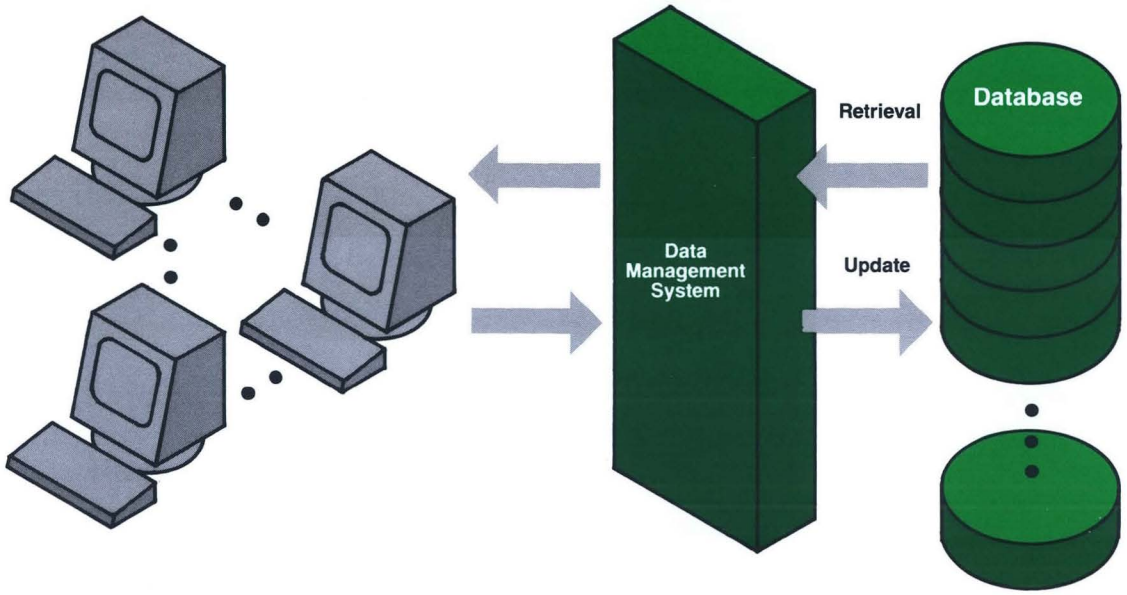
PERKIN-ELMER

Series 3200

DMS/32

Data Management System

DMS/32 provides retrieval & update of data in several different files, by multiple users.



Product Description

Perkin-Elmer's Data Management System (DMS/32) is the industry's first data management system specifically designed for the on-line transaction processing environment. DMS/32 is a component of the Reliance transaction processing and database management system. DMS/32 provides high-performance, multi-keyed data update and retrieval from multiple user files in a single database. DMS/32 manages real-time concurrent data access by maintaining database integrity through automatic on-line recovery features, record and file locking, and facilities for rapid restoration of the database in the event of any system failure.

DMS/32 is well-suited for applications involving the retrieval and update of data in several different files by multiple users. Programming simplicity and consistent high performance make DMS/32 an excellent choice for business applications such as inventory control, order entry, and materials management.

Features

These facilities make DMS/32 an outstanding solution to the most complex data processing problems:

- Unlimited number of secondary keys
- Random and sequential access
- Data access by full, approximate and generic keys
- Consistent, rapid response
- Multiple files with multiple extents
- Concurrent access control
- Ensured data integrity
- On-line rollback of failed transactions
- Automatic recovery from any system failure
- Central control of the database
- Easy use through COBOL, FORTRAN, RPG II, and Perkin-Elmer's Common Assembler Language, CAL MACRO
- Flexible and dynamic disk space management for peak performance and growth
- Compaction of data records
- On-line database back up
- Dynamic allocation and deletion of secondary indices

Data Access in DMS/32

DMS/32 is designed for consistently high-performance data access in an on-line environment. DMS/32 provides an indexed file organization in which records are referenced by multiple keys including:

