MCBA®

Mini-Computer Business Applications Inc.

Digital Equipment Corporation

MCBA®
2441 Honolulu Avenue, Montrose, California 91020

SOFTWARE REFERENCE MANUAL

INVENTORY MANAGEMENT

DIBOL, RELEASE 7

SEPTEMBER, 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).

Checked by _____ Date 10 Jun 88

HOW TO USE THIS MANUAL

This User's Manual is organized into six sections:

1. Introduction
2. Getting Started
3. Operations Reference Section
4. User Instructions
5. Glossary
6. Index

At the front of this manual there is a Table of Contents listing all six sections. Subsections are listed under sections, and where the subsections are broken up into separate documents (in Sections 2 and 4) these documents are listed under the subsection title.

Page numbers are of the form:

S.ss.pp

where "S" is the section number (1 through 6),
"ss" is the sub-section number within a section, and
"pp" is the page number within a sub-section.

This manual assumes that executable (runnable) program files for this package are available to you. If you received source code, then you will have received a separate "Software Reference Manual" which contains the information relating to compiling source code. Given executable program files, the "Getting Started" section guides the user through installation and loading initial data into the data files.

This User's Manual was written with three categories of users in mind: the Installer, the Operations Manager, and the Data Processing Operator. We assume the Installer will make all necessary assignments at the operating system level, and, if this is the first MCBA package installed on this system, the Installer will set up the MCBA environment for all MCBA packages.

The Operations Manager is the person with a thorough knowledge of the operation of this MCBA package. The Operations Manager sets up initial data for the package and then manages the continuing operation of the package after it has been put into production.

The Data Processing Operator routinely enters data and produces and distributes reports. The portions of this manual useful to the Data Processing Operator are (1) the Introduction section, (2) the Getting Started subsection on "Using the Package", (4) the User Instructions section, (5) the Glossary, and (6) the Index.

When you first receive an MCBA package, all users should read Section 1, Introduction. The Installer should then work through the Getting Started subsection on "Installing the Package".

The Operations Manager should then load the initial package data using the Getting Started subsection on "Setting Up the Package".

The Getting Started subsection on "Using the Package" is for both the Operations Manager and the Data Processing Operator. It contains MCBA's system menu and package menu documentation and a document entitled "Periodic Processing" (if applicable) which outlines processing done daily, weekly, monthly, etc.

When all instructions in the Getting Started section have been accomplished, the remaining sections of the User's Manual serve as a reference guide to support continuing operations. Section 3 is the Operations Reference Section for the Operations Manager. Section 4 provides details for the operation of each application within the package.

In section 4, User Instructions, subsections are numbered to correspond with the selection numbers of applications as listed on the package menu (except that applications on the second screen of the menu have subsection numbers that follow in sequence after subsection numbers for first-screen applications).

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

Application Overview,
Run Instructions,
Screen Field Descriptions,
Sample Screens,
Error Recovery Procedures,
Report Field Descriptions,
Sample Reports, and
File Load Sheets.

Not all of these documents will appear in every application sub-section.

Section 5 is the glossary containing definitions of technical terms used in package and application documentation.

Section 6 is the index for this manual. The index is extremely useful for locating documentation on particular subjects, on data fields, or on files used by the package.

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
```

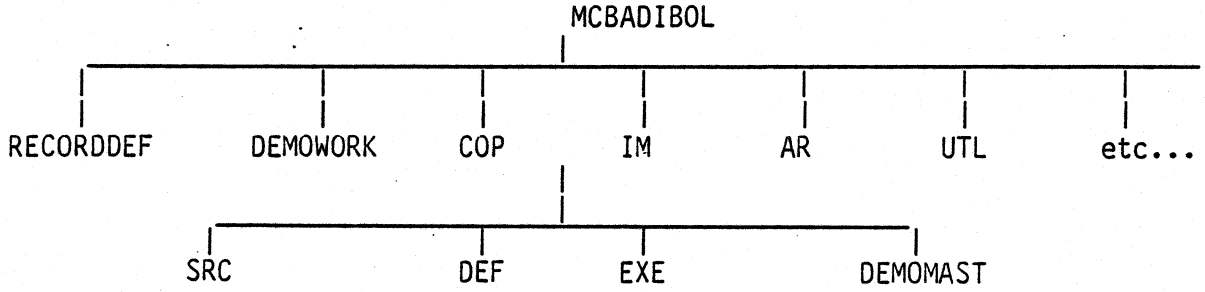
This should be inserted into the system manager's "SYSTARTUP.COM" file.

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 UTIL2O.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 UTIL2O.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.)

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 MTO: 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 .TMI 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 DRO: as the drive you originally assigned to OUT. If your drive is different, be sure to substitute its name whenever DRO: 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 DRO:[201,1]IN
ADD LOGICAL DRO:[201,1]OUT
^Z
```

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

```
ASSIGN/SYSTEM DRO:[201,1] IN
ASSIGN/SYSTEM DRO:[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 - INVENTORY MANAGEMENT, 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 updates for the MCBA system itself.

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. All reported problems in the previous release of I/M have been addressed.

RELEASE NOTES - INVENTORY MANAGEMENT, DIBOL RT-11 RELEASE 7

Release 7 of MCBA's Accounting, Distribution and Manufacturing system addresses problems created by Digital's DIBOL-83. In addition the Security System utilities have been significantly enhanced.

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 - INVENTORY MANAGEMENT, 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 Inventory Management package itself. In addition the Security System utilities have been significantly enhanced.

Enhancements for the I/M Package

The following changes have been made to the Item Master file:

1. The Quantity on-Order field has been increased from a D5 to D6.

2. The number of vendors have been increased from 1 to 3 and numeric vendor numbers are right-justified.

3. Item numbers may be optionally right-justified.

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 sort utility DMSORT.

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.

10. The MESARA.xxx message files are now compressed into one MESARA.DDF file. Each record is exactly one block long to prevent record locking conflicts. The file is variable in size to support from 1 to 98 terminals (1-98 users).

11. Each package allows the user to print reports, display, spool or print on a local printer. If you select either "spool file" or print to a queued printer and answer "N" (no) to the question "DELETE AFTER PRINTING", the report will be saved on a disk file and can later be reprinted using the "PRINT SPOOLED REPORTS" function contained in the Special Functions menu in each package.

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 RECO01 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 INVENTORY MANAGEMENT FILES

This program (INITIM) may be run upon request to initialize the master, index and transaction type files in the I/M 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 requests 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 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 ITMMAS and ITMIDX are always created as a pair with the same number of records, since ITMIDX is the index to the ITMMAS 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 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 ITMIDX 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.

This page intentionally left blank.

INVENTORY MANAGEMENT MENU

This program (IMMENU) displays the main menu for the Inventory Management 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 Inventory Management 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 Inventory Management package have the following format:

Bxttss.ccc

"B" is the fixed designation for the MCBA Inventory Management 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 Inventory Management reports and the programs that create them are as follows:

A - Item Master File - Base Data	- IMFPRT
B - Item Master File - Management Data	- IMFPR2
C - Item Master File - Manufacturing Data	- IMFPR3
D - Item Master File - All Data	- IMFPR4
E - Usage Exceptions Report	- USAGEX
F - Inventory Transaction Edit List	- RECEDT
G - Inventory Transaction Posting Report	- RECRPT
H - LIFO Inventory Evaluation Report	- LIFRPT
I - Items Purchased - By Vendor	- PURPRT
J - Items Purchased - By Item	- PURRPT
K - Stock Status Report by Item	- ITMSTS
L - Stock Status Report by Location	- LOCSTS
M - ABC Analysis Report	- ABCRPT
N - Cycle Count Worksheet	- CYCRPT
O - Reorder Advice Report by Item	- ADVICE
P - Reorder Advice Report by Location	- ADVLOC
Q - Reorder Advice Report by Vendor	- ADVPRT
R - Physical Count Edit List	- TAGEDT
S - Physical Count Update Audit List	- TAGAUD

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

CE0201.MCB

This page intentionally left blank.

DEVICE TABLE ASSIGNMENTS

Record #	File Name	Package	File Description
40	INVTAG	I/M	Inventory Tag File (Physical Count Tags)
41	ITMMAS	I/M	Item Master File
42	ITMIDX	I/M	Item Master Index
43	INVTRX	I/M	Inventory Transaction Work File
53	PURCHS	I/M	Purchase Order File
56	SELINV	I/M	Selected Inventory Items File (Temporary)
91	PRDSTR	BOMP	Product Structure File
92	PRDIDX	BOMP	Where Used Index
130	CYCIDX	I/M	Cycle Count Index (Temporary)
131	ABCIDX	I/M	ABC Analysis Index (Temporary)

This page intentionally left blank.

LIST OF PROGRAMS BY APPLICATION

1. Initialize Inventory Management files

   INITIM

2. Inventory Management Menu

   IMMENU

3. Item Master File Maintenance

   IMFMNT (IMFM1, IMFM2, IMFM3)
   IMFPRT
   IMFPR2
   IMFPR3
   IMFPR4
   ORGIMF
   SRTIID
   IMFCNT (PRSSRT, SRTPRS)
   RBPRST
   SRTPR1
   RBPRS2

4. Inventory Location Control

   LOCMNT

5. Print Usage Exceptions Report

   USAGEX

6. Inventory Trx Entry & Editing

   RECENT
   SRTREC
   RECEDT
   RECRPT
   UPDIMF
   CLRREC
   TRXCNT

7. Purchase Entry & Editing

   PURCH (PCMNU)
   ORDENT
   SRTPCH
   PURPRT
   SRTPHS
   PURRPT

8. Stock Status Inquiry

STKSTS

9. Print Stock Status Report

PSTKST
LOCSTS
ITMSTS

10. Print ABC Analysis Report

ABCSEL
ABCRPT
SRTABC
UNRABC
ABCSET

11. Print Cycle Count Worksheet

CYCSEL
SRTCYC
CYCRPT

12. Print Reordering Advice Report

ADVICE (AMENU, ADVEN, ITMSL)
SRTADV
ADVPRT
STADVL
STAVLC
ADVLOC

13. Physical Count Entry & Editing

TAGMNT	STTGB1
SRTTAG	TAGAUD
UNPRTG	PSTTAG
TAGEDT	CLRTAG
ORGTAG	

14. Special Functions

IMSFMN	CLRYOE
CLRMOE	LIFRPT
CLRMO2	LIFROL
IMSPOL	IMFILS

FILE USAGE MAP

LEGEND

O = Output
U = Updated
I = Input
D = Deleted
P = Protected
C = Use Count Set
* = Happens only under certain conditions

	ITMMAS	ITMIDX	INVTRX	PURCHS	INVTAG	ABCIDX	CYCIDX	SELINV	PRDSTR	PRDIDX
INITIM	O	O	O	O	O					
IMMENU										
IMFMNT	UC I*P*	UC I*P*							UC*	UC*
IMFPRT	IC	IC							UC* P*	
IMFPR2	IC	IC								
IMFPR3	IC	IC								
IMFPR4	IC	IC								
ORGIMF	UP	UP								
SRTIID	P* C*	UP								
IMFCNT	UP* C*									
LOCMNT	UC	IC								
USAGEX	IC	IC								
RECENT	UC	IC	IP UC							
SRTREC			UP							
TRXCNT			UP							
RECEDT	IC		IC							
RECRPT	IC		IP							
UPDIMF	UC	IC	IP							

LEGEND

O = Output
U = Updated
I = Input
D = Deleted
P = Protected
C = Use Count Set
* = Happens only under certain conditions

	ITMMAS	ITMIDX	INVTRX	PURCHS	INVTAG	ABCIDX	CYCIDX	SELINV	PRDSTR	PRDIDX
CLRREC			UP							
PURCH				IP UP						
ORDENT	UC	UC		UC						
SRTPCH				UP						
PURPRT	IC			UC						
SRTPHS				UP						
PURRPT	IC			UC						
STKSTS	IC	IC								
PSTKST	IC									
ITMSTS	IC	IC								
LOCSTS	IC	IC								
ABCSEL	IP					OPU D*				
SRTABC	P					UP				
ABCRPT	IP					IP D*				
ABCSET	UP					IP D				
CYCSEL	IP						OP D*			
SRTCYC	P						UP			

LEGEND

O = Output
U = Updated
I = Input
D = Deleted
P = Protected
C = Use Count Set
* = Happens only under certain conditions

	ITMMAS	ITMIDX	INVTRX	PURCHS	INVTAG	ABCIDX	CYCIDX	SELINV	PRDSTR	PRDIDX
CYCRPT	IP						IP D			
ADVICE	IC	IC						OP		
SRTADV								UP		
ADVPRT	IC							IP D		
STADVL								UP		
STAVLC								UP		
ADVLOC	IC							IP D		
TAGMNT	IC	IC P*			IP UC					
SRTTAG					UP					
UNPRTG					UP					
ORGTAG					UP					
TAGEDT	IC	IC			IC					
STTGBI	P				UP					
TAGAUD	IP	IC			IP					
PSTTAG	UP	IC			IP					
CLRTAG					UP					
IMSFMN										

LEGEND

O = Output
U = Updated
I = Input
D = Deleted
P = Protected
C = Use Count Set
* = Happens only under certain conditions

	ITMMAS	ITMIDX	INVTRX	PURCHS	INVTAG	ABCIDX	CYCIDX	SELINV	PRDSTR	PRDIDX
CLRMOE	UP	IC								
CLRMO2	UP	IC								
CLRYOE	UP	IC								
LIFRPT	IC	IC								
LIFROL	UP	IP								
IMSPOL										
IMFILS	I		I	I	I					
PRSSRT									UP	IP
RBPRST	IC	IC							UP	UP
RBPRS2	IC	IC							UP	IP

MODIFYING THE DEFAULT NUMBER OF LOCATIONS, PRICES OR VENDORS

The Item Master file contains three sets of arrays that sometimes need expansion (or reduction) for a particular end user's needs. These three arrays are for locations, prices and vendors.

The Locations Array

The locations array is the heart of MCBA's multi-location capability. The first location and its associated fields can be entered via the Item Master File Maintenance in the I/M package. All locations can be entered and maintained via the Inventory Location Control application, also in the I/M package. The fields involved are:

- LOC - a 2-character alphanumeric field that contains the location abbreviation.
- QTYONH - a 6-digit numeric field that contains the quantity on hand for the item at this location.
- QTYCOM - a 6-digit numeric field that contains the quantity allocated (committed) for the item at this location.
- QTYONO - a 6-digit numeric field that contains the quantity on order for the item at this location.
- REOLVL - a 5-digit numeric field that contains the reorder level for the item at this location.
- ORDUPT - a 6-digit numeric field that contains the maximum order quantity (order up-to) for the item at this location.
- PIKSEQ - a 3-character alphanumeric field that contains the bin location (picking sequence) for the item at this location.

All of the Quantity fields are whole numbers.

The array sizes for each of these fields must be identical. As shipped from MCBA, five locations can be entered for each item. The maximum number of locations that can be accommodated by the system is 99.

The Prices Array

The prices array contains prices by customer type. 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 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.

The Vendors Array

The vendors array contains the minimum order quantity for each of the vendors that supply this item. The first vendor's information can be entered via the Item Master File Maintenance application in the I/M package. The second and subsequent vendors must be entered via the Item Vendor Maintenance application in the Purchase Order and Receiving (P/O) package.

VENDOR - a 3-character alphanumeric field that contains the vendor number.
MINORD - a 5-digit numeric field that contains the minimum order for the item from this vendor. It is a whole number, expressed in the same units as the Purchase Unit of Measure.

The array sizes for these two fields must be identical. As shipped from MCBA, a maximum of three vendors per item can be entered. The maximum number of vendors per item that the system is designed to accommodate is 10.

Steps to Modify any Array

To change any one of the three sets of arrays (locations, prices, or vendors), you must make the same change to each of the fields associated with this set, so that the dimension (size) of each field's array is the same.

As an example, let us assume you wish to expand the prices array from 5 to 10. You would have to perform the following steps:

1. Locate all the record definition files for the Item Master file. They are all recognizable because they have the form RD041x.DEF, where "x" is a letter. RD041A.DEF is the standard record definition for the file. RD041B.DEF is the record definition for the control record of the file. RD041C.DEF contains all bracket characters. RD041S consists of one field that is initialized with a value that represents the size of each Item Master record. There will be others that have specialized uses that are clarified by the comments within the record definition itself.

2. Inspect RD041A.DEF. Notice that the field definition for the PRICCD field is 5A2 and that for the PRICE field is 5D8. These must be changed to 10A2 and 10D8 respectively.

3. Inspect RD041B.DEF. Notice the undefined field that is defined as 5A10. This is a filler field that occupies as much space as the prices array in the actual data records. This must be changed to 10A10.

4. Make similar changes to each of the other RD041x.DEF files.

5. If this is a new installation, you may proceed standardly with your installation. When you initialize your I/M files, you will be prompted for the number or prices, locations and vendors you wish to support. Take the default for locations and vendors, but answer "10" for prices.

6. If this is an existing installation with a significant number of items, then you will have to write a conversion program to convert your existing file into the expanded file. Use both the old and new record definitions to assist you. Be sure to expand the control record, data records and bracket records.

7. No other changes to source code should be necessary, unless you are at the same time making other changes, such as enlarging the size of the fields themselves, rather than just the array dimension size.

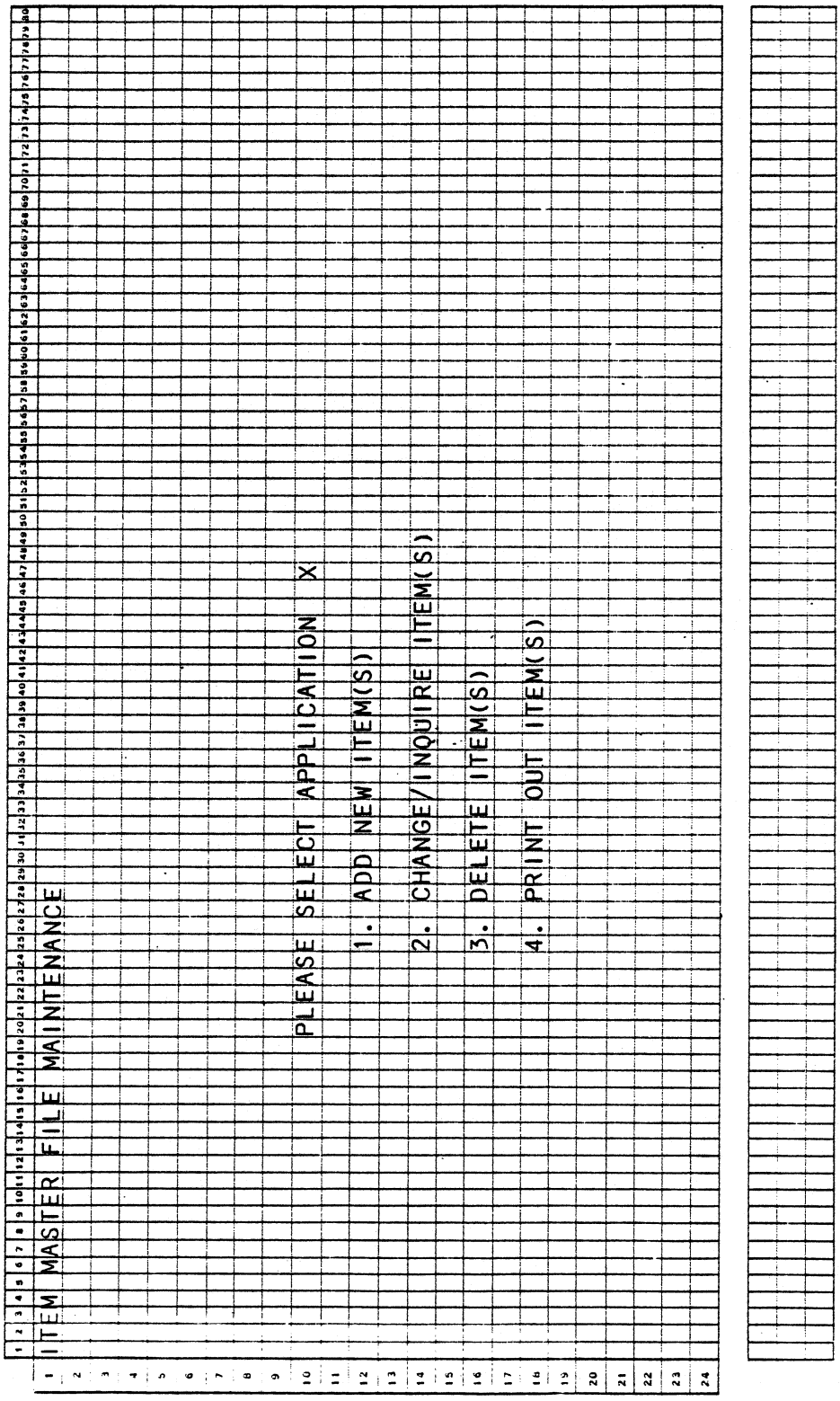
8. Recompile and relink the I/M package using the newly edited .DEF files. Also recompile any other packages that are installed that access the Item Master file.

If you wish to make different changes to the prices array, or to make changes to either the locations or vendors array, use the above steps as a guide. They are equally applicable for either expanding or reducing these arrays.

This page intentionally left blank.

SCREEN FORMATS

Program: IMFMNT:MMENU



```
ITEM MASTER FILE MAINTENANCE
ADD,CHANGE/INQUIRE,DELETE

 1. ITEM #      XXXXXXXXXXXXXXX                    13. WEIGHT              XXXXXX

 2. DESCR       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX     14. UNIT MEASURE        XX

 3. PROD CAT    XX          4. USER DEF CODE  XX    15. QTY SOLD MTD        XXXXXX-

 5. ON-HAND     XXXXXX-                             16. QTY SOLD YTD        XXXXXX-

 6. ALLOCATD    XXXXXX                              17. SALES $ MTD         XXXXXXXX-

 7. ON-ORDER    XXXXXX-                             18. SALES $ YTD         XXXXXXXXX-

 8. ORD UPTO    XXXXXX                              19. COST MTD            XXXXXXXX-

 9. REO LVL     XXXXX      10. BIN NO  XXX          20. COST YTD            XXXXXXXXX-

11. AVG COST    XXXXXXXXX                           21. BACKORDERABLE ? X

12. PRICE       XXXXXXXX                            22. TAXABLE ?           X

FIELD # TO CHANGE  XX
```

```
ITEM # ALREADY ON FILE
NO DELETION.  ITEM IS A COMPONENT OF A BILL.
NO DELETION. QUANTITIES ARE NOT ZERO.
```

STRUCTURES DELETED
YOU MUST REORG ITMMAS FILE BEFORE FURTHER DELETIONS CAN OCCUR.

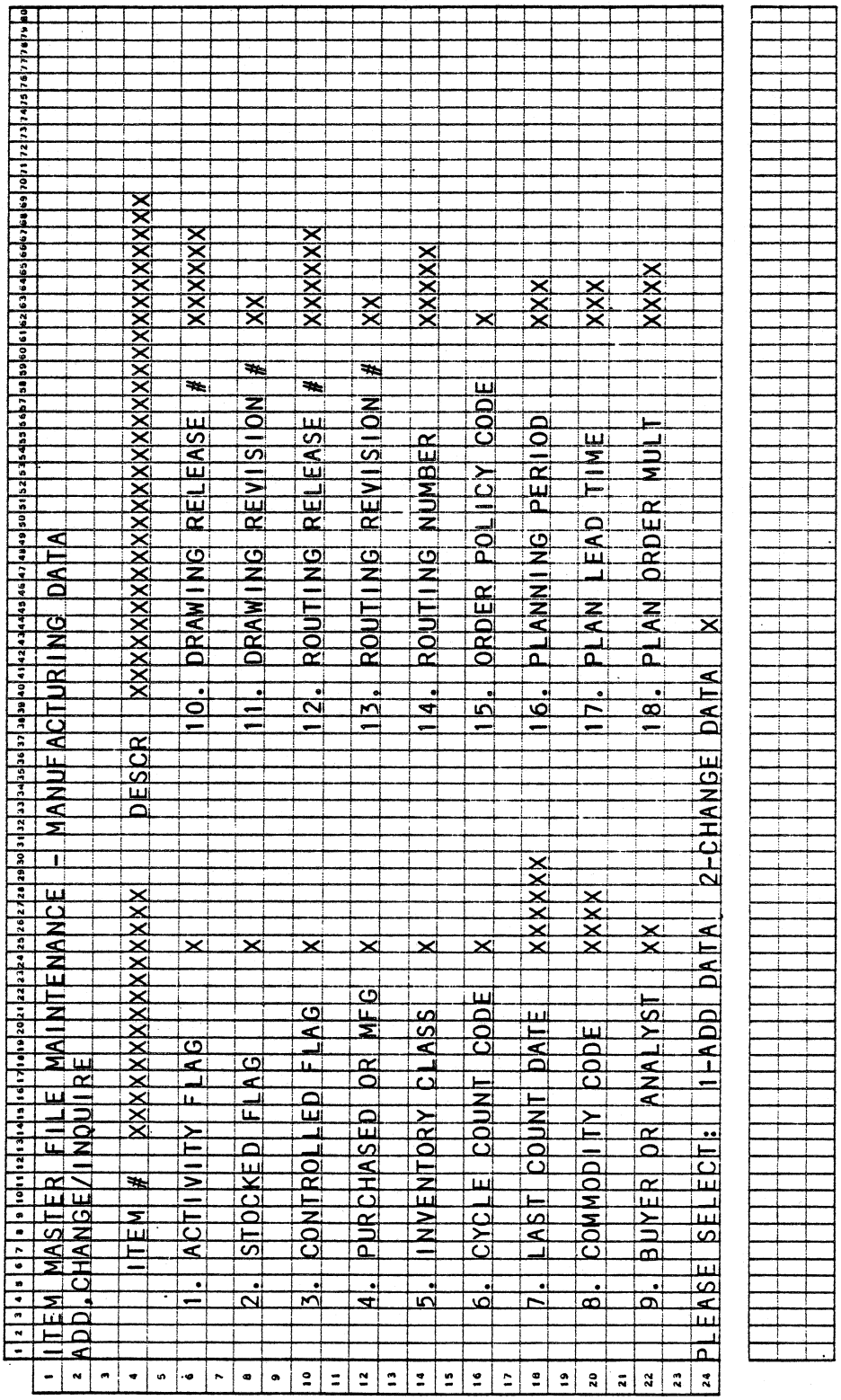
Remarks: IF BACKSPACE KEY IS PRESSED FOR "FIELD # TO CHANGE", UPDATE RECORD AND STAY ON THIS SCREEN.

```
ITEM MASTER FILE MAINTENANCE - MANAGEMENT DATA
ADD,CHANGE/INQUIRE

          ITEM #  XXXXXXXXXXXXXXX     DESCR  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

