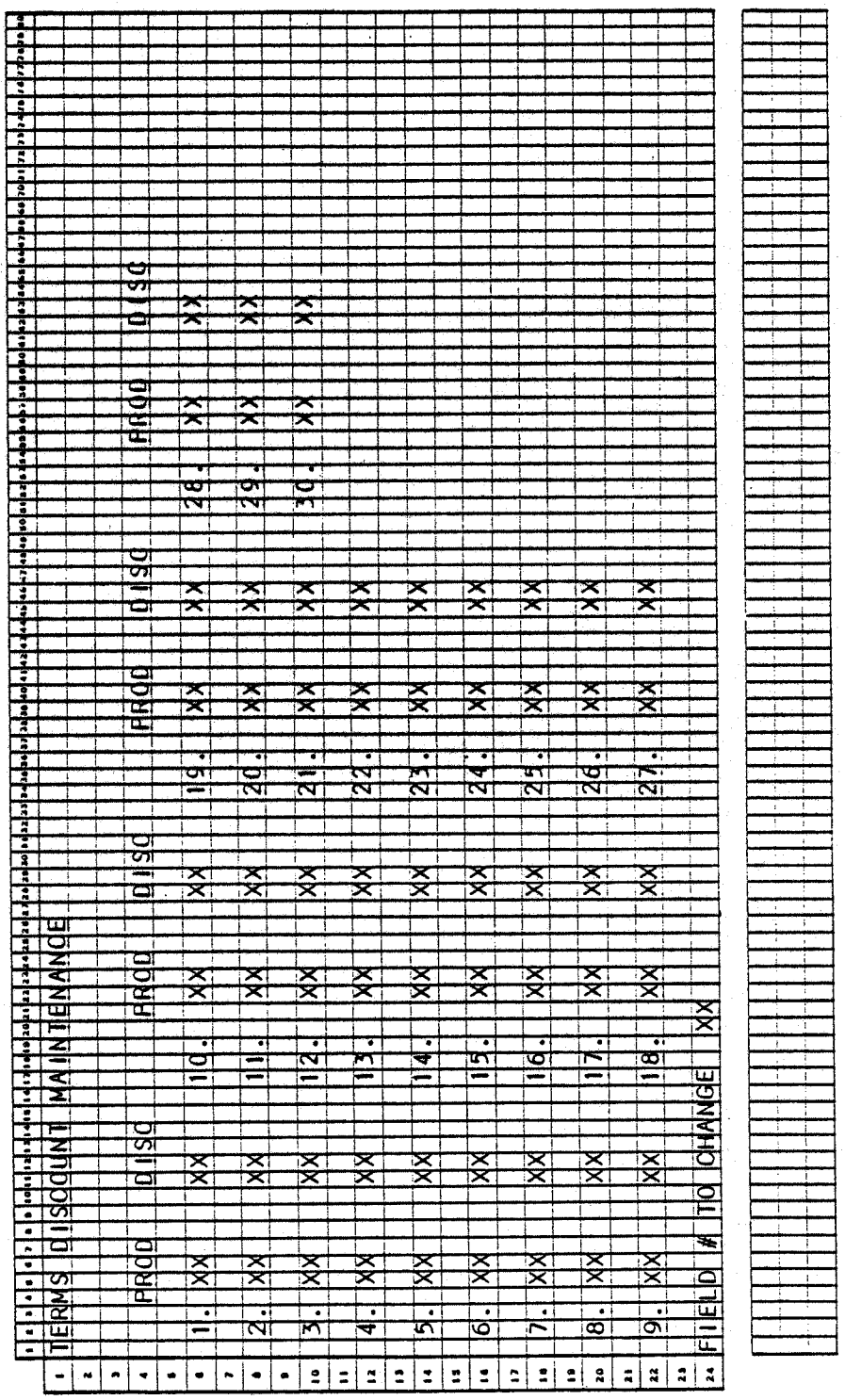


This page intentionally left blank.

CUSTOMER ORDER PROCESSING PACKAGE
TERMS DISCOUNT BY PRODUCT CATEGORY MAINTENANCE APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: TERMSD



TERMS DISCOUNT BY PRODUCT CATEGORY MAINTENANCE APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Terms (line item) discount array maintenance.

Input: CUSIDX Files Updated: CUSMAS Output: None

Enter Module From: SPCFUN When Done Return To: SPCFUN

Programs in Module: TERMSD

Program Functions and Notes:

TERMSD

Display all existing product categories and corresponding discounts from the CUSMAS control record.

Accept additions, changes, or deletions to the array.

If the operator selects a number for which there is no category, and that number is not the next available slot, direct him to that slot, regardless of what number he has input.

Allow an existing category to be deleted by blank for product category. After a code is deleted, "SQUEEZE" the array so that all active categories occupy the first positions. For instance, if there are three product categories in the array, they must occupy TPRDCD(1), TPRDCD(2), and TPRDCD(3).

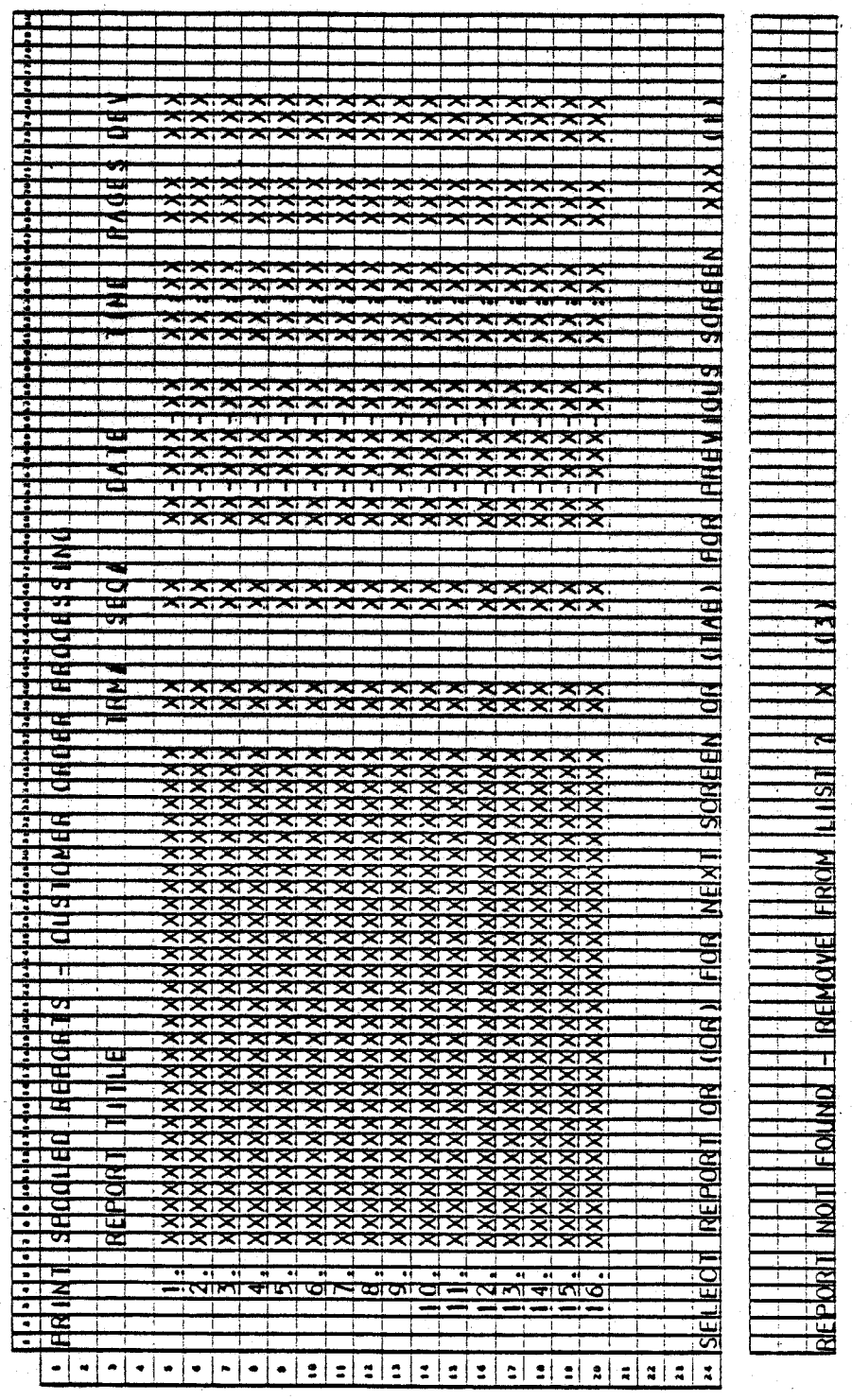
If a new product category is added, or an existing category is changed, check to be sure this category does not already exist in the array. Do not allow duplicate categories to be entered.

Allow the program to begin running only if exclusive use of the CUSIDX and CUSMAS file is obtainable.

CUSTOMER ORDER PROCESSING PACKAGE
PRINT SPOOLED REPORTS APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: CPSPOL



Remarks: (1) THE "NEXT SCREEN" AND "PREVIOUS SCREEN" PORTIONS OF THIS MESSAGE ARE DISPLAYED ONLY IF THEY ARE APPLICABLE. (2) THE "STARTING PAGE" PROMPT IS DISPLAYED IF "P" IS SELECTED; THE "ARE YOU SURE ?" PROMPT IS DISPLAYED IF "D" IS SELECTED. (3) THIS MESSAGE IS DISLAYED IF THERE IS AN ENTRY IN THE SPOOLER DIRECTORY, BUT THE SPOOLED REPORT WAS NOT FOUND ON THE DEVICE LIST.PRINT SPOOLED REPORTS - CUSTOMER ORDER PROCESSING.

PROGRAM SPECIFICATIONS

Function: Allows the user to inspect the list of reports that have been spooled, and print or delete any one he chooses.

