Professional™
300 series

PRO/RMS–11:
An Introduction

Order No. AA-P098A-TK

Developer's Tool Kit

digital
software

PRO/RMS–11:
An Introduction

Order No. AA-P098A-TK

November 1982

This document introduces the concepts and operations of Record Management Services (RMS-11) for the Professional Operating System (P/OS).

OPERATING SYSTEM AND VERSION: P/OS V1.0 or later
SOFTWARE VERSION: RMS-11 Version 2.0

digital equipment corporation · maynard, massachusetts

First Printing, November 1982

Printed in U.S.A.

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

ZK2170

HOW TO ORDER ADDITIONAL DOCUMENTATION

In Continental USA and Puerto Rico call 800-258-1710

In New Hampshire, Alaska, and Hawaii call 603-884-6660

In Canada call 613-234-7726 (Ottawa-Hull)
800-267-6146 (all other Canadian)

DIRECT MAIL ORDERS (USA & PUERTO RICO)*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire 03061

*Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575)

DIRECT MAIL ORDERS (CANADA)

Digital Equipment of Canada Ltd.
940 Belfast Road
Ottawa, Ontario K1G 4C2
Attn: A&SG Business Manager

DIRECT MAIL ORDERS (INTERNATIONAL)

Digital Equipment Corporation
A&SG Business Manager
c/o Digital's local subsidiary or approved distributor

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Northboro, Massachusetts 01532

CONTENTS

FIGURES

TABLES

PREFACE

DOCUMENT OBJECTIVES

This document introduces the major concepts of RMS-11: record formats, file organizations, and record access modes.

It also introduces the RMS-11 operations and defines key terms required for understanding RMS-11 capabilities and functions.

This document does not provide reference or usage information. You should read it, however, before you proceed to the other RMS-11 documents. See the Associated Documents section below.

INTENDED AUDIENCE

This document is intended for all users of RMS-11, including MACRO-11 and high-level language programmers. Nonprogramming users, programming users who are new to the Professional Operating System, and programming users who are new to RMS-11 will find this document useful.

ASSOCIATED DOCUMENTS

The PRO/RMS-11 Macro Programmer's Guide manual is a reference document for MACRO-11 programmers that describes the macros and symbols that make up the interface between a MACRO-11 program and the RMS-11 operation routines. Chapter 4, Operations, of this manual provides a background for the reference information.

CHAPTER 1

INTRODUCTION

The PDP-11 computer allows you to store large amounts of information. The combination of the software supplied by DIGITAL and the software that you write allows you to access and manipulate that information. This means that you can store, retrieve, and process (modify and delete) data according to your requirements.

All businesses, for example, need to store detailed and up-to-date personnel information. At a minimum, this information would include each employee's name, address, and social security number.

The employer will want to be able to retrieve this information in some orderly fashion, perhaps in alphabetical order by employee name or by social security number for tax purposes.

The employer also must be able to readily process the information: add information on new employees, delete information on employees who leave, and modify information that is incorrect or changes (the employee moves).

The ability to store and retrieve information and to process that information readily and in some orderly fashion implies that the information is stored in some orderly fashion, such as in records. A record is a logical unit of data, that is, an item or collection of items that are related in some manner.

The information on each employee -- name, address, and social security number -- would constitute one record. The information on each manufactured part in inventory -- part name, number, and price -- would constitute another, different type of record.

To keep records of one type separate from records of another type, records are organized into files. A file contains groups of records of the same type. One or more files, depending on the amount of data, would contain all the records of a specific type.

In a small business, all the employee records would be stored in one file; all the inventory records would be stored in another.

How the data is used helps determine how the records are stored in and retrieved from files so they can be processed, or used for a specific purpose. Data storage and retrieval is also called access.

To obtain a complete report on all employees, the employer might simply get each record, one after another. New records might be put at the end of the file, one after another. To withhold state income tax, the employer might use the address data to identify the records of employees who reside in that state.

The software that provides data access is RMS-11. RMS-11 is a set of routines that allows your programs to gain access to and process -- add, modify, and delete -- records and files. RMS-11 provides the connection between your program and the stored data that your program requires.

This document is an introduction to RMS-11. Data storage is described briefly in Chapter 2. File structure and access are described in Chapter 3.

The RMS-11 operation routines that provide file and record processing are introduced in Chapter 4.