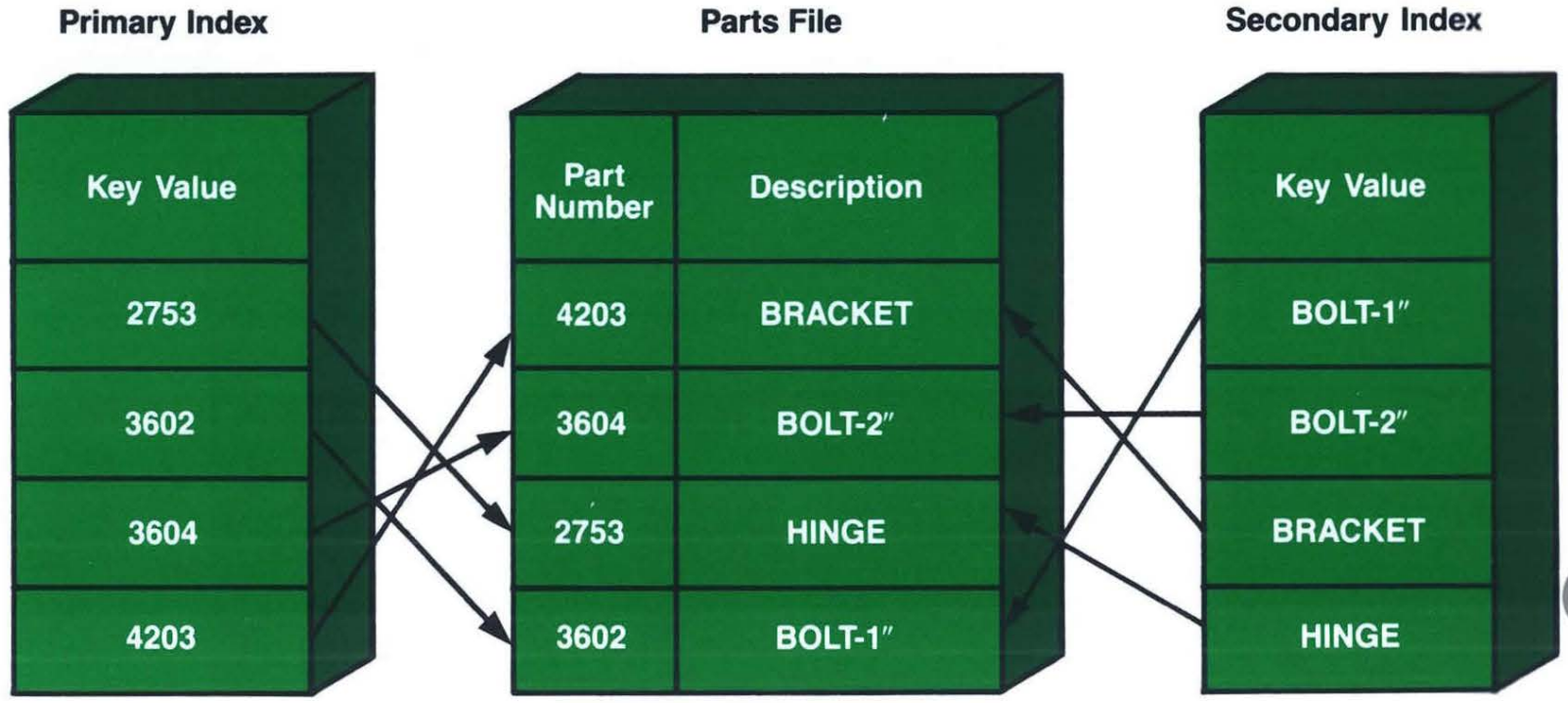
- One primary key which is unique for each record and is the identifier for that record.
- Virtually unlimited number of secondary keys which may be non-unique and which provide alternate paths to the same data.

As shown in Figure 1, the parts file is accessible through part number (primary key) or part description (secondary key). Any other field(s) in the record may also be defined as secondary keys, such as quantity or vendor. Users may retrieve records sequentially, randomly or dynamically (random access followed by sequential). Sequential access may be in ascending or descending order, depending on user needs.

Records may be randomly accessed by specifying a full or partial key value for any of the keys. For example, an inventory request for a display of items whose on-hand quantity equals or exceeds 100 units would specify "quantity equals 100" as an approximate key. Ascending sequential access from that point would fulfill the request, whether or not a record with that exact quantity existed.

A generic key may also be specified. A generic key specifies the leftmost portion of a key. This allows record retrieval based upon groups within a key. In Figure 1, specifying BOLT as a generic key would commence the retrieval of records whose description begins with that word. Using DMS/32's generic key facilities, all types of BOLTs in the database would be retrieved, simply and efficiently.

Figure 1
Multiple Indices in DMS/32



Data Compaction

For efficient use of disk space, the records of a file can optionally be held in a compressed form. Each record is transformed, prior to being written to the database, by removing all redundant data to ensure that the minimum possible amount of data is stored on disk. Similarly, when a record is retrieved from the database, it undergoes a complementary transformation to return the record to its original form.