Input: Files Updated: SPLDIR Output: Any spooled report

Enter Module From: SPCFUN When Done Return To: SPCFUN

Programs in Module: CPSPOL

Program Functions and Notes:

CPSPOL

The program gets the Company code of the current terminal by reading record 99 of the DEVICE.DDF file. It reads through the Spooler Directory file (SPLDIR) from beginning to end and puts the record number of all directory entries corresponding to Customer Order Processing reports for the user's Company code into an array in memory.

It then displays the first 16 reports, using this stored array to locate the records on the SPLDIR file.

If the user selects to print a report, the program tries to open the file. If it is unsuccessful, it displays the "NOT FOUND" message. If the user then says he wants to remove the report from the list, the SPLDIR record is marked for deletion with "OO" in the SPLDEL field.

If the report is found on the expected device, the PRSPL subroutine is used to handle the printing of the report. On exit from the program, if any reports were deleted, the SPLDIR file is reorganized to remove the deleted records.

CUSTOMER ORDER PROCESSING PACKAGE
FILE DEFINITIONS
DIBOL JUL-84

GENERAL FILE DEFINITION DATA

There are several distinct types of files used in the MCBA accounting and manufacturing packages, each type having a similar set-up and use no matter which particular package it is in. There are also some files which do not conveniently fit into any of these categories.

The file types to be described here generally are:

1. Standard Master File with Index
2. Standard Transaction File
3. Temporary Index File

Please note that except for Order Line and Order Header files in the Customer Order Processing package, the MCBA DIBOL accounting and manufacturing packages do not use ISAM files. This is basically to keep individual program size down (the use of ISAM files adds significantly to the size of programs), and to increase program execution speed within individual programs.

1. Standard Master File with Index

Examples of this type of file would be the Customer Master file and Customer Index (CUSMAS, CUSIDX) in Accounts Receivable; the Employee Master file and Employee Index (EMPMAS, EMPIDX) in Payroll, etc.

Both the Master file and its Index are permanent files (they are two separate files which may reside on different physical devices). After the file is initially set up, the information in it remains fairly stable with time (as compared to the other types of files to be mentioned here).

The first record of the Master file is called the control record. It does not contain an actual data record of the file (such as customer information) but contains only information about the file itself. The last 18 characters of the control record of a Master file always have the same format:

Organized Count	ORGCNT	,D5)	
Record Count	RECCNT	,D5)	18 characters
Maximum Count	MAXREC	,D5)	
Delete Count	DELCNT	,D3)	

The significance of these are as follows:

The need for a reorganization is recognized by the Master File Maintenance program by inspecting the value of DELCNT. The need for a sort is recognized by the Master File Maintenance program by comparing the values of ORGCNT and RECCNT. If both a reorganization and a sort need to be done, the reorganization is done first, then the sort, all in the same job stream. The last three letters of its name are usually "CNT" (e.g. CUSCNT, EMPCNT).

a) ORGCNT - This gives the number of records in the Index file that are known to be in sorted order. After a sort is done on the index in the normal course of processing, the "CNT" program mentioned in b) above (e.g. CUSCNT, EMPCNT), sets ORGCNT = RECCNT, indicating that the Index file is at this time completely sorted. As new Master records are added by the Master File Maintenance program, RECCNT is increased, while ORGCNT remains the same. ORGCNT includes the control record.

b) RECCNT - This is the count of the number of valid data records in the file. It includes the control record and all logically deleted records. It does not include dummy bracket records. When a new data program, RECCNT is checked and the new record is added to RECCNT + 1. RECCNT + 1 was the first dummy bracket record in the file, and it is over-written by the new data record. The value of RECCNT is then incremented by 1. A record is also added to the Index file in the same position (the Index file is also padded - pre-extended - with dummy bracket records).

c) MAXREC - This gives the maximum available space for data records in the file. It is set by the File Initialization program when the file is originally created (e.g. by INITAR, INITPR, etc.). It includes the control record. There is always exactly one more record on the file than the count given by MAXREC. This is to ensure that the very last record of the file (and its Index file) is <u>always</u> a dummy bracket record.

d) DELCNT - This is the count of logically deleted records now on the Master file. When the delete function of the Master File Maintenance application is used, the indicated record is not physically deleted at this time. Rather, it is marked as <u>logically</u> deleted - ']]]DEL' is inserted into a predetermined location in the Master File record, and '00000' is inserted in a predetermined location in the corresponding Index record (usually in the Record Number field, i.e. the field that gives the record number of the corresponding Master File record). When DELCNT reaches a certain point (usually 50 or 95), the Master File Maintenance program senses this fact, and automatically invokes the reorganization program to physically purge these records from both the Master file and the Index file.

The first record of the Index file for the Master file is usually a record of blanks (this fact should be remembered if you are inspecting the Index file using an Editor, or using PIP).

The Index file and the Master file are kept in synchronization. When a new record is added to the Master file, a corresponding record pointing to it is added to the Index file. When a record is marked as deleted on the Master file, its corresponding Index record is also marked as deleted.

The Index record for a Master record usually contains the key value and the record number of the Master record in the Master file.

2. <u>Standard Transaction File</u>

Examples of this type of file would be the Sales Transaction file (SALESO.YYY) in Accounts Receivable; the Payroll Work file (PAYWRK.YYY) in Payroll, etc.

A Transaction file is a permanent file in that the file itself is never deleted by any program once it has been initially created by the File Initialization program. However, the data held by it is very volatile, and is completely cleared from it on a regular basis.

The first record of a Transaction file is its control record, just as for a Master file. The information contained in the control record is identical to that in the control record for a Master file.

See the Program Specifications for the Standard Transaction File Entry for more details on the use of this type of file. The normal data flow through this file is as follows:

a) Records are added, changed, and deleted in a way similar to that for a Master file (except there is no Index). The records in the file are in the order that they are entered - they are not in sorted order during the entry and editing phase.

b) An Edit List of the contents of the Transaction file may be printed at any time, and this Edit List will again be in entry order.

c) When the user decides to post the transactions in the file, the Transaction file is sorted. Data is transferred from the Transaction file to one or more of the main files of the system; and the Transaction file is cleared to one record, the control record, with the remainder being repadded with dummy bracket records.

   Thus, after posting is complete, the Transaction file is in exactly the same condition as when it was created by the File Initialization program.

3. Temporary Index File

Examples of this type of file would be the TMPIDX file used in the Accounts Receivable package to produce the Alphabetical Customer List; the TVNIDX file used in Accounts Payable to produce the Alphabetical Vendor List, etc.

It is an Index to a Master file or to another main file based on a particular key, which is not the key of the permanent Index file (if there is one). It is usually used to produce a print-out of a file in a sort order which is different from the normal sort order of the main file or its permanent index. It can also be used to do other types of sequential processing on a Main file in an order other than its normal order (such as with the PURIDX file in the "Purge" application of Accounts Receivable).

The form of a record in the Temporary Index file is simply some key field, obtained from the Main File record, plus a pointer to that record. It is created for a specific application in a "Build Index" type program, sorted on the key by a standard MCBA Sort program, and then used by a print-out or other type program after it is sorted. Once the application is completed, the last program of the application deletes the Temporary Index file in its entirety.

This page intentionally left blank.

A/R DISTRIBUTION ACCOUNT FILE
(ARACCT)

Description: A/R Distribution Account File. This is a master file containing the G/L account numbers that are used in the A/R package. An account number must be in this file before it can be used by any other A/R program.

File Status: Master Record # in DEVICE.DDF: 07 Rec. Size: 37
+ End of Record

The first record of the file is a standard control record.

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
G/L Account Number	ARACNO	D7	XXXX-XXX
Account Description	ARACDS	A30	

NOTE: This is a master file without an index. Deleted records are marked with "000000" in the last 6 characters of the ARACDS fields.

This page intentionally left blank.

A/R TAX CODE FILE
(ARTCDE)

Description: A/R Tax Code File. This is a permanent file of all Tax codes.

File Status: Key Index Record # in DEVICE.DDF: 169 Rec. Size: 69
+ End of Record

The first record of the file is a Control record.

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Tax Code	ARTCOD	A3	
Tax Code Description	ARTDSC	A30	
Tax Percent	ARTPRT	3D5	XX.XXX%
G/L Account Number	ARTGLN	3D7	XXXX-XXX
** CONTROL RECORD **			
(Unused)		A51	
Organized Record Count	ORG169	D5	
Record Count	REC169	D5	
Maximum Record Count	MAX169	D5	
Delete Record Count	DEL169	D3	

See the General File Definition section for an explanation of the above fields.

Field Descriptions

ARTCOD - Tax Code. This is a three character code used to identify the Tax Code. For example, "CAL" might be used to represent the California sales tax.

ARTSDC - Tax Code Description. The 30 character description of the Tax code. In the above example, this might be "California Sales Tax".

ARTPRT - Tax Percent. The tax percent. In the case of the California sales tax, this would be 6%.

ARTGLN - G/L Account Number. This is the account to which any tax amounts charged to this customer will be posted.

This page intentionally left blank.