 1. RECOM MIN ORDER    XXXXX              10. LEAD TIME              XXX

 2. ECONOMIC ORD QTY   XXXXXX             11. LAST COST              XXXXXXXX

 3. AVERAGE USAGE      XXXXXX             12. VENDOR NUMBER          XXXX

 4. USAGE WGHT FCT     XX                 13. ORDER MINIMUM          XXXX

 5. SAFETY STOCK       XXXXX              14. ORDER MULTIPLE         XXX

 6. SAFETY FACTOR      XX                 15. TARGET MARGIN          XX

 7. AVERAGE ERROR      XXXXX              16. PURCHASED U OF M       XX

 8. SUM OF ERRORS      XXXXX-             17. PURCH TO INV RATIO     XXXX

 9. USAGE FILTER       XX

PLEASE SELECT: 1-ADD DATA  2-CHANGE DATA  X
```

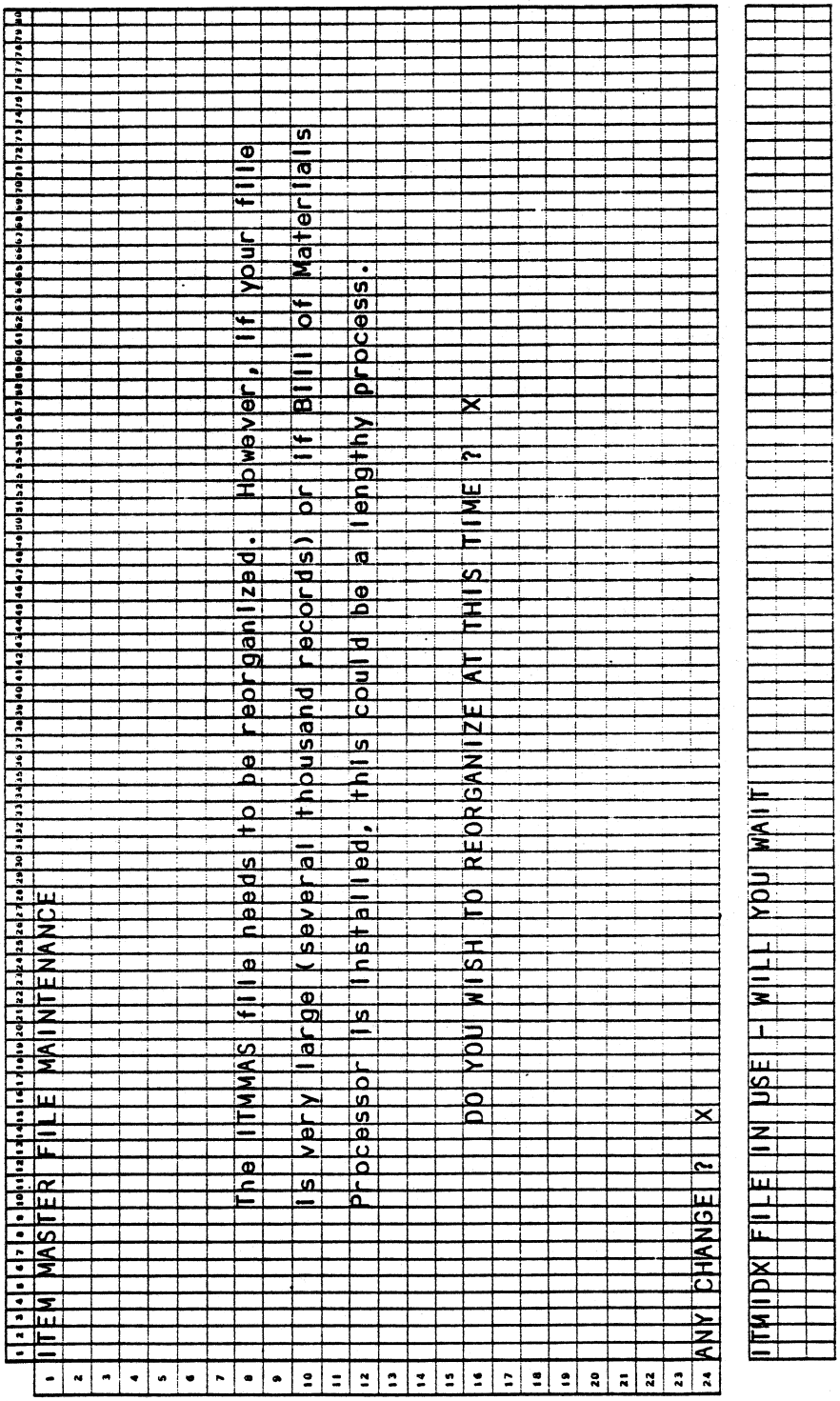
```
DO YOU WISH ALL ADDITIONS/CHANGES MADE TO BE PERMANENT ?  X
```

Remarks: IF BACKSPACE KEY IS PRESSED TO "FIELD # TO CHANGE", UPDATE RECORD AND RETURN TO SCREEN 1.

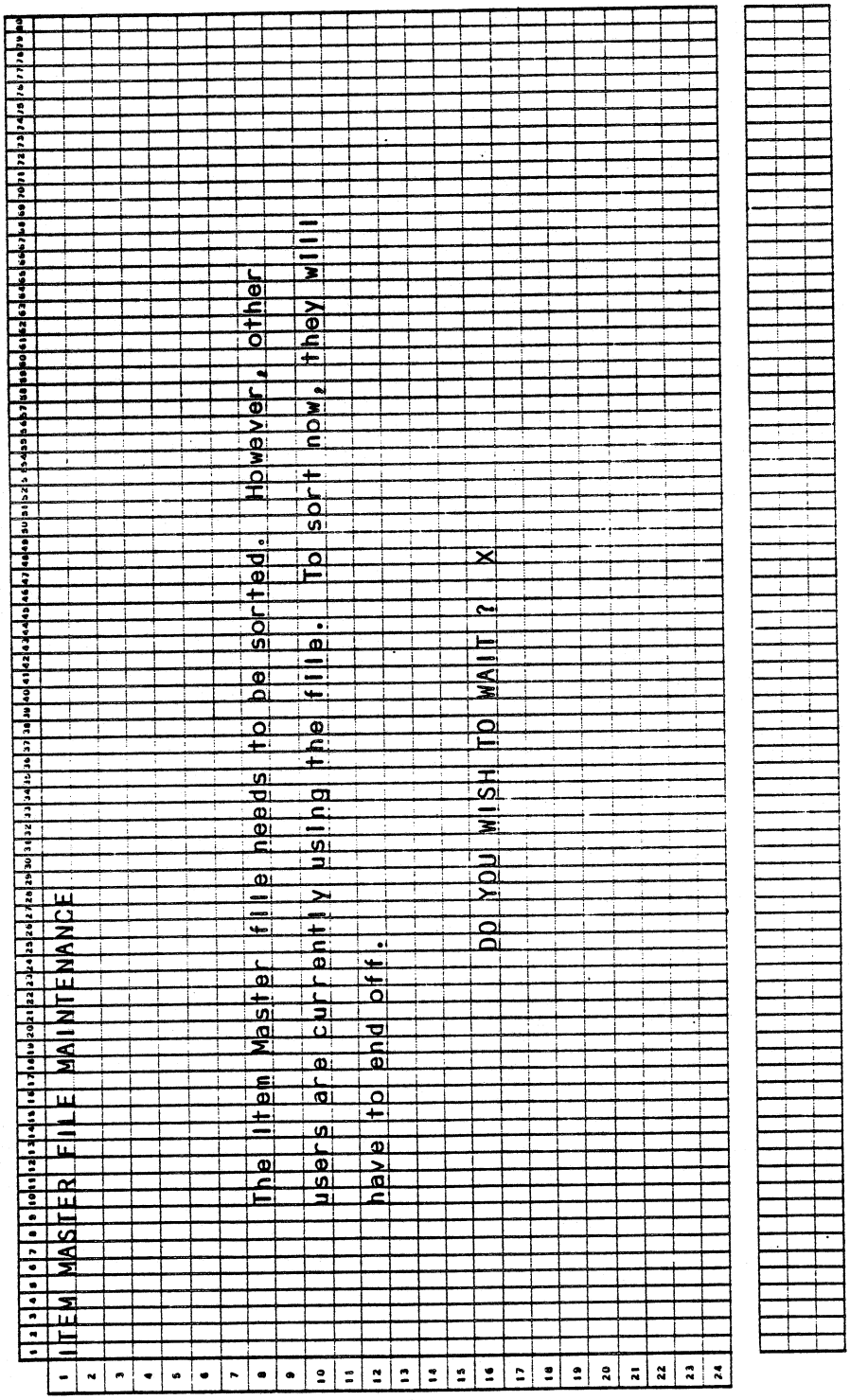
Program: IMFMNT:IMFM3



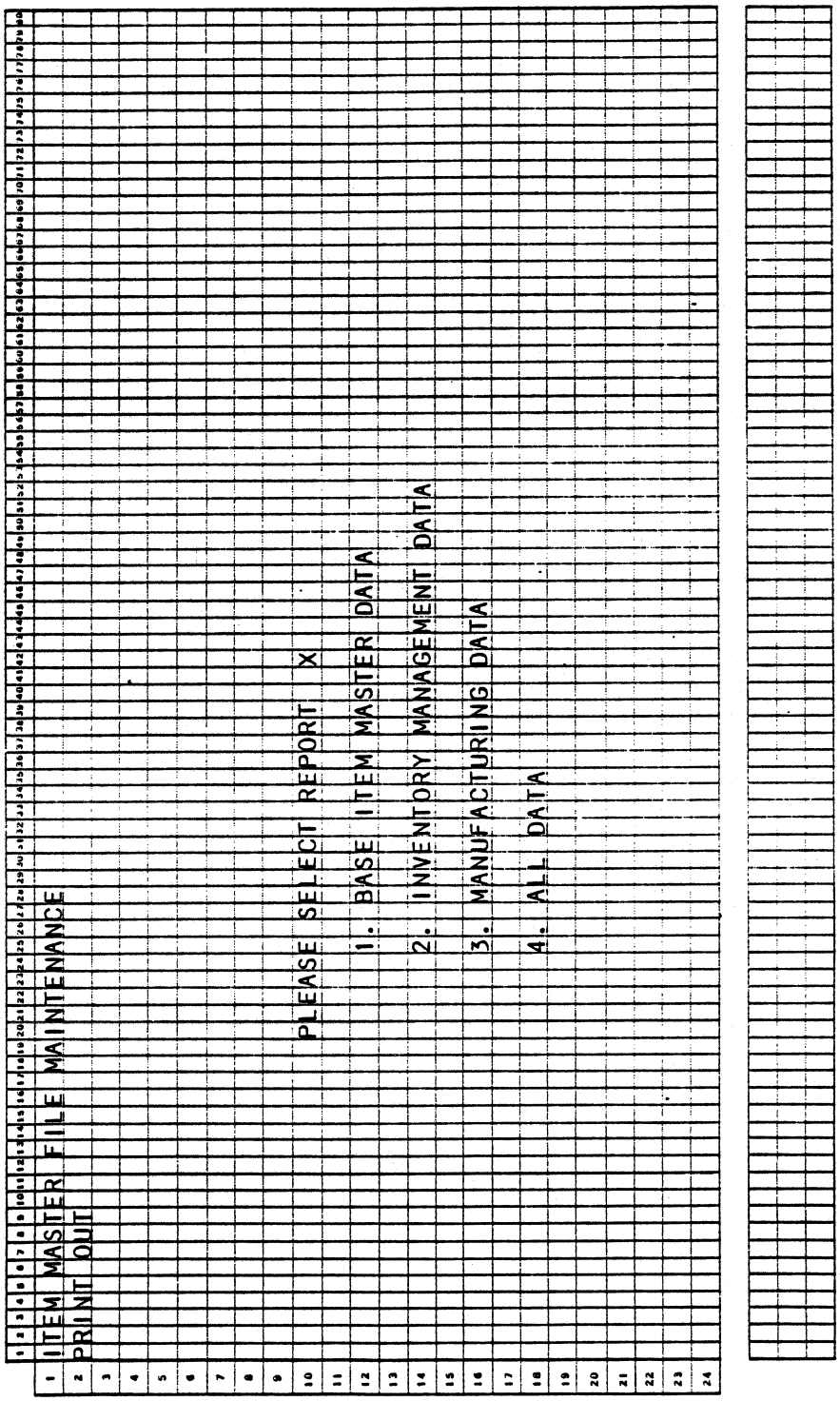
Program: IMFMNT



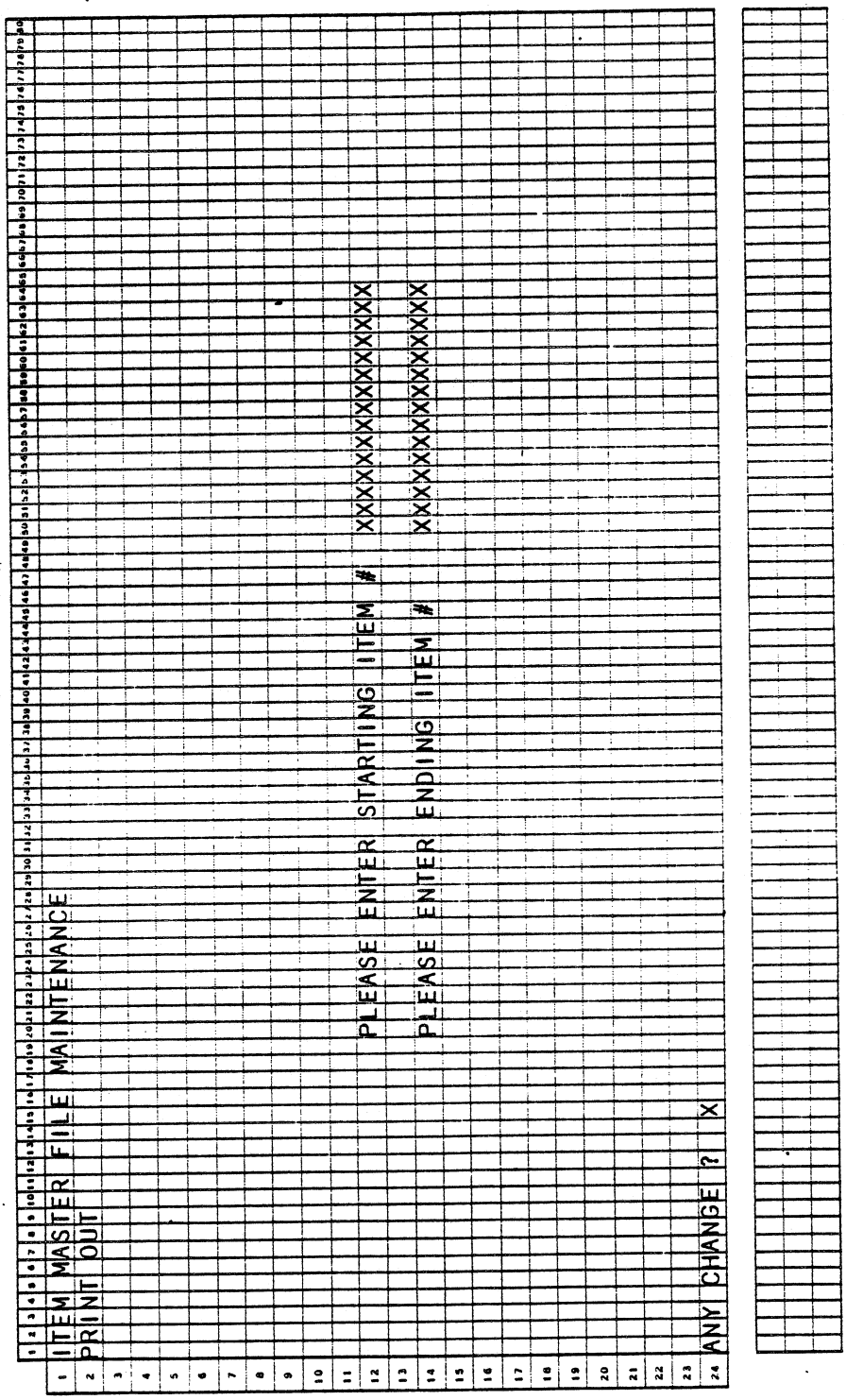
Program: IMFMNT



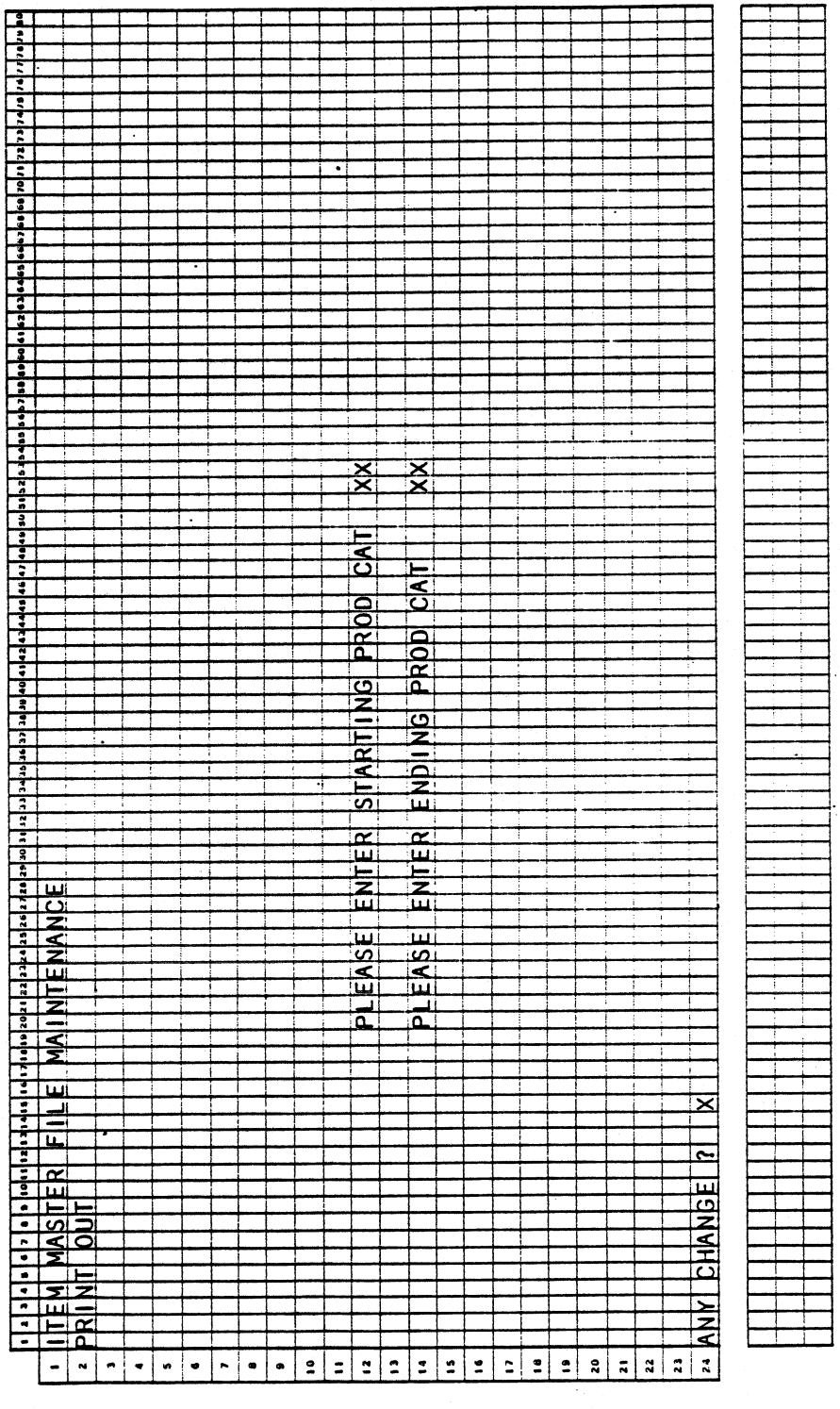
Program: IMFPRT



Program: IMFPRT:STENO



Program: IMFPRT:STENO



ITEM MASTER FILE MAINTENANCE APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Maintains the Item Master file.

Input: KBD

Files Updated: ITMMAS
ITMIDX

Output: Item File Print-Outs
Base Data
Management Data
Manufacturing Data
All Data

Enter Module From: IMMENU

When Done Return To: IMMENU

Programs in Module: IMFMNT, IMFM1, IMFM2, IMFM3, IMFPRT, IMFPR2, IMFPR3, IMFPR4, ORGIMF, SRTIID, IMFCNT, PRSSRT, SRTPRS, PRSCNT, RBPRST, SRTPRI, RBPRS2

Program Functions and Notes:

IMFMNT

Because of the size of the ITMMAS record, data entry is handled on three separate screens. Subroutines IMFM1, IMFM2, and IMFM3, respectively handle each screen.

Uses subroutine MMENU to display the Item Master submenu.

If print-out is selected, chains to appropriate program.

After opening the ITMMAS and ITMIDX files in update mode, the program reads the control record and checks to see if BOMP is installed. If so, it opens PRDSTR and PRDIDX also.

Add/Change Modes:

Each screen is handled identically. Checks returning values of INXCTL from IMFM1, IMFM2 and IMFM3. If the ABORT key has been pressed, returns to the head of that screen's paragraph. If the END key has been pressed, it does not write out the record. The program returns to the Item Master submenu if it is already in screen 1 or returns to screen 1 if in screens 2 or 3.

Then checks for the value of CNGCTL. If the END key has been pressed, the program writes out the record and returns to screen 1. Otherwise, it displays the next screen.

For add mode, sets the values of the first location and the manufacturing location to the default location in the ITMMAS control record. Also sets

the Stocked flag to "S" (Stocked), Controlled flag to "C" (Controlled), Activity flag to "A" (Active), Purchased flag to "P" (Purchased), and the purchase to inventory ratio to 1. These values can later be changed by the operator if screens 2 and 3 are entered at the same time. After all data has been entered, checks once again for the existence of the item number being added. Then updates RECCNT in control record, writes out new records to ITMIDX and ITMMAS, frees the buffer and redisplays screen 1.

Delete Mode:

Does not allow any deletions if DELCNT is greater than or equal to 95.

After returning from IMFM1, does the following validation:

If BOMP is installed, searches PRDIDX with a binary search option for the item number to be deleted. If successful, backs up through the file to find the first occurrence of the item number as a component. Then sequentially reads through all matches of the item number to be deleted and component item number in PRDIDX. Also checks the unsorted records. If any non-deleted match is found, disallows deletion and redisplays screen 1.

If okay, then checks to ensure all quantity-on-hand amounts at all locations are zero. If any are nonzero, disallows deletion and redisplays screen 1. Otherwise, marks records in ITMIDX and ITMMAS for deletion and updates DELCNT in control records.

If okay, does a binary search of PRDSTR file and finds the first occurrence of the item number as a parent. Marks all such records for deletion, making sure to search the unsorted records also. Keeps a running record of the number of records deleted. When complete, updates PDELCT in control record of PRDSTR, frees the buffer and redisplays screen 1.

Ending Off:

Upon ending off, if the files were opened and if the deleted count is greater than or equal to 50, displays a message informing the operator that the file needs reorganizing and allows the option of doing so now or later. Tries to protect ITMIDX, ITMMAS, and (optionally) PRDSTR and PRDIDX files by XCALL FILES with a switch of 3. If unsuccessful, chains to IMMENU.

If the delete count is less than 50, checks the unsorted count. If greater than or equal to 50, tries to protect ITMIDX file. If successful, chains to sort. If unsuccessful, displays a message and gives the operator the option of waiting until the file can be protected.

IMFM1

Subroutine to handle standard maintenance entry of screen #1. Called from IMFMNT.

IMRM2

Subroutine to handle standard maintenance entry of screen #2. Called from IMFMNT.

IMFM3

Subroutine to handle standard maintenance entry of screen #3. Called from IMFMNT.

IMFPRT

A print program for Item Master File Print-Out - Base Data per Report Format.

IMFPR2

A print program for Item Master File Print-Out - Management Data per Report Format.

IMFPR3

A print program for Item Master File Print-Out - Manufacturing Data per Report Format.

IMFPR4

A print program for Item Master File Print-Out - All data per Report Format.

ORGIMF

Chain to this program to purge deleted records from the Item Master file when the deleted record count is greater than 50.

Use buffered technique of reading and writing file over itself, skipping deleted records. Do this for the Master file, then the index.

Updates control record.

If, in the ITMMAS record, TYPSYS is greater than or equal to 3 (BOMP is installed), the PRDSTR and PRDIDX files must be updated to reflect the new relative record numbers in ITMMAS. Do this by chaining to SRTIID, thenITMCNT, then PRSSRT, then SRTPRS, then RBPRST, then SRTPRI, and finally RBPRS2.

SRTIID

Sort the file ITMIDX. Use the field IITMNO as the key.

IMFCNT

Program to update the organized record count in the ITMMAS control record after the sort is completed.

NOTE:

If MCBA's BOMP package is installed, the following sort and update sequence is done after a reorganization of the ITMMAS file.

PRSSRT

Sets up counters for SRTPRS.

SRTPRS

Sort the Product Structure file by item number and sequence number.

PRSCNT

Updates counters in the Product Structure file.

RBPRST

Since the ITMIDX file and the PRDSTR file are both in item number sequence, do a step to compare these two files and when a match is found, input the new relative record number into the PRDSTR record. Also write a new record into the PRDIDX file.

During this process remove any deleted records from the PRDSTR file. They are marked in CITMNO (1.6) with]]]DEL.

Write bracket records to pad out the remainder of both the PRDSTR file and the PRDIDX file.

SRTPRI

Sort the PRDIDX file using CMPITM as the key.

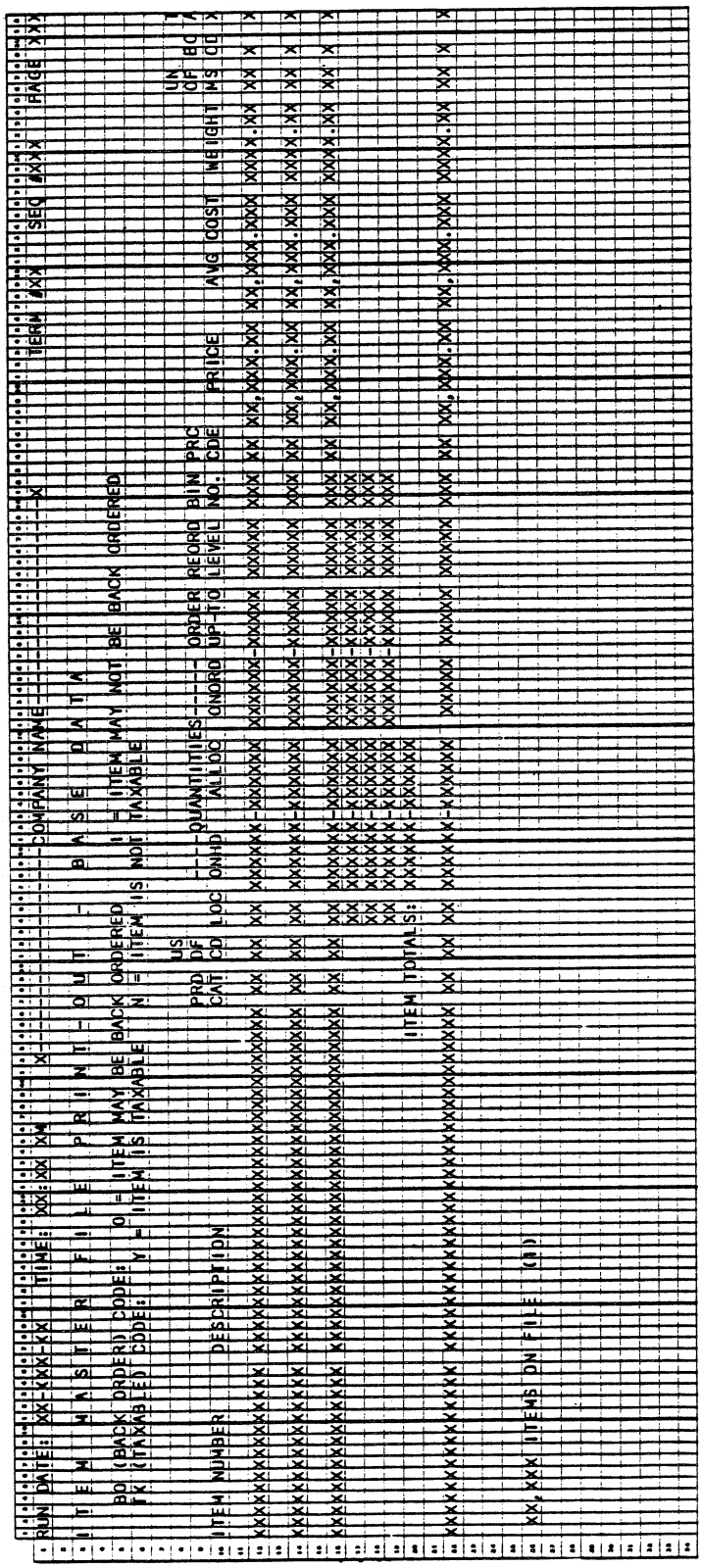
RBPRS2

Use a similar compare to the one used in RBPRST, scan the ITMIDX and the PRDIDX files.

When a match is found update the CRECNO field in the PRDSTR file.

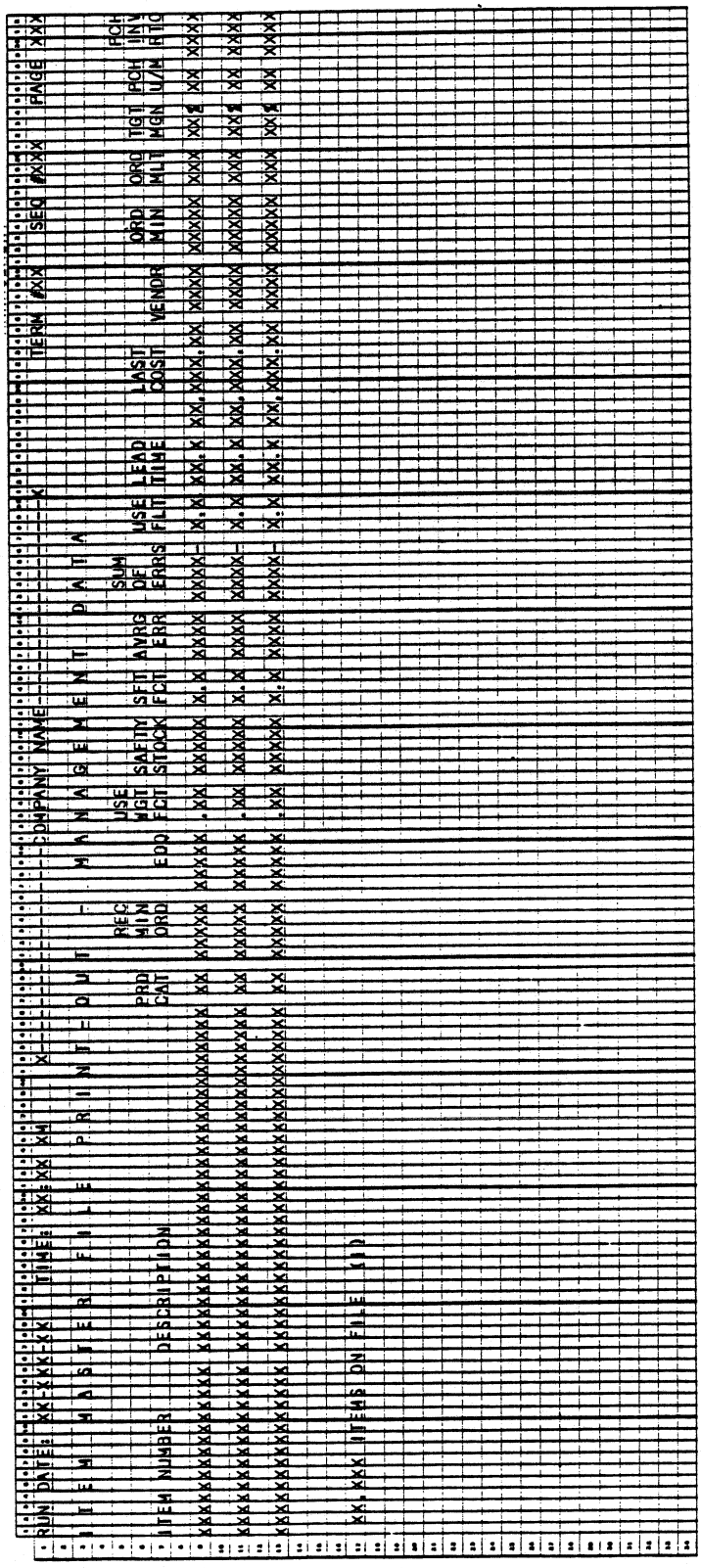
ITEM MASTER FILE MAINTENANCE APPLICATION
DIBOL JUN-84
REPORT FORMATS

Program: IMFPRT



Remarks: (1) ONLY PRINTS WHEN ALL ITEMS ARE SELECTED TO PRINT ON THE REPORT.

Program: IMFPR2



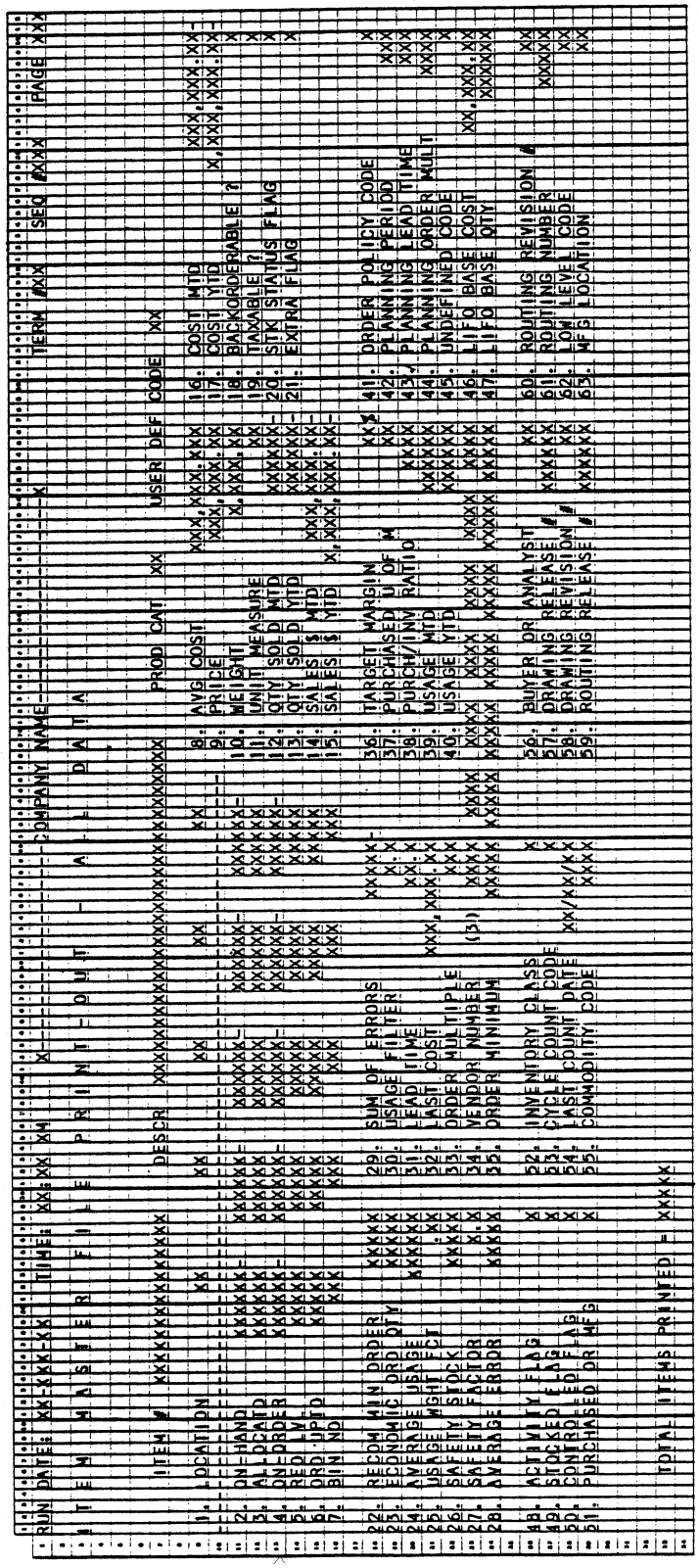
Remarks: (1) ONLY PRINTS WHEN ALL ITEMS ARE SELECTED TO PRINT ON THE REPORT.

Program: IMFPR3

```
RUN DATE: XX-XXX-XX   TIME: XX:XX XM      X-----------------------COMPANY NAME-----------------------X           TERM #XX    SEQ #XXX    PAGE XXX

I T E M   M A S T E R   F I L E   P R I N T - O U T   -   M A N U F A C T U R I N G   D A T A

                                                                      PUR      CYC LAST          ORD                                        LOW
                                                        PRD ACT          DR  ITM CNT COUNT    CMD  BUY  POL DRAWNG DRW ROUTNG RTE ROUTE  LEV MFG
ITEM NUMBER     DESCRIPTION                              CAT FLG STK CTL MFG CLS COD DATE     COD  ANAL COD REL #  REV REL #  REV NUMBER COD LC

XXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XX  X   X   X   X   X   X  XX/XX/XX XXXX XX   X   XXXXXX XX  XXXXXX XX  XXXXX  XX  XX
XXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XX  X   X   X   X   X   X  XX/XX/XX XXXX XX   X   XXXXXX XX  XXXXXX XX  XXXXX  XX  XX
XXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XX  X   X   X   X   X   X  XX/XX/XX XXXX XX   X   XXXXXX XX  XXXXXX XX  XXXXX  XX  XX