The amount of compaction obtainable depends on the structure and contents of the record. On average, however, a record can be held in 70% of the space which it originally occupied. The major savings are achieved by characters and zeros, and by storing common character pairs as single special characters.

Consistent Response

In a transaction processing system, users expect consistently high performance. Since a major portion of fulfilling user transactions is the update and retrieval of data in the database, the data management system must ensure rapid response, regardless of the mode of file usage (i.e., the key used) or the volatility of the database over time (e.g., response at 4:30 P.M. versus response at 8:30 A.M.). DMS/32 fulfills these needs through balanced indexed structures, distributed free space, and continuous reorganization. DMS/32 is not preferential to primary or secondary index keys as far as speed of record access is concerned. The index structures of a DMS/32 database provide consistent response, regardless of the path specified for access. Thus, all users of a DMS/32 system enjoy the same high performance regardless of their view of the structure of the database.

Furthermore, consistency of response is assured over time, regardless of file activity. DMS/32's distributed free space allows records to be updated and inserted without the use of overflow areas and without building chain pointers to inserted records. The access paths to both "old" and "new" records are the same and response to later requests is consistent. Space from deleted records is immediately available for reuse. Continuous reorganization is automatic and eliminates the need for periodic off-line reorganizations. With DMS/32, records can be added, changed and deleted while files remain well organized and users' responses are consistently at a peak.

Physical Structure of The Database

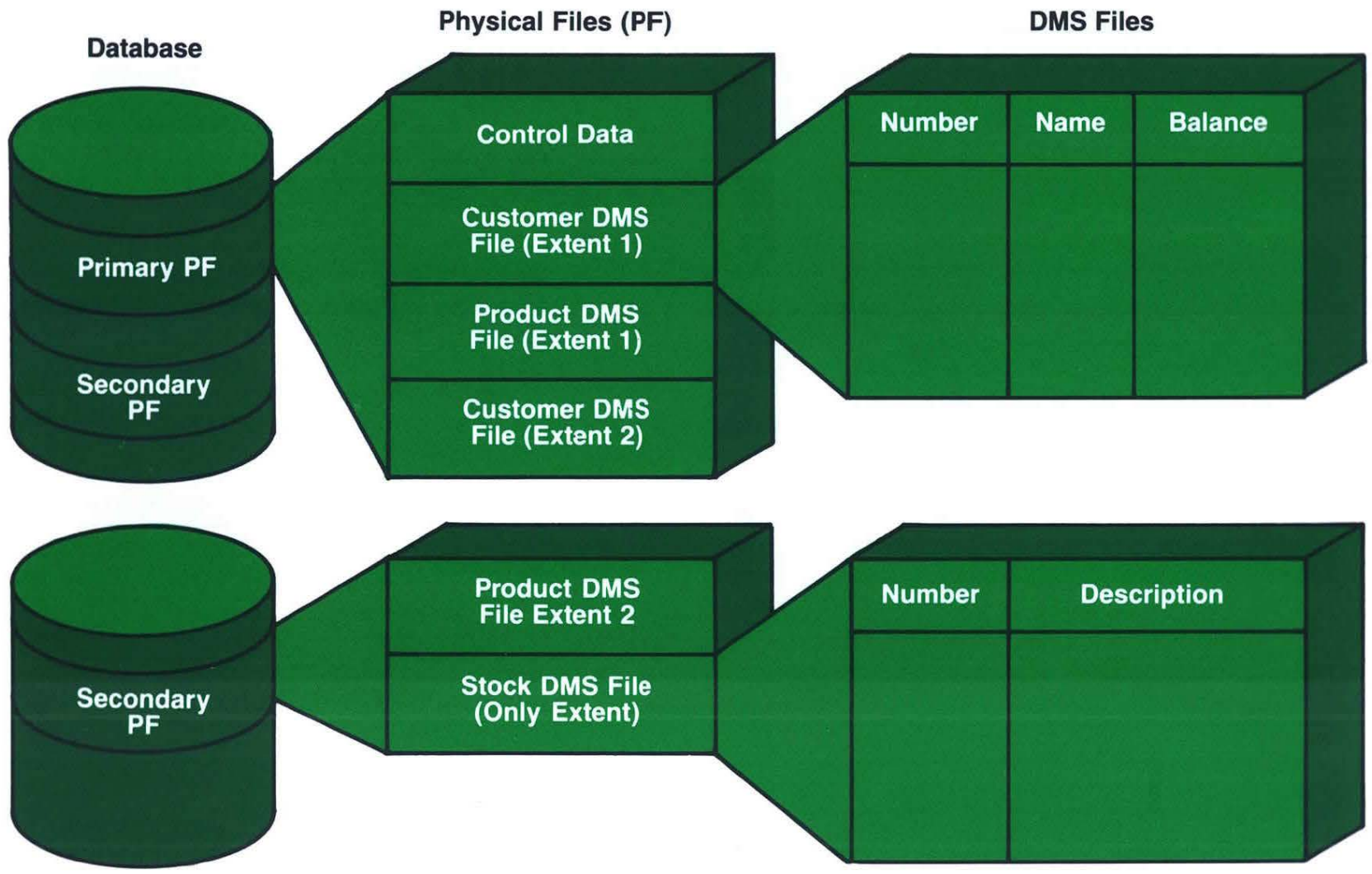
A DMS/32 database is contained in one or more physically contiguous placement areas on one or more disks. These placement areas correspond to files under OS/32, Perkin-Elmer's multitasking 32-bit operating system. Using multiple OS/32 files to contain the database allows virtually unlimited database size. The distribution of DMS files over multiple disk drives allows the user to achieve optimum performance.