BACK ORDER FILE
(BAKORD)

Description: Back Order File

File Status: Work Record # in DEVICE.DDF: 51 Rec. Size: 140
+ End of Record

Is the first record of this file used as a control record? Yes

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Item Number	BITMNO	A15	
Description	BDESCR	A30	
Product Category	BPRDCT	A2	
Customer Number	BCUSNO	D6	
Name	BCUSNM	A25	
Order Number	BORDNO	D6	
Order Date	BORDDT	D6	YYMMDD
(Unused)	BSCHDT	D6	
Quantity Ordered	BQTYOR	D4	
Quantity Back Ordered	BQTYBO	D4	
Location	BLOC	A2	
Unit Price	BUNPRC	D7	$XX,XXX.XX
Order Discount	BODISC	D2	XX%
Discount (Line)	BLDISC	D2	XX%
Separate Ship-to	BSHPTO	A1	N=No, Y=Yes
Stocked Flag	BSTOKD	A1	S=Stocked, Blank=Non-Stocked
Order Type	BORTYP	A1	O=Item not backorderable

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Customer's Purchase Order	BCUSPO	A10	
Job Number	BJOBNO	A10	
** CONTROL RECORD **			
(Unused)		A105	
By Scheduled or Order Date ?	DATFLG	A1	
Starting Date for Report	STDATE	D6	
Ending Date for Report	ENDATE	D6	
Report Type	RPTTYP	D1	
Location(s) Reported	RPTLOC	A2	
By Customer Report Flag	CUSRPT	D1	
Organized Record Count	ORG051	D5	
Record Count	REC051	D5	
Maximum Record Count	MAX051	D5	
Deleted Record Count	DEL051	A3	

BACK ORDER INDEX FILE
(BOINDX)

Description: Index to BAKORD File

File Status: Work Record # in DEVICE.DDF: 52 Rec. Size: 34
+ End of Record

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Item Number	BIITMN	A15	
Location of Order	BILOC	A2	
Date of Order	BIORDT	D6	YYMMDD. See Note 1.
Customer's Number	BICUNO	D6	
Relative Record Number in BAKORD	IRC051	D5	

NOTES:

1. This may be Scheduled Date if operator so requests.

2. Sorted by Item Number, then Location, then Order Date (see Note 1), if the report is requested "BY ITEM". Sorted by Customer Number, then Item Number, then Order Date (See Note 1), if the report is requested "BY CUSTOMER".

This page intentionally left blank.

COP CONTROL FILE
(COPCTL)

Description: COP Control File. Contains defaults and accounting data used in Customer Order Processing.

File Status: Permanent Record # in DEVICE.DDF: 60 Rec. Size: 209
+ End of Record

Is the first record of this file used as a control record? No

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Default Billing Comments	ORDCOM	2A35	
Last Order Number Used	LSTORD	D6	
Last Invoice Number Used	LSTINV	D6	
Last Credit Memo Number Used	LSTCRM	D6	
(Unused)		A6	
G/L Distributions Flag	DSTFLG	A1	Y/N
(Unused)		A4	
Multiple A/R Accounts Flag	MLARFG	A1	Y/N
Default A/R Account No	DEFARA	D7	XXXX-XXX
(Unused)		A14	
Multiple Misc. Accounts Flag	MLMSFG	A1	Y/N
Default Misc Account No	DEFMSA	D7	XXXX-XXX
(Unused)		A14	
Multiple Sales Tax Accounts Flag	MLSTFG	A1	Y/N
Default Sales Tax Account No	DEFSTA	D7	XXXX-XXX
(Unused)		A14	
Multiple Freight Accounts Flag	MLFRFG	A1	Y/N
Default Freight Account No	DEFRET	D7	XXXX-XXX
(Unused)		A36	

This page intentionally left blank.

CREDIT MEMO HEADER FILE
(CRMHDR)

Description: File is sorted in Credit Memo Number sequence

File Status: Master Record # in DEVICE.DDF: 46 Rec. Size: 322
+ End of Record

Is the first record of this file used as a control record? Yes

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Credit Memo Number	CCRMNO	D6	
Memo Date	CCRMDT	D6	MM/DD/YY
This Memo Applies to Invoice Number	CAPLTO	D6	
Customer Number	CCUSNO	D6	
Ship-to Number	CSHPTO	D4	
Customer Name	CCUSNM	A30	
Customer Address, Line 1	CADD1	A30	
Customer Address, Line 2	CADD2	A30	
Customer Address, Line 3	CADD3	A30	
Salesman Number	CSLMAN	D2	
Location of Inventory	CLOC	A2	
Customer's Purchase Order Number	CCUSPO	A10	
Memo Discount Percent	CDSCNT	D2	XX%
Comment Lines	CCOMNT	2A35	
Amount of Sale	CSALAM	D8	$XXX,XXX.XX
A/R Account Number	CARACT	D7	XXXX-XXX
Miscellaneous Amount	CMISC	D6	$X,XXX.XX
Miscellaneous Account Number	CMSACT	D7	XXXX-XXX
Sales Tax Amount	CTAX	3D7	$XX,XXX.XX

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Sales Tax Account Number	CTXACT	3D7	XXXX-XXX
Freight Amount	CFRGHT	D6	$X,XXX.XX
Freight Account Number	CFRACT	D7	XXXX-XX
Cost of Goods	CCOST	D8	$XXX,XXX.XX
Commission Amount	CCOMDU	D7	$XX,XXX.XX
** CONTROL RECORD **			
(Unused)		A314	
Organized Record Count	ORG046	D5	
Records Count	REC046	D5	
Maximum Record Count	MAX046	D5	
Deleted Record Count	DEL046	D3	

CREDIT MEMO LINE ITEM FILE
(CRMLIN)

Description: Credit Memo Line Item File

File Status: Master Record # in DEVICE.DDF: 47 Rec. Size: 78
+ End of Record

Is the first record of this file used as a control record? Yes

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Credit Memo Number	CLCRNO	D6	
Item Number	CLITEM	A15	
Description	CLDESC	A30	
Product Category	CLPDCD	A2	
Quantity Credited	CLQTY	D4	
Location	CLLOC	A2	
Unit of Measure	CLUOFM	A2	
Unit Price	CLPRCE	D7	$XX,XXX.XX
Line Discount	CLDISC	D2	XX%
Unit Cost	CLCOST	D7	$XX,XXX.XX
Discount Allowed Flag	CDAFLG	A1	
** CONTROL RECORD **			
(Unused)		A60	
Organized Record Count	ORG047	D5	
Record Count	REC047	D5	
Maximum Record Count	MAX047	D5	
Deleted Record Count	DEL047	D3	

This page intentionally left blank.

CUSTOMER INDEX FILE
(CUSIDX)

Description: Customer Index File. This is the index to the Customer Master file. It is one of the four files that make up the main part of the Accounts Receivable data base.

File Status: Master Index Record # in DEVICE.DDF: 02 Rec. Size: 10
+ End of Record

The first record is blank and never used.

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Customer Number	ICUSNO	D6	
Record Number is CUSMAS File	IRC001	D5	

Field Description

ICUSNO - Customer Number. This is the key of the file and matches the CUSNO field on the CUSMAS file.

IRC001 - Record Pointer to CUSMAS record. When a customer record is deleted by the CUSMNT program this field is set to zero.

See the General File Definitions section for more information on the Standard Master file and its Index file.

This page intentionally left blank.

CUSTOMER MASTER FILE
(CUSMAS)

Description: Customer Master File. This file contains one record for each customer. It is one of the four files that make up the main part of the Accounts Receivable data base (the other three being the CUSIDX, AROPEN, and ARDIST files). It contains the fairly static data connected with a customer, as well as cumulative historical data showing the overall activity for the customer.

File Status: Master Record # in DEVICE.DDF: 01 Rec. Size: 217 Data + End of Record

*See note concerning record size at the end of this File Definition.

The first record is used as a control record.

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Customer Number	CUSNO	D6	
Name	NAME	A25	
Address Line #1	ADD1	A25	
Address Line #2	ADD2	A21	
City	CITY	A15	
State	STATE	A2	
Zip Code	ZIP	A10	
Phone Number	PHONE	D10	XXX-XXX-XXXX
Salesman Number	SLSMAN	D2	
Territory	TERR	A2	
Inventory Location	CUSLOC	A2	
Customer Type	CUSCD	A2	
Customer Code-2	CUSCD2	A2	Not used at this time.
Sales $ Month-to-Date	SALMTD	D8	$XXX,XXX.XX-

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Sales $ Year-to-Date	SALYTD	D8	$XXX,XXX.XX-
Cost $ Month-to-Date	COSMTD	D8	$XXX,XXX.XX-
Cost $ Year-to-Date	COSUTD	D8	$XXX,XXX.XX-
Tax Code	TAXFLG	A3	
Credit Limit	CRDLMT	D6	$XXX,XXX.-
Account Balance	OSTDCR	D8	$XXX,XXX.XX
Terms	TERMS	A1	
Account Balance Method	BALMTH	A1	"O" = open item "B" = balance forward
Statement Flag	STMFLG	D1	1 = gets a statement 2 = does not get a statement
External Flag	EXTFLG	A1	Not used at this time.
Product Code Array	PRDCD	10A2	
Discount Percent Array	DISCNT	10D2	XX%
** CONTROL RECORD **			
(Unused)		A11	
(Unused)	PAD1	nA4	See Field Descriptions.
Detailed G/L Distribution Flag	DETDST	A1	Y/N
Multiple Cash Accounts ?	MLTCSH	A1	Y/N
Default Cash Account	DEFCSH	D7	XXXX-XXX
Multiple Discount Accounts ?	MLTDSC	A1	Y/N
Default Discount Account	DEFDSC	D7	XXXX-XXX

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Number of Product Discounts	NUMDSC	D2	
Multiple A/R Accounts ?	MLTAR	A1	Y/N
Default A/R Account	DEFAR	D7	XXXX-XXX
Multiple Sales Accounts ?	MLTSLS	A1	Y/N
Default Sales Account	DEFSLS	D7	XXXX-XXX
Multiple Other Charges Accounts ?	MLTOCH	A1	Y/N
Default Other Charges Account	DEFOCH	D7	XXXX-XXX
Multiple Tax Accounts ?	MLTTAX	A1	Y/N
Default Tax Account	DEFTAX	D7	XXXX-XXX
Multiple Freight Accounts ?	MLTFRT	A1	Y/N
Default Freight Account	DEFFRT	D7	XXXX-XXX
Default Finance Charges Account	DEFFCH	D7	XXXX-XXX
Terms Product Category	TPRDCD	30A2	
Terms Discount Percentage	TDISC	30D2	
Delete Flag	DFL001	D1	
Sort Flag	SFL001	D1	
Organized Record Count	ORG001	D5	
Record Count	REC001	D5	
Maximum Record Count	MAX001	D5	
Deleted Record Count	DEL001	D3	

Field Descriptions

CUSNO - This is the unique key assigned to each customer. CUSNO - "999999" signifies a "Miscellaneous Customer" used in several applications, and should be entered by the user in addition to his regular customers.

NAME - Customer name. The first six characters of this field are set to "]]]DEL" by the CUSMNT program if the record has been deleted.