  XX,XXX ITEMS ON FILE  (1)
```

Remarks: (1) ONLY PRINTS WHEN ALL ITEMS ARE SELECTED TO PRINT ON THE REPORT.

Program: IMFPR4

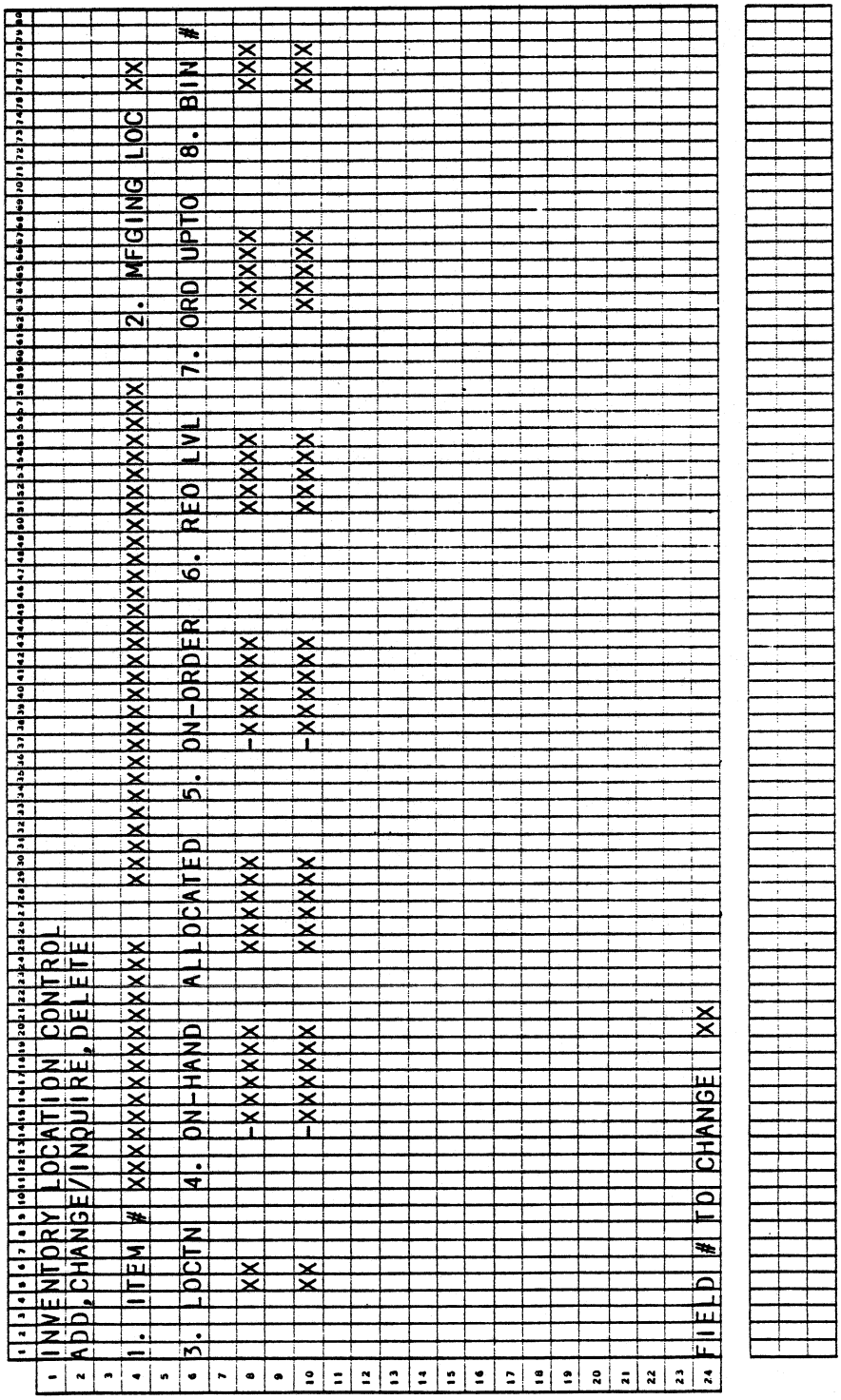


Remarks: (1) PROGRAM WILL PRINT TWO (2) ITEMS PER PAGE UNLESS SINGLE ITEMS ARE INDIVIDUALLY SELECTED.
(2) IN SOME CASES, THERE WILL BE MORE THAN 5 LOCATIONS - BUT THIS PROGRAM WILL ONLY PRINT THE FIRST 5.
(3) UP TO 7 VENDOR/ORDER MINIMUM COMBINATIONS MAY BE PRINTED, DEPENDING ON THE NUMBER SET AT BUILD TIME.
(4) THIS PROGRAM ONLY PRINTS THE FIRST PRICE IN THE PRICE ARRAY.

This page intentionally left blank.

SCREEN FORMATS

Program: LOCMNT



Remarks: DISPLAY SCROLLS TO HANDLE ANY SCREEN OVERFLOW OF LOCATIONS FOR AN ITEM.

PROGRAM SPECIFICATIONS

Function: Inventory Item maintenance - location related fields.

Input: KBD
ITMIDX
ITMMAS

Files Updated: ITMMAS

Output: None

Enter Module From: IMMENU

When Done Return To: IMMENU

Programs in Module: LOCMNT

Program Functions and Notes:

LOCMNT

Display the header as outlined on the Screen Format.

Accept the item number and search the ITMIDX file for it.

If found, display description, stocking location, and the first element of each of the following arrays:

LOC, QTYONH, QTYCOM, QTYONO, REOLVL, ORDUPT, and PIKSEQ

Allow changes to be made to any of these fields excluding quantity allocated (except when deleting location).

If location is changed check to be sure the new location does not already exist in the location array. Do not allow a blank location to be entered. Also, check that the original location is not the manufacturing location. If it is, display a notice that the stocking location is changed to the new location input.

Continue displaying locations one at a time until an empty (or blank) location is found.

Allow input of information for a location, always checking so that duplications in the location array do not occur.

Allow a location to be deleted only if quantity on-hand, quantity allocated, and quantity on-order is equal to zero.

Allow maintenance of the manufacturing location, but always check to be sure the location indicated as the manufacturing location exists in the location array.

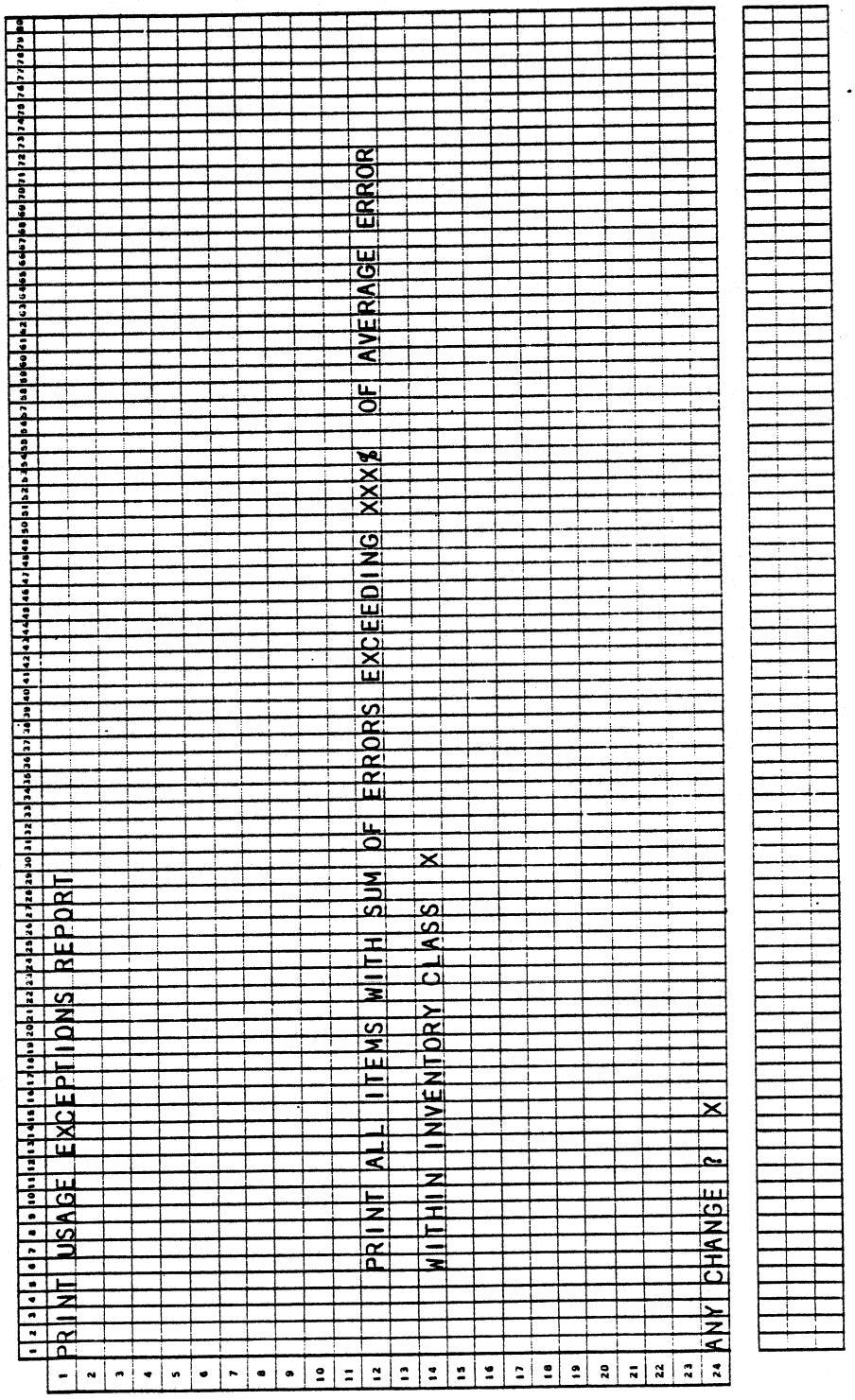
If location is changed and control W is input for the new location, delete this location, if and only if, on-hand, allocated, and on-order = 0. After deleting this location "squeeze" the other locations (if any) to the front of the array. Never allow a location to be blank in the middle of the locations array.

Read the ITMMAS control record to determine the maximum number of locations for any one record.

This page intentionally left blank.

SCREEN FORMATS

Program: USAGEX



PROGRAM SPECIFICATIONS

Function: Print Inventory Usage Exceptions Report.

Input: KBD
ITMIDX
ITMMAS

Files Updated:

Output: Inventory Usage Exceptions Report

Enter Module From: IMMENU

When Done Return To: IMMENU

Programs in Module: USAGEX

Program Functions and Notes:

USAGEX

Display per Screen Format and accept input for the report parameters.

Assume the Item Master Index is in order and read through it.

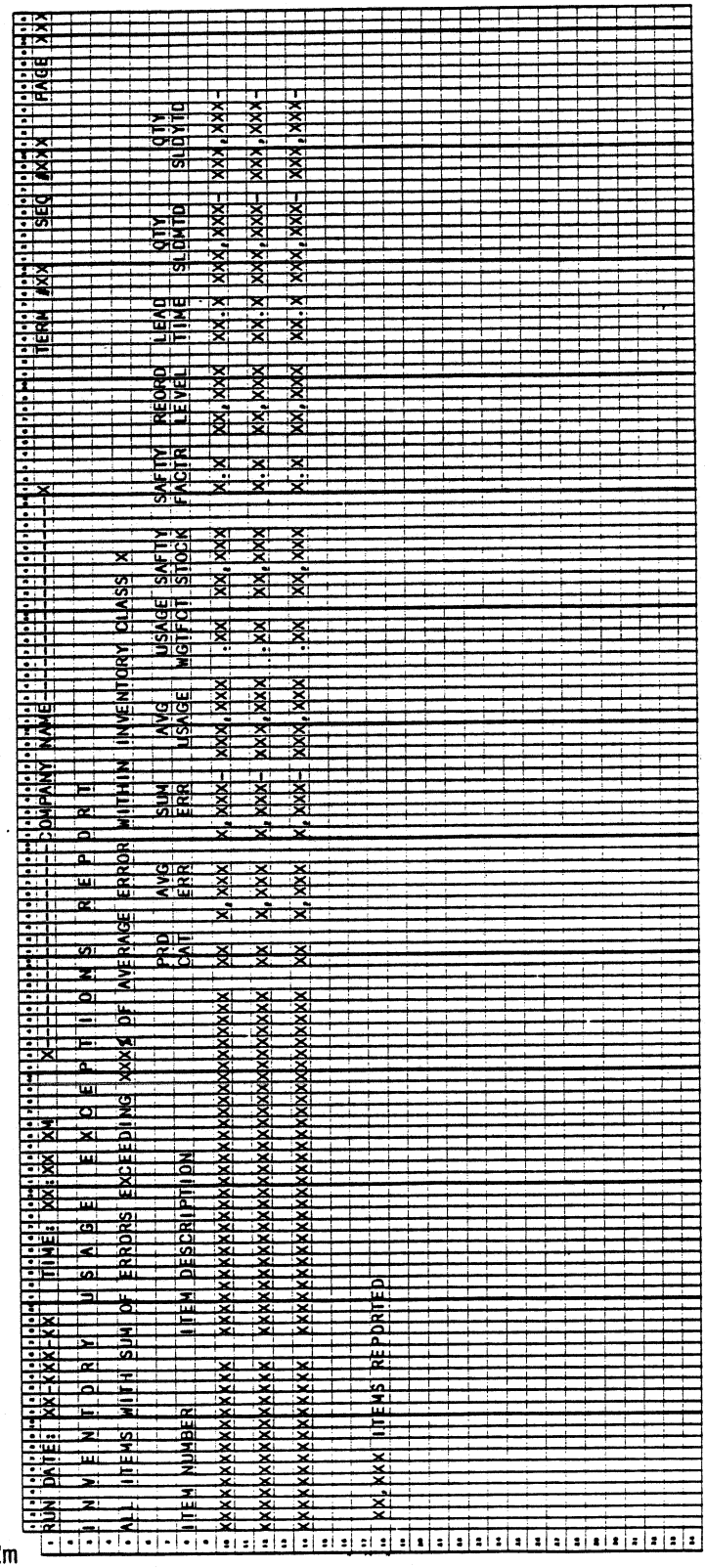
If the Item record has 0 average error, ignore it. Calculate the ratio* and compare the absolute value of it to the percentage entered by the operator. If the ratio exceeds the entered percentage, print that item.

Use the default location which is stored in the ITMMAS control record.

$$*\text{the ratio} = \frac{(\text{Sum of Errors})\ (100)}{\text{Average Error}}$$

NOTE: The Sum of Error (SUMERR) and Average Error (AVGERR) are fields in each Item Master record.

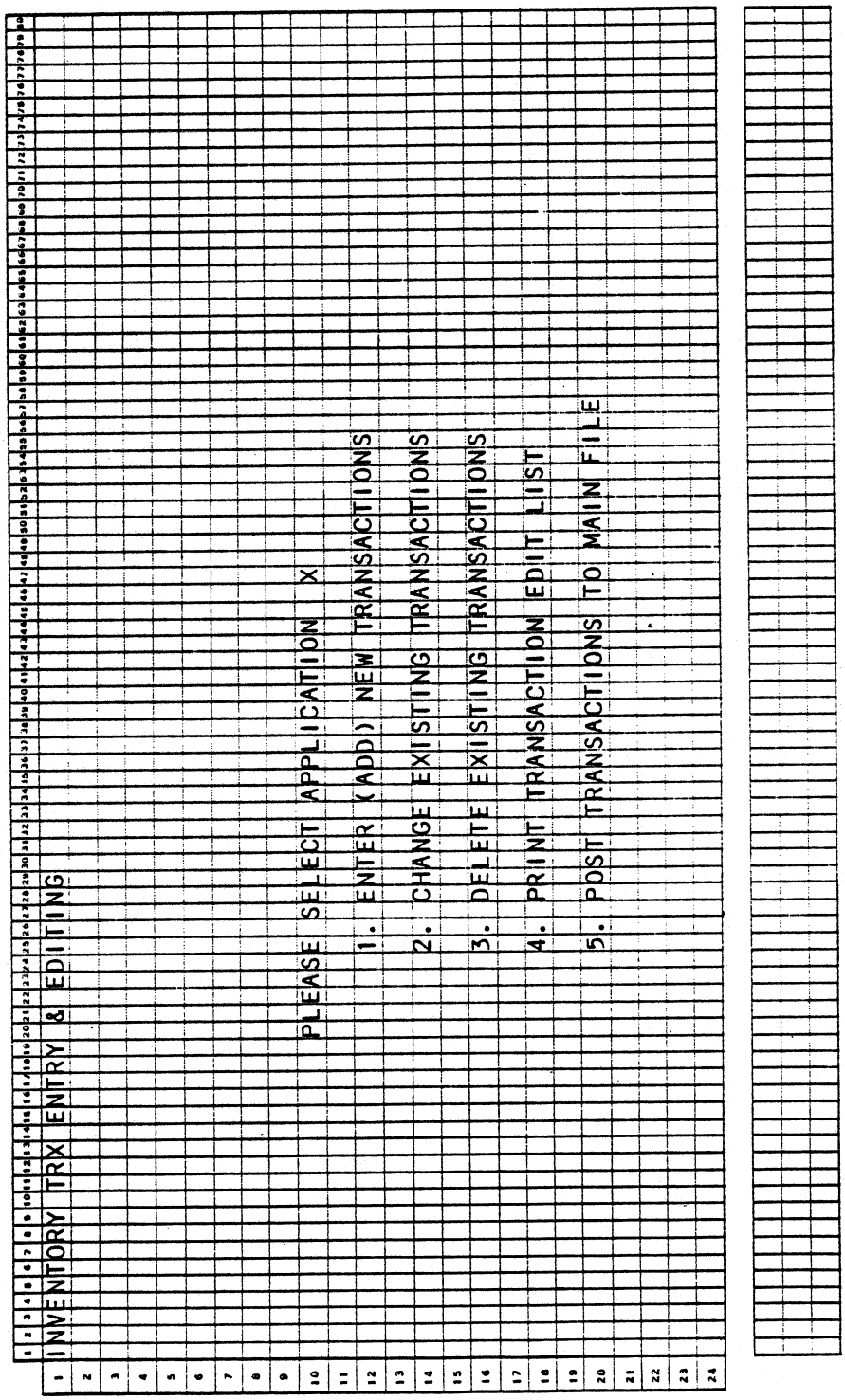
Program: USAGEX



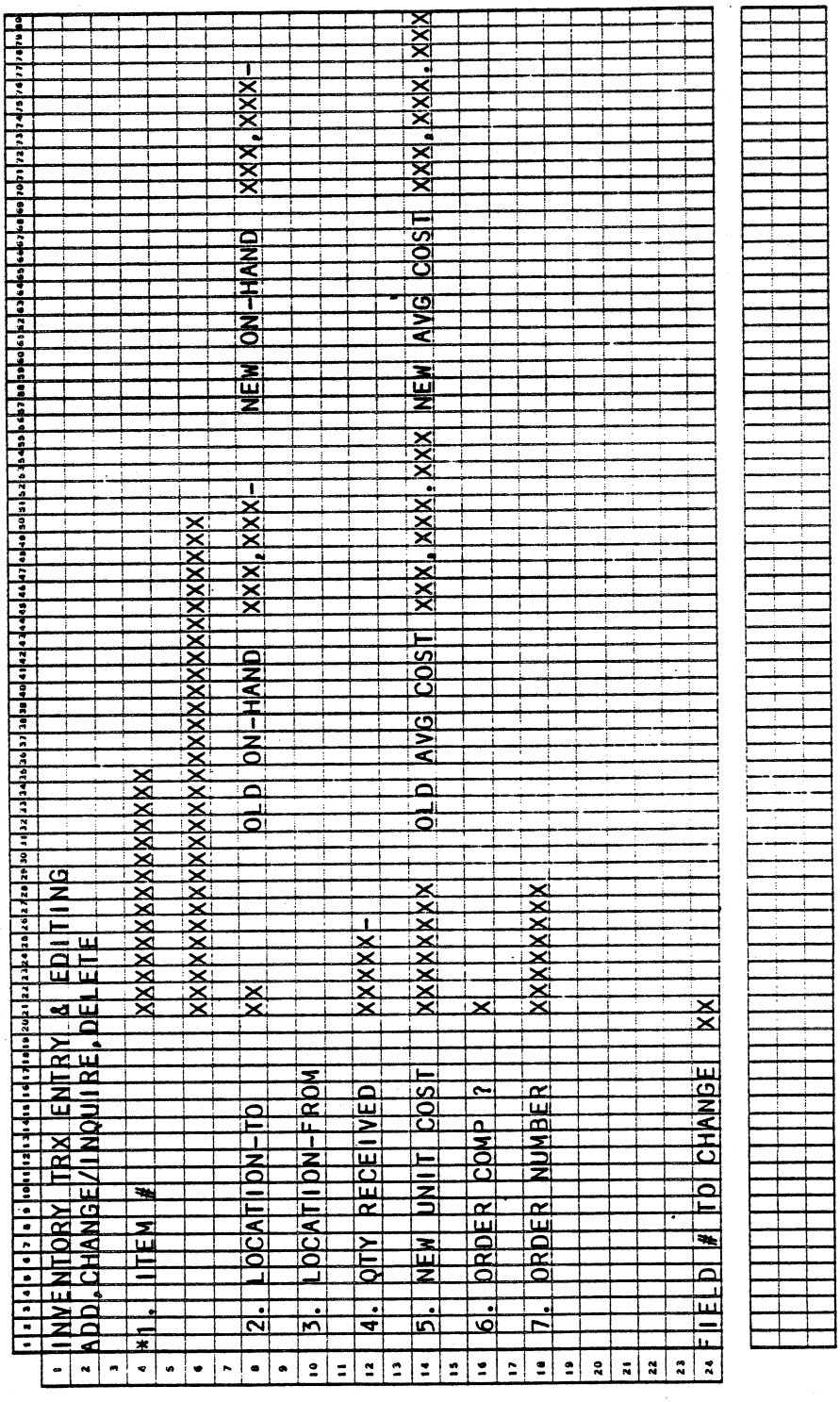
This page intentionally left blank.

SCREEN FORMATS

Program: RECENT:TMENU



Program: RECENT



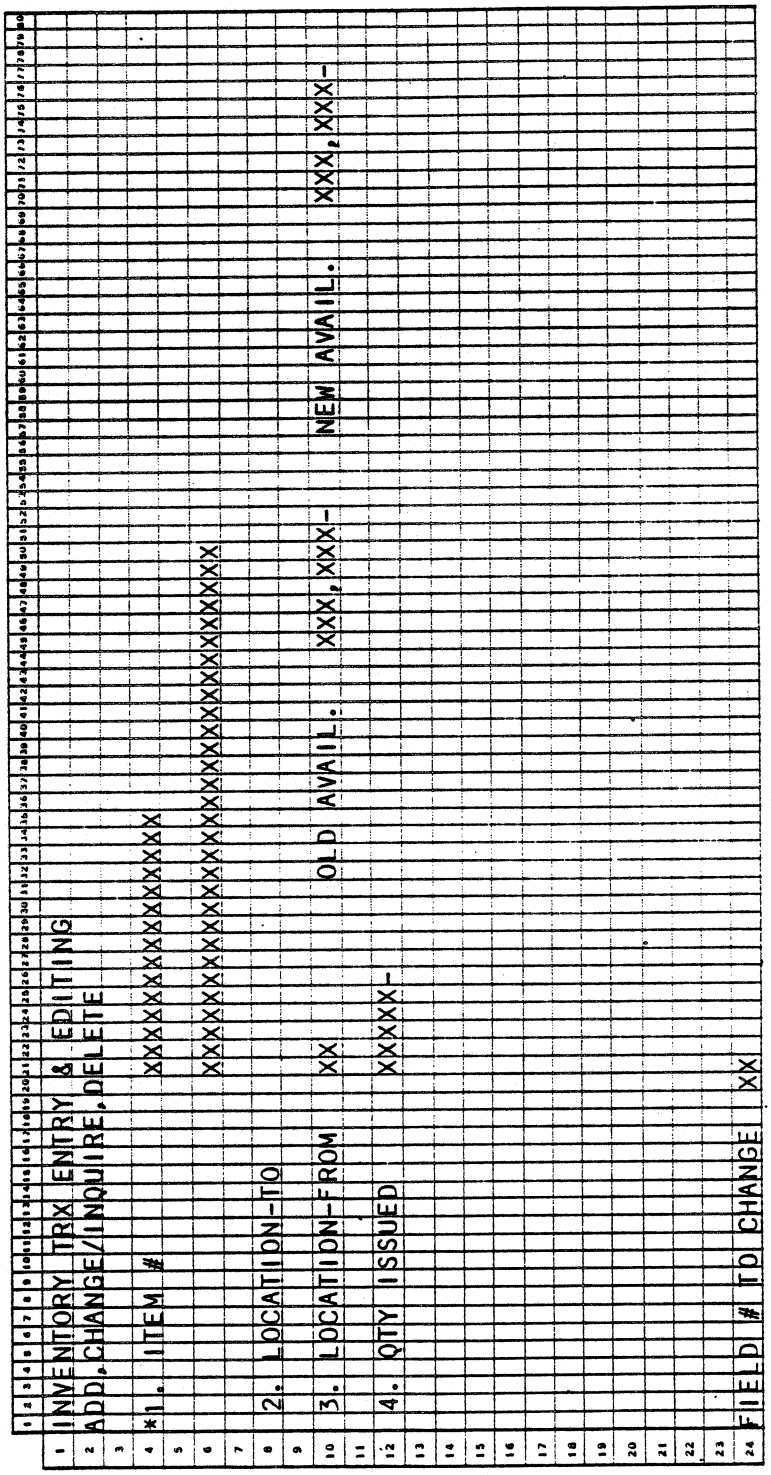
Remarks: THIS DISPLAY IS FOR RECEIVINGS ONLY. DETECT RECEIVINGS BY HAVING NO INPUT FOR LOCATION-FROM. FOR TRANSFERS AND ISSUES, SEE OTHER VIDEO TERMINAL FORMATS. "*" DISPLAYS IN CHANGE AND DELETE MODES AND INDICATES KEY FIELDS NEEDED TO RECALL A TRANSACTION. NO CHANGE IS ALLOWED TO "*" FIELDS.

Program: RECENT

```
INVENTORY TRX ENTRY & EDITING
ADD,CHANGE/INQUIRE,DELETE

 1. ITEM #                 XXXXXXXXXXXXXXX

                           XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

 2. LOCATION-TO            XX          OLD ON-HAND  XXX,XXX-    NEW ON-HAND  XXX,XXX-

 3. LOCATION-FROM          XX          OLD AVAIL.   XXX,XXX-    NEW AVAIL.   XXX,XXX-

 4. QTY TRANSFERRED        XXXXX-

FIELD # TO CHANGE  XX
```

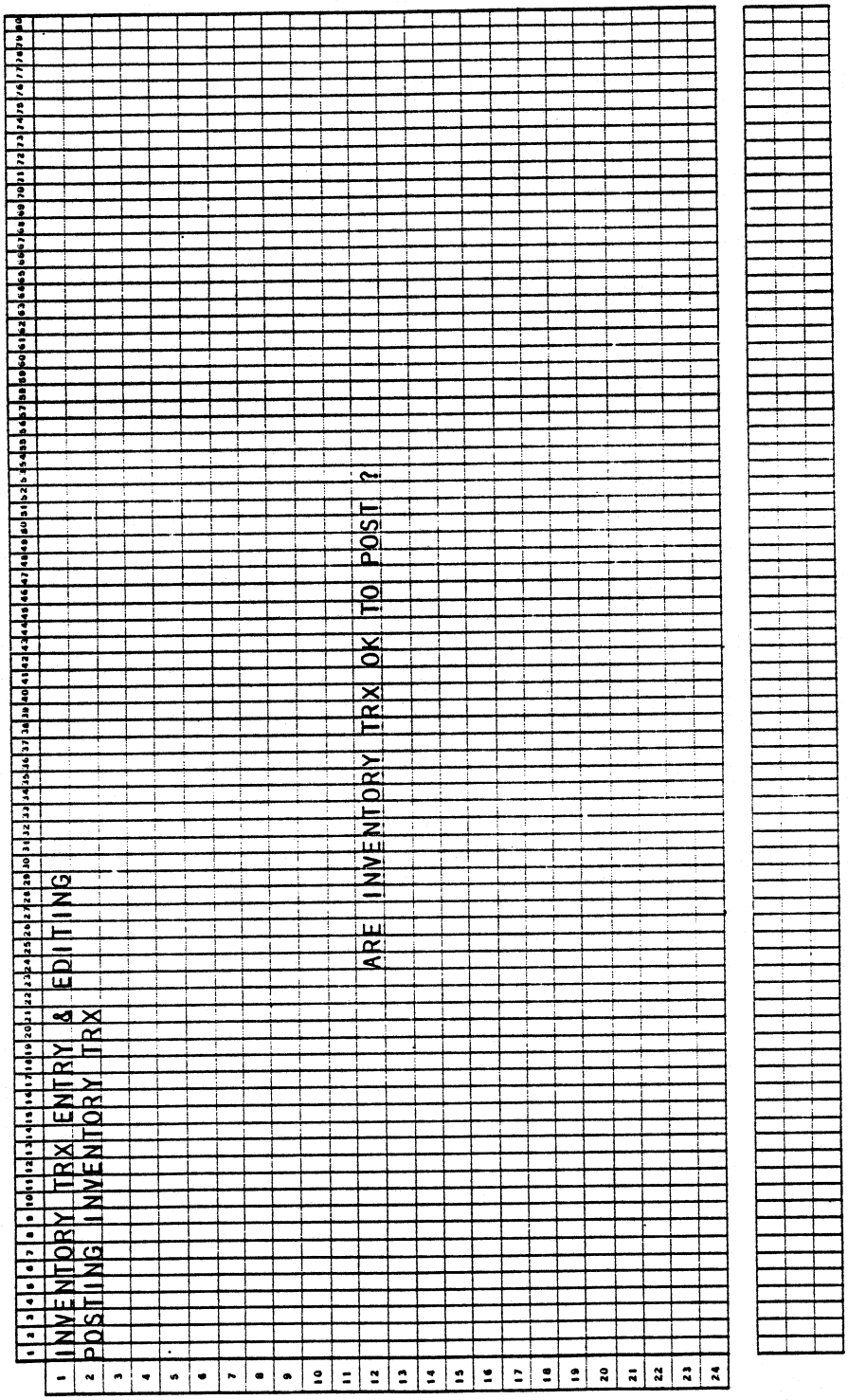
Remarks: THIS DISPLAY IS FOR TRANSFERS ONLY. DETECT TRANSFERS BY HAVING INPUT FOR LOCATION-TO AND LOCATION-FROM. FOR RECEIVINGS AND ISSUES, SEE OTHER VIDEO TERMINAL FORMATS. "*" DISPLAYS IN CHANGE AND DELETE MODES AND INDICATES KEY FIELDS NEEDED TO RECALL A TRANSACTION. NO CHANGE IS ALLOWED TO "*" FIELDS.

Program: RECENT



Remarks: THIS DISPLAY IS FOR ISSUES ONLY. DETECT ISSUES BY HAVING NO INPUT FOR LOCATION-TO. FOR RECEIVINGS AND TRANSFERS, SEE OTHER DISPLAY FORMATS.
"*" DISPLAYS IN CHANGE AND DELETE MODES AND INDICATES KEY FIELDS NEEDED TO RECALL A TRANSACTION. NO CHANGE IS ALLOWED TO "*" FIELDS.

Program: RECENT



PROGRAM SPECIFICATIONS

Function: Handles receivings, issues, and inventory transfer transactions including entry, editing, and posting.

See the Inventory Transaction Grid at UPDIMF for a summary of changes made to On-hand, On-order, and Allocated fields.

* Only the Allocation fields of controlled items (CNTRLD = C) are updated by definition.

Input:	Files Updated:	Output:
KBD	INVTRX	Inventory Receivings
ITMMAS	ITMMAS	Transfers, and Issues
ITMIDX		Reports
INVTRX		

Enter Module From: IMMENU When Done Return To: IMMENU

Programs in Module: RECENT, SRTREC, RECEDT, RECRPT, UPDIMF

Program Functions and Notes:

This will be a Standard Transaction Maintenance module.

Add mode, change mode, delete mode will be handled in program RECENT. Edit list will be program RECEDT. Posting journal will be program RECRPT. Inventory record update program will be UPDIMF.

RECENT

Open the files ITMIDX, INVTRX, and ITMMAS. Open the file INVTRX on a second channel for input or searching purposes.

Use subroutine TMENU for mode selection.

Display the top part of the transaction screen; do not display RCVD, TRFR, or ISSUE.

If in change or delete mode, display * next to index #1.

Accept input for the item number. Search for the item in inventory.

In add mode, display the description and accept the "to" location.

If no "to" location is input, accept the "from" location. If there is no "to" or "from" location, return to the "to" location.

Discern transaction type as follows: if there is a "to" location and no "from" location, treat the transaction as a receipt of goods. If there is a "from" location but no "to" location, treat the transaction as an issue. If there is both a "to" and "from" location, treat the transaction as an inventory transfer.

If it is a receipt of goods, display the remainder of the receipt screen and accept input.

If it is a transfer transaction, display the transfer screen.

If a non-recorded location is input in either the "to" or "from" field, and if there is room in the Item Master record to do so, create that location for that Item Master record and note on the screen that this action has been taken.

Compute the new average unit cost for receipt transactions as follows: if the quantity received when added to the total quantity on-hand for that item is equal to zero, put the new unit cost into the Average Unit Cost field. If the quantity on-hand is a negative amount, temporarily change it to a positive amount for the purposes of these calculations. Calculate the new average by taking the (total on-hand) multiplied by the (old average unit cost) plus the quantity received times the new unit cost; divide that sum by the quantity received plus on-hand.

For issues and transfers, the quantity allocated is increased at the "from" location by the quantity issued or transferred.

If the quantity received plus on-hand is zero:

New Average Cost = Net Unit Cost.

Else:

New Average Cost =

$$\frac{\text{(On-Hand of all locations) (Old Avg Cost) + (Qty Received) (New Unit Cost)}}{\text{On-Hand of all locations + Qty Received.}}$$

In change or delete mode, remember to update that quantity allocated if there is a "from" location.

SRTREC

Sort the INVTRX file by using the transaction type as the major key and the item number as the minor key.

RECEDT

Write a standard reporting program using the Report Format that says Edit List. Break to the next report on change of transaction type. Do not print headers if no transactions of this type exist in the file.

Print receipts first, transfers second, and issues third.

RECRPT

Identical program to RECEDT except the report title says Report.

UPDIMF

Update the Item Master file with the transactions from the INVTRX file.

Remember to decrease the quantity allocated for transfer and issue transactions for the "from" location.

If for some reason a transaction cannot be posted, print a line on the printer indicating such.

Inventory Transaction Grid

	Issues (From)	Transfer (From)	Transfer (To)	Receipt (To)
Qty On-Hand	-	-	+	+
Qty On-Order				-
Qty Allocated*	-	-		
Usage MTD**	+			
Usage YTD**	+			
Avg Unit Cost				X
Last Unit Cost				X

* Controlled items only
** Stocked items only.

Legend:
- subtract transactions qty
+ add transactions qty
X calculate

The grid shows the Item Master fields that are updated at post-time due to transaction type. NOTE: At entry-time, quantity allocated at the location-from location is increased by both transfer and issue transactions.

CLRREC

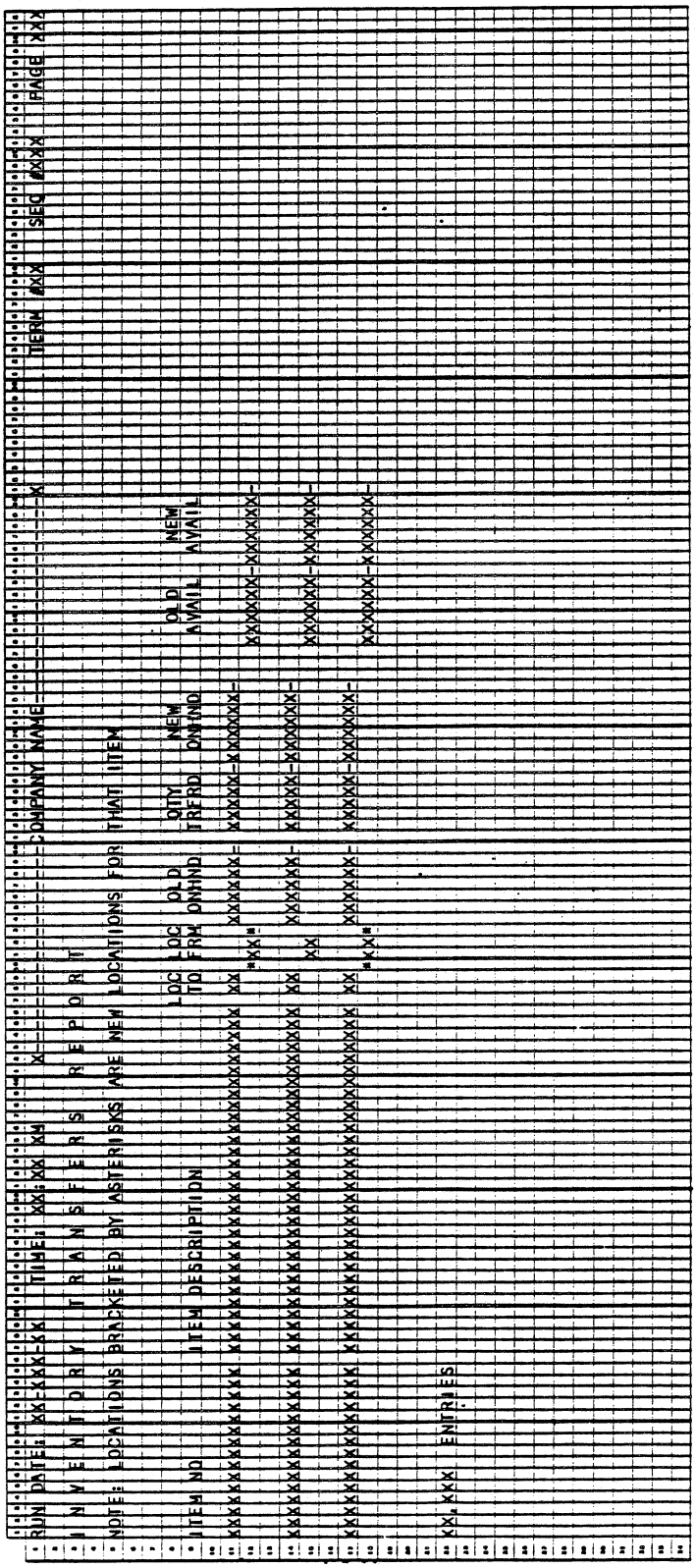
Clear the INVTRX file to all bracket records.

Program: RECRPT