Within the DMS/32 database, DMS files can be allocated with one or more physically contiguous areas called extents. A DMS file's extents can be contained in any number of the OS/32 physical files. Each DMS file contains fixed length records and indices containing user-defined keys. Indices point to the records. Figure 2 portrays the database structure and shows the tremendous flexibility of the database organization.

Physical retrieval of data is in fixed length blocks called pages. Each page contains data and free space. The distributed free space is used to efficiently insert and update records. Continuous space reclamation and reorganization ensure efficient operations in storage usage and performance. Data records within the pages are compacted, further reducing disk storage requirements.

DMS/32 automatically checks at the start of each session that all required OS/32 files and any associated log files are available and are proper compatible versions.

Figure 2
Database Structure



Transaction Units

The Transaction Unit concept is central to the unique operational capabilities of DMS/32. A business transaction normally includes the updating of multiple files in a database. For example, when a sale is made from inventory, a number of ledger updates must be made, including adjusting the customer's accounts receivable record and debiting inventory for the quantity allocated. If all of the requisite updates are not completed, the database will contain inconsistencies and the financial files will be out of balance. In the example, the result could be erroneous inventory levels or improper customer bills.

In DMS/32, the transaction unit is the unit of database integrity. A transaction unit includes a group of updates all of which must be properly completed to ensure database consistency. DMS/32 automatically locks records affected in a transaction unit and unlocks them following that transaction unit's completion. Commands to end or fail transaction units allow the programmer to state simply whether prior updates in the transaction unit should be accepted or rejected. Updates from failed transaction units are automatically rolled back with minimal overhead to the system.

The transaction unit approach is a breakthrough in on-line system design. A comparison with other approaches to the concurrent transaction problem shows that DMS/32 truly offers the simplest and highest performance solution for multi-user systems.

Concurrent Use of DMS/32

DMS/32 is designed for simple and efficient use in a multiuser environment. The problems of managing user contention for the same data, controlling user access to the database, recognizing deadlock situations, and ensuring the consistency and on-line integrity of data are handled by DMS/32. In the past, application developers have discovered that implementing on-line, concurrent database applications has involved more thought and coding related to concurrency than to the actual application. Concepts and techniques built into DMS/32 ease the development of transaction processing systems, allowing programmers and analysts to focus on the application problem to be solved. The DMS/32 transaction unit concept and implementation offers unprecedented ease in the development of application transactions.

Old Cures– New Problems

Application designers normally have had to build into their application programs the procedures for ensuring against database inconsistencies. The result has been oversized and overly complex programs using coding sequences reflective more of concurrency concerns than the solution to the application problem. One common approach in the past has been the journalling or staging of database updates until the transaction is completed. In a journalling system, the staged updates are not applied to the database if the transaction fails. If the transaction is successful, the entire database is usually locked while all journalized updates are applied to the files.

There are numerous drawbacks to this approach. Managing the journalizing process is frequently the obligation of the application program. Shared use of the entire database is prevented while all updates are applied from the journal. Most significantly, successful updates require writing both to the journal and to the database. In effect, "good" transactions are penalized in order to allow for "bad" transactions. Since the large majority of transactions are successful, the entire system incurs a very large performance overhead in exchange for some automation of the integrity process.