CHAPTER 2

DATA STORAGE

The data that your programs use typically is stored on mass-storage devices, called disks. The operating system software controls these hardware devices, and allows your programs to access the data stored on them. Each device is controlled by a device driver, or software that handles the input/output (I/O) -- the writing of data into and reading of data out of storage.

The Files-11 Ancillary Control Processor (FCP) is the P/OS file control processor that catalogues and maintains files on the disks, and issues I/O requests to the device drivers.

The smallest unit of information stored on hardware devices is the bit. A bit is an area of disk or tape surface for which the magnetic orientation can be changed to one of two values, conventionally designated 0 and 1.

Information typically is grouped into units of 8 bits, called bytes. Bytes are used, for example, to represent alphanumeric characters in memory with the DEC multinational character set. Other ways of representing data, particularly numeric data, may require 2 or more bytes. A word, for example, consists of 2 bytes (or 16 bits).

DISK STRUCTURE

A disk is made up of one or more circular platters, arranged one above the other to form a cylinder. A disk is mounted on a drive, which is the electromechanical unit for reading and writing data on the platters. The disk platters spin and the read/write heads of the drive move across the platters to access the data. Data is usually written on both sides of a platter.

Figure 1 shows how data is stored on a disk. On most disks supplied by DIGITAL, a sector consists of 512 bytes. A track consists of all the sectors at a single radius on one disk platter, and a cylinder consists of all the tracks at the same radius on all the platters.

The disk drive can access all tracks on a single cylinder without changing the position of the read/write heads. Any sector on a disk thus has a unique physical address: its cylinder number, its track number, and its sector number on that track.