```
RUN DATE: XX-XXX-XX   TIME: XX:XX XM    X-----------------COMPANY NAME------------------X          TERM #XX   SEQ #XXX   PAGE XXX

I N V E N T O R Y   R E C E I V I N G S   R E P O R T

NOTE: LOCATIONS BRACKETED BY ASTERISKS ARE NEW LOCATIONS FOR THAT ITEM

                                                      LOC       OLD      QTY     NEW   ORD  OLD      NEW     OLD AVG   NEW UNIT     NEW AVG   ORDER
ITEM NO         ITEM DESCRIPTION                      TO       ONHND    RECVD   ONHND  CPL ONORD    ONORD     COST       COST        COST     NUMBER
XXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*XX*    XXXXXX-  XXXXX-XXXXXX-  X  XXXXXX-XXXXXX-XX,XXX.XXX-XX,XXX.XX-XX,XXX.XXX-XXXXXXXXX
XXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*XX*    XXXXXX-  XXXXX-XXXXXX-  X  XXXXXX-XXXXXX-XX,XXX.XXX-XX,XXX.XX-XX,XXX.XXX-XXXXXXXXX
XXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*XX*    XXXXXX-  XXXXX-XXXXXX-  X  XXXXXX-XXXXXX-XX,XXX.XXX-XX,XXX.XX-XX,XXX.XXX-XXXXXXXXX

XX,XXX  ENTRIES
```

Remarks: EDIT LIST HAS SAME FORMAT. TITLE PRINTS "EDIT LIST" INSTEAD OF "REPORT". EDIT LIST PROGRAM IS RECEDT.
(1) TOTAL IS PRINTED WHEN NEXT RECORD READ IS A TRANSFER TRANSACTION.

Program: RECRPT



Remarks: EDIT LIST HAS SAME FORMAT. TITLE PRINTS "EDIT LIST" INSTEAD OF "REPORT". EDIT LIST PROGRAM IS RECEDT.

Program: RECRPT

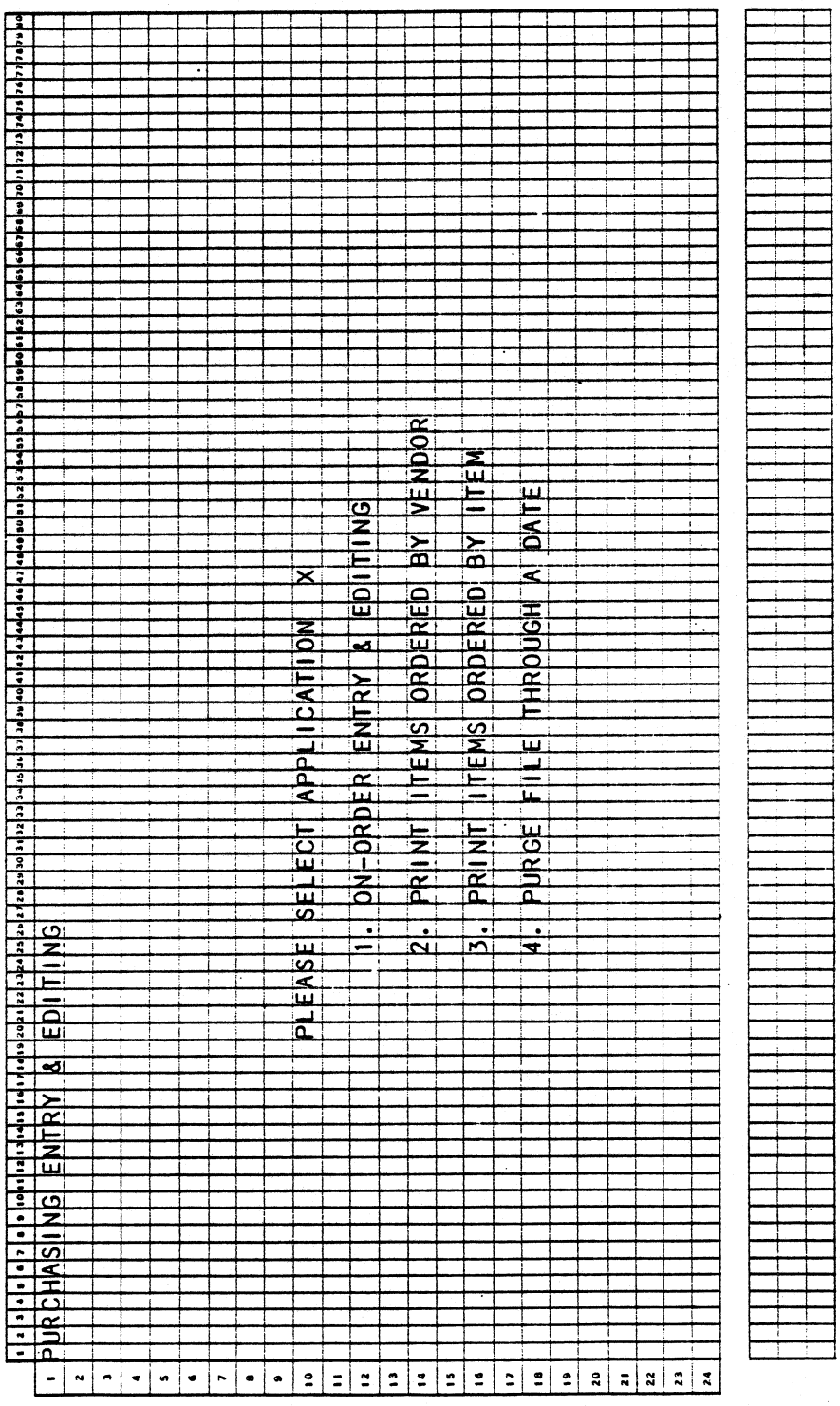
```
RUN DATE: XX-XXX-XX   TIME: XX:XX XM    X-----------------COMPANY NAME-----------------X          TERM #XX   SEQ #XXX   PAGE XXX
I N V E N T O R Y   I S S U E S   R E P O R T
NOTE: LOCATIONS BRACKETED BY ASTERISKS ARE NEW LOCATIONS FOR THAT ITEM

                                                      LOC  OLD       QTY    NEW       ORDER
ITEM NO           ITEM DESCRIPTION                    FRM  AVAIL     ISSUE  AVAIL     NUMBER
XXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX      *XX*XXXXXX-  XXXXX-XXXXXX-  XXXXXXXXXX(1)
XXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX       XX XXXXXX-  XXXXX-XXXXXX-  XXXXXXXXXX(1)
XXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX       XX XXXXXX-  XXXXX-XXXXXX-  XXXXXXXXXX(1)

XX,XXX  ENTRIES
```

Remarks: EDIT LIST HAS SAME FORMAT. TITLE PRINTS "EDIT LIST" INSTEAD OF "REPORT". EDIT LIST PROGRAM IS RECEDT.
(1) WILL BE BLANK UNLESS YOU HAVE SHOP FLOOR CONTROL INTERFACE.

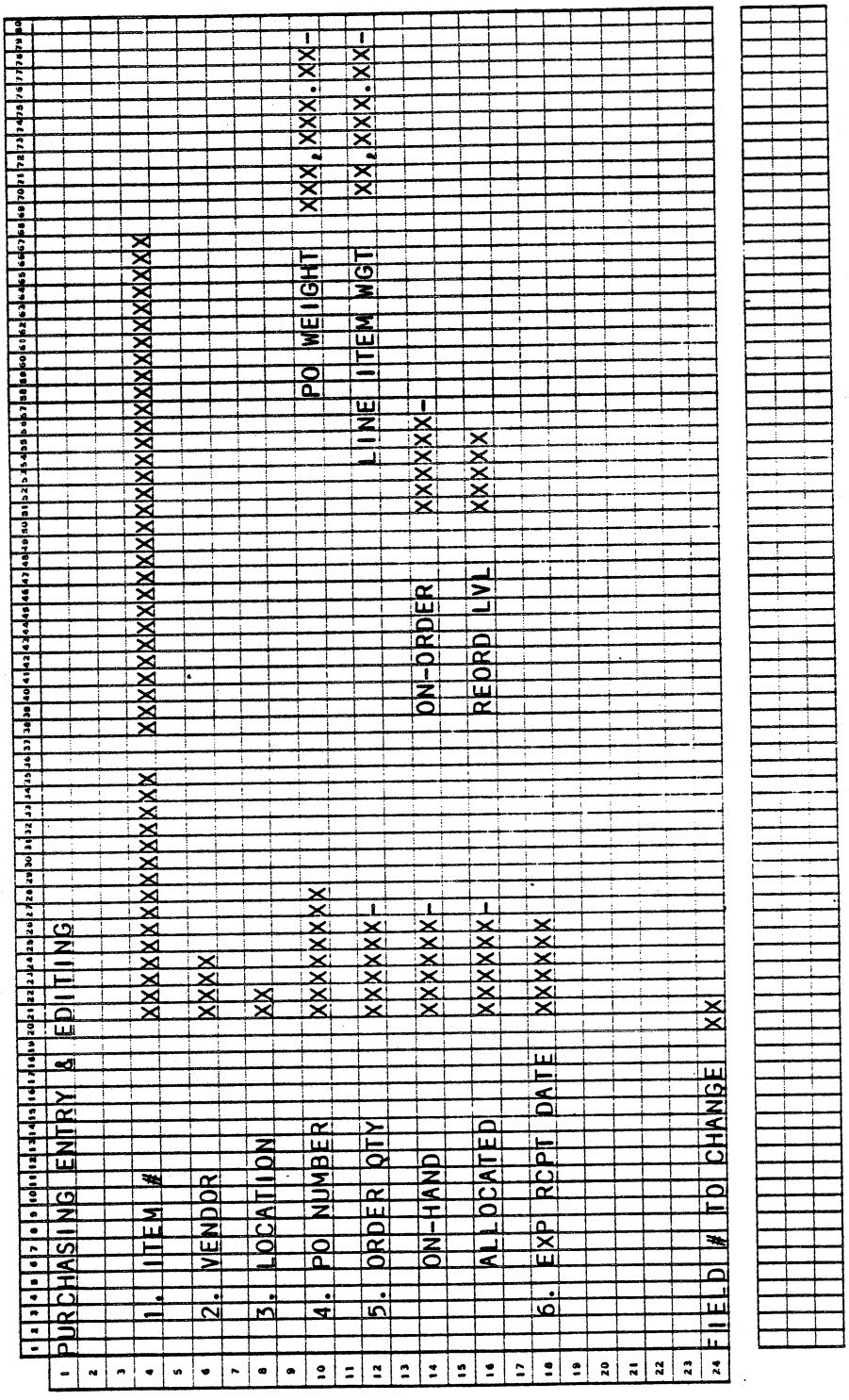
This page intentionally left blank.

SCREEN FORMATS

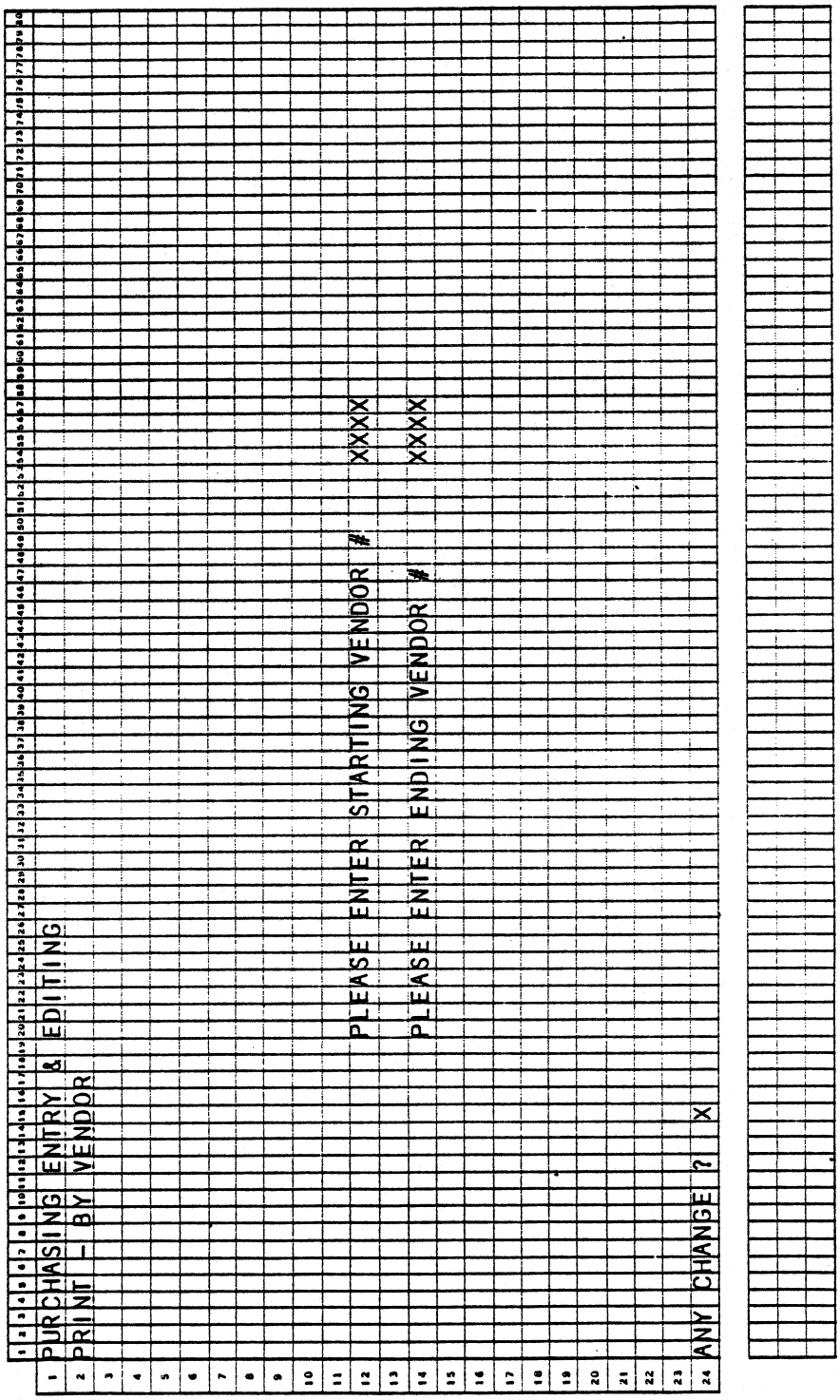
Program: PURCH (PCMNU)



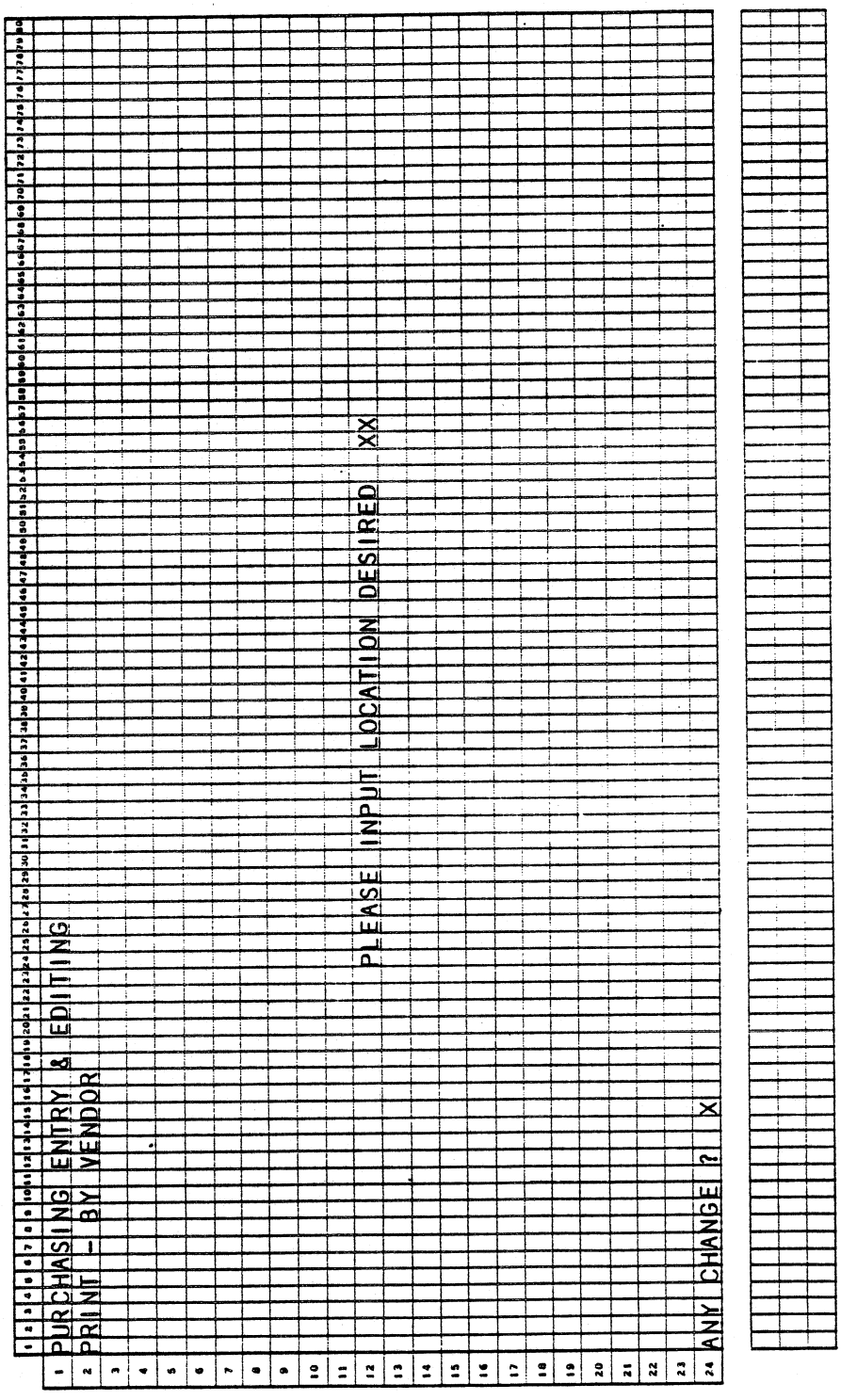
Program: ORDENT



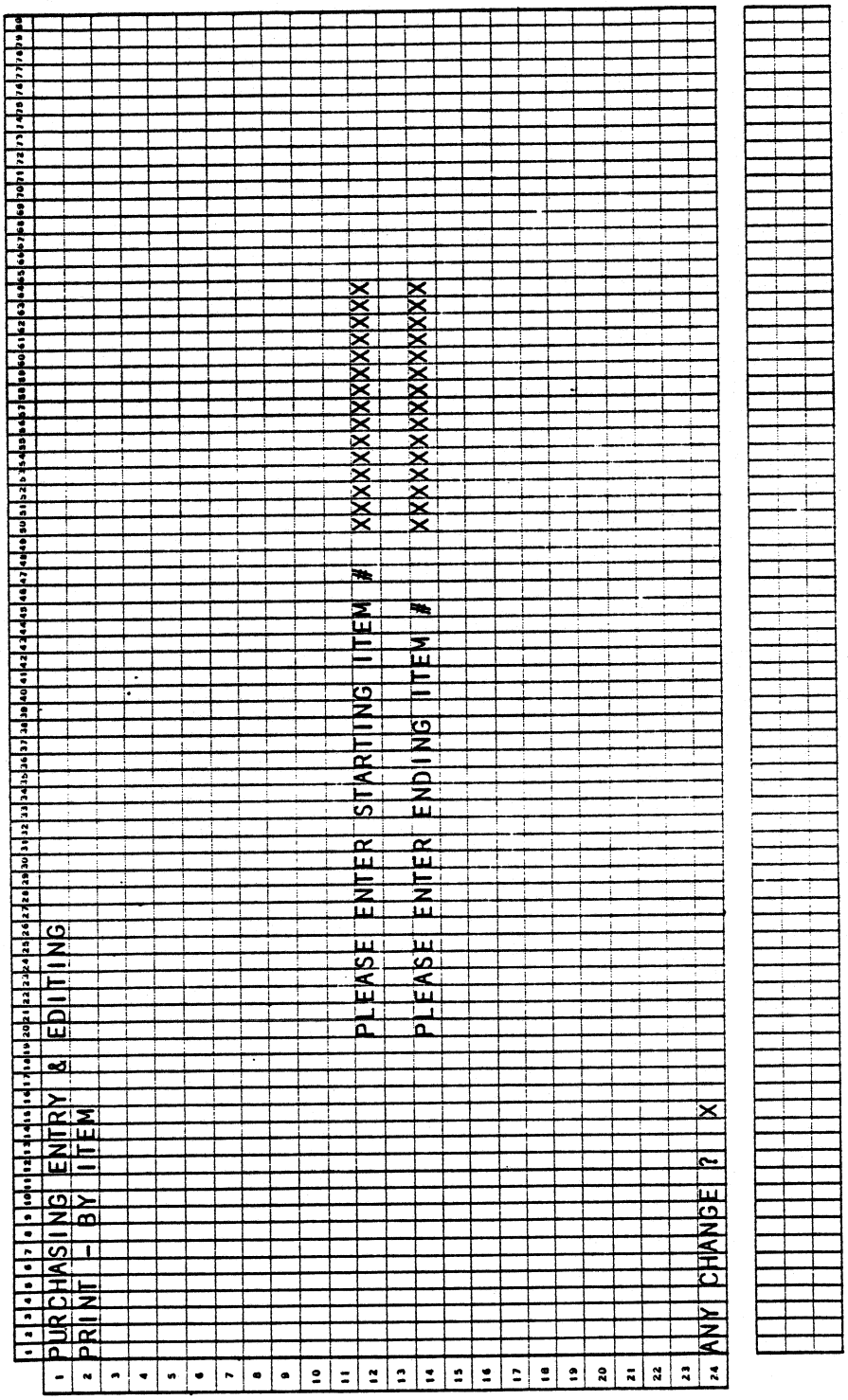
Program: PURPRT (STENO)



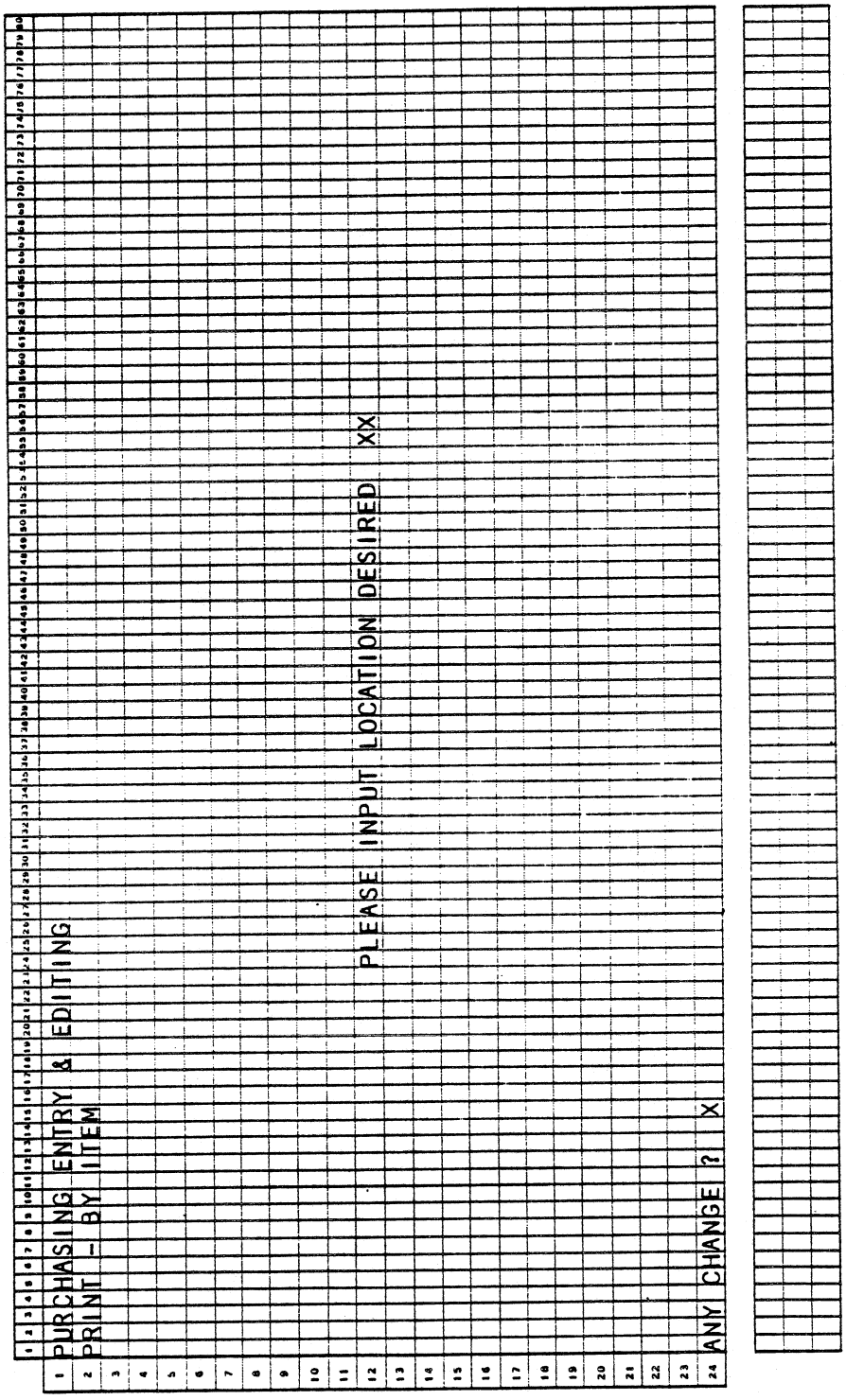
Program: PURRPT



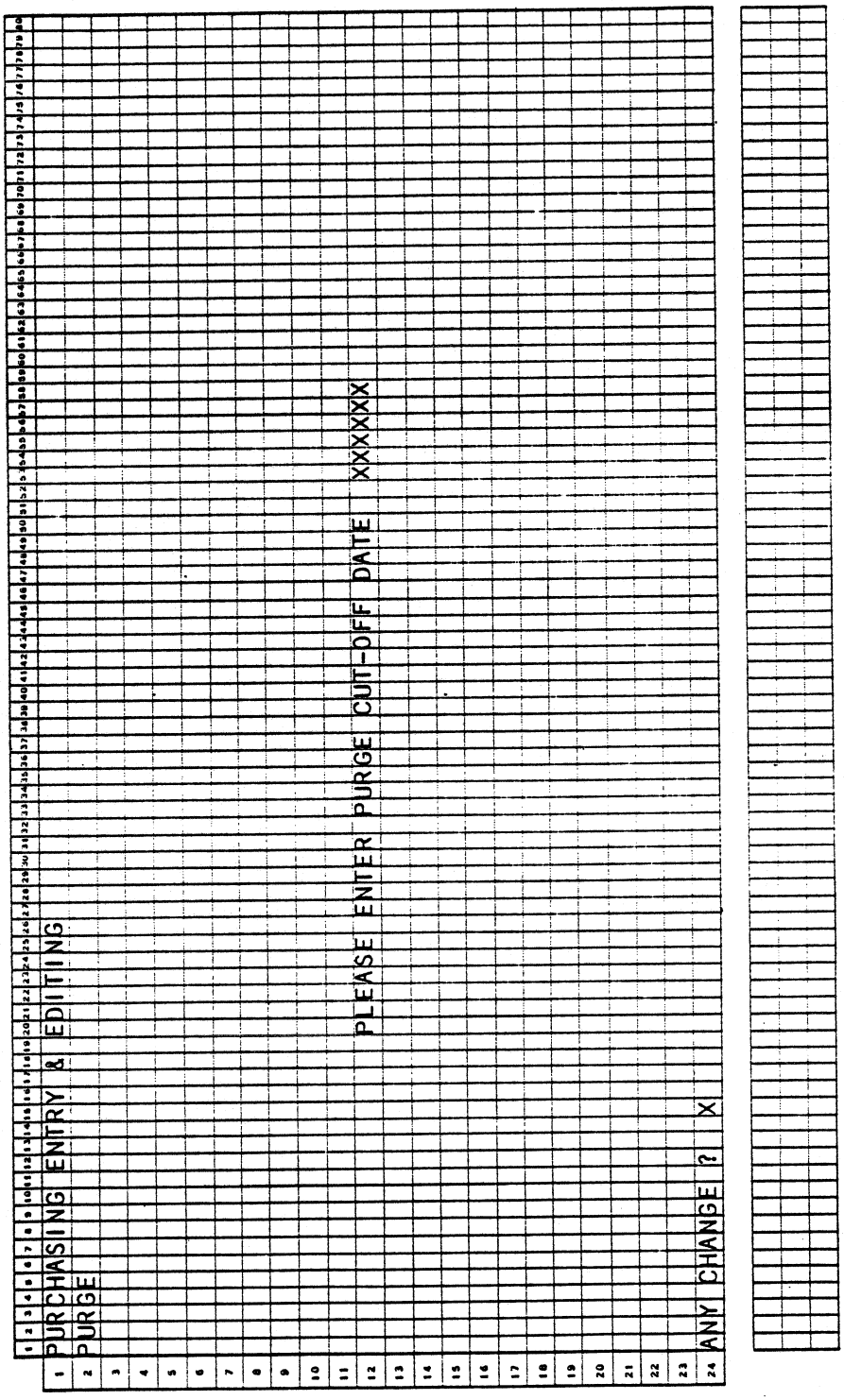
Program: PURRPT (STENO)



Program: PURPRT



Program: PURCH



PROGRAM SPECIFICATIONS

Function: Allows entry of items on-order, prints a report of items ordered by vendor or by item, and selectively purges and reorganizes the Purchasing file by date.

This is a simplistic Purchase Order handler; for a much more complete handling see MCBA's Purchase Order & Receiving (P/O) package.

This module accepts the quantity ordered as it is entered (no conversion) and increases on-order by that amount for the specified item and location.

(NOTE: The On-Order field is reduced by a receipt in Inventory Transaction posting.)

Input:	Files Updated:	Output:
KBD	PURCHS	Items Purchased Report by Vendor
PURCHS	ITMMAS	
	ITMIDX	Items Purchased Report by item

Enter Module From: IMMENU When Done Return To: IMMENU

Programs in Module: PURCH, ORDENT, SRTPCH, PURPRT, SRTPHS, PURRPT

Program Functions and Notes:

PURCH

Display a submenu (PCMNU) to select:

1. On-Order Entry & Editing
2. Print Items Ordered by Vendor
3. Print Items Ordered by Item
4. Purge File Through a Date

If 1 is selected, stop to ORDENT.

If 2 is selected, stop to SRTPCH.

If 3 is selected, stop to SRTPHS.

If 4 is selected, ask for purge cut-off date. Default to the system date.

Read through the PURCHS file, rewriting to the top of the file and records with a date greater than the purge date. Fill the remainder of the file with right bracket records and update the control record.

Return to the submenu.

Allow end (press the END key) from the submenu. If the END key is pressed, return to IMMENU.

ORDENT

A standard transaction entry (add mode only) program which reads:

1. Item #
2. Vendor
3. Location
4. P.O. Number
5. Order (quantity)
6. Expected Receipt Date

Calculate and display P.O. weight and total weight. Display quantities on-hand, allocated, on-order, and reorder level.

Purchase Orders may be added only. Changes to orders may be made only to fields and only at the time of initial entry. A negative order may be entered to cancel an existing order, but you may not cancel (delete) or change an existing order.

Allow "FIELD # TO CHANGE".

Stop to PURCH.

SRTPCH

Sort the PURCHS file in order by vendor, P.O. number, and item number.

Stop to PURPRT.

PURPRT

Write a standard print program of items purchased in order by vendor. Ask for starting and ending vendor number and desired location. Accept END key in Vendor Number field to end. Chain to PURCH.

SRTPHS

Sort the PURCHS file in order by item number, year, month and day, and vendor.

Stop to PURRPT.

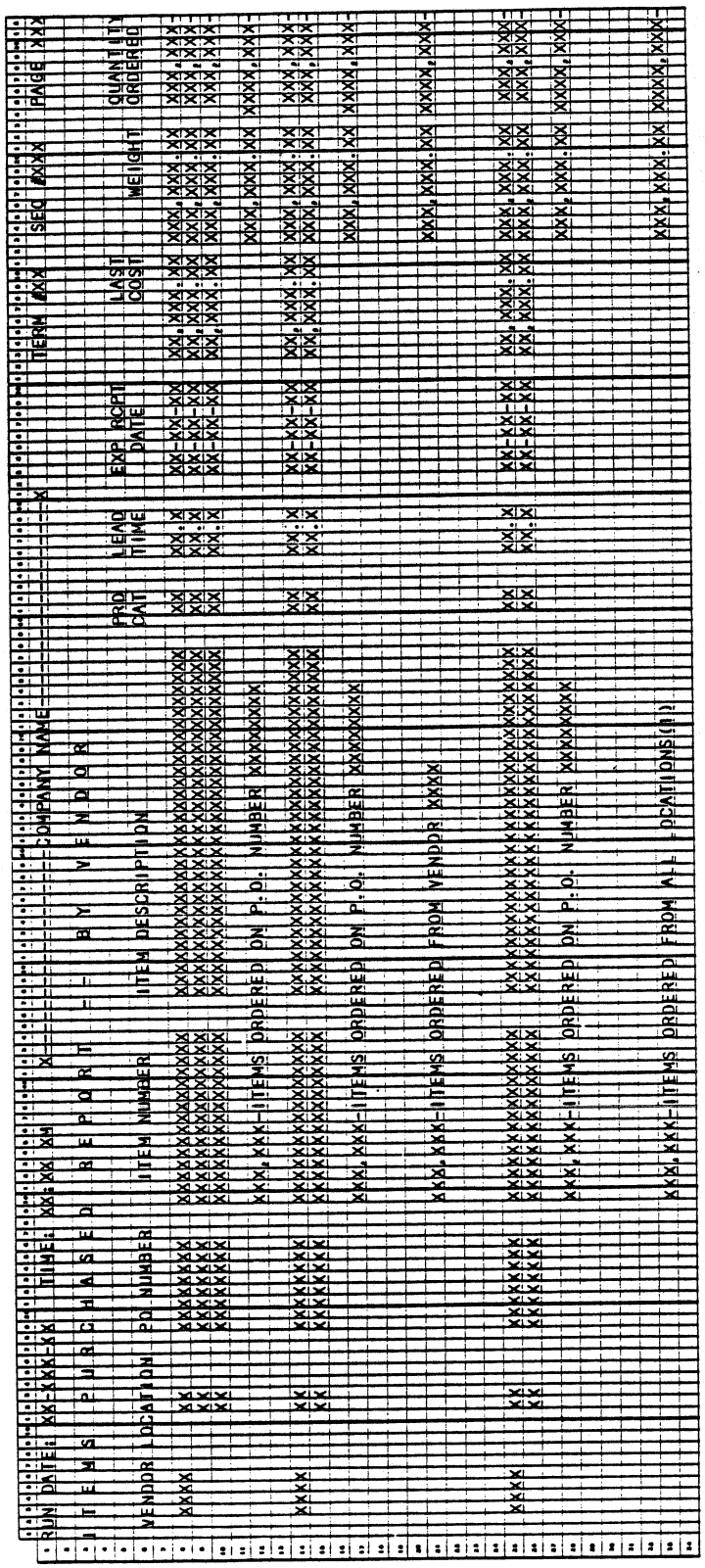
PURRPT

Write a standard print program of items purchased in order by item.

Ask for starting and ending item number and desired location. Accept END key in Item Number field to end, then chain to PURCH.

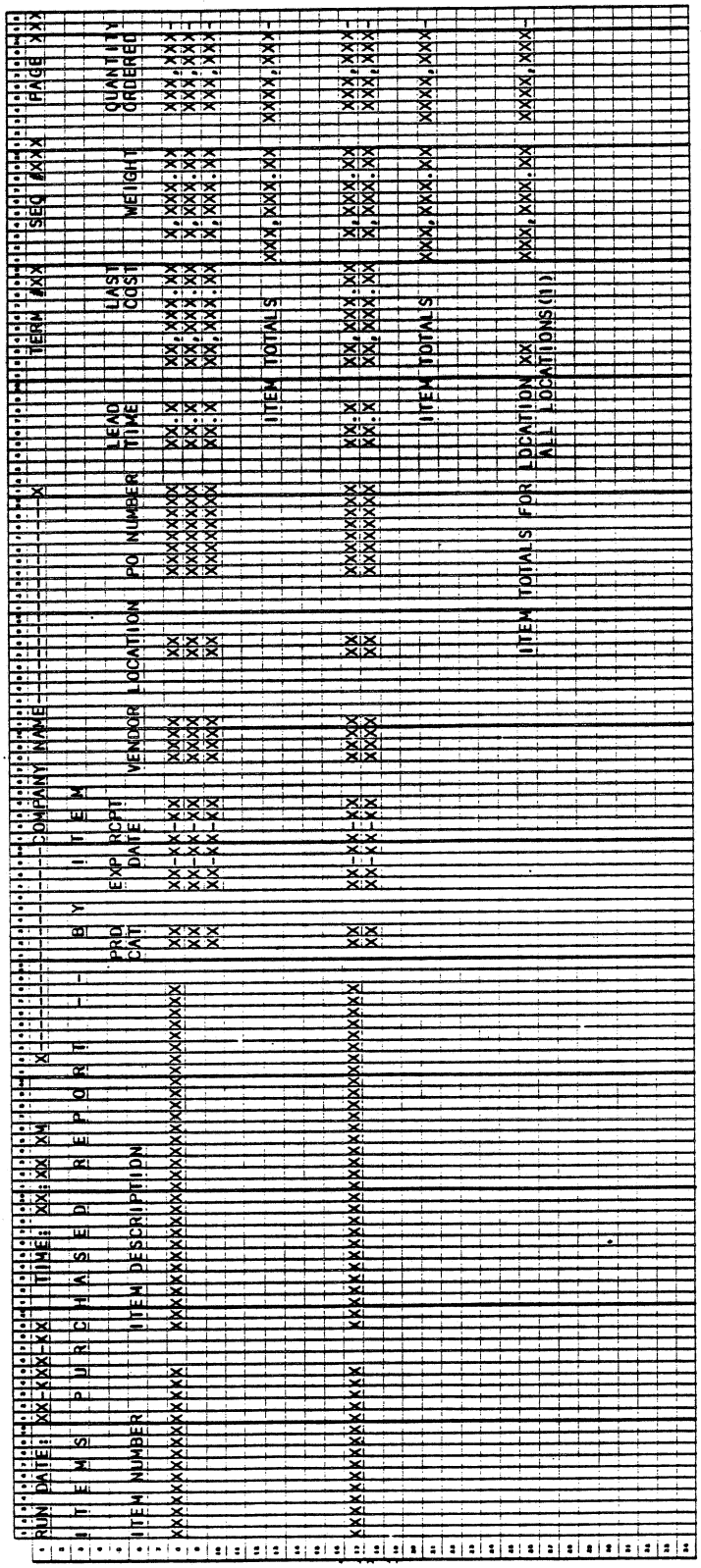
PURCHASE ENTRY & EDITING APPLICATION
DIBOL JUN-84
REPORT FORMATS

Program: PURPRT



Remarks: QUANTITY ORDERED MAY BE NEGATIVE.
1) "LOCATION XX" REPLACES "ALL LOCATIONS" WHERE SPECIFIC LOCATION IS INPUT.

Program: PURRPT



SCREEN FORMATS

Program: STKSTS