The DMS/32 Solution

With DMS/32, successful transactions are accorded maximum efficiency, while failed transactions incur the minimal overhead of the integrity system.

Updates are applied directly to the database next to the previous version of the updated record. For efficiency, WRITEs to DMS/32 are performed asynchronously, allowing the application program to continue processing. Locking is performed at the record level, not at the file or database level. (Optional file locking is permitted for sensitive off-line applications). Following successful completion of the transaction, the new versions of the updated records are immediately available for use. The space occupied by the old versions is reclaimed for reuse. If the transaction fails, control pointers are modified to indicate the old versions are still active. The space occupied by the new versions is reclaimed and record locks are cancelled. Thus, top efficiency is afforded to successful transactions. Failed transactions bear a minimal overhead. At no time is the entire database locked: other transactions not using the specifically updated records proceed while DMS/32 asynchronously removes the effects of the failed transaction.

Programming Ease

Due to the transaction unit approach, application programming is greatly simplified and undue side effects of a concurrent database are eliminated. All data records affected by a transaction unit are locked for the duration of that transaction unit. This prevents concurrent modification of the same record by two separate users.

An example of this concurrent processing problem is a situation where user 1 and user 2 retrieve the same inventory record. Initially, the record indicates that quantity-on-hand is 100 units. User 1 requires 60 units for a customer, and updates the quantity-on-hand to 40 units. User 2, meanwhile, requires 70 units and updates the record to reflect that 30 units are remaining. In fact, of course, the two users have unknowingly overdrawn the stock, and have committed the same inventory to two customers.

In DMS/32, such a situation cannot occur. Updates are only performed on locked records, and locks are maintained until the end of the transaction unit. The other user attempting to access the locked record would be rejected until the first transaction unit was completed.

The commands to END or FAIL a transaction unit automatically remove all record locks, freeing the programmer from concerns about inadvertently leaving "orphan" locks on records.

Programmers can use simple programming sequences to process the DMS/32 database. These sequences can reflect the application flow without being artificially reordered to allow for update rollback at the transaction level. The resulting code is far simpler to write, read and maintain. There is no need to write recovery programs or rollback subroutines. At any point where the user decides to cancel the transaction, a single FAIL command will undo all updates and restore database consistency.

Program-level or system-level failures are automatically handled by DMS/32. If a user program fails, DMS/32 rolls back the effects of that transaction unit. If there is a more widespread system failure, such as a power failure or hardware/software malfunction, DMS/32 rolls back active transaction units for all users when it is reinstated. The rollback is automatic and cannot be bypassed, thus ensuring database consistency in all circumstances.

Run Units

A DMS/32 run unit represents a user's right-of-access to the database, either a terminal user's session or a batch program. Thus, each DMS/32 user is uniquely identified for purposes of access control and recovery. A run unit consists of one or more consecutive transaction units.

Disk Logging

To safeguard against the loss of a database as a result of media failure, DMS/32 optionally copies the contents of updated pages to a disk log file associated with the database. The log file, which is normally held on a separate disk volume from its database, contains only the latest copy of all pages updated successfully since the last security copy of the database and, thus, generally requires significantly less disk space than the database.

Each time a security copy of the database is taken, a new log for the database can be created by specifying a log filename. When DMS/32 detects that a log is nearly full, it informs the operator who can respond by adding an extension file to the log.

Rollforward Reconstruction

In the event of media corruption, a DMS/32 database can be reconstructed by applying the contents of the appropriate log files to a security backup of the database. If, during rollforward, more than one log file is required, DMS/32 prompts for successive logs as its needs them. At the same time, DMS/32 checks that the logs are supplied in the correct order and are for the database concerned.

Because the log files include only the latest updated versions of new or modified records, the rollforward process is exceptionally fast. Thus, system availability is maximized, regardless of any problem with the physical database.

File Space Management

DMS files and secondary placement areas can be created, modified or deleted at any time while the database is operational. The operations may be performed either by using the File Space Management Utility, or by issuing supplied CAL MACROS.

DMS/32 can automatically extend DMS files when they become full. The extension is performed in such a way that application programs updating the DMS file are not affected. Automatic extension is an option controlled by information supplied when the file was created or modified.

Database Structure

Information held by DMS/32 regarding the database, placement areas, log and DMS files can be obtained by interrogating special DMS information files. Data from the information files can be retrieved using the File Space Management Utility, or by programs using a subset of the DMS/32 retrieval commands.