ADD1 - Address Line 1.

ADD2 - Address Line 2.

CITY - City

STATE - State. It is recommended that you use the standard Post Office abbreviations for the states.

ZIP - Zip Code. This is 10 digit alpha field to accommodate the new 9 digit U.S. Zip code.

PHONE - Phone Number.

SLSMAN - Salesman number. This is used to produce the Sales Analysis by Salesman Report. It is the default salesman number used in the Sales Entry application.

TERR - Territory. This is used to produce the Sales Analysis Report by Territory.

LOC - Inventory Location. This field is used by the Customer Order Processing (COP) package. It is the default inventory location used when entering orders for this customer.

CUSCD - Customer Type. This is used to produce the Sales Analysis Report by Customer Type. It is also used for various purposes within the MCBA Customer Order Processing (COP) package.

CUSCD2 - Customer Code-2. This is a second Customer Type field which is available for use at user's option. It is not currently used in the A/R package.

SLSMTD - Sales Dollars Month-to-Date. Once the value is initially entered in CUSMNT, it is kept up-to-date via the ACMSLS program (during sales posting), and should be set to zero at the end of the month using the CLRMAR program.

SLSYTD - Sales Dollars Year-to-Date. Similar to SLSMTD. It is set to zero at the end of the year by running the CLRYAR program.

CSTMDT - Cost Dollars Month-to-Date. This is the cost of sales accumulated from the cost figure entered in Sales Entry. Its handling is the same as SLSMTD.

CSTYTD - Cost Dollars Year-to-Date. Compares to SLSYTD.

NOTE: - The above four fields contain all the data that is used in printing the Sales Analysis Reports.

TAXFLG - Taxable Code. This field will either be blank or correspond to a valid Tax code in the A/R Tax Code file. Default tax amounts are calculated using the tax percentages referenced for this code.

CRDLMT - Credit Limit. This field is used in the ARTBAL and ARTBL2 programs to determine whether or not a customer has exceeded his credit limit, in which case a warning is printed next to his balance. It is used in conjunction with the OSTDCR field, by the COP package.

OSTDCR - Account Balance. This field is updated by the ACMCSH program when posting cash receipts. It is used in the COP package to display a warning message, when appropriate, when entering orders for this customer. It is set by the ARTBAL and ARTBL2 programs, and updated by the ACMSLS and ACMCSH programs.

TERMS - Terms. These are the customer's default terms. This field will contain a valid code from the Terms Code file (ARTERM).

BALMTH - Accounting Balance Method. See the A/R Glossary for an explanation of Open Item and Balance Forward accounting. This field is used extensively throughout the package.

STMFLG - Statement Flag. If this is "2" the STMENT program will not print a statement for this customer.

PRDCD - Product Code Array,

DISCNT - Discount Array - Thses two arrays are synchronized, and are set up by the Product Discount Maintenance application. The arrays give a correspondence between product categories used in the COP package, and the usual discount allowed to this customer for the product category.

NOTE: The CUSMAS file is normally set up to handle 10 Product code, discount pairs. A question is asked about this by the INITAR program. If you wish to have more than 10 such pairs, minor modifications must be made to the source code in every program that uses the CUSMAS file.

Additional Control Record Fields

PAD1 - This is a field put into the control record for the convenience of the user in case he wants to use more than 10 product discounts per customer. Originally it is the source code only as a comment, with field size nA4. If the user wishes, say, 14 discounts per customer, he would remove the ";" indicating a comment, and change "n" to "4".

DETDST - Detailed G/L Distribution Flag. This flag is set by INARGL if the user answers "Y" to the first G/L Distribution question. If this flag is "N", then no G/L distribution tracking or reporting will be done by the system at all.

MLTCSH - Multiple Cash Accounts. If the user typically uses more than one G/L account number for cash, this would be set to "Y". Then, in the Cash Entry & Editing application, add mode, the G/L distribution default screen will initially show "Y" to the Multiple Cash Accounts question. This can be changed to "N" before actually beginning the entry of the current batch of cash receipts transactions.

DEFCSH - Default Cash Account. This is the seven-digit default cash G/L number which will be used automatically by the Cash Entry & Editing application if the user has answered "N" to Multiple Cash Accounts. If the answer to Multiple Cash Accounts is "Y", this account number will display on the screen as the default, but can be overridden by the user if he wishes.

The remaining MLT--- and DEF---fields are used for default purposes in the Sales Entry & Editing, Cash Entry & Editing, and Calculate Finance Charges applications. Only MLTCSH and DEFCSH are described here. The other fields function in exactly similar ways.

NUMDSC - Number of Product Discounts. This is the size of the Product Discount array (PRDCD), which is normally 10.

TRPDCD - Terms Product Category. Entry to this field is via Terms Discount Maintenance in Customer Order Processing.

TDISC - Terms Discount Percentage. This is the line item discount percentage associated with the Terms Product Category, also entered via Terms Discount Maintenance in Customer Order Processing.

See the General File Definition section for an explanation of the remaining fields beginning with DFL001.

DETAIL SALES HISTORY FILE
(SLSHST)

Description: Detail Sales History file: contains one record per line item of every invoice generated by Order Entry and Credit Memo Entry. This file is sorted by Customer Number, Product Number, Invoice Date and may be the source file for detailed sales reports. This data is posted to the Sales Summary file and then purged.

File Status: History Record # in DEVICE.DDF: 55 Rec. Size: 77
+ End of Record

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Invoice Date	HINVDT	D6	
Invoice Number	HINVNO	D6	
Order Date	HORDDT	D6	
Back Order Flag	HBOFLG	D1	
Taxable Flag	HTXFLG	A1	
Customer Number	HCUSNO	D6	
Customer Type	HCUSCD	A2	
Customer Code 2	HCUSC2	A2	
Location	HLOC	A2	
Salesman Number	HSLSMN	D2	
Territory Code	HTERR	A2	
Item Number (End-Item Number)	HITMNO	A15	
Product Category	HPRDCD	A2	
Product Code 2	HPRDC2	A2	
Quantity Sold	HQTY	D5	
Net Price	HSALE	D8	

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Cost	HCOST	D8	
Status of Record	POSTED	D1	0=not yet posted 1=selected for posting but not yet posted 2=already posted

DETAIL SALES HISTORY WORK FILE
(SLHWRK)

Description: Detail Sales History Work File

File Status: Master Record # in DEVICE.DDF: 57 Rec. Size: 77
+ End of Record

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Invoice Date	WINVDT	D6	
Invoice Number	WINVND	D6	
Order Date	WORDDT	D6	
Back Order Flag	WBOFLG	D1	
Taxable Flag	WTXFLG	A1	
Customer Number	WCUSNO	D6	
Customer Type	WCUSCD	A2	
Customer Code 2	WCUSC2	A2	
Location	WLOC	A2	
Salesman Number	WSLSMN	D2	
Territory Code	WTERR	A2	
Item Number (End-Item Number)	WITMNO	A15	
Product Category	WPRDCD	A2	
Product Code 2	WPRDC2	A2	
Quantity Sold	WQTY	D5	
Net Price	WSALE	D8	
Cost	WCOST	D8	
Status of Record	WPSTED	D1	0 - Not Yet Posted

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
			1 - Selected for posting but not yet posted 2 - Already posted
** CONTROL RECORD **			
(Unused)		A57	
Delete Flag	DFL057	D1	
Sort Flag	SFL057	D1	
Organized Record Count	ORG057	D5	
Record Count	REC057	D5	
Maximum Record Count	MAX057	D5	
Deleted Record Count	DEL057	D3	

INVENTORY TRANSACTION FILE
(INVTRX)

Description: Inventory Transactions - receipts, transfers and issues. First record in file is control record. File sorted in transaction type, then Item Number sequence.

File Status: Transaction Record # in DEVICE.DDF: 43 Rec. Size: 116
+ End of Record

Is the first record of this file used as a control record? Yes.

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Item Number	RITMNO	A15	
Transaction Type	TRXTYP	D1	0=receiving, 1=transfer, 2=issue.
Item Description	RDESCR	A30	
Location Inventory Goes To	LOCTO	A2	Blanks for an issue. See Note 1.
Location Inventory Comes From	LOCFRM	A2	Blanks for a receipt. See Note 2.
New Location Established	NEWLOC	D1	1=new to-loc, 2=new from-loc, 3=both lines new
To-Location, Old On-Hand Quantity	TOOONH	D6	See Note 1.
To-Location, Old On-Order Quantity	TOOONO	D6	See Note 1.
From-Location, Old On-Hand Quantity	FROONH	D6	See Note 2.
From-Location, Old On-Order Quantity	FROONO	D6	See Note 2.
Old Average Unit Cost	OLDAVG	D9	$XXX,XXX.XXX See Note 3.
Quantity Received, Issued, etc.	QTYRCD	D5	

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
New Per Unit Cost	NEWCST	D8	$XXX,XXX.XX See Note 3.
New Average Unit Cost	NEWAVG	D9	$XXX,XXX.XX See Note 3.
Order Reference Number	PONUM	A9	See Note 3.
Order Complete ?	ORDCMP	A1	Y = Yes, N = No. See Note 3.
** CONTROL RECORD **			
(Unused)		A103	
Record Count	RECCNT	D5	
Maximum Record Count	MAXREC	D5	
(Unused)		A3	

NOTES:

1. These fields only for receivings or transfer transactions.
2. These fields only for issues or transfer transactions.
3. These fields used only for receiving transactions.
4. The SORT keys are TRXTYP, RITMNO (16 bytes).

ITEM INDEX FILE
(ITMIDX)

Description: Index for Item Master file (ITMMAS).

File Status: Master Record # in DEVICE.DDF: 42 Rec. Size: 22
+ End of Record

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Item Number	IITMNO	A15	
Relative Record Number in ITMMAS file	IRC041	D5	
Product Category	IPRCAT	A2	

NOTE:

While the first record in this file is not a control record, it is a blank record, to correspond with the control record of ITMMAS.

The SORT key is IITMNO (15 bytes).