```
 1  STOCK STATUS INQUIRY
 2
 3
 4    1. ITEM #          XXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 5
 6    2. LOCATION        XX
 7
 8      ON-HAND          XXX,XXX,XXX-
 9
10      ALLOCATED        XXX,XXX,XXX
11
12      ON-ORDER         XXX,XXX,XXX-
13
14      REORDER LEVEL    XX,XXX
15
16
17
18
19
20
21
22
23
24 INQUIRE ANOTHER LOCATION ?  X
```

PROGRAM SPECIFICATIONS

Function: Display inventory stock status on the screen.

Input: KBD
ITMMAS
ITMIDX

Files Updated: None

Output: None

Enter Module From: IMMENU

When Done Return To: IMMENU

Programs in Module: STKSTS

Program Functions and Notes:

STKSTS

Display a screen as per the Screen Format.

Accept an item number. If the item is found automatically, display description.

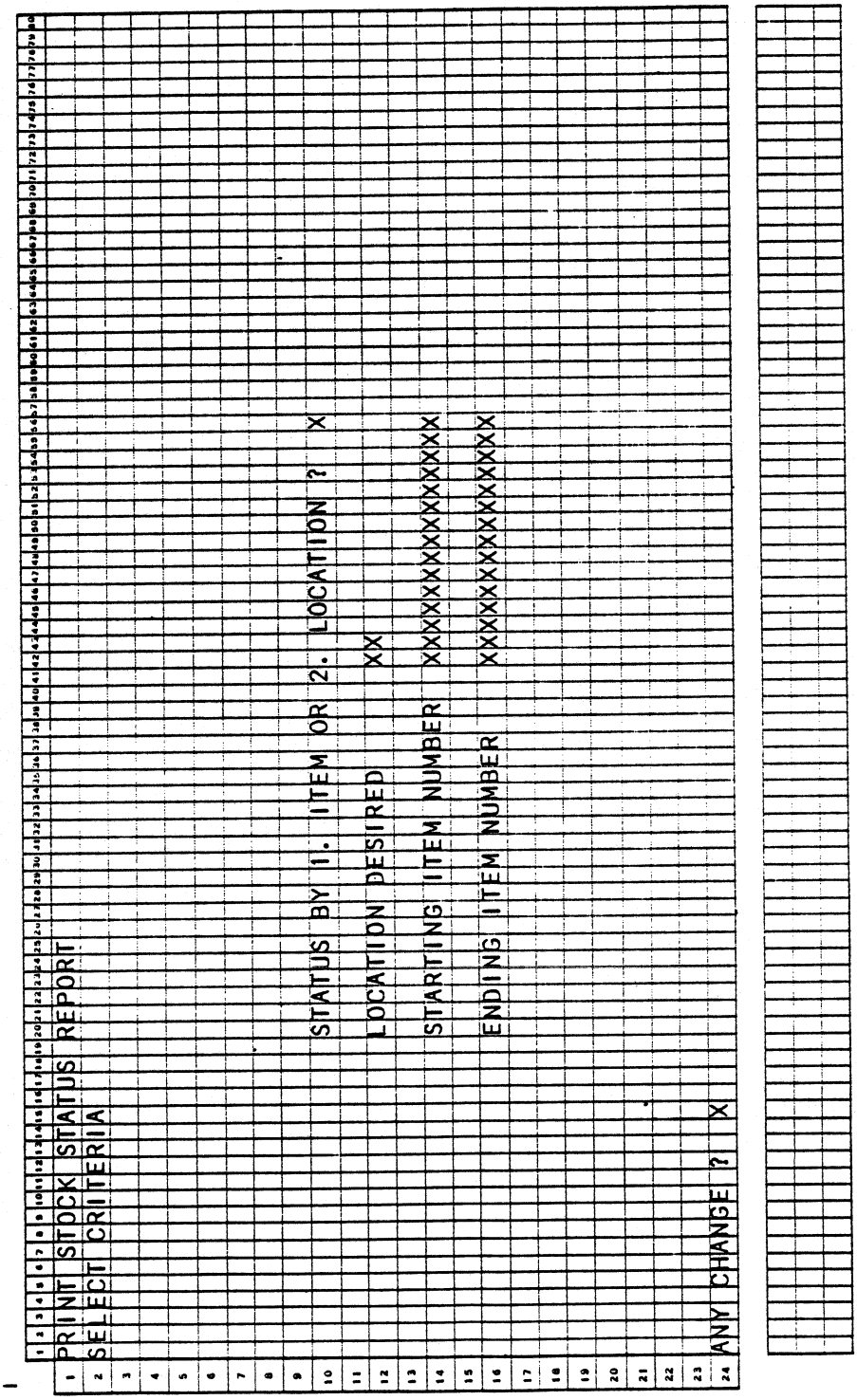
Accept a location. If blanks are input or the RETURN key is pressed for this location, redisplay the word "ALL" and sum all active on-hands, allocateds, and on-orders and display those quantities. Only displays the default reorder level.

If one location is input, scan the record for that location. When it is found, display the above fields for that location including reorder level.

Accept change to location or item number.

SCREEN FORMATS

Program: PSTKST



Remarks: STARTING AND ENDING ITEM NUMBER IS DISPLAYED ONLY IF #1, STATUS BY ITEM, IS SELECTED.

PRINT STOCK STATUS REPORT APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Prints Inventory Stock Status Reports.

Input: KBD
ITMMAS
ITMIDX

Files Updated:

Output: Stock Status Report by Item
Stock Status Report by Location

Enter Module From: IMMENU

When Done Return To: IMMENU

Programs in Module: PSTKST, ITMSTS, LOCSTS

Program Functions and Notes:

PSTKST

Display a screen per the Screen Format.

Accept a choice of one or the other reports at each sitting. Accept a location or, if the by item report was selected, blanks default to all locations. Accept a starting and ending item number for the report.

Assume the file ITMIDX is in order and read it sequentially to find the starting item number.

ITMSTS

Print program to output the Stock Status Report by Item per the Report Format. Test for items below reorder (REO) level or out-of-stock (O/S) and indicate the results on the report. If the quantity on-hand minus the quantity allocated is less than or equal to the reorder level, print "REO", and if it is negative, print "O/S" under the stock status heading on the report.

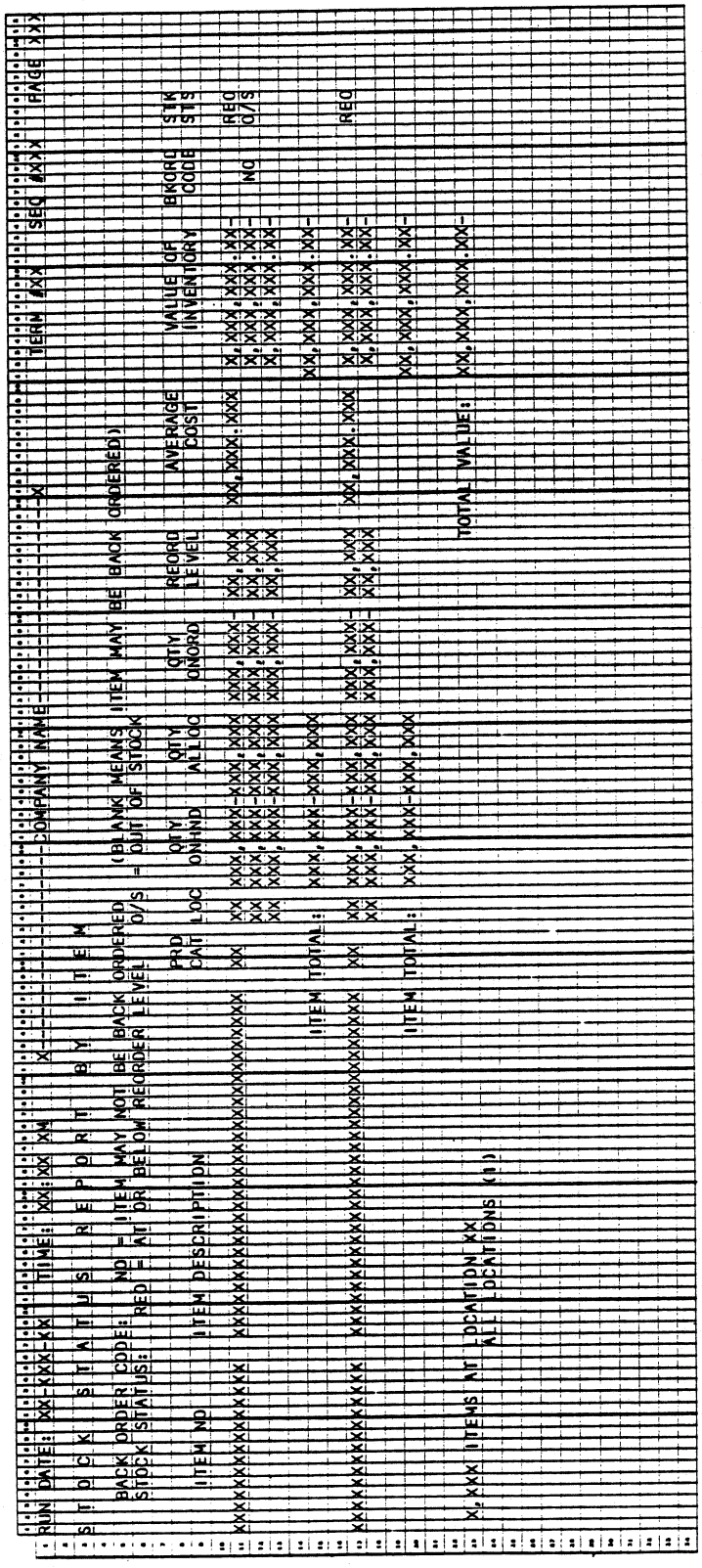
Do not print the item total line if only one location is reported.

LOCSTS

Print program to output the Stock Status Report by Location per the Report Format. Same calculations for below reorder level and out-of-stock as in ITMSTS. Print the result on the report as in ITMSTS.

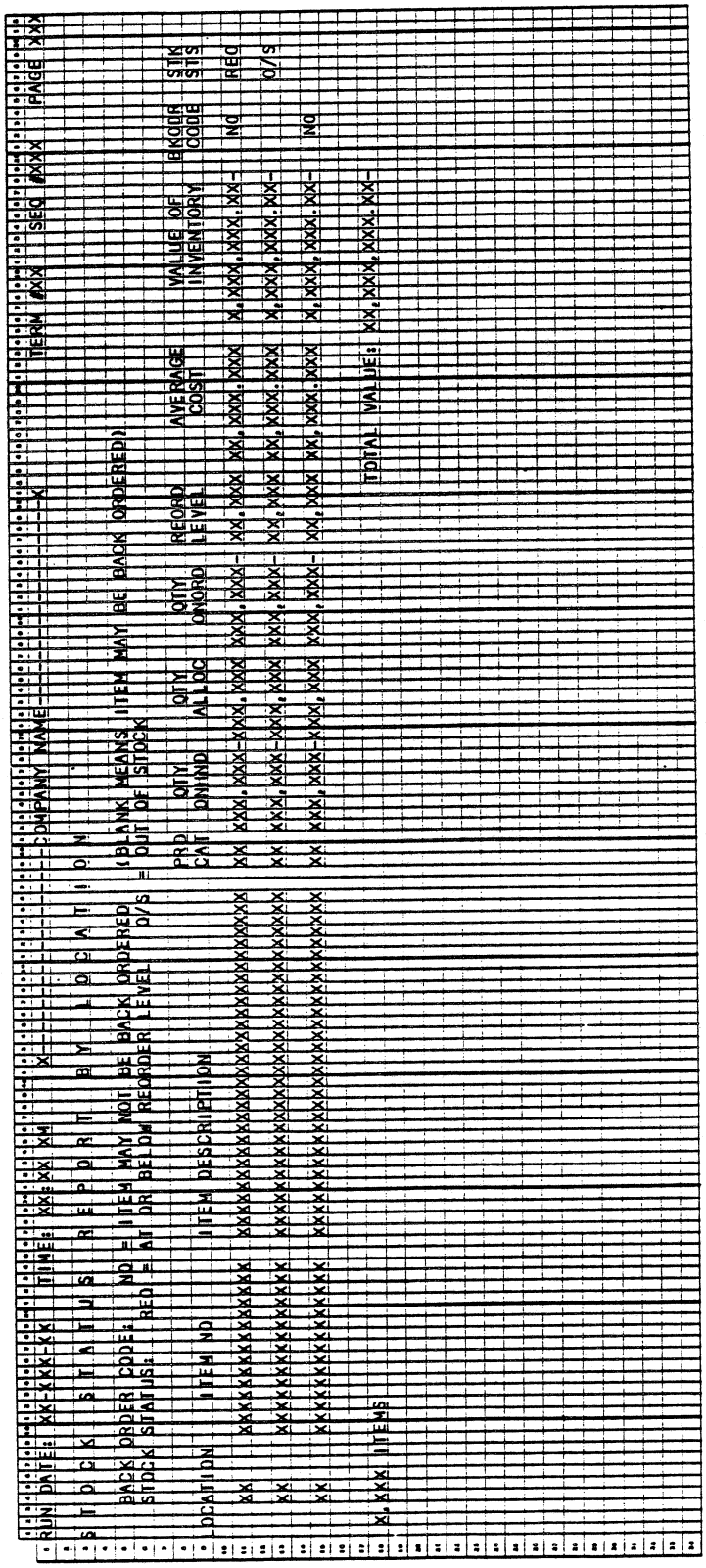
PRINT STOCK STATUS REPORT APPLICATION
DIBOL JUN-84
REPORT FORMATS

Program: ITMSTS



Remarks: (1) REPLACES "LOCATION XX" WHEN "ALL" LOCATIONS REQUESTED.

Program: LOCSTS



Program: ABCRPT

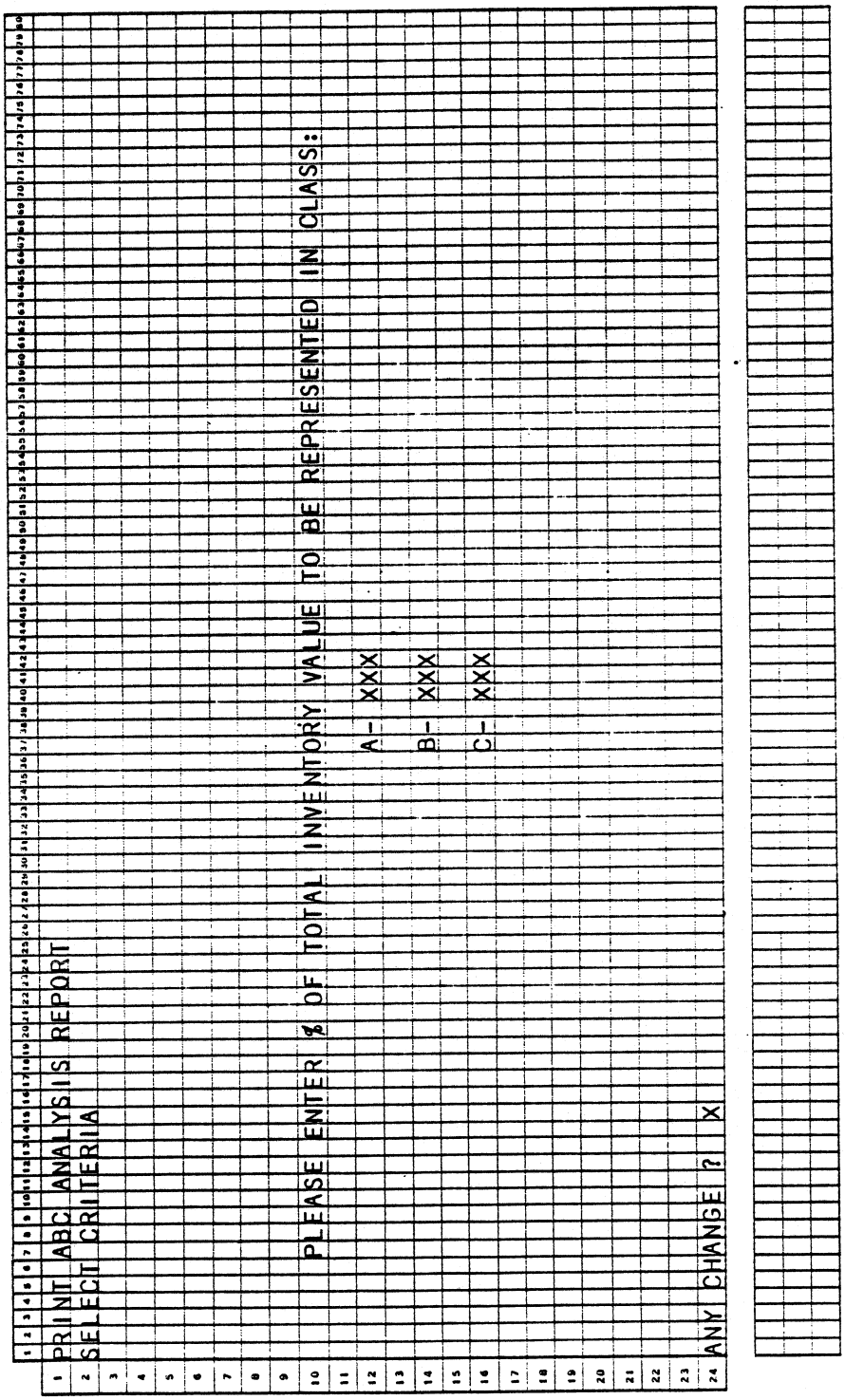
```
RUN DATE: XX-XXX-XX   TIME: XX:XX XM      X-----------------COMPANY NAME-----------------X          TERM #XX   SEQ #XXX    PAGE XXX

A B C   A N A L Y S I S   R E P O R T

CLASS A -- TOP XXX%, CLASS B -- NEXT XXX%, CLASS C -- BOTTOM XXX% OF TOTAL INVENTORY VALUE
                                                        UNITS USAGE     AVERAGE      DOLLARIZED        --INVENTORY CLASS--   DOLLARIZED USAGE
   ITEM NUMBER                  DESCRIPTION                 Y-T-D           COST          USAGE          PRESENT   SUGGESTED       SUB-TOTAL

XXXXXXXXXXXXXXX    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX     XXX,XXX    XXX,XXX.XXX XXX,XXX,XXX.XXX      X          X         XXX,XXX,XXX.XXX