DMS/32 Operation

A DMS/32 database is created by the Generate Utility. This utility allocates the primary placement area and establishes the control information for the database.

The DMS/32 session initialization command loads the DMS/32 software and establishes the DMS/32 environment. This includes:

- preparing the data areas and the database for use,
- recovering the consistency of the database after a system failure,
- rolling back the transaction units that were interrupted by a system failure.

These operations are performed without operator intervention; DMS/32 detects if the system failed last time the database was used.

The only circumstances that require extra information from the operator during initialization are:

- to specify a new log file,
- to inhibit logging,
- to reconstruct, or rollforward, the database following a disk media failure.

A DMS/32 system is terminated by a QUIESCE command entered at the system console. This brings the system to an end when all the current run units have been detached from the database. Typically, this occurs in a batch environment at the end of all the current application programs, and in a transaction processing environment at the end of all the current business transactions. Alternatively, by using one of the optional forms of the QUIESCE command type, the system can be ended after all the current transaction units have completed, or immediately. When used in the Reliance system, DMS/32 quiescence is synchronized with the transaction processor. The Reliance CONTROL-QUIESCE transaction terminates DMS/32 activity for each terminal as the terminal completes a logical unit of work as defined by the transaction designers. In this way, DMS/32 gradually quiesces as the transaction terminals arrive at natural stopping points.

System Requirements

Minimum Hardware Requirement

Any Perkin-Elmer 32-bit processor with a minimum of 512KB, 80MB disk, and a console device.

Minimum Software Requirement

OS/32 Release 7.2 or higher.

Product Numbers

S71-022 DMS/32 Group I
S72-022 DMS/32 Group II
S73-022 DMS/32 Group III
Note: DMS/32 is also supplied as a component of the Reliance software system, part number
S71-035 Reliance Group I
S72-035 Reliance Group II
S73-035 Reliance Group III
and the Reliance PLUS System, part number
S71-054 Reliance PLUS Group I
S72-054 Reliance PLUS Group II
S73-054 Reliance PLUS Group III

Related Documentation

29-714 DMS/32 Introduction
29-715 Data Management with COBOL Programmers Reference Manual
29-716 DMS/32 CAL Programmers Reference Manual
29-717 DMS/32 System Programming and Operations Manual
48-045 Data Management with FORTRAN Programmers Reference Manual
48-061 Data Management System/32 File Space Management Utility Manual

Worldwide Sales Offices

U.S.A Offices

ALABAMA: Huntsville; ARIZONA: Phoenix; CALIFORNIA: Los Angeles, San Diego, San Francisco, Santa Clara, Tustin; COLORADO: Denver; CONNECTICUT: Fairfield, Hartford; FLORIDA: Orlando; GEORGIA: Atlanta; ILLINOIS: Chicago, Springfield; MARYLAND: Rockville; MASSACHUSETTS: Boston; MICHIGAN: Detroit; MISSOURI: St. Louis; NEW JERSEY: Cherry Hill, West Long Branch; NEW MEXICO: Albuquerque; NEW YORK: Binghamton, Lake Success, New York City, Rochester; NORTH CAROLINA: Raleigh; OHIO: Cleveland, Dayton; OKLAHOMA: Tulsa; PENNSYLVANIA: Pittsburgh; TEXAS: Dallas, Houston; VIRGINIA: Richmond; WASHINGTON: Seattle.

Major Subsidiaries

AUSTRALIA: Brisbane, Canberra, Melbourne, North Ryde, North Sydney, Perth, and NEW ZEALAND: Auckland, Wellington; BELGIUM: Brussels; CANADA: Calgary, Montreal, Ottawa, Toronto, Vancouver; FRANCE: Bois d'Arcy Cedex, Bordeaux, Grenoble, Lille, Lyon, Toulouse; GERMANY: Dusseldorf, Frankfurt, Munich, Stuttgart, Uberlingen, and AUSTRIA: Vienna; GREECE: Athens; HONG KONG; ITALY: Milan; THE NETHERLANDS: Gouda; THE REPUBLIC OF SINGAPORE: Singapore; SPAIN: Madrid; UNITED KINGDOM: Manchester, Slough. Other countries are served by a network of distributors.

The information contained herein is intended to be a general description and is subject to change with product enhancement.

PERKIN-ELMER

Data Systems Group
2 Crescent Place
Oceanport, N.J. 07757
(201) 870-4712
(800) 631-2154 (U.S.A. Only)

EVERYWARE™

PB304084
Printed in U.S.A.