This page intentionally left blank.

ITEM MASTER FILE
(ITMMAS)

Description: Item Master file. Record size varies with price, location, and vendor array dimensions. First record on file is a control record. File is in order in which items were entered (index is in item number sequence).

File Status: Master Record # in DEVICE.DDF: 41 Rec. Size: 533 + End of Record*

*See Notes 1, 2 and 3.

The first record is used as a control record.

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Item Number	ITEMNO	A15	
Item Description	DESCR	A30	
Product Category	PRDCAT	A2	
User Defined Code	USRDEF	A2	
Last Unit Cost	LSTCST	D8	$XXX,XXX.XX
Average Unit Cost	AVGCST	D9	$XXX,XXX.XXX
Target Margin	TGTMGN	D2	XX%
Price Code (Customer Type)	PRICCD	5A2	See Note 1.
Unit Price	PRICE	5D8	$XXX,XXX.XX See Note 1.
Location	LOC	5A2	See Note 2.
Quantity On-Hand	QTYONH	5D6	XXX,XXX- See Note 2.
Quantity Allocated	QTYCOM	5D6	XXX,XXX See Note 2.
Quantity On-Order	QTYONO	5D6	XXX,XXX- See Note 2.
Reorder Level	REOLVL	5D5	XXX,XXX See Note 2.
Order Up-To-Level	ORDUPT	5D6	See Note 2.

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Picking Sequence (Bin Number)	PIKSEQ	5A3	See Note 2.
Recommended Minimum Order Quantity	RECMIN	D5	XX,XXX
Economic Order Quantity	EOQ	D6	XXX,XXX
Average Monthly Usage	AVGUSE	D6	XXX,XXX
Usage Weighting Factor	USEWGT	D2	.XX
Safety Stock	SAFSTK	D5	XX,XXX
Safety Factor	SAFFAC	D2	X.X
Average Forecast Error	AVGERR	D5	XX,XXX
Sum of Forecast Errors	SUMERR	D5	XX,XXX-
Usage Filter	USEFLT	D2	X.X
Vendor Lead Time	LEADTM	D3	XX.X, in months
Vendor Weight	WEIGHT	D6	X,XXX.XX
Selling Unit of Measure	SUOFM	A2	
Purchase Unit of Measure	PUOFM	A2	
Purchase to Stock Conversion Factor	PSRAT	D8	XXXX.XXXX
Usage Month-to-Date	USEMTD	D6	XXX,XXX
Usage Year-to-Date	USEYTD	D6	XXX,XXX
Quantity Sold Month-to-Date	QTYMTD	D6	XXX,XXX-
Quantity Sold Year-to-Date	QTYYTD	D6	XXX,XXX-
Sales $ Month-to-Date	SLSMTD	D8	$XXX,XXX.XX-
Sales $ Year-to-Date	SLSYTD	D9	$X,XXX,XXX.XX-
Cost of Sales, Month-to-Date	CSTMTD	D8	$XXX,XXX.XX-
Costs of Sales, Year-to-Date	CSTYTD	D9	$X,XXX,XXX.XX-
Backorder Code	BOCODE	D1	0=backorder. 1-no backorder.

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Taxable Flag	TXFLAG	A1	Y = taxable. N = non-taxable.
Stock Status Flag	SSFLAG	D1	
Extra Flag	EXFLAG	A1	
Vendor	VENDOR	3A4	XXXX See Note 3.
Minimum Order from this Vendor	MINORD	3D5	XX,XXX See Note 3.
Order Multiple	ORDMLT	D3	XXX
Activity Flag	OBSFLG	A1	O=obsolete, A=active, F=forecasted.
Stocked Flag	STOKED	A1	S=stocked N=non-stocked.
Controlled Flag	CNTRLD	A1	C=controlled N=non-controlled.
Purchased or Manufactured Flag	PRCHCD	A1	P=purchased, M=manufactured.
Inventory Class	INVCLS	A1	A, B, or C: or user defined.
Cycle Count Code	CYCTCD	D1	User defined
Last Counted Date	LSTCNT	D6	MM/DD/YY.
Commodity Code	COMCOD	A4	
Low Level Code	LLCODE	D2	
Buyer/Analyst	BUYER	A2	
Engineering Drawing Release Number	DRWREL	A6	
Engineering Drawing Revision Number	DRWREV	A2	
Routing Release Number	RTEREL	A6	
Routing Revision Number	RTEREV	A2	

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Routing Number	RTENUM	A5	
Manufacturing Location	MFGLOC	A2	
Order Policy Code	ORDPOL	A1	
Planning Period	PLNPER	D3	
Planning Lead Time	PLNLT	D3	
Planning Order Multiple	PLNMLT	D4	
Undefined Code	UNDEF	A1	
LIFO Base Quantity	LIFOBQ	D6	
LIFO Base Cost	LIFOBP	D7	
(Unused)		A26	
** CONTROL RECORD **			
(Unused)		A75	
Reset Pointers in SPC Flag	SPCFLG	D1	1 = Reset, 0 = No Reset.
Right Justify Numeric Item Numbers	JSTIFY	D1	
Default Location	DFLTLO	A2	
(Prices Array)		5A10	See Note 1.
(Locations Array)		5A34	See Note 2.
(Vendors Array)		3A9	See Note 3.
Default Product Codes Array	DPRDCD	45A2	See Note 4.
Default Discounts Array	DDISC	45D2	See Note 4.
Type of Manufacturing System	TYPSYS	D1	See Note 5.
Dimension of Prices Array	NUMPRC	D2	
Dimension of Locations Array	NUMLOC	D2	
Dimension of Vendors Array	NUMVEN	D2	

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Delete Flag	DFL041	D1	
Sort Flag	SFL041	D1	
Organized Record Count	ORG041	D5	
Record Count	REC041	D5	
Maximum Record Count	MAX041	D5	
Deleted Record Count	DEL041	D3	

NOTES:

1. These two arrays must be of the same dimension. You may have 1-42 prices per item. Change these two arrays in all programs using them and recompile.
2. These seven arrays must be of the same dimension. You may have 1-99 locations per item. Change these seven arrays in all programs using them and recompile.
3. These two arrays must be of the same dimension.
4. These are referred to as "trade discounts".
5. 1 = Inventory Management only.

   2 = I/M and Customer Order Processing installed.

   3 = I/M and Bill of Material Processor installed.

   4 = I/M, BOMP, and COP installed.

This page intentionally left blank.

ORDER HEADER FILE
(ORDHDR)

Description: File is in Order Number Sequence

File Status: Master Record # in DEVICE.DDF: 44 Rec. Size: 396
+ End of Record

File Type: ISAM

Is the first record of this file used as a control record? No

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Order Number	OORDNO	D6	
Order Location	OLOC	A2	
Customer Type Code	OCUSCD	A2	
Order Date	OORDDT	D6	MM/DD/YY
Customer Number	OCUSNO	D6	
Customer Name	OCUSNM	A25	
Ship-to Number	OSHPTO	D4	
Ship-to-Name	OSHPNM	A30	
Ship-to-Address Line 1	OSHAD1	A30	
Ship-to Address Line 2	OSHAD2	A30	
Ship-to Address Line 3	OSHAD3	A30	
Customer's Purchase Order Number	OPONO	A10	
Job Number	OJOBNO	A10	
Collect or Prepaid Code	OCLPPD	D1	
Terms of Payment Code	OTERMS	D1	
Terms Discount Percentage	OTRMSD	D4	XX.XX%
Terms Discount Allowed	ODSCAL	D6	$X,XXX.XX
Ship Via Code	OSHVIA	A1	

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Invoice Number	OINVNO	D6	
Invoice Date	OINVDT	D6	MM/DD/YY
Amount of Sale	OSALE	D8	$XXX,XXX.XX
A/R Account Number	OARACT	D7	XXXX-XXX
Miscellaneous Charge Amount	OMISC	D6	$X,XXX.XX
Miscellaneous Charges Account Number	OMSACT	D7	XXXX-XXX
Sales Tax Amount	OTAX	3D7	$XX,XXX.XX
Sales Tax Account Number	OTXACT	3D7	XXXX-XXX
Freight Charges Amount	OFRGHT	D6	$X,XXX.XX
Freight Charges Account Number	OFRACT	D7	XXXX-XXX
Order Discount Percentage	ODISC	D2	XX%
Salesman Number	OSLMAN	D2	
Order Comments	OCOMNT	2A35	
Cost of Order	OCOST	D8	$XXX,XXX.XX
Commission Due	OCOMDU	D7	$XX,XXX.XX
Selected for Billing Flag	OFLAG	D1	0=Not Selected, 1=Selected, 2=Invoice printed
Order Type	ORDTYP	A1	
Order Sequence	ORDSEQ	D2	
(Unused)		A4	

ORDER LINE ITEM FILE
(ORDLIN)

Description: File is in Order Number, then Line Sequence Order

File Status: Master Record # in DEVICE.DDF: 45 Rec. Size: 122
+ End of Record

File Type: ISAM

Is the first record of this file used as a control record? No

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Order Number	LORDNO	D6	
Line Print Sequence	LINSEQ	D2	
Item Number	LITMNO	A15	
Item Description	LDESCR	A30	
Quantity Ordered	LQTYOR	D4	
Quantity Shipped	LQTYSH	D4	
Quantity Back Ordered	LQTYBO	D4	
Location	LLOC	A2	
Product Category	LPRDCD	A2	
Unit of Measure	LUOFM	A2	
Price	LPRICE	D7	$XX,XXX.XX
Order Discount	LODISC	D2	XX%
Line Discount	LDISC	D2	XX%
Line Status	LSTATS	D1	1=Backorder Status, B/Oable item, 2=Backorder Status, Non-B/Oable item.
Picking Sequence (Bin Number)	LPICSQ	A3	

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Unit Cost	LCOST	D7	$XX,XXX.XX
Select for Billling Flag	LFLAG	D1	0=Not Selected 1=Selected, 2=Invoiced, 3=Remove from File
Item Weight	LITMWT	D6	X,XXX.XX
Item Tax Flag	LTXFLG	D1	0=Non-Taxable, 1=Taxable
Stocked Item Flag	LSTOKT	A1	S=Stocked, Blank=Non-Stocked
Expected Ship Date (Request Date)	LEXSDT	D6	YYMMDD
Promise Date	LPRMDT	D6	YYMMDD
(Unused)		A8	

PICKING TICKET INDEX FILE
(LINIDX)

Description: Picking Ticket Index

File Status: Index Record # in DEVICE.DDF: 48 Rec. Size: 11
+ End of Record

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Order Number	LXORDN	D6	
Picking Sequence	LXPIKS	A3	
Line Sequence	LXLSEQ	A2	

This page intentionally left blank.