XXXXXXXXXXXXXXX    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX     XXX,XXX    XXX,XXX.XXX XXX,XXX,XXX.XXX      X          X         XXX,XXX,XXX.XXX
```

Remarks: CHAINS TO ABCSET IF USER DESIRES THAT SUGGESTED INVENTORY CLASSES BE SET UP. REPORTED IN DECREASING DOLLARIZED VALUE - HIGHEST VALUE FIRST.

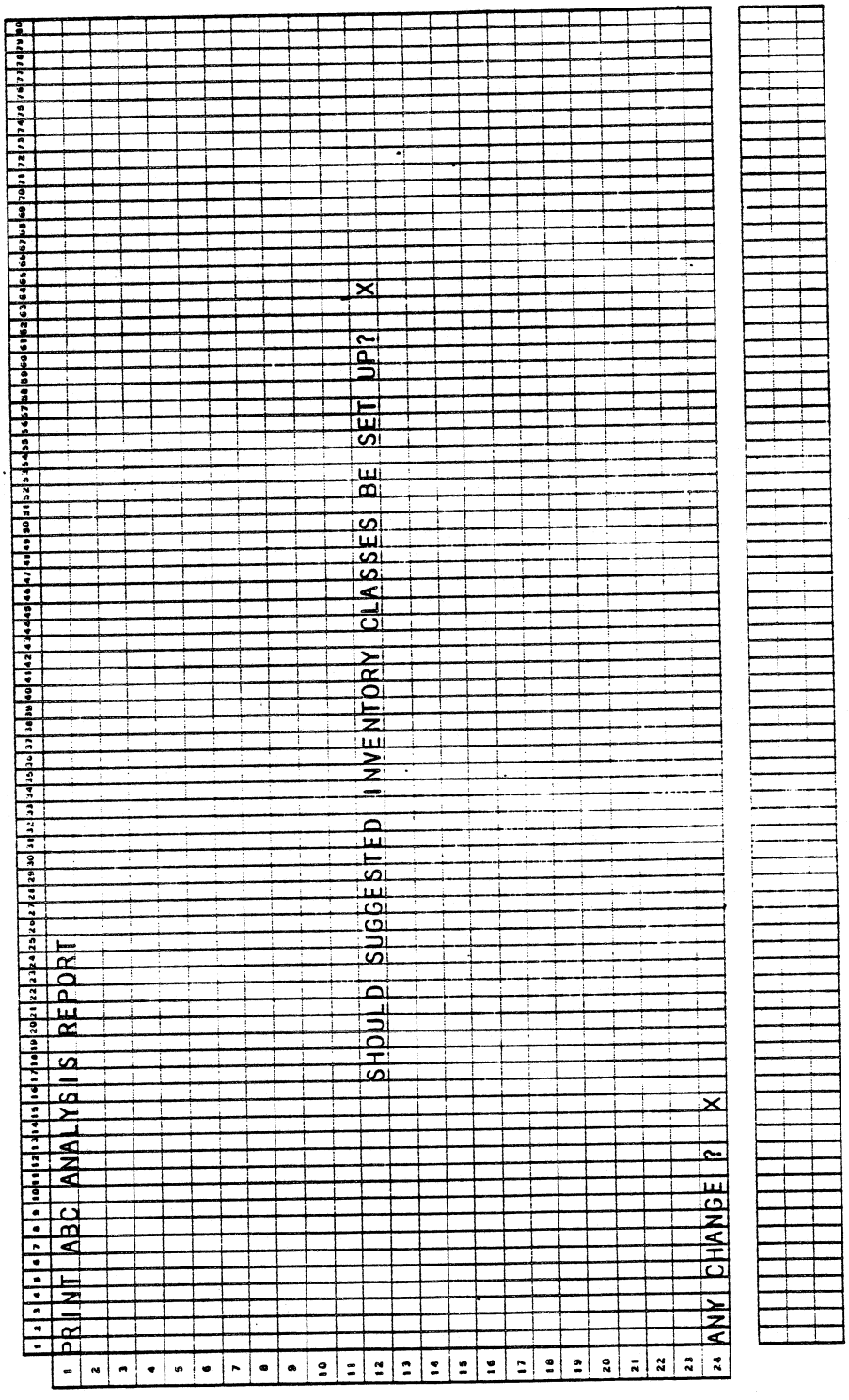
This page intentionally left blank.

SCREEN FORMATS

Program: ABCSEL



Program: ABCRPT



PRINT ABC ANALYSIS REPORT APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Ask operator for inventory class percentage, build index for report.

Input: KBD
ITMMAS
ABCIDX

Files Updated: ABCIDX
ITMMAS

Output: ABC Analysis Report

Enter Module From: IMMENU

When Done Return To: IMMENU

Programs in Module: ABCSEL, SRTABC, ABCRPT, ABCSET

Program Functions and Notes:

ABCSEL

Open and protect files. Display Select Criteria, input and edit percentage of total inventory value to be represented by inventory classes A, B, and C. (TOP n percent will be represented by class A, NEXT m percent by B, LOWEST p percent by C, in descending order of average inventory value*.)

Average Inventory Value = Average Unit Cost * Usage Year-to-Date. (NOTE: Usage Year-to-Date is updated by invoices from COP and issues from I/M.)

If no criterion is entered for inventory classes A, B or C, they default to A=80%, B=15%, and C=5%. This says: find those items whose dollarized usage (average unit cost X usage Year-to-Date) makes up the top 80% of the total dollarized usage for all items and, in a work file, label those as "A" items; find those items which contribute to the next 15% and label them "B" items; and the rest are "C" items. The rule of thumb is the "A" items will make up 20% of all items (thereby the 80-20 rule). This small, but valuable percentage of items, is theoretically easier to manage more closely than 100%.

Write control records onto the file (first is a dummy - we do not have total average value yet).

Write an index for active Item Master records. (Keep track of the Average Total Active Record's inventory value.)

Rewrite the first control record with a total average inventory value of active items.

SRTABC

Sort ABCIDX in descending order of average item inventory value.

Set top in (specified) percentage of inventory average value items to Class A, next m (also specified) percentage of items to Class B, last (100-m-n) percentage to Class A.

Delete index.

ABCRPT

Print per the Report Format. If the printer is busy, unprotect ITMMAS, delete ABCIDX.

After printing the ABC Analysis Report, ask on the screen if the user wants to update the Item Master record with the new inventory classes.

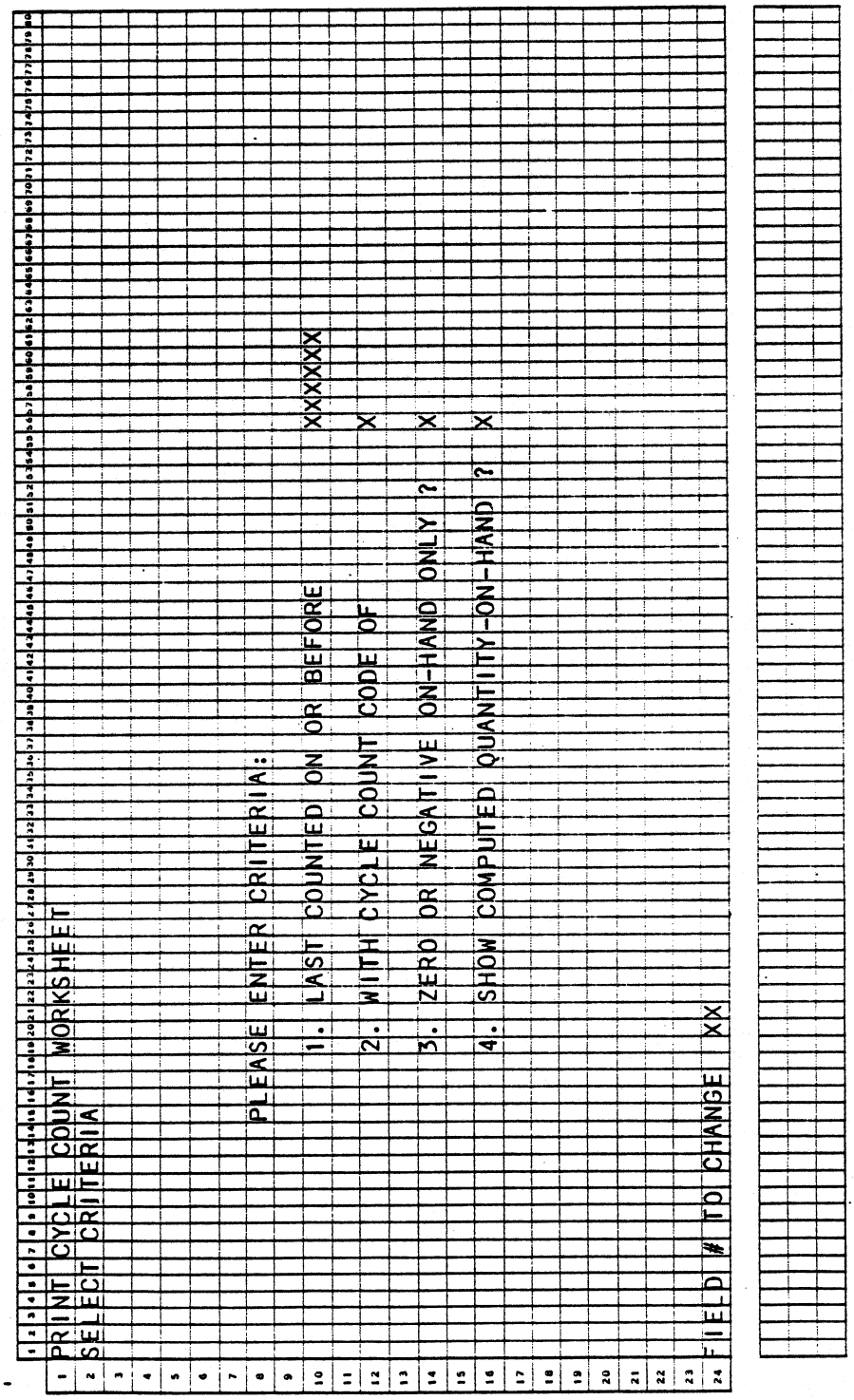
Yes - execute ABCSET
No - delete ABCIDX, unprotect ITMMAS - chain to IMMENU

ABCSET

Reset the Inventory Class field for each item reported on the ABC Analysis Report.

SCREEN FORMATS

Program: CYCSEL



PRINT CYCLE COUNT WORKSHEET APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: To provide for selection criteria; to read in that criteria; then cull out those items that fit the criteria and print them in a worksheet format. This worksheet is expected to be used as a turnabout document to cycle count the items on it. It will then be used as input to the Physical Count Entry and Editing program.

Input: KBD
ITMMAS

Files Updated: CYCIDX

Output: Cycle Count Worksheet

Enter Module From: IMMENU

When Done Return To: IMMENU

Programs in Module: CYCSEL, SRTCYC, CYCRPT

Program Functions and Notes:

CYCSEL

Open and protect files. Display a screen per the Screen Format.

Build the cycle count index accordingly.

Only include active Item Master records (at the default location) that meet all three selection criteria.

The default location is in the Item Master control record.

SRTCYC

Sort the Cycle Count Index in ascending order by part bin number.

CYCRPT

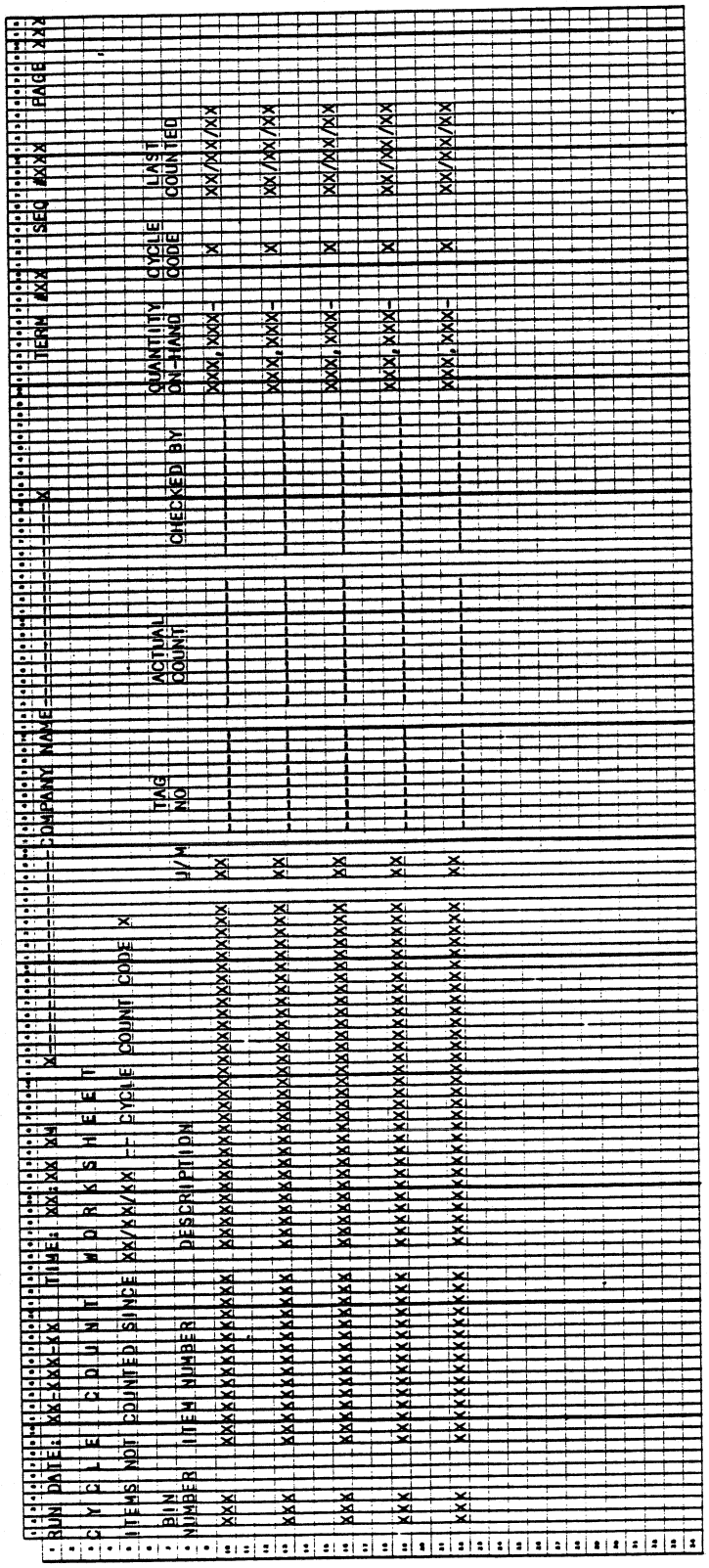
Print the Cycle Count Worksheet according to the Report Format.

If a printer is not available, the operator will not wait. Run UNRCYC to delete the index, unprotect the item master, return to menu.

Delete the index after running the report.

PRINT CYCLE COUNT WORKSHEET APPLICATION
DIBOL JUN-84
REPORT FORMATS

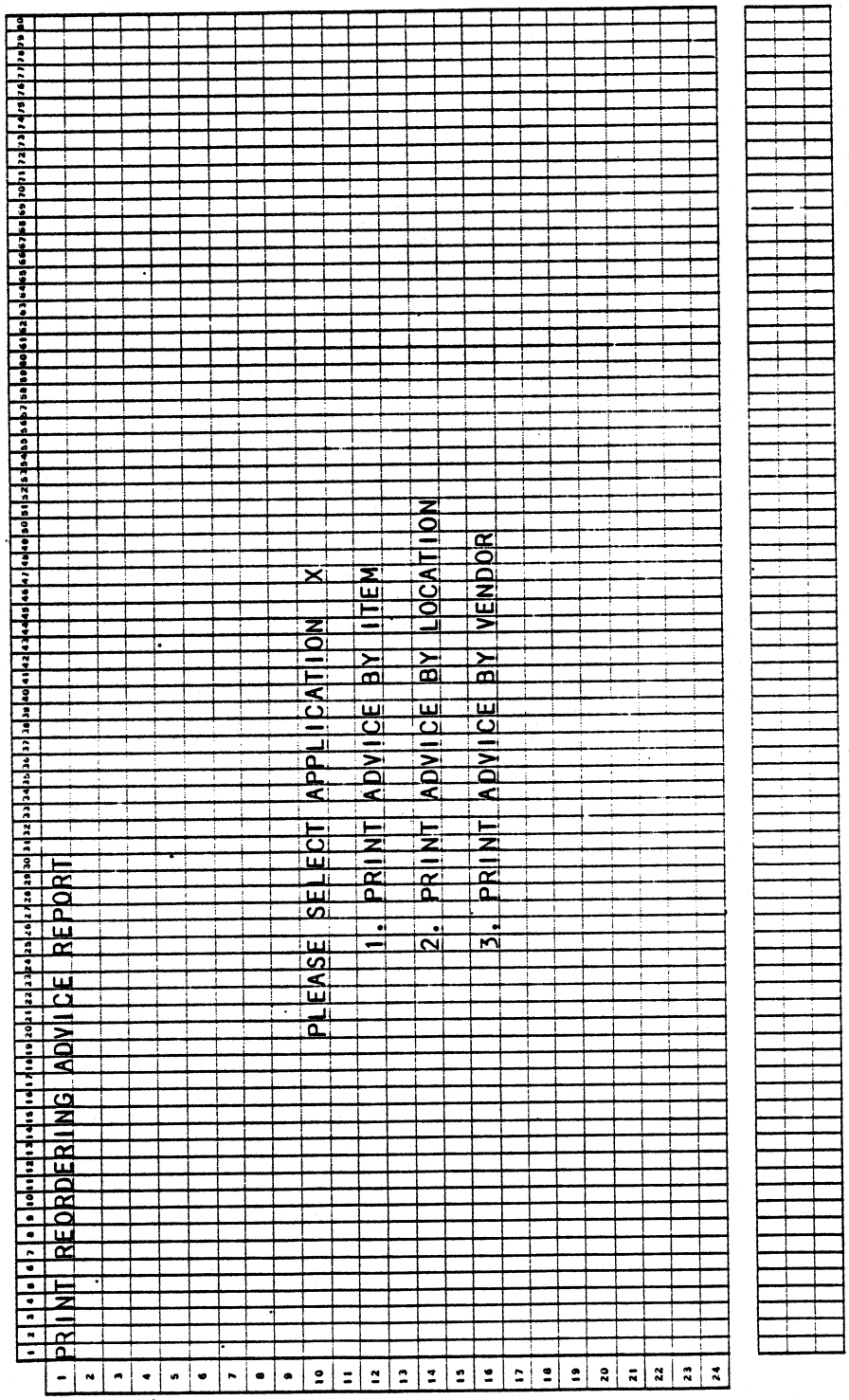
Program: CYCRPT



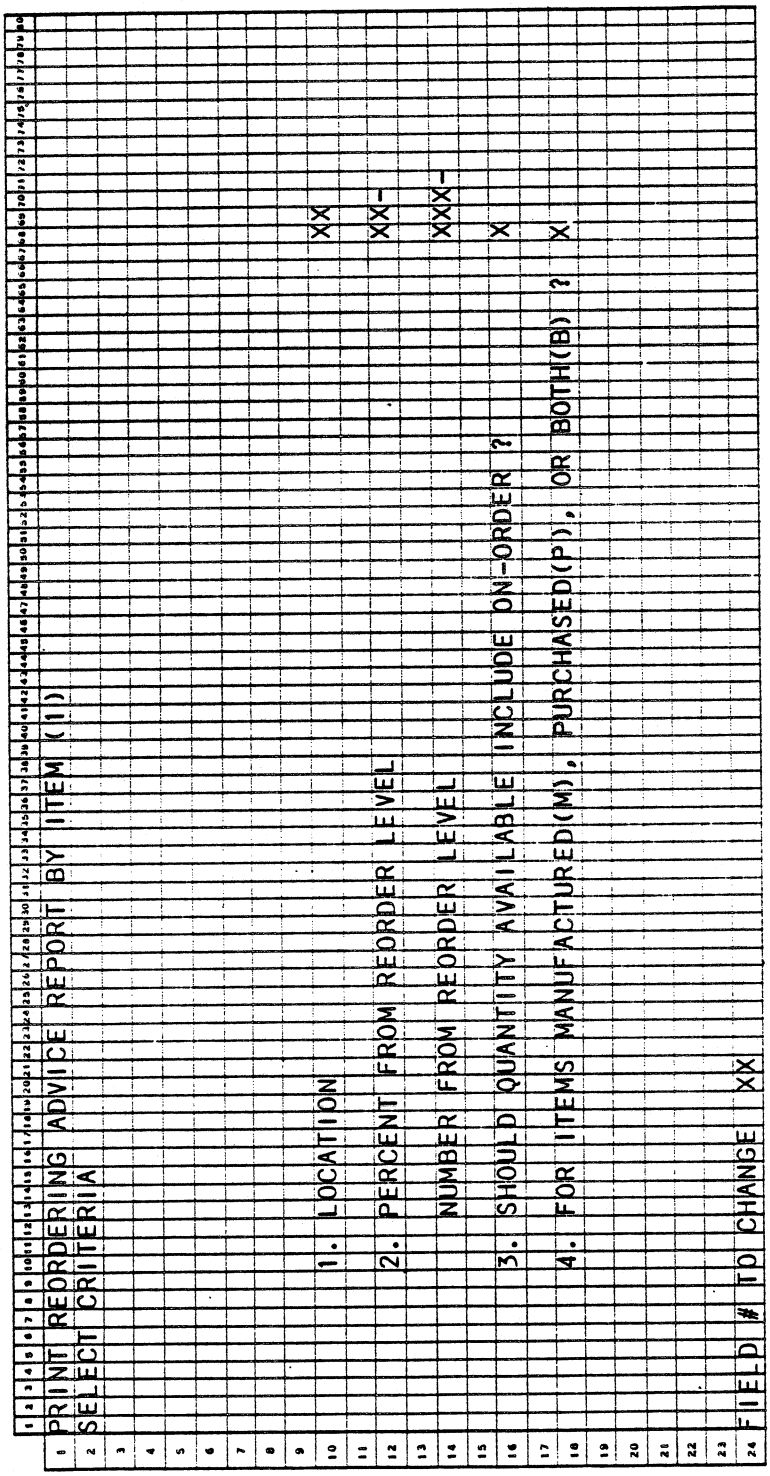
This page intentionally left blank.

SCREEN FORMATS

Program: ADVICE (AMENU)

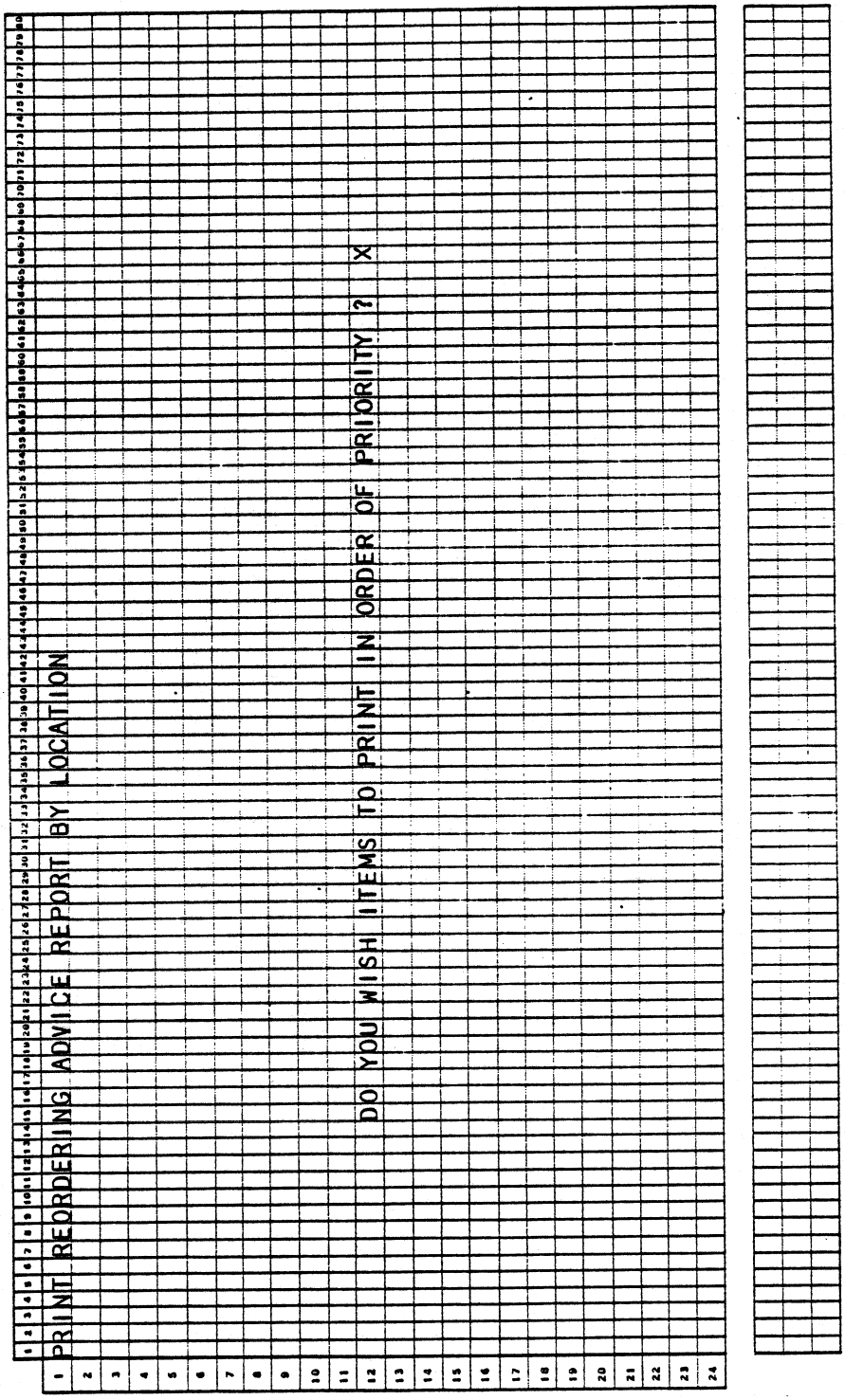


Program: ADVICE (AMENU)

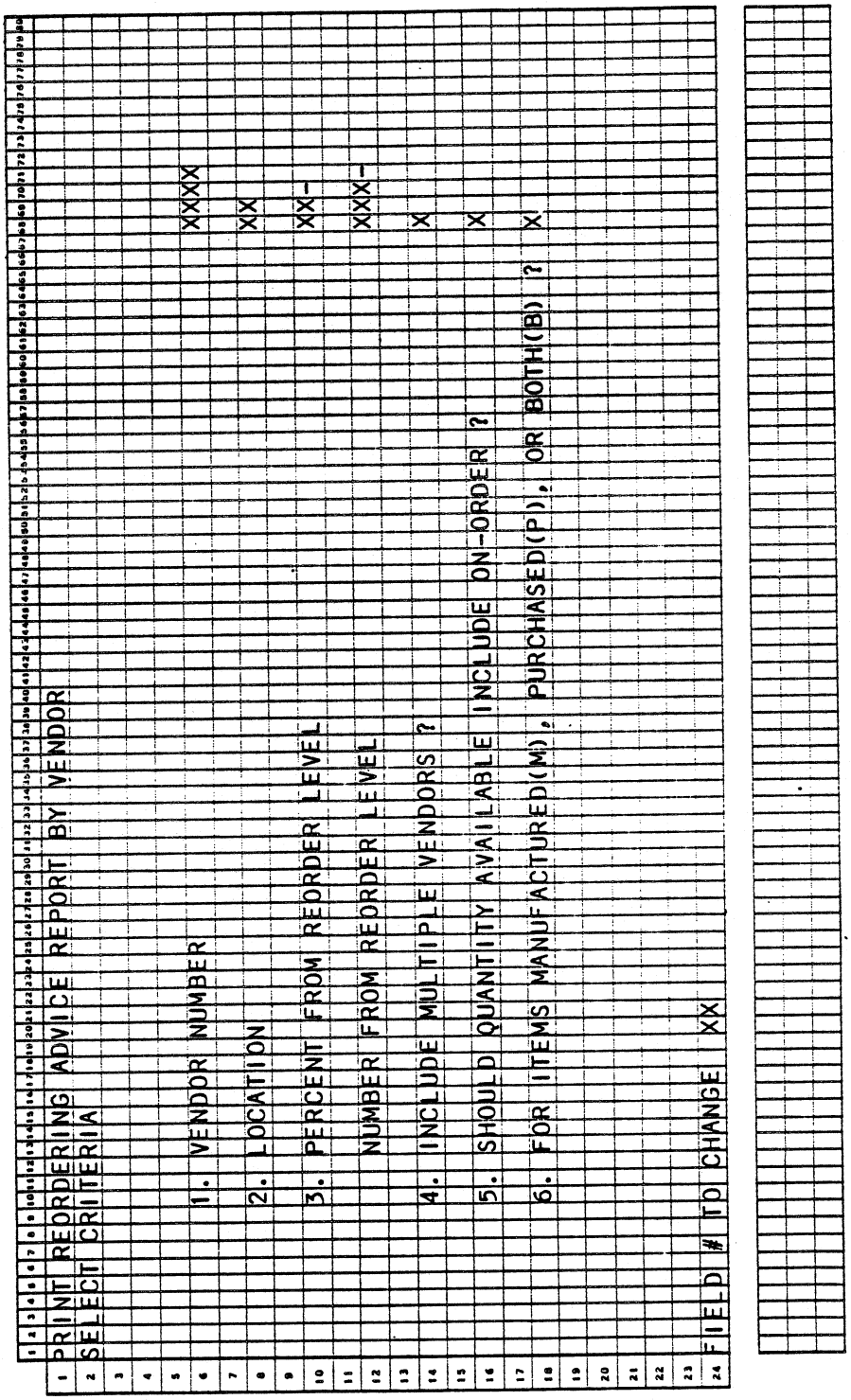


Remarks: RETURN DEFAULTS TO "ALL" FOR LOCATION AND TO "B" FOR #4. DISPLAYS ONLY IF ADVICE BY ITEM OR BY LOCATION WAS SELECTED.
(1) DISPLAYS "LOCATION" WHEN #2 IS SELECTED FROM THE PRIOR SUBMENU.

Program: ADVICE



Program: ADVICE (ITMSL (ADVEN))



Remarks: RETURN DEFAULTS TO "ALL" FOR LOCATION AND TO "B" FOR #6. DISPLAYS ONLY IF ADVICE BY VENDOR IS SELECTED.

PRINT REORDERING ADVICE REPORT APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Provides a report that shows what items need to be reordered i.e., purchase and/or manufacturing orders need to be originated.

Input: KBD
ITMMAS
ITMIDX

Files Updated: SELINV

Output: Reorder Advice Report by Vendor
Reorder Advice Report by Item
Reorder Advice Report by Location

Enter Module From: IMMENU

When Done Return To: IMMENU

Programs in Module: ADVICE, (AMENU), (ADVEN), (1 TMSL), SRTADV, ADVPRT, STADVL, STAVLC, ADVLOC

Program Functions and Notes:

ADVICE

Subroutine AMENU displays a submenu to select a Reordering Advice Report by Item, Location or Vendor. It also handles the criteria selection screens for reports by item and location.

When Reordering Advice Report by Vendor is selected in AMENU, subroutine ADVEN handles the criteria selection screen.

Subroutine ITMSL builds the SELINV file for printing the Reordering Advice Report by Vendor.

Items not stocked or not controlled are ignored. Items that do not have an Order Policy code of "R" are ignored.

The Adjusted Reorder Level = Reorder Level + (entered %) (Reorder Level)
or, Adjusted Reorder Level = Reorder Level + (entered amount)

Note: Reorder level is synonymous with reorder point.

Compute a Priority = $\frac{\text{Adjusted Reorder Level - Availability}}{\text{Adjusted Reorder Level}}$

where: Available = On-Hand + On-Order - Allocated

If the entered location does not equal the default location, compare the adjusted reorder level against the quantity available, by item. Print this location only.

If the entered location is equal to "ALL", sum the total availability and compare with the default location's adjusted reorder level. Print all locations with item total. For "ALL" locations, the total availability is ("ALL" locations on-hand) less ("ALL" locations allocations) plus (the default location's on-order) - by item. (It is assumed a remote location's "on-order" is on order from the default location.)

If the selected location is equal to the default location, it is treated as "ALL" locations.

Print exceptions only on the Reordering Advice by Item Report or build the SELINV file if report by location or vendor was selected.

SRTADV

Sort SELINV, created in ADVICE, by the multiple vendor factor and record number of the ITMMAS file.

ADVPRT

Print the Reordering Advice Report by Vendor using the SELINV file.

STADVL

Sort SELINV, created in ADVICE, by location only.

STAVLC

Sort SELINV, created in ADVICE, by location and priority (see prior page).

ADVLOC

Print the Reordering Advice Report by Location using the SELINV file.

Recommended Minimum Order Quantity (REQMIN) is read out of the Item Master record if the reorder level is equal to available quantity.

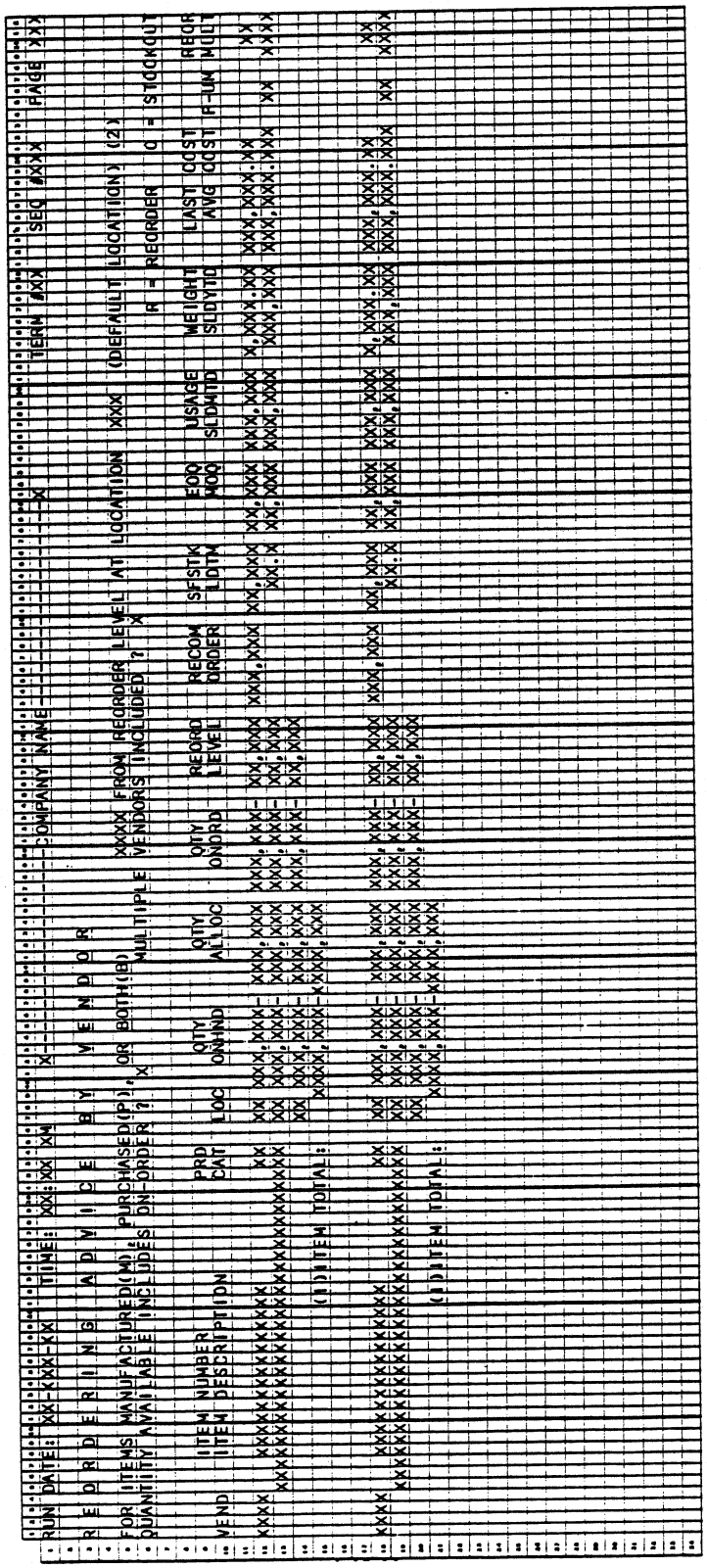
However, if available quantity is below Reorder Level, the REQMIN will be increased by the difference between these two. Delete the SELINV file.

Default Location Consideration:

For the default location, available is calculated using the sum of all locations on-hand quantities less all locations allocation quantities. If the user requests on-order be added into available, only the default location's on-order is used. Satellite or remote warehouse on-order quantities are assumed to be on-order from the default location.

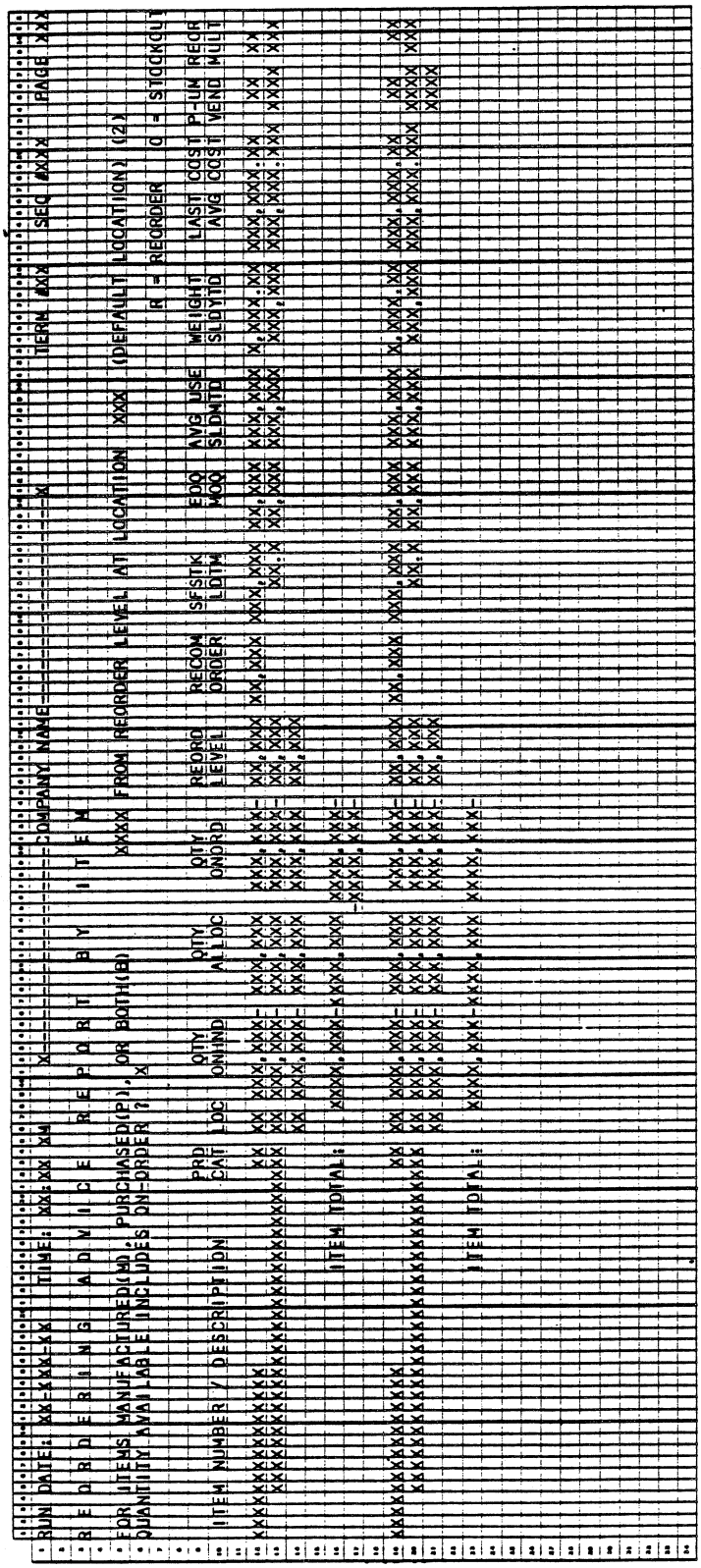
PRINT REORDERING ADVICE REPORT APPLICATION
DIBOL JUN-84
REPORT FORMATS

Program: ADVPRT

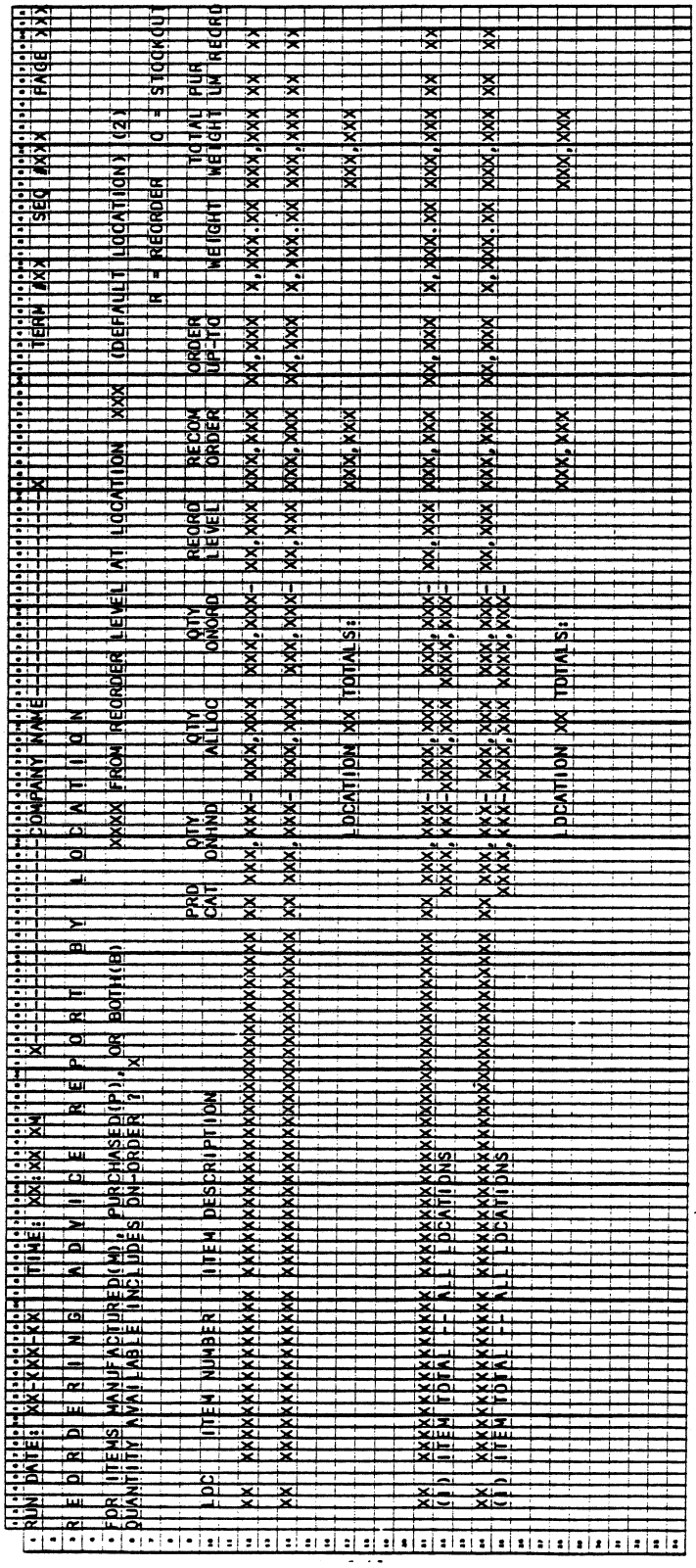


Remarks: SEE ITMSL AND ADVEN FOR REPORT ITEM SELECTION.
(1) ONLY PRINTS IF "ALL" OR DEFAULT LOCATION IS CHOSEN.
(2) ONLY PRINTS IF DEFAULT LOCATION IS CHOSEN.

Program: ADVICE



Program: ADVLOC



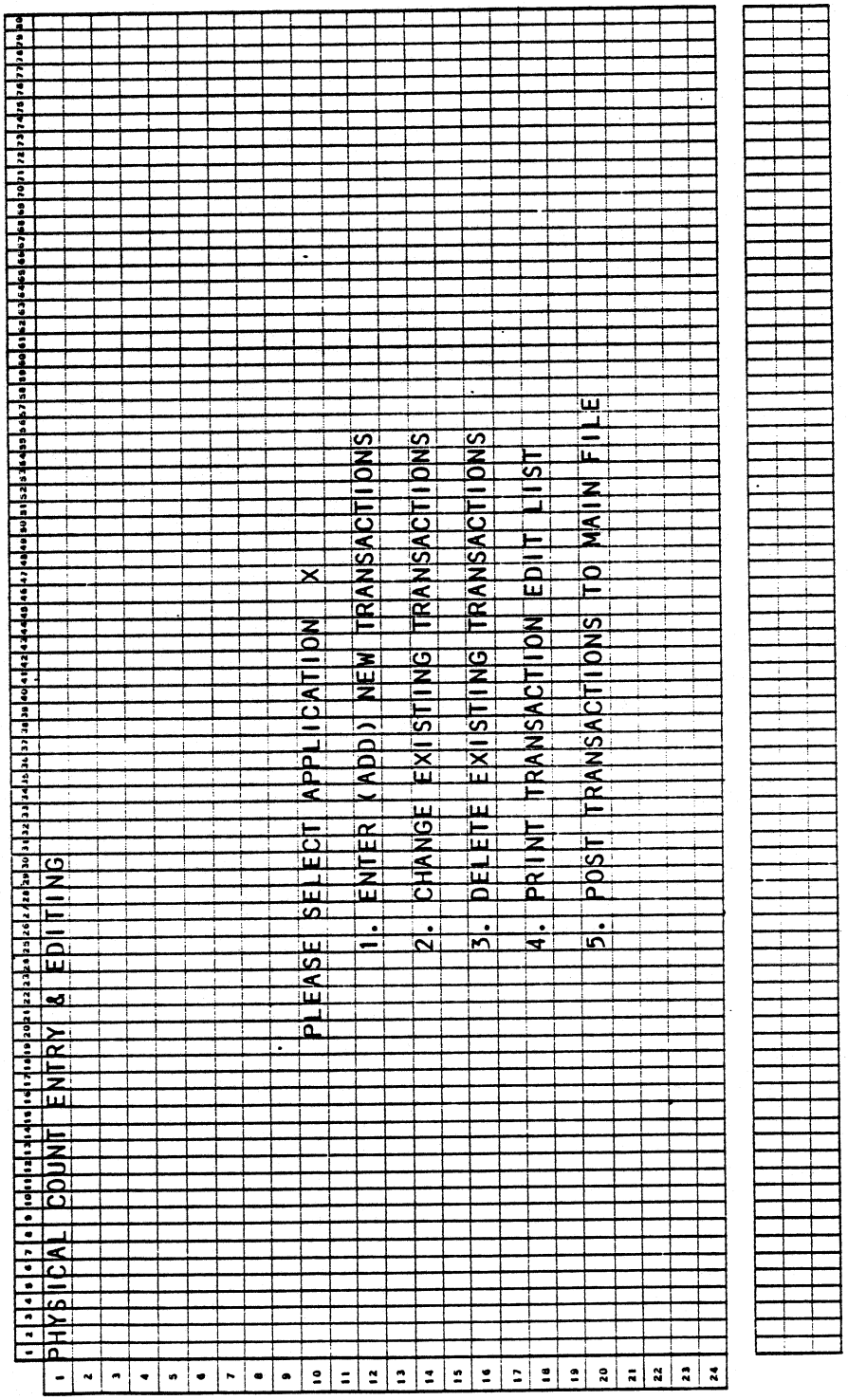
Remarks: (1) ITEM TOTAL LINE WILL PRINT ONLY IF LOCATION IS THE DEFAULT LOCATION.
2) ONLY PRINTS IF LOCATION IS DEFAULT LOCATION.

This page intentionally left blank.

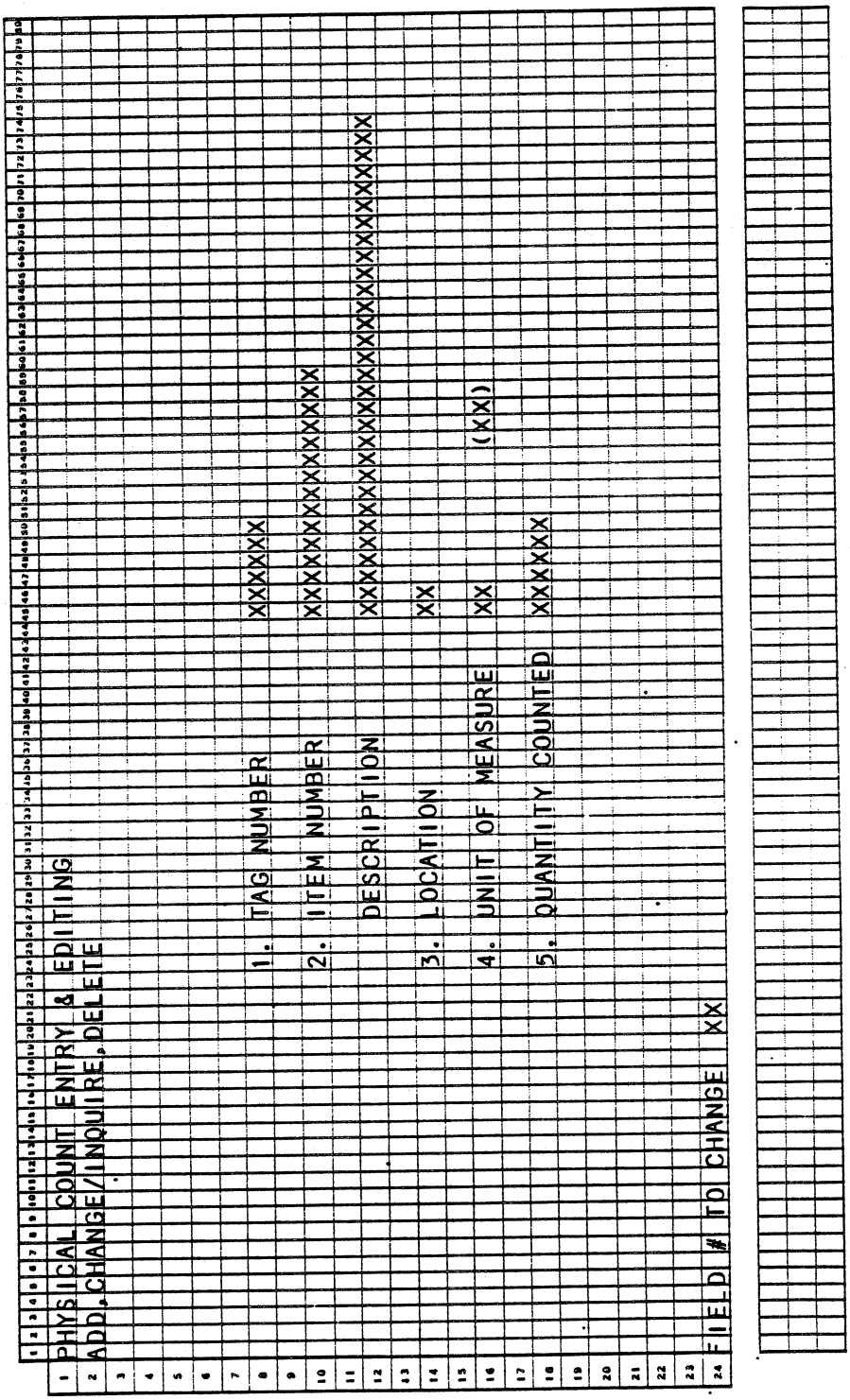
INVENTORY MANAGEMENT PACKAGE
PHYSICAL COUNT ENTRY & EDITING APPLICATION
DIBOL JUN-84

SCREEN FORMATS

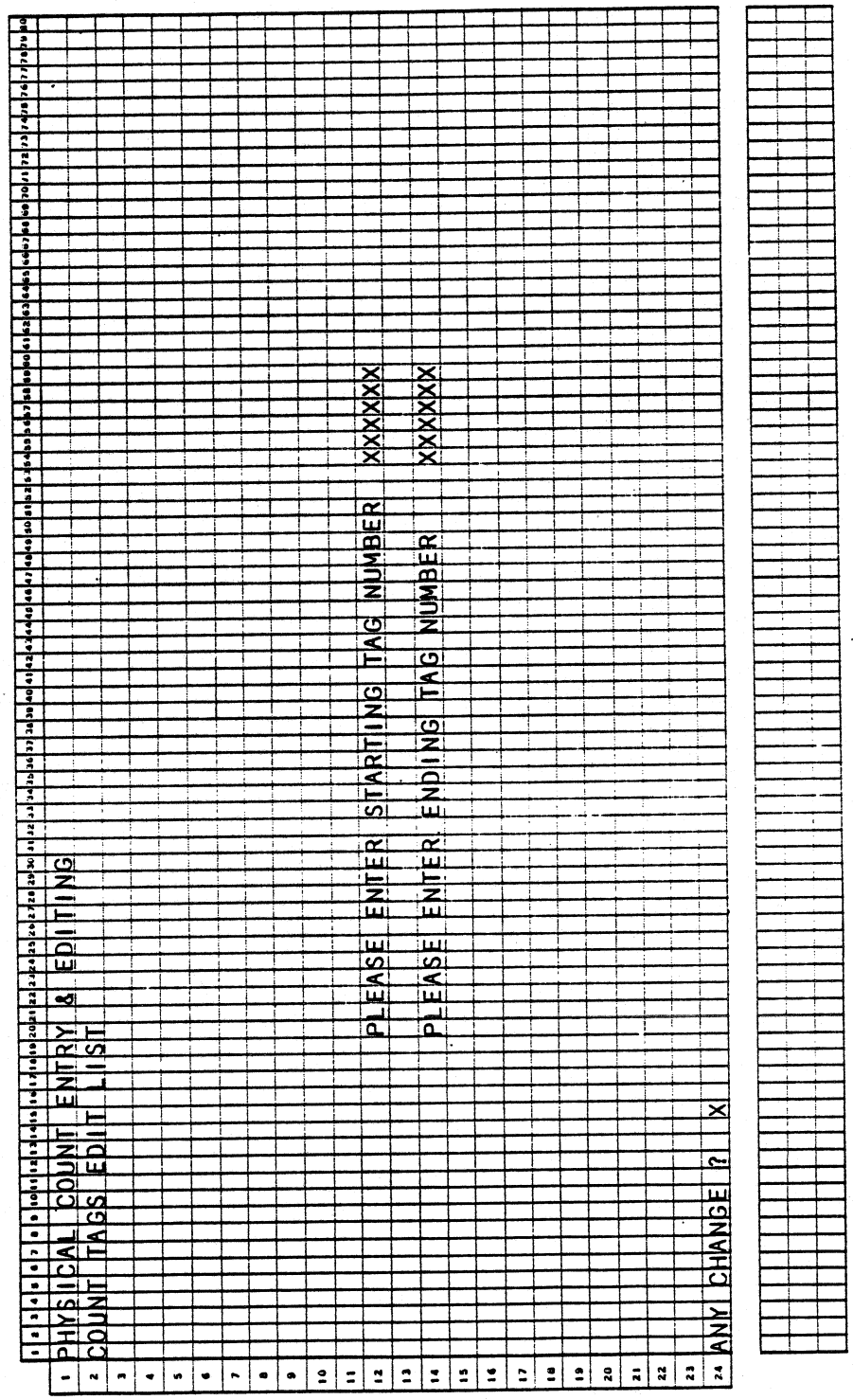
Program: TAGMNT (TMENU)



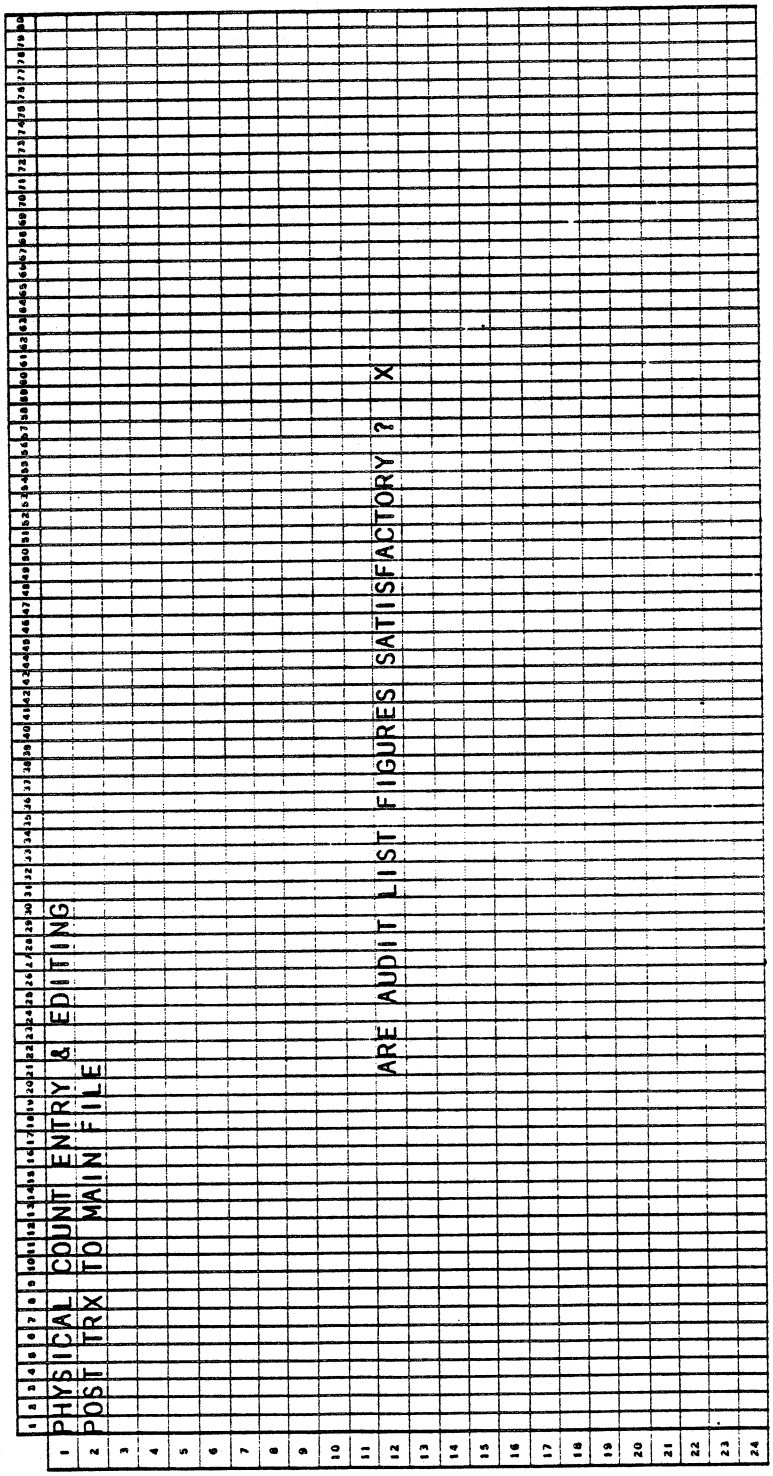
Program: TAGMNT



Program: TAGEDT (STENO)



Program: TAGAUD



Remarks: IF "Y", CHAIN TO PSTTAG - UPDATE ITMMAS ON-HAND FIELD WITH TAG COUNTS.
IF "N", CHAIN TO SORT - PUT TAGS BACK INTO TAG NUMBER SEQUENCE.

PHYSICAL COUNT ENTRY & EDITING APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Allows entry and editing of on-hand values by item and by location from either physical inventory counts or cycle counts.

Input:	Files Updated:	Output:
KBD	INVTAG	Count Tags Edit List
INVTAG	ITMMAS	Physical Count Update
ITMIDX		Audit List
ITMMAS		

Enter Module From: IMMENU When Done Return To: IMMENU

Programs in Module: TAGMNT, SRTTAG, ORGTAG, UNPRTG, TAGEDT, STTGBI, TAGAUD, PSTTAG, CLRTAG

Program Functions and Notes:

TAGMNT

Standard Transaction File Maintenance allowing add, change, delete, edit list, and post functions. Accept only numeric inventory tag numbers, and treat each one as if it were unique. Verify the item number, location, and unit of measure against the Item Master file. If the number of records flagged for deletion exceeds 200, go to ORGTAG. If Post is selected, protect INVTAG, ITMMAS, increment user count of ITMIDX, and chain to STTGBI.

SRTTAG

Sort the INVTAG file into ascending sequence by tag number.

ORGTAG

Remove deleted records in INVTAG and reorganize the file. Return to TAGMNT upon completion.

UNPRTG

Update the control record counts and remove file protection after the sort. Receive message of next program to chain to and chain to next program.

TAGEDT

An edit list of tags. Print one line per item. Flag each break in tag sequence numbers. See Report Format.

STTGBI

Sort the INVTAG file into order by location within item number.

TAGAUD

Report each tag per the Report Format. Show the old/new quantity on-hand and the old/new dollar value of inventory. Do not allow posting for invalid item numbers or dissimilar units of measure. If no match is found for the location, assume the first location of the array. Flag multiple counts at the same location as a warning. Ask the question "ARE REPORT FIGURES SATISFACTORY ?" at the report end. If the operator accepts the figures, chain to PSTTAG to post the tags to the Item Master file. If not, then chain to SRTTAG to resort the INVTAG file back into tag sequence number for possible further editing.

PSTTAG

Post (replace) the quantity on-hand amounts within the Item Master file with the values obtained from each INVTAG record. Upon completion of the posting process, unprotect ITMMAS, decrement ITMIDX user count, and chain to CLRTAG.

CLRTAG

Clear the INVTAG file to all bracket records. Reset the control record counts to correspond to an empty file.

PHYSICAL COUNT ENTRY & EDITING APPLICATION
DIBOL JUN-84
REPORT FORMATS

Program: TAGEDT