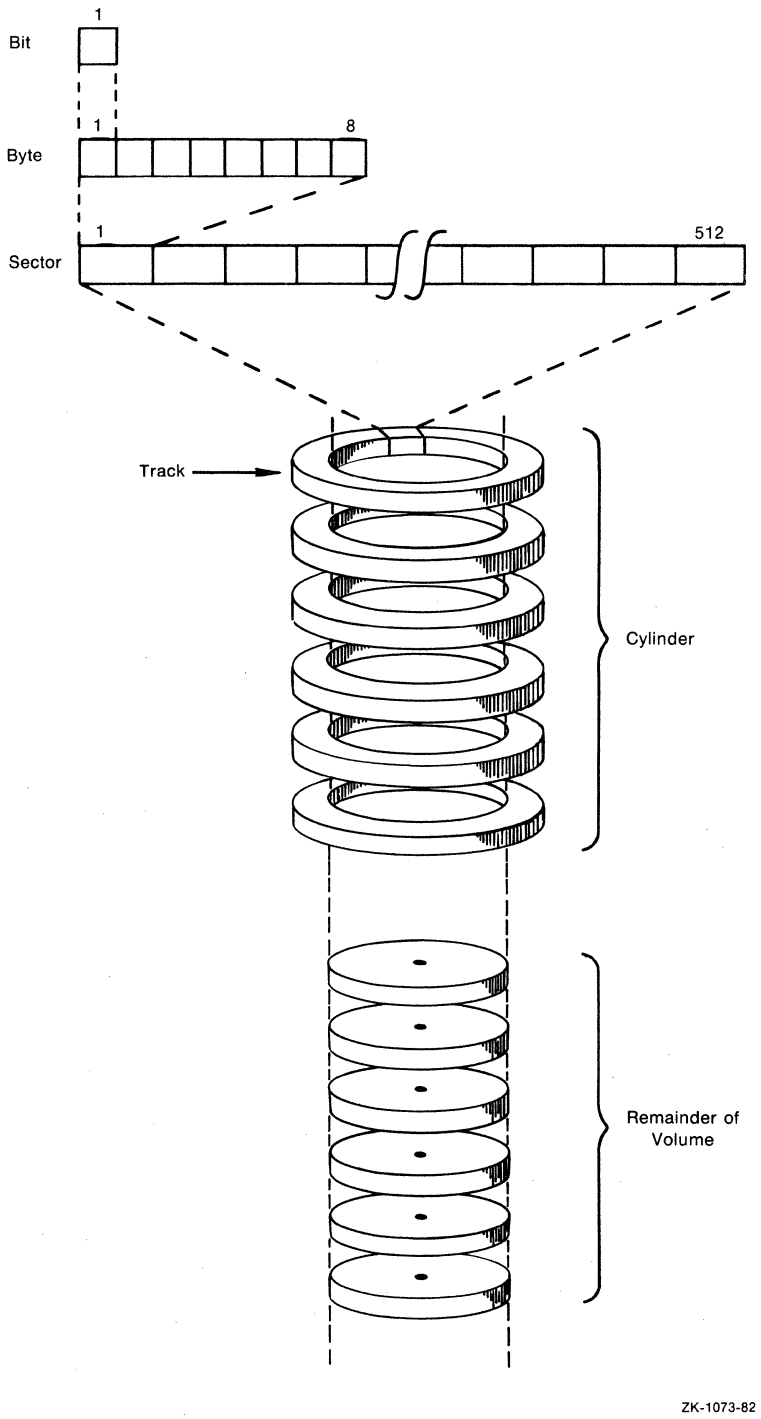

Figure 1: Physical Disk Structure

The device drivers impose a simpler logical addressing structure on each disk. They treat the disk as a single, logically contiguous, series of data units, called blocks. A block contains 512 8-bit bytes. Logical blocks are numbered sequentially, from 0 to n-1, where n is the number of blocks on the disk. Figure 2 illustrates this logical structure. Blocks, which are logically adjacent, frequently correspond to sectors, which are physically adjacent on the disk.

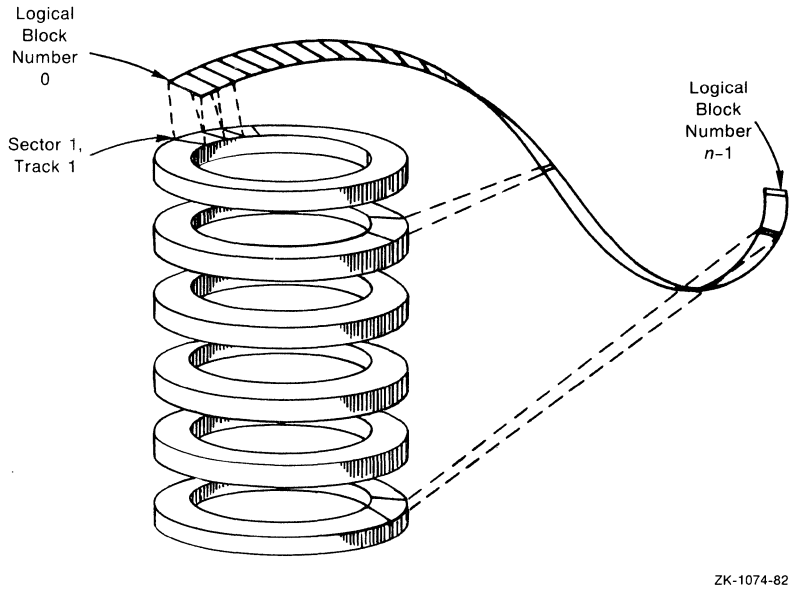

Figure 2: Logical Disk Structure

On disk, a file is simply a series of blocks, which contain your data organized into records. You can store multiple files on a disk. The file control processor treats each file as a unit, ignoring any blocks on the disk except those in the file being processed.

Figure 3 illustrates how blocks in a file -- called virtual blocks -- may be mapped to logical blocks. Virtual blocks are numbered sequentially in a file from 1 to n, where n is the number of blocks in a file.

The blocks in a file, however, need not be logically contiguous. As files are created or extended, the file control processor may allocate blocks to the file that are not next to each other on the disk. The blocks in a file, then, are virtually contiguous.

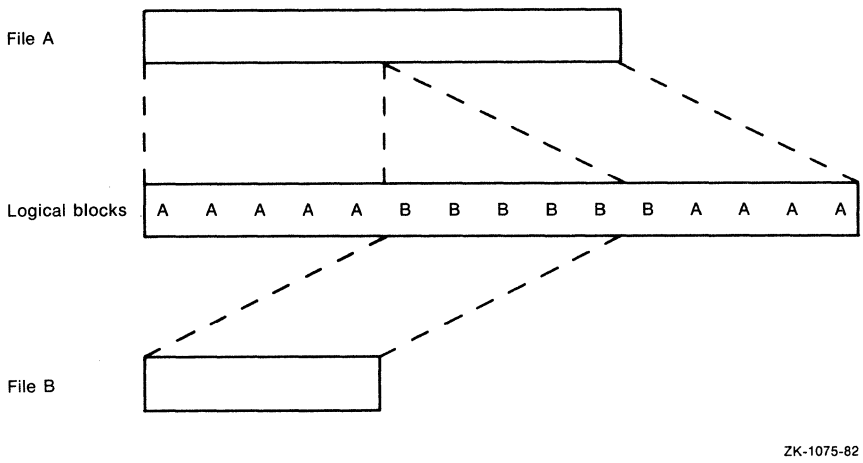

Figure 3: Virtual-to-Logical Block Mapping

Note that a virtual block number (VBN) and a logical block number (LBN) refer to the same physical unit of disk storage space. But although a virtual block also has an LBN, a logical block has a VBN only if it is allocated to a file.