PRODUCT CATEGORY ACCOUNT FILE
(PRDACT)

Description: Product Category Account file. Associates a sales account number with a product category. It is used by the POSTAR and CRMAR programs when posting invoices and credit memos to the A/R package, to obtain the appropriate sales distribution account numbers.

File Status: Master Record # in DEVICE.DDF: 69 Rec. Size: 39
+ End of Record

The first record of this file is a standard control record.

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Product Category	PRDCAT	A2	
Sales Account Number	PACTNO	D7	XXXX-XXX
Sales Account Description	PACTDS	A30	

This page intentionally left blank.

PRODUCT CATEGORY INDEX FILE
(SAPIDX)

Description: File is sorted in Product Category, then Item Number sequence

File Status: Work Record # in DEVICE.DDF: 49 Rec. Size: 22
+ End of Record

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Item Number	SITMNO	A15	
Record Number in ITMMAS file	SRECNO	D5	
Product Category	SPRCAT	A2	

NOTES:

1. First record in file is a blank record, corresponding to the first blank record in ITMIDX.

This page intentionally left blank.

PRODUCT STRUCTURE FILE
(PRDSTR)

Description: This file contains product structures. Each record contains a parent item number and one of its components.

File Status: Master Record # in DEVICE.DDF: 91 Rec. Size: 62
+ End of Record

The first record of this file is used as a control record.

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Parent Item Number	PITMNO	A15	
Sequence Number	SEQNUM	D2	
Component Item Number	CITMNO	A15	
Quantity-Per Parent	QTYPER	D7	XXX.XXXX
Operation where Component Attaches to Parent	ATCHOP	A2	
Shrinkage/Scrap Factor	SCRFAC	D4	XXX.X%
Activity Flag	OBSFLG	A1	O=obsolete A=active F=forecasted or planned
Parent Address in ITMMAS File	PRECNO	D5	
Component Address in ITMMAS File	CRECNO	D5	
User Defined	EFFECT	D6	
** CONTROL RECORD **			
(Unused)		A43	
Where Used Flag	WUFLAG	D1	
Organized Record Count	ORG091	D5	
Record Count	REC091	D5	
Maximum Record Count	MAX091	D5	

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Deleted Record Count	DEL091	D3	

NOTE:

Position 18 through 23 = ']]]DEL' for a deleted record.

Field Description

PITMNO - Parent Item Number. It must exist in the Item Master file before it can be entered into PRDSTR. PITMNO is the primary sort field used when sorting or listing Product Structure (Sequence Number is secondary).

SEQNUM - Sequence Number. It can be used to specify the component sequence (within the above parent) when sorting or listing this particular structure.

CITMNO - Component Item Number. It must exist in the Item Master file before it can be entered into PRDSTR. Components are not part of the sort and will be listed as entered unless the associated SEQNUM is non-blank.

QTYPER - Quantity of this component per one of this parent. A negative quantity-per is usually only used for modular bills. The four decimal places can only be used for components coded in the Item Master as non-controlled.

ATCHOP - The operation, in this parent's routing, where this component is first used or attached.

SCRFAC - The percentage of this component that is anticipated to be lost due to scrap, shrinkage, etc.

OBSFLG - Activity Flag. Normally, a structure is active (A). It can be obsolete (O) so as not to be used in any new cycle of action nor to be deleted. Lastly, a structure can be forecast (F), (used by MRP); also referred to as a planning bill.

PRECNO - The relative record number, in the Item Master file, of this parent.

CRECNO - The relative record number, in the Item Master file, of this component.

EFFECT - User defined because no function is coded to use it as is. The user may want to specify an effectivity date (from or to) for this parent-component relationship and then write his own code (or modification) to use it.

PRODUCT STRUCTURE INDEX FILE
(PRDIDX)

Description: File is sorted in component item number, then relative record number sequence.

File Status: Index Record # in DEVICE.DDF: 92 Rec. Size: 20
+ End of Record

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Component's Item Number	CMPITM	A15	
Relative Record Number of this Structure in PRDSTR File	IRC091	D5	

NOTES:

The first record in the file is not a control record, but it is blank to correspond to the control record in PRDSTR.

A component will have as many PRDIDX records as it has parents, i.e., this is the Where-Used file.

Field Description

CMPITM - Component item number that exists in the Item Master file and is also part of one or more structures.

IRC091 - The relative record number of a PRDSTR record which contains a relationship between this component and a parent item.

This page intentionally left blank.

SALESMAN FILE
(SALMAN)

Description: Salesman File. This is a non-standard file containing information about the user's salesmen. It always has exactly 99 records, with the Salesman Number field always pre-set to the relative record number, no matter how many salemen have actually been entered. The remainder of the record is pre-extended with right brackets until the salesman is actually entered in the program SALMNT.

File Status: Permanent Master* Record # in DEVICE.DDF: 54 Rec. Size: 151
+ End of Record

* Non-standard

The first record of the file is a data record, no a control record.

FIELD	FIELD NAME	TYPE/ SIZE	FORMAT
Salesman Number	SLSNO	D3	
Salesman Name	SLSNM	A25	
Address Line 1	SLSAD1	A25	
Address Line 2	SLSAD2	A21	
City	SLSCTY	A15	
State	SLSST	A2	
Zip Code	SLSZIP	A10	XXXXX-XXXX
Phone Number	SLSTNO	D10	XXX-XXX-XXXX
Customer Type Array	SLSCST	8A2	
Commission Percent Array	SLSCOM	8D3	XX.X%

Field Descriptions

SLSNO - Salesman Number. This is always pre-set to the relative record number of this record in the file.

SLSNM - Salesman Name. A deleted record is marked with "]]]DEL" in the first six characters of this field.

SLSAD1 - Address Line 1.

SLSAD2 - Address Line 2.

SLSCTY - City.

SLSST - State.

SLSZIP - Zip Code.

SLSTNO - Telephone Number.

SLSCST - Customer Type Array. Used in conjunction with the next array to associate customer types (Customer codes) with usual commission percents. This is used in the COP package to automatically calculate the commission due on an invoice.

SLSCOM - Commission Percent Array. Used in conjunction with the above array.

SALES SUMMARY FILE
(SLSSUM)

Description: Sales Summary file: contains one record for each Customer/Item combination that has had sales activity during the past year. Records in this file are in random sequence, and are normally accessed via the SLSIDX file.

File Status: History Record # in DEVICE.DDF: 58 Rec.Size: 351
+ End of Record

Is the first record of this file used as a control record? Yes

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
** SUMMARY RECORD **			
Customer Number	SSCUS	D6	
Item Number	SSPROD	A15	
Current Period Sales	SSSALE	D8	
Current Period Units Sold	SSUNIT	D6	
Current Period Cost	SSCOST	D8	
Last Period Sales	SSLSTS	D8	
Last Period Units Sold	SSLSTU	D6	
Last Period Cost	SSLSTC	D8	
Year-to-Date Sales by Month	SSSYTD	13D8	
Year-to-Date Units Sold by Month	SSUYTD	13D6	
Year-to-Date Costs by Month	SSCYTD	13D8	
** CONTROL RECORD **			
(Unused)		A21	
Current Year	SSYEAR	D4	(e.g. 1981)
Current Period	SSPER	D2	

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Reporting Period Descriptions	SSDESC	13A10	
		A176	
Organized Record Count	ORG058	D5	
Record Count	REC058	D5	
Maximum Record Count	MAX058	D5	
Deleted Record Count	DEL058	D3	

NOTES ON SALES SUMMARY FILE

Definition of variables:

SSSALE, SSUNIT, and SSCOST contain the current period sales, units and costs. Current period sales history figures on the reports are taken directly from these variables.

SSLSTS, SSLSTU, and SSCSTC contain the last period (i.e., last month) sales, units, and costs. These variables are used for certain Year-to-Date calculations.

SSSYTD is a 13 X 8 array
SSUTYD is a 13 X 6 array
SSCYTD is a 13 X 8 array

Most users will use 12 of these "buckets", one for each month. Some users (those with 13 sales periods per year) will use all 13 buckets. These buckets do not shift, rather one bucket is written over during a month-end shift or a year-end shift.