```
RUN DATE: XX-XXX-XX   TIME: XX:XX XM   X--------------COMPANY NAME--------------X   TERM #XX   SEQ #XXX   PAGE XXX

C O U N T   T A G S   E D I T   L I S T

TAG NUMBER   ITEM NUMBER       DESCRIPTION                      LOCATION   UNIT-OF-MEASURE   QUANTITY COUNTED
XXXXXX       XXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XX         XX                XXX,XXX
XXXXXX       XXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XX         XX                XXX,XXX
(1) ******   TAG SEQUENCE BREAK
XXXXXX       XXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XX         XX                XXX,XXX
             TOTAL NUMBER OF TAGS THIS RUN XXX,XXX
```

Remarks: (1) THIS PRINTS IF THERE IS A SEQUENTIAL BREAK IN THE TAG NUMBERS.

Program: TAGAUD

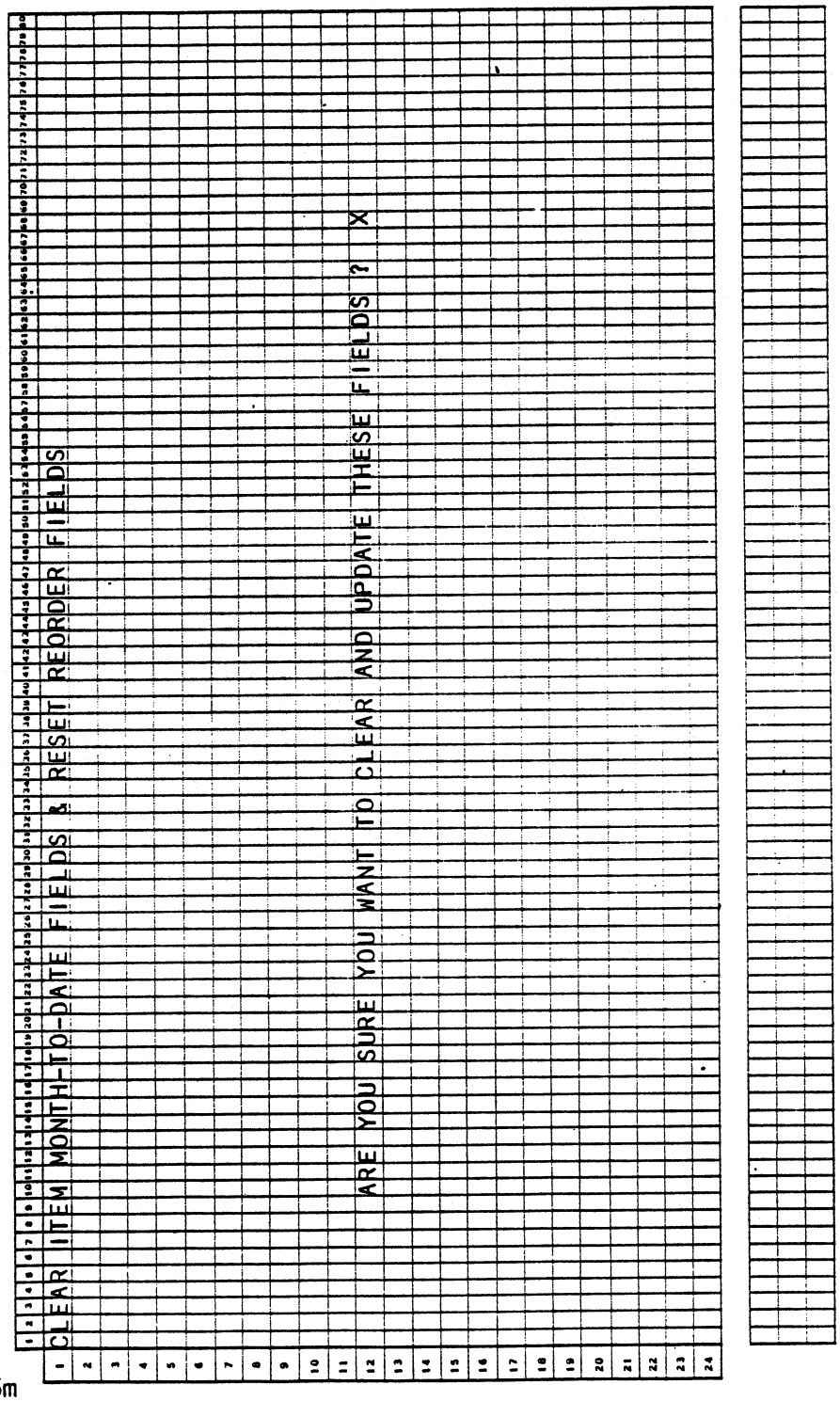
```
RUN DATE: XX-XXX-XX   TIME: XX:XX XM    X-----------------COMPANY NAME----------------X        TERM #XX   SEQ #XXX   PAGE XXX
P H Y S I C A L   C O U N T   U P D A T E   A U D I T   L I S T
           ITEM NUMBER        DESCRIPTION                                  OLD          COUNTED         DOLLAR
                              TAG NUMBER   LOCATION    UOM    QTY-ON-HAND    QTY-ON-HAND        CHANGE       POSTING COMMENTS
                                                                                                                  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
           XXXXXXXXXXXXXXX    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                XXXXXX        XX        XX        XXX,XXX-       XXX,XXX      X,XXX,XXX.XX-
                                XXXXXX        XX        XX        XXX,XXX-       XXX,XXX      X,XXX,XXX.XX-  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                      LOCATION XX  SUBTOTAL:      XXX,XXX-       XXX,XXX      X,XXX,XXX.XX-
                                XXXXXX        XX        XX        XXX,XXX-       XXX,XXX      X,XXX,XXX.XX-

                                              ITEM TOTAL:         XXX,XXX-       XXX,XXX      X,XXX,XXX.XX-

                                                                                                                  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
           XXXXXXXXXXXXXXX    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                XXXXXX        XX        XX        XXX,XXX-       XXX,XXX      X,XXX,XXX.XX-
                                              ITEM TOTAL:         XXX,XXX-       XXX,XXX      X,XXX,XXX.XX-