Here is what a 12-period array would look like in late June after all sales had been entered for June and when the June sales reports are about to be run:

1	2	3	4	5	6	7	8	9	10	11	12
JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
80	80	80	80	80	79	79	79	79	79	79	79
YTD	YTD	YTD	YTD	YTD	YTD	YTD	YTD	YTD	YTD	YTD	YTD

After all June Sales reports have been run, a month-end shift is run. SSPER is then set to 7 and all new sales are posted to July. Here is what the array would now look like:

1	2	3	4	5	6	7	8	9	10	11	12
JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
80	80	80	80	80	80	79	79	79	79	79	79
YTD	YTD	YTD	YTD	YTD	YTD	YTD	YTD	YTD	YTD	YTD	YTD

All reports which print month-to-month sales histoy comparisons use the various fields fields in the SALES SUMMARY FILE in the following way:

1) If the current period (SSPER) = 1)

This period last year sales = SSSYTD(1)
This period last year units = SSUYTD(1)
This period last year costs = SSCYTD(1)

This period this year sales = SSSALE
This period this year units = SSUNIT
This period this year costs = SSCOST

This period last year Year-to-Date sales = SSSYTD(1)
This period last year Year-to-Date units = SSUTYD(1)
This period last year Year-to-Date costs = SSCYTD(1)

This period this year Year-to-Date sales = SSSALE
This period this year Year-to-Date units = SSUNIT
This period this year Year-to-Date costs = SSCOST

2) (If the current period (SSPER) = 2)

This period last year sales = SSSYTD(2) - SSSYTD(1)
This period last year units = SSUYTD(2) - SSUYTD(1)
This period last year costs = SSCYTD(2) - SSCYTD(1)

This period this year sales = SSSALE
This period this year units = SSUNIT
This period this year costs = SSCOST

This period last year Year-to-Date sales = SSSYTD(2)
This period last year Year-to-Date units = SSUYTD(2)
This period last year Year-to-Date costs = SSCYTD(2)

This period this year Year-to-Date sales = SSLSTS + SSSALE
This period this year Year-to-Date units = SSLSTU + SSUNIT
This period this year Year-to-Date costs = SSLSTC + SSCOST

3) (If the current period (SSPER) is greater than 2)

This period last year sales = SSSYTD(SSPER) - SSSYTD(SSPER-1)
This period last year units = SSUYTD(SSPER) - SSSUYTD(SSPER-1)
This period last year costs = SSCYTD(SSPER) - SSCYTD(SSPER-1)

This period this year sales = SSALE
This period this year units = SSUNIT
This period this year costs = SSCOST

This period last year Year-to-Date sales = SSSYTD(SSPER)
This period last year Year-to-Date units = SSUYTD(SSPER)
This period last year Year-to-Date costs = SSCOST(SSPER)

This period this year Year-to-Date sales = SSSYTD(SSPER-2)+SSLSTS+SSSALE
This period this year Year-to-Date units = SSUYTD(SSPER-2)+SSLSTU+SSUNIT
This period this year Year-to-Date costs = SSCYTD(SSPER-2)+SSLSTC+SSCOST

% MRG is always calculated by taking the appropriate sales and costs figures and plugging them into the following formula:

$$\%\ MRG = \frac{SALES - COSTS}{SALES} \times 100$$

SALES SUMMARY INDEX FILE
(SLSIDX)

Description: Sales Summary Index: is in one-one correspondence with the Sales Summary file. Sort sequence is Customer Number/Item Number. Other fields in this file are used in selection for reporting. Sources of the selection fields are the current Item Master and Customer Master*.

File Status: Work Record # in DEVICE.DDF: 59 Rec. Size: 42
+ End of Record

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
** SUMMARY INDEX **			
Customer Number	SICUS	D6	
Item Number	SIPROD	A15	
Customer Type	SICCD	A2	See Note.
Customer Code 2	SICCD2	A2	See Note.
Product Category	SIPCD	A2	See Note.
Product Code 2	SIPCD2	A2	See Note.
Territory	SITERR	A2	See Note.
Salesman	SISMAN	D2	See Note.
Location	SILOC	A2	See Note.
Status	SISTAT	A2	See Note.
Record Number	IRC058	D5	

NOTES:

The updates of these fields happen at posting time. If the Item Master or Customer Master no longer exists, the last values are retained.

This page intentionally left blank.

SALES TRANSACTION FILE
(SALESO)

Description: Sales Transaction File. This file is used solely by the Sales Entry and Editing application. It holds all sales transactions from the time that they are entered until they are finally posted to the AROPEN file, at which point this file is cleared to one control record.

File Status: Transaction Record # in DEVICE.DDF: 04 Rec. Size: 299 + End of Record

The first record of the file is used as a control record.

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Document Number	SDOCNO	D6	
Document Type	SDOCTP	D1	
Document Date	SDOCDT	D6	MMDDYY
Customer Number	SCUSNO	D6	
Customer Name	SNAME	A25	
Salesman	SSLMAN	D2	
Document Due Date	SDDCDU	D6	MMDDYY
Sale Amount	SSLAMT	D8	$XXX,XXX.XX
A/R Account Number	SARACT	D7	XXXX-XXX
Other Charges Amount	SMISC	D6	$X,XXX.XX
Other Charges Account Number	SMSACT	D7	XXXX-XXX
Tax Amount	STAX	3D7	$XX,XXX.XX
Tax Account Number	STXACT	3D7	XXXX-XXX
Freight Amount	SFRGHT	D6	$X,XXX.XX
Freight Account Number	SFRACT	D7	XXXX-XXX
Default Discount Amount	SDISAL	D6	$X,XXX.XX
Apply-to Document Number	SAPLNO	D6	

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Cost Amount	SCOST	D8	$XXX,XXX.XX
Commission Amount	SCOMM	D7	$XX,XXX.XX
Sales Account Number Array	SDACTS	9D7	XXXX-XXX
Sales Distribution Amount Array	SDAMTS	9D8	$XXX,XXX.XX
External Flag	SEXFLG	A1	
Posting Flag	SPSTFL	D1	
** CONTROL RECORD **			
(Unused)		A228	
Detailed G/L Distribution Flag	SDETDS	A1	Y/N
Multiple A/R Accounts ?	SMLAR	A1	Y/N
Default A/R Account	SARAC	D7	XXXX-XXX
Multiple Sales Accounts ?	SMLSLS	A1	Y/N
Default Sales Account	SSLSAC	D7	XXXX-XXX
Multiple Other Charges Accounts ?	SMLOCH	A1	Y/N
Default Other Charges Account	AOCHAC	D7	XXXX-XXX
Multiple Tax Accounts ?	SMLTAX	A1	Y/N
Default Tax Account	STAXAC	D7	XXXX-XXX
Multiple Freight Accounts ?	SMLFRT	A1	Y/N
Default Freight Account	SERTAC	D7	XXXX-XXX
Default Finance Charges Account	SFCHAC	D7	XXXX-XXX
(Unused)		A5	
Organized Record Count	ORG004	D5	
Record Count	REC004	D5	
Maximum Record Count	MAX004	D5	
Deleted Record Count	DEL004	D3	

Field Descriptions

SDOCNO - Document Number. This is the identifying number of the transaction (invoice number for a sale, etc.).

SDOCTP - Document Type. See ADOCTP for the AROPEN file. Document Types 1, 3, 4 and 5 may be entered via the Sales Entry and Editing application; document type 2 is reserved for the Cash Receipts Entry and Editing application, which uses the CASH file.

SDOCDT - Document Date. This is stored in MMDDYY format.

SCUSNO - Customer Number.

SNAME - Customer Name. This is taken automatically from the Customer record. unless SCUSNO = "999999" in which case the user enters the name directly. A deleted Sales Transaction record is flagged with "000000" in the first 6 positions of this field.

SSLMAN - Salesman Number.

SDOCDU - Document Due Date. This is the date that the invoice is due. It is calculated using the customer's term code as defined in the Terms Code Maintenance application.

SSLAMT - Sale Amount. This is the main invoice, CR memo, finance charge or DR memo amount. For a CR memo, if the amount is positive, it represents a positive credit.

SARACT - A/R Account Number. This is the A/R account number to debit for this document.

SMISC - Other Charges. For example, special handling.

SMSACT - Other Charges Account Number.

STAX - Sales Tax. Up to three individual sales tax amounts can be used. These correspond to the 3 tax percentages entered per the Tax code.

STXACT - Sales Tax Account Number. Up to three individual Sales Tax account numbers can be specified. These correspond to the 3 G/L account numbers entered per the Tax code.

SFRGHT - Freight Charge.

SFRACT - Freight Charges Account Number.

NOTE: The above three quantities are added together and posted to the AOTHER field of the AROPEN record, by the PSTSLS program.

SDISAL - Default Discount Amount. This field is transferred from the COP package when invoices are posted to Accounts Receivable. It cannot be entered directly in A/R. It goes into the ADISC field of the AROPEN record.

SAPLNO - Apply-to Number. See A/R Glossary for a detailed explanation. For a Balance Forward customer this is always -1.

SCOST - Cost Amount. This is the cost of the sale, and is used to update the CSTMTD and CSTYTD fields of the Customer Master record, by the ACMSLS program.

SCOMM - Commission Amount. This commission will be posted to the COMDUE file, for the Salesman (SSLMAN) given on this document.

SDACTS - Sale Accopunt Number Array. This holds the account numbers for the sales distributions for this document.

SDAMTS - Sales Distribution Amount Array. These are the sales distribution amounts corresponding to the accounts in SDACTS.

SEXFLG - Extra Flag. Not used at this time.

SPSTFL - Posting Flag. Not used at this time.

Additional Control Record Fields

SDETDS - Detailed G/L Distribution Flag. This is similar to the DETDST flag in the control record of the CUSMAS file, and should always have the same value. If INITAR is run to recreate the SALES file alone, INARGL should be run and then ended immediately, in order to reset this flag to its proper value.

The SML --- and S---AC fields serve a similar purpose to the corresponding fields in the CUSMAS control record, but they are the defaults for the current run of sales transactions only. Once the current sales transactions are posted, the G/L distribution defaults revert back to the system-wide defaults stored in the CUSMAS file.

ORG004, REC004 and MAX004 are described in the General File Definition Data section.

SAVE DETAIL SALES HISTORY FILE
(SSVDSH)

Description: This is an optional file which is used if the user selects (as an installation option) to save the Purge Detail Sales History (see Note 1). The file is written over on each purge run, and contains records in the same format as the SLSHST file.

File Status: History Record # in Device.DDF: 88 Rec. Size: 77
+ End of Record

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT

** OPTIONAL FILE **

Record layout is identical to
SLSHST file

NOTES:

1. Because of the variable requirements of users, it is the installer's requirement to set up a procedure to permanently store this data after the purge process completes.

This page intentionally left blank.

SHIP-TO CODE FILE
(SHIPTO)

Description: Ship-to Code File

File Status: Master Record # in DEVICE.DDF: 171 Rec. Size: 133
+ End of Record

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Customer Number	SHCSNO	D6	
Ship-to Number	SHTONO	D4	
Ship-to Name	SHTONA	A30	
Ship-to Address	SHTOAD	3A30	
Ship-to Tax Code	SHTOTC	A3	
** CONTROL RECORD **			
(Unused)		A115	
Organized Record Count	ORG171	D5	
Record Count	REC171	D5	
Maximum Record Count	MAX171	D5	
Deleted Record Count	DEL171	D3	

This page intentionally left blank.