        TOTAL ITEMS: XX,XXX            TOTAL TAGS COUNTED: XXX,XXX
```

Remarks: POSTING COMMENTS INCLUDE: "WARNING - MULT COUNT SAME LOC". "** NO POSTING ** BAD ITEM **", "LOC NOT FOUND - ASSUMING LOC 1".

INVENTORY MANAGEMENT PACKAGE
CLEAR ITEM MONTH-TO-DATE FIELDS & RESET
REORDER FIELDS APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: CLRMOE



CLEAR ITEM MONTH-TO-DATE FIELDS & RESET
REORDER FIELDS APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Clears Item Master file records of month-to-date figures and recalculates six values that impact reordering advice.

Input: ITMMAS Files Updated: ITMMAS Output: None

Enter Module From: IMSFMN When Done Return To: IMSFMN

Programs in Module: CLRMOE

Program Functions and Notes:

CLRMOE

Ask the operator the question "ARE YOU SURE YOU WANT TO CLEAR AND UPDATE THESE FIELDS ?"

On an "N" answer, do no processing and return to the Special Functions menu.

Read the ITMMAS control record and store the default location and number of locations.

Sequentially read ITMMAS; if a record is deleted, ignore it and read another. If it is a valid record, proceed.

Recalculate average forecast error, sum of forecast errors, average monthly usage, recommended minimum order quantity, safety stock, and reorder level (see below). The only location considered is the default location. Find the default location from the Item Master control record.

Set USEMTD, QTYMTD, SLSMTD, CSTMTD to 0. Write the record back to the ITMMAS file.

The initial calculation is the resetting reorder fields section is for a qualified Month-to-Date quantity (MTDQTY).

MTDQTY is initially set to the current month's forecasted usage (AVGUSE) times the usage filter (USEFLT).

Then MTDQTY is compared to zero; it is also compared to the actual month's usage (USEMTD).

If MTDQTY = 0 or is greater than USEMTD, MTDQTY is reset = USEMTD.

Else it stays as is (USEFLT x AVGUSE). MTDQTY is used in several of the following calculations. NOTE: This is ensuring that the actual monthly usage (USEMTD) is not blindly used. If USEMTD exceeds some peak usage determined with the usage filter (USEFLT X AVGUSE) and average usage, the actual usage is dampened (reducted) as shown.

Sum of Forecast Errors (SUMERR)

New SUMMERR = Old SUMERR + (AVGUSE-MTDQTY)

NOTE: SUMERR can be negative.

Sum of Errors (SUMERR), in this case forecast errors, is a way of locating bad forecasts. A biased forecast will continuously, month after month, generate errors on one (biased) side of actual usage, slowly building up a large sum. By setting limits of same kind, the user can find those actual usage exceptions that are most out of line. Then the user can adjust those items forecasted by changing USEWGT. Or he can change the usage filter.

Average Forecast Error (AVGERR)

The new average forecast error (New AVGERR) is found via exponential smoothing as follows:

New AVGERR = Old AVGERR (1 - USEWGT) + USEWGT (AVGUSE-MTDQTY)*.

*The absolute value of (MTDQTY-AVGUSE) represents a qualified deviation between actual usage this month and forecasted usage this month.

USEWGT is the usage weighting factor (USEWGT) from the Item Master record. It is the "alpha" quantity in conventional exponential smoothing equations.

NOTE: AVGERR cannot be negative; it is strictly a deviation measurement.

Average Monthly Usage (AVGUSE)

Use exponential smoothing to get new Forecasted Average Monthly Usage.

New AVGUSE = Old AVGUSE (1 - USEWGT) + (USEWGT) x (MTDQTY).

USEWGT is the usage weighting factor.

Recommended Minimum Order Quantity (RECMIN)

New RECMIN = New AVGUSE x LEADTM.

Safety Stock (SAFSTK)

New SAFSTK = Safety Factor x New AVGERR.

Reorder Level (REOLVL)

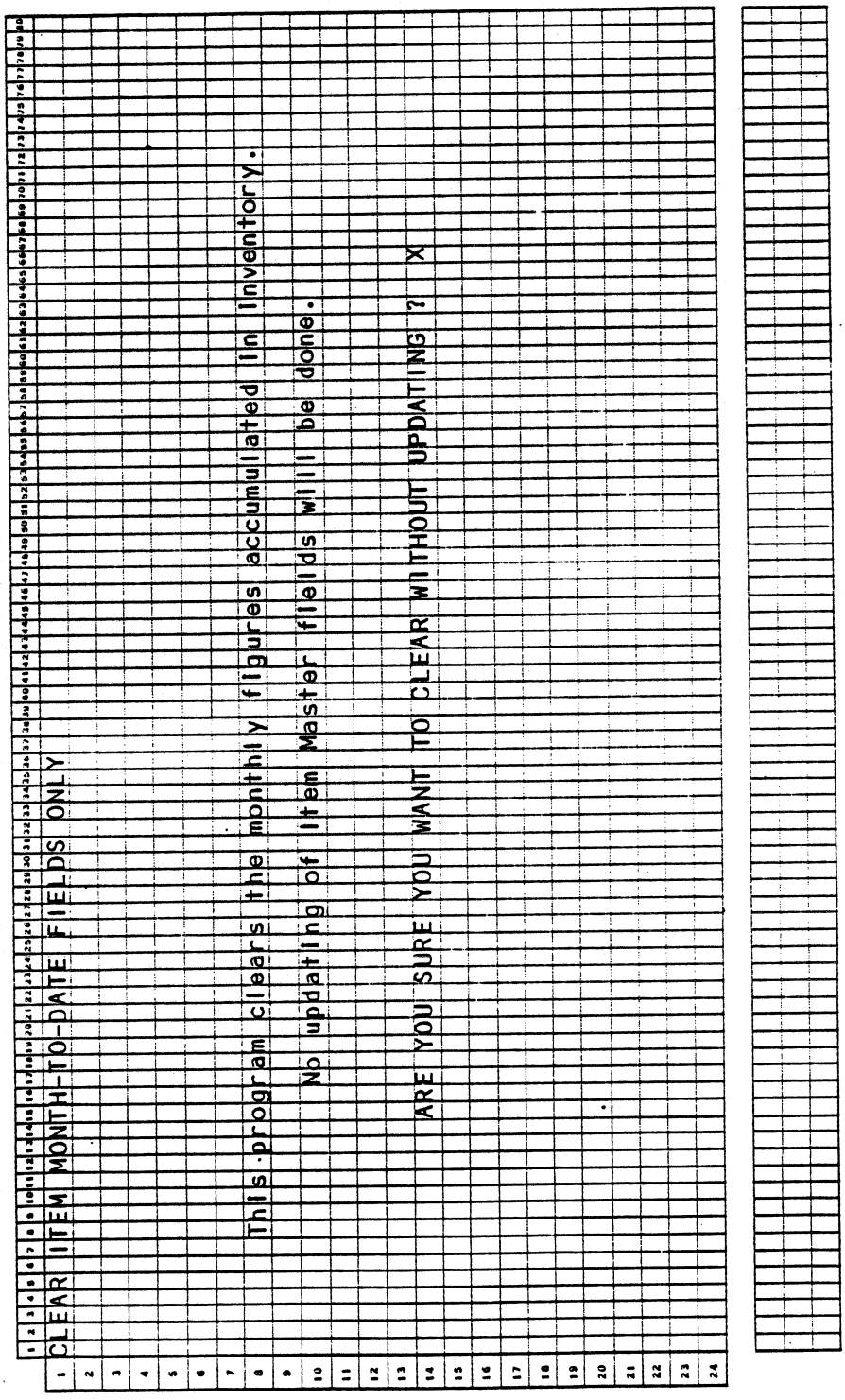
This is also known as the Reorder Point calculation. It is only recalculated each month for items that use the reorder point method of order generation.

This page intentionally left blank.

INVENTORY MANAGEMENT PACKAGE
CLEAR ITEM MONTH-TO-DATE FIELDS ONLY APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: CLRM02



PROGRAM SPECIFICATIONS

Function: Sets the four Month-to-Date fields in each Item Master record to zero.

Input: ITMMAS Files Updated: ITMMAS Output: None

Enter Module From: IMSFMN When Done Return To: IMSFMN

Programs in Module: CLRM02

Program Functions and Notes:

CLRM02

Ask the operator the question "ARE YOU SURE YOU WANT TO RUN THIS PROGRAM ?".

On an "N" answer, do no processing, and return to the Special Functions menu.

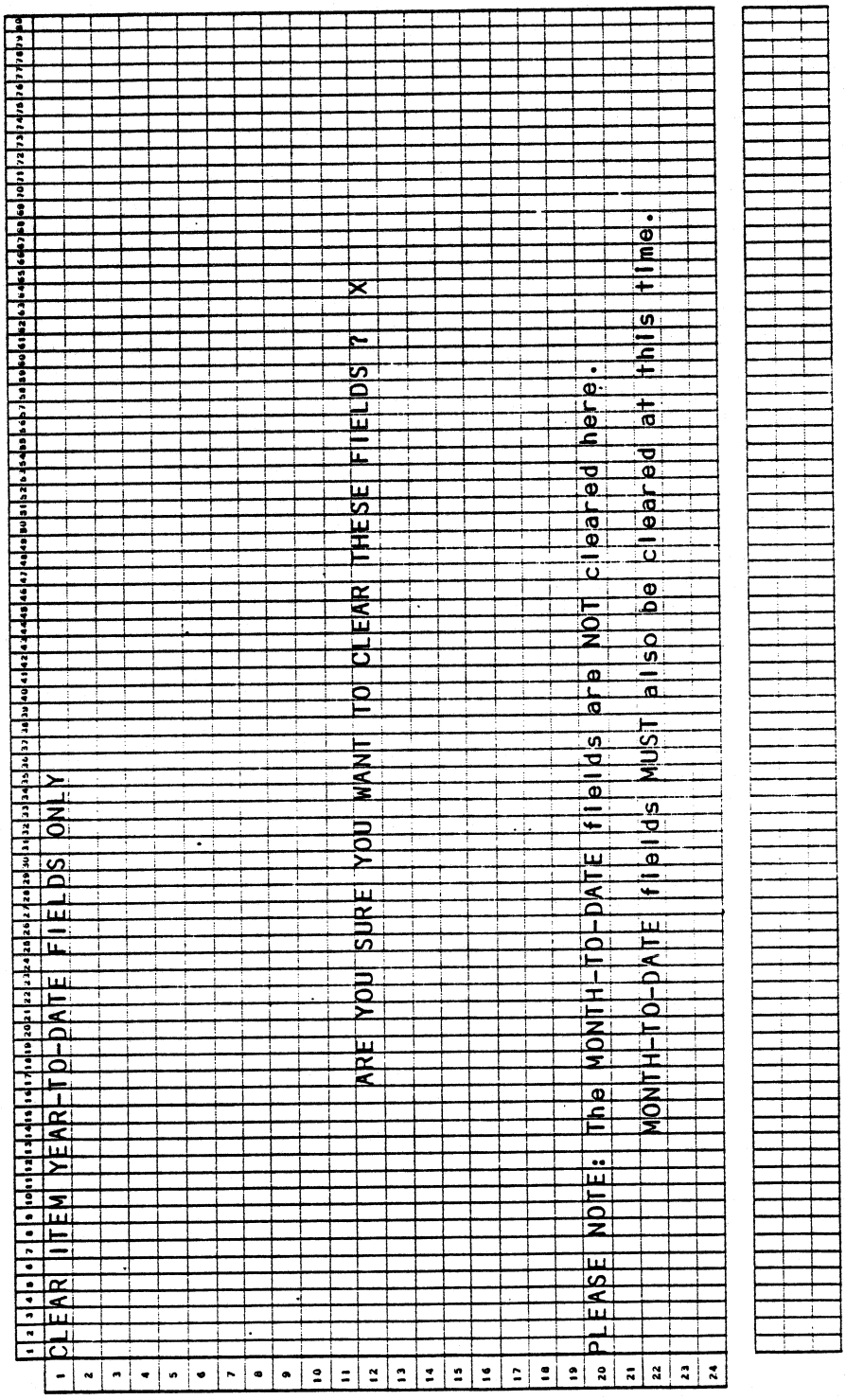
Read the ITMMAS control record to determine if this installation has Inventory Management fields (that is, TYPSYS = 2, 4 or 6).

Sequentially read ITMMAS. If a record is deleted, ignore it. If it is a valid record, proceed.

Clear USEMTD, QTYMTD, SLSMTD, CSTMTD to 0. Write the record back to the ITMMAS file.

SCREEN FORMATS

Program: CLRYOE



CLEAR ITEM YEAR-TO-DATE FIELDS ONLY APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: Clears Item Master file records of year-to-date figures.

Input: ITMMAS Files Updated: ITMMAS Output: None

Enter Module From: IMSFMN When Done Return To: IMSFMN

Programs in Module: CLRYOE

Program Functions and Notes:

CLRYOE

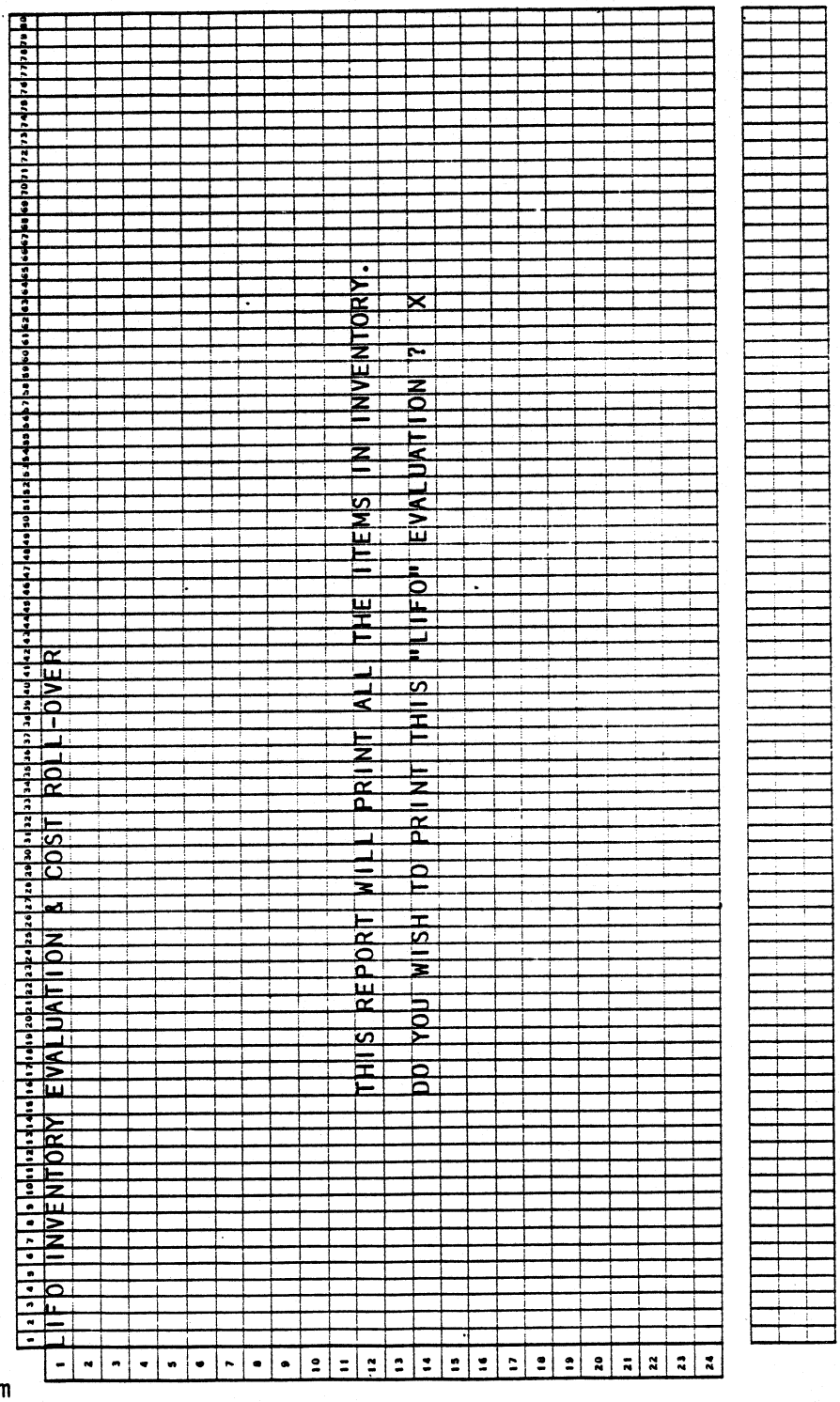
Read ITMMAS sequentially. Skip the control record, but on all other active (i.e., non-deleted) records set the USEYTD, QTYYTD, SLSYTD, CSTYTD fields to 0.

Rewrite cleared records to the ITMMAS file.

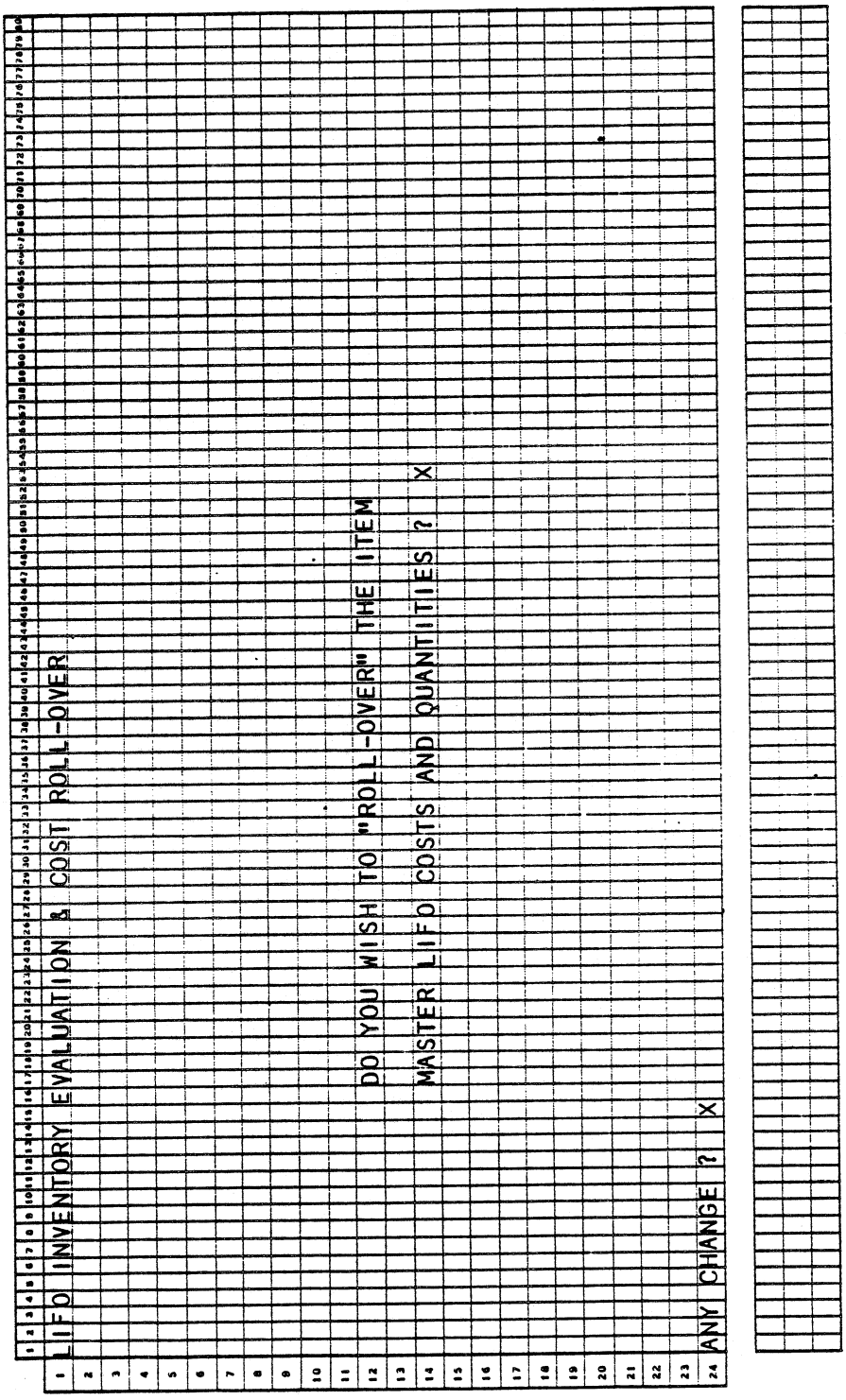
INVENTORY MANAGEMENT PACKAGE
LIFO INVENTORY EVALUATION & COST
ROLL-OVER APPLICATION
DIBOL JUN-84

SCREEN FORMATS

Program: LIFRPT

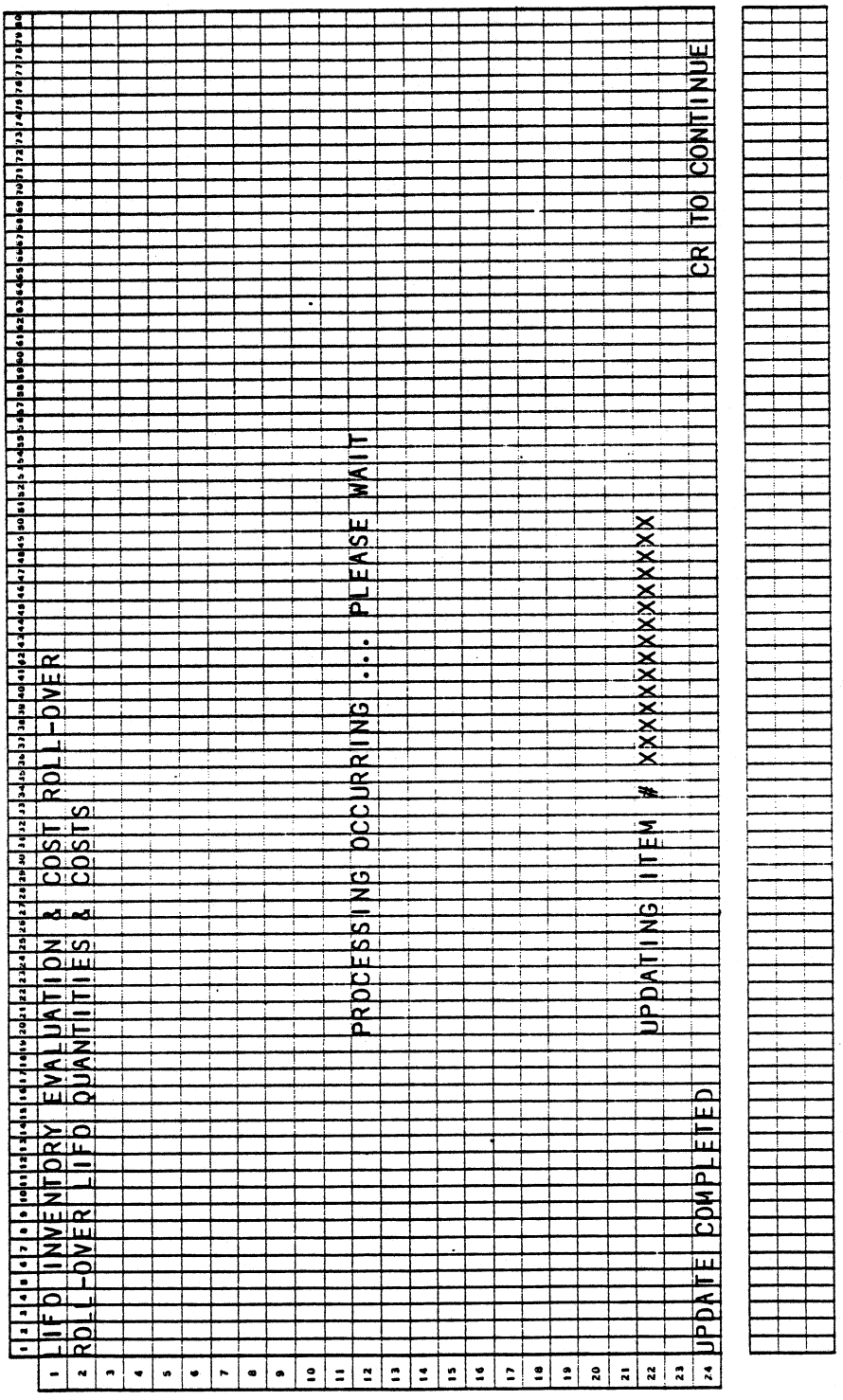


Program: LIFRPT



Remarks: ACCEPT ONLY "Y" OR "N" - NO DEFAULT ANSWERS.

Program: LIFROL



LIFO INVENTORY EVALUATION & COST
ROLL-OVER APPLICATION
DIBOL JUN-84

PROGRAM SPECIFICATIONS

Function: This module provides a "LIFO" evaluation of current inventory, plus a "roll-over" feature which records the current LIFO base quantity and LIFO base cost for each Item Master record. Two programs are used. LIFRPT, which actually prints the report, and LIFROL which performs the optional "roll-over".

Input: KBD
ITMMAS
ITMIDX

Files Updated: ITMMAS

Output: LIFO Inventory Evaluation

Enter Module From: IMSFMN

When Done Return To: IMSFMN

Programs in Module: LIFRPT, LIFROL

Program Functions and Notes:

LIFRPT - LIFROL

Print a line for each inventory item. Print all inventory items. Show the current LIFO base cost (LIFOBP) and LIFO base quantity (LIFOBQ), as well as the current quantity on-hand (QTYONH) and current average cost (AVGCST). Use the following equations to determine new LIFO values and net dollar changes in inventory value.

If current on-hand quantity is greater than the LIFO base quantity:

.New LIFO base cost =

$$\frac{((QTYOHN - LIFOBQ) * AVGCST) + (LIFOBQ * LIFOBP)}{QTYONH}$$

.New LIFO base quantity = QTYONH

Else

.New LIFO base cost = no change
.New LIFO base quantity = QTYONH

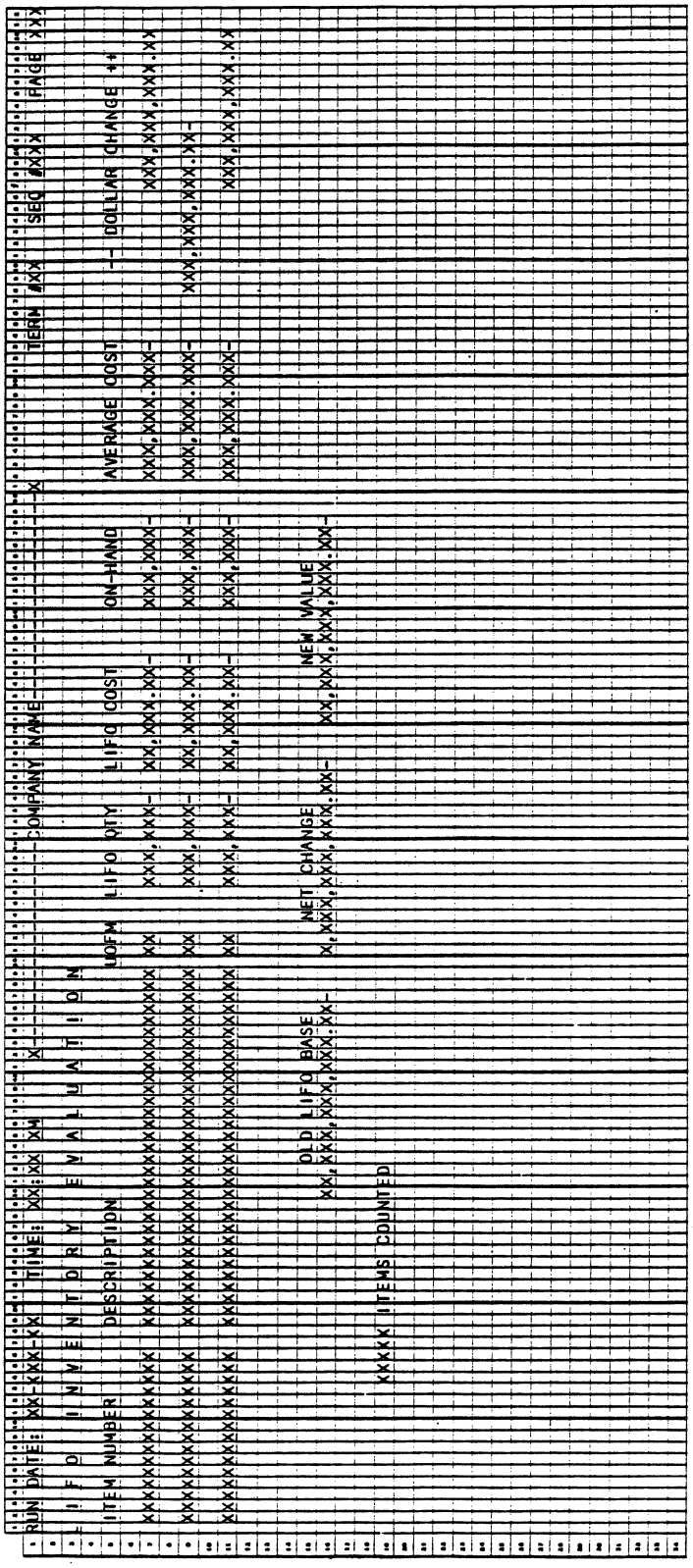
Use this same report criterion for both the LIFO evaluation report and the actual LIFO roll-over process, to adopt the new figures.

LIFO inventory evaluation means all items are valued (using new LIFOBQ * new LIFOBP) and then summed.

This sum represents the inventory value in dollars using a LIFO calculation technique. Taxes are paid on inventory and either the LIFO value or the straight value (On-Hand * avg Unit Cost) can be used. In times of currency depreciation (inflation), some companies find the LIFO value to be a lower number than the straight value. Therefore, taxes would be less.

LIFO INVENTORY EVALUATION & COST
ROLL-OVER APPLICATION
DIBOL JUN-84
REPORT FORMATS

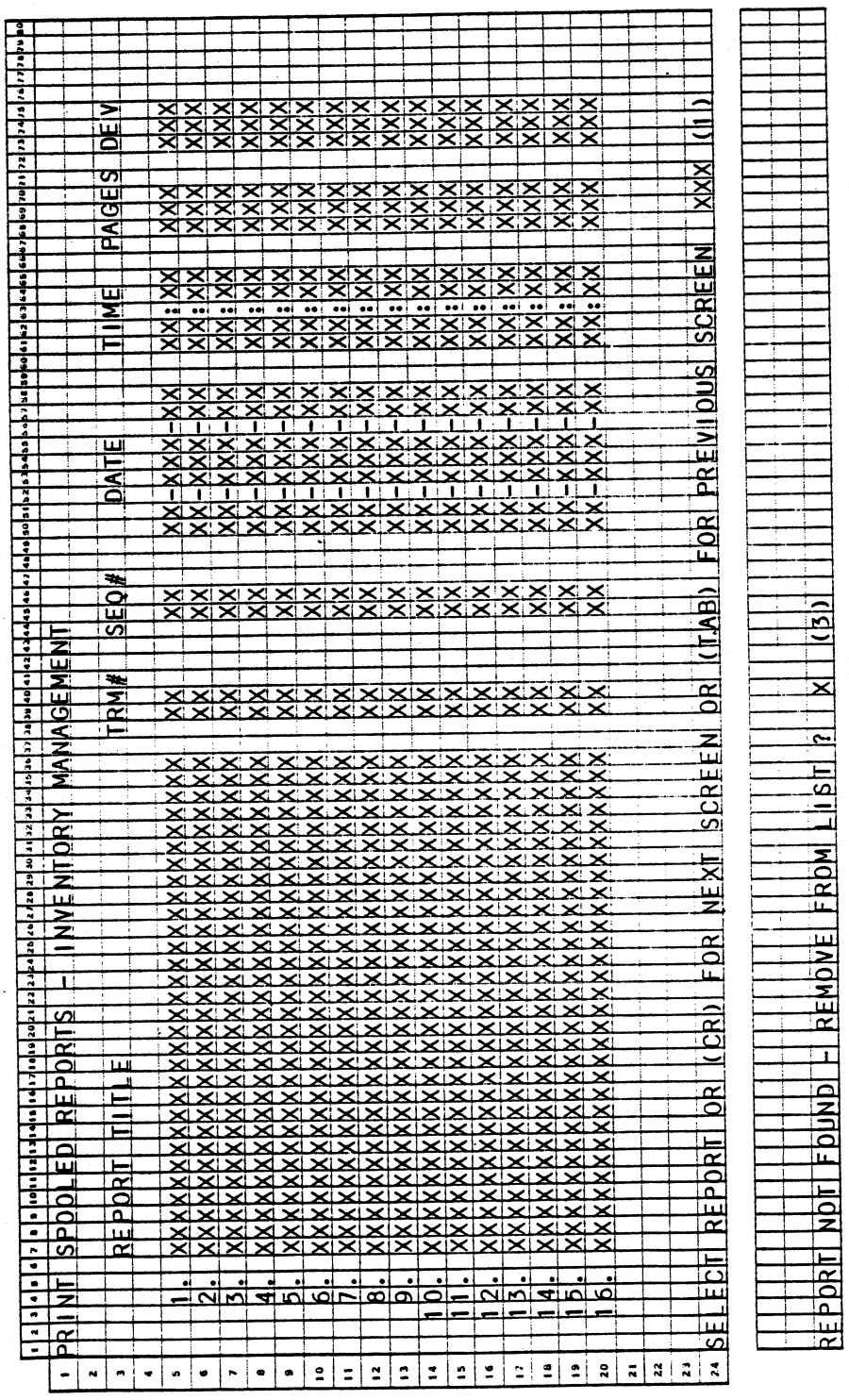
Program: LIFRPT



This page intentionally left blank.

SCREEN FORMATS

Program: IMSPOL



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.

PROGRAM SPECIFICATIONS

Function: Allows the user to inspect the list of reports that have been spooled, and print.

Input: Files Updated: SPLDIR Output: Any spooled report

Enter Module From: IMSFMN When Done Return To: IMSFMN

Programs in Module: IMSPOL

Program Functions and Notes:

IMSPOL

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 General Ledger reports for the user's Company code into an array in memory.

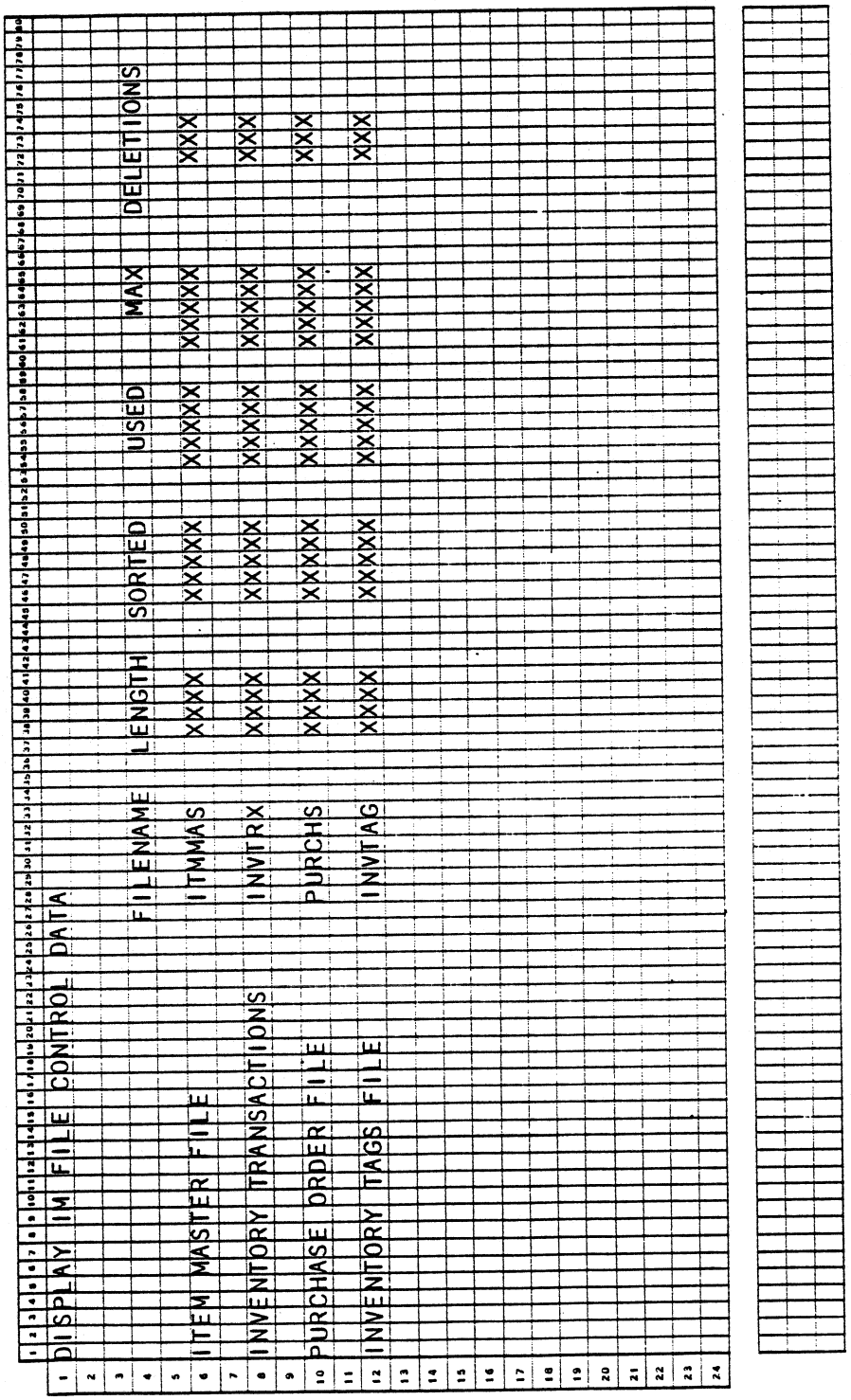
It then displays the first 16 reports, using this stored array to locate the records on the SPLDIR file.

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.

SCREEN FORMATS

Program: IMFILS



PROGRAM SPECIFICATIONS

Function: Displays the control record of all permanently created files in the I/M package.

Input: Files Updated: None Output: None

Enter Module From: IMSFMN When Done Return To: IMSFMN

Programs in Module: IMFILS

Program Functions and Notes:

IMFILS

This program has a table of the DEVICE.DDF record numbers for all the files in the Inventory Management package. It opens these files one at a time, does an RSTAT to get the record lengths, then displays the data in the control record.

INVENTORY MANAGEMENT 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 always 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 logically 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. Standard Transaction File

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.

ABC INDEX WORK FILE
(ABCIDX)

Description: 4 control records; 1) total value of inventory at average value cost; 2) percentage of total value to be used in Class A; 3) percentage of total value to be represented in Class B; 4) percentage in Class C.

File Status: Work Record # in DEVICE.DDF: 131 Rec. Size: 20
+ End of Record

Is the first record of this file used as a control record ? Yes.

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Value at Average Use	VALAVU	D15	See Note 1.
Item Master Record Number	ABC041	D5	
** PERCENTAGE CONTROL RECORD (3) **			
(Unused)		A17.	
Percentage of Total Value	ABCPCT	D3	See Note 2.
** TOTAL INVENTORY VALUE CONTROL RECORD (1) **			
Total Value of Inventory	TOTVAL	D15	See Note 3.
(Unused)		A5	

NOTES:

1. At average use and cost.
2. First record holds percentage in Class A, Second - B, Third - C.
3. The SORT key is TOTVAL (15 bytes).

This page intentionally left blank.

CYCLE COUNT ITEMS FILE
(CYCIDX)

Description: Cycle Count Items Selected.

File Status: Work Record # in DEVICE.DDF: 130 Rec. Size: 8
+ End of Record

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Parts Bin Number	BINNUM	A3	
Item Master Record Number	CYC041	D5	

The SORT key is BINNUM (3 bytes).

This page intentionally left blank.

INVENTORY TAG FILE
(INVTAG)

Description: Inventory Tag file used by cycle count transactions and physical count transactions.

File Status: Work Record # in DEVICE.DDF: 40 Rec. Size: 39
+ End of Record

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Tag Number	TAGNO	D6	XXXXXX
Tag Item Number	TAGITM	A15	
Quantity On-Hand	QTYOH	D6	XXX,XXX
Tag Unit of Measure	TUOM	A2	
Tag Location	TLOC	A2	
(Unused)		A8	

The SORT key is TAGNO (6 bytes).

This page intentionally left blank.

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.

PURCHASE ORDER FILE
(PURCHS)

Description: Purchase Order file.

File Status: Work Record # in DEVICE.DDF: 53 Rec. Size: 89
+ End of Record

Is the first record of this file used as a control record? Yes

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Vendor Number	PVEND	A4	
Location	PLOC	A2	
Purchase Order Number	PPONUM	A8	
Item Number	PITEM	A15	
Item Description	PDESCR	A30	
Product Category Code	PPRDCT	A2	
Lead Time	PLDTM	D3	XX.X in months
Expected Receipt Date	PRCTDT	D6	MMDDYY
Last Unit Cost	PLSTCT	D7	$XX,XXX.XX
Item Weight	PWGT	D6	X,XXX.XX
Quantity Ordered	PORDED	D6	
** CONTROL RECORD **			
(Unused)		A71	
Organized Record Count	ORG053	D5	
Record Count	REC053	D5	
Maximum Record Count	MAX053	D5	
Deleted Record Count	DEL053	D3	

The SORT keys are PVEND, PPONUM, PITEM (27 bytes) for printing items ordered by vendor.

The SORT keys are PITEM, PRCTDT, PVEND (25 bytes) for printing items ordered by item.

SELECTED INVENTORY ITEMS FILE
(SELINV)

Description: Selected Inventory Items file.

File Status: Work Record # in DEVICE.DDF: 56 Rec. Size: 33
+ End of Record

FIELD DESCRIPTION	FIELD NAME	TYPE/ SIZE	FORMAT
Selected Record Address	SEL041	D5	
Multiple Vendor Flag	SMULTV	D1	
Sorting Location	SRTLOC	A2	
Sorting Factor	SRTFAC	D5	
Quantity On-Hand	QTYOH	D6	XXX,XXX
Quantity Allocated	QTYCM	D6	XXX,XXX
Quantity On-Order	QTYOO	D6	XXX,XXX
Reorder Level Flag	R	A1	
Out-of-Stock Flag	O	A1	

NOTES:

The SORT keys are SMULTV, SEL041 for printing advice by item.

The SORT key is SRTLOC for printing advice by location.

The SORT key is SRTLOC and priority for printing by vendor.

This page intentionally left blank.