SHIP VIA CODE FILE
(SHPVIA)

Description: Ship Via Code File

File Status: Master Record # in DEVICE.DDF: 172 Rec. Size: 20
+ End of Record

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Ship Via Code	SHPVCD	A1	
Ship Via Description	SHPVDS	A15	
(Unused)		A4	
** CONTROL RECORD **			
(Unused)		A2	
Organized Record Count	ORG172	D5	
Record Count	REC172	D5	
Maximum Record Count	MAX172	D5	
Deleted Record Count	DEL172	D3	

This page intentionally left blank.

SHOP ORDER FILE
(SHPORD)

Description: Shop Order File - Header Record (type 1). There are four different types of records in this file. The header records are directly indexed through the SHPIDX file. Other record types sequentially follow the header record in order by key: Location, Shop Order Number, Operation Number, Sequence Number, Record Type. The keys are common to all records in the Shop Order file.

File Status: Master Record # in DEVICE.DDF: 101 Rec. Size: 170
+ End of Record

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
** HEADER RECORD **			
Location of Shop Order (Plant)	SOLOCN	A2	
Shop Order Number	SONUMB	A9	See Note 1.
Operation Number	SOOPNO	D2	
Sequence Number	SOSEQ	D2	(Blanks for header record)
Record Type	SORECT	D1	= 1 for header record.
Shop Order Header Status Code	SOSTS	A1	
Item Being Built (Parent Item)	SOITEM	A15	
Buyer or Analyst	SOBA	A2	
Last Transaction Posting Date	SOTRXD	D6	MMDDYY.
Quantity Ordered (of Parent)	SOQTY	D6	
Quantity Completed (of Parent)	SOQTYC	D6	
Shop Order Start Date	SOSDTE	D6	MMDDYY.
Shop Order Due Date	SODDTE	D6	MMDDYY.
Unit of Measure	SOUM	A2	
Standard Accumulated Man-Hours	SOACCS	D8	XXX,XXX.XX

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Actual Accumulated Man-Hours	SOACCA	D8	XXX,XXX.XX
Order Type	SOTYPE	A1	"B" = Base, "T" = Trial, "P" = Productive.
Shop Order Reference Number	SOREF	A8	
Description of Item Being Built	SOIDES	A30	
Engineering Release Number	SOERNO	A6	
Engineering Revision Number	SODREV	A2	
Routing Release Number	SORTNO	A6	
Routing Revision Number	SORREV	A2	
Lead Time (In Months)	SOLEAD	D3	XX.X
ABC Code (for Inventory Value)	SOABC	A1	
Stocked/Non-Stocked Flag	SOSTOK	A1	"S" = Stocked (default), "N" = Non-stocked
Controlled/Non-Controlled Flag	SOCNTL	A1	"C" = Controlled (default), "N" = Non-Controlled.
Job Number	SOJOB	A6	
Job Sequence Number	SOJOBS	A4	
Shop Order Completion Date	SOCDTE	D6	MMDDYY.
Scheduling Method	SOMTHD	A1	"F" = Forward (default), "B" = Backward, "M" = Manually.
(Unused)	SOCNTP	A1	
(Unused)		A9	
** OPERATION RECORD **			
Location (i.e. Plant) of Shop Order	OPLOCN	A2	

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Shop Order Number	OPNUMB	A9	See SOEDT for format.
Operation Sequence Number	OPOPNO	D2	
Sequence Number	OPSEQ	D2	00 for operations.
Record Type	OPRECT	D1	2 for operations.
Operation Status Code	OPSTS	A1	
Quantity Ordered at Operation	OPQTYO	D7	
Quantity Completed at Operation	OPQTYC	D7	
Quantity Rejected at Operation	OPQTYR	D7	
Quantity Scrapped at Operation	OPQTYS	D7	
Operation Start Date	OPSDTE	D6	MMDDYY.
Operation Due Date	OPDDTE	D6	MMDDYY.
Operation Unit of Measure	OPUM	A2	
Standard Value Added Man-Hours Per Piece	OPHRS	D8	XXXX.XXXX.
Actual Value Added Man-Hours (Total)	OPHRA	D8	XXXXXXX.X.
Labor Grade	OPLG	A2	
Operation Type	OPOPTP	A1	
Operation Sub-Type	OPSUBT	A1	
Operation Reference Number	OPREF	A8	
Operation Description	OPDESC	A30	
Queue Lead Time on Entry to this W/C	OPLEAD	D3	XX.X Days.
Count Point Flag	OPCPOP	A1	"M" = Material Relief, "Y" = Final Count Point.
Operation Priority	OPPRTY	D3	
Department	OPDEPT	A2	

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Work Center	OPWC	A2	
Standard Value Added Machine-Hours Per Piece	OPMCHS	D6	XX.XXXX.
Actual Value Added Machine-Hours (Total)	OPMCHA	D6	XXXX.XX.
Machine Number	OPMHNO	A4	
User Defined Code	OPUSER	A4	
Number of Men	OPMEN	D2	XX.
Work Center Load Code	OPWCLD	A1	"1" = Add time once per order, "2" = Add time once per piece.
Setup Code	OPSET	A1	Used to accumulate like setups.
Estimated or Standard	OPESTH	A1	
(Unused)		A15	
** MATERIALS RECORD **			
Location (i.e. Plant) of Shop Order	COLOCN	A2	
Shop Order Number	CONUMB	A9	See SOEDT for format.
Operation Number Material is Used at	COOPNO	D2	"00" = Posted to S/O header, Any Other = Posted to that operation.
Material Sequence Number	COSEQ	D2	
Record Type	CORECT	D1	3 for materials.
Material Status Code	COSTS	A1	
Material Item Being Used (Component)	COITEM	A15	
Need Date of Material	CONDTE	D6	MMDDYY.
Quantity of Component Per Parent	COQTYP	D7	XXXX.XXX.

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Scrap Factor	COSCRP	D3	Not used.
Component's Unit of Measure	COUM	A2	
Component's Buyer or Analyst	COBA	A2	
Component's Description	CODESC	A30	
Component Reference Number	COREF	A8	
Stocked/Nonstocked Flag (S/N) for Component	COSTK	A1	
Controlled/Noncontrolled Flag (C/N) for Component	COCNTL	A1	
Bin Location (Where Found in Plant)	COPRIM	A3	
Is this Item a Substitute for Another	COSUB	A1	"Y" = Yes "N" = No.
Quantity Used (i.e. Issued)	COQTYU	D6	
Material Code	COMCD	A2	
Department Number Where Material Joins Parent	COOPDP	A2	
Work Center Number Where Material Joins Parent	COOPWC	A2	
Part Substituted For	COSUB4	A15	
Remaining Quantity Allocated	COQTYA	D6	
(Unused)		A41	
** NOTE RECORD **			
Location (i.e. Plant) of Shop Order	NOLOCN	A2	
Shop Order Number	NONUMB	A9	
Operation Number that Note is Tied to	NOOPNO	D2	
Note Sequence Number	NOSEQ	D2	
Record Type	NORECT	D1	4 for notes.

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
"X" Indicates Cancelled	NOSTS	A1	
1st Line of Note	NONTE1	A30	
2nd Line of Note	NONTE2	A30	
3rd Line of Note	NONTE3	A30	
4th Line of Note	NONTE4	A30	
(Unused)		A33	
** CONTROL RECORD**			
(Unused)		A128	
Does System Interface to Standard Product Routing	SRFLAG	D1	"0" = No "1" = Yes
If I/M Installed, Handle Issues Thru I/M ?	IMISSU	D1	"0" = No "1" = Yes
Are Material Issues Automatic at Count-Point Operation ?	SFISFL	D1	"0" = No "1" = Yes.
Does System Interface to Inventory Management ?	IMFLAG	D1	"0" = No "1" = Yes.
Does System Interface to Job Cost ?	JCFLAG	D1	"0" = No "1" = Yes.
Default Location	SHPLOC	A2	(User Defined)
Delete Flag (unused)	DFL101	D1	
Sort Flag (unused)	SFL101	D1	
SHPIDX Organized Record Count	ORG102	D5	
SHPIDX Record Count	REC102	D5	
SHPIDX Maximum Record Count	MAX102	D5	

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Organized Record Count	ORG101	D5	
Record Count	REC101	D5	
Maximum Record Count	MAX101	D5	
Deleted Record Count	DEL101	D3	

NOTES:

1. Shop order number is formatted by the SOEDT subroutine, which is designed to be easily modified by the end user if desired. Initial format is straight A9 field.

This page intentionally left blank.

SHOP ORDER INDEX FILE
(SHPIDX)

Description: Shop Order Index File - partial index to Shop Order file. Each shop order has one index record which points to the Shop Order Header record. All detail records are then accessed sequentially following the header record.

File Status: Master Record # in DEVICE.DDF: 102 Rec. Size: 16
+ End of Record

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Location of Shop Order (Plant)	ISOLOC	A2	
Shop Order Number	ISONO	A9	
Relative Record Number of Shop Order Header in Shop Order File	ISOREC	D5	
** CONTROL RECORD **			
(Unused)		A11	
(Unused)		D5	

This page intentionally left blank.

TERMS CODES FILE
(ARTERM)

Description: This is the Terms Codes file.

File Status: Permanent Record # in DEVICE.DDF: 170 Rec. Size: 26
+ End of Record

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Term Code	ARTRCD	A1	
Term Description	ARTRDS	A15	
Due Days	ARTRDD	D3	
Discount Days	ARTRDY	D3	
Discount Percent	ARTRDP	D4	XX.XX%
** CONTROL RECORDS **			
(Unused)		A8	
Organized Record Count	ORG170	D5	
Record Count	REC170	D5	
Maximum Record Count	MAX170	D5	
Delete Record Count	DEL170	D3	

This page intentionally left blank.