To access files, the file control processor translates VBNs to LBNs and issues an I/O request to the device driver. The device driver, in turn, translates the LBNs to the physical location (cylinder, track, and sector) that is to be read or written.

Disk storage offers two main advantages: 1) it allows both sequential and random access and 2) it allows access sharing. Sequential access means that virtual blocks containing data are stored and retrieved consecutively, one after another.

On the other hand, random access (also called direct access) means that a specific block containing data can be located and retrieved without a search of all the blocks that precede it in the file. The time needed to access the data may thus be improved over sequential access.

In addition, disk storage allows access sharing. This means that more than one user can access the same disk at one time, and more than one user can be allowed to open the same file at one time.

CHAPTER 3

FILE STRUCTURE

The operating system software -- file control processor, device drivers -- handles files. Your programs, however, must be able to access the records within the files so they can process the data within the records.

RMS-11 allows you to define the internal structure of files (the size and arrangement of records within files) and provides operations that allow your programs to read and write records in files. RMS-11 thus provides the interface between the operating system and your programs.

You define the internal structure of a file when you create it by selecting record format and file organization.

Your selection of file organization may have to take into account the access modes that you will use to store and retrieve your data.

This chapter introduces the concepts of record format, file organization, and access modes. Chapter 4 introduces the RMS-11 operations.

RECORD FORMATS

RMS-11 does not handle, or process, data within records. Your program does that. However, to retrieve or store a record for your program, RMS-11 must know how large that record is. The record formats that allow you to define for RMS-11 the size of your data records are:

- Fixed length
- Variable length
- Variable length with fixed control (VFC)
- Stream

These are the standard RMS-11 record formats. In addition, you can specify undefined as the record format for non-RMS-11 files.

Fixed-Length Records

In a file that contains fixed-length records, every record is the same size. The size (number of bytes) is fixed at file-creation time and cannot be changed for the life of the file.

Typically, fixed-length records are used to contain data that is always the same length. This is the most efficient use, although your program need not supply data for every byte of a fixed-length record. RMS-11 will simply store and return the fixed-size record regardless of whether every byte in the record contains data that is meaningful to you. Your program must be able to recognize that some bytes in the record are "unneeded."

For example, you could use a fixed-length record to contain information on manufactured parts including, say, part number, date of manufacture, and price. Each of these fields would always be the same length and would always be filled.

If you also used a field for a part name, that field would vary in length with the names of the parts. However, as long as most of the fields contain data whose length does not vary, fixed-length record format may still be the most efficient way to store your data.

Variable-Length Records

In a file that contains variable-length records, records can be of different lengths, up to a maximum size that you specify. This maximum is fixed at file-creation time and cannot be changed for the life of the file.

Because the number of bytes of data per record is not fixed, each variable-length record is only as long as it needs to be to contain the record data. To keep track of the length of each record for reading and writing, RMS-11 maintains a length field which is prefixed to each record.

Thus, a variable-length record will require slightly more space than a fixed-length record to hold exactly the same amount of data. However, if the number of bytes of data in a record must vary, fixed-length records would waste space in unneeded bytes, and variable-length records may therefore be more efficient.

Figure 4 compares the fixed-length and variable-length record formats.

Typically, variable-length records contain text data that varies in length: for example, the names and addresses of employees. Using fixed-length records for this information could result in many unneeded bytes and inefficient use of storage space.

Variable-Length with Fixed Control (VFC) Records

A VFC record is much like a variable-length record except that in each record in the file a fixed-length control area precedes the variable-length data. This format allows you to construct records with additional data that labels, or identifies, the contents of the variable-length portion of the record.

For example, in a text file, each line of text constitutes the variable-length portion of a record. The fixed-length control area of each record contains a line number that indicates the sequence in which the line occurs in the text file: line 3, say, in a text file of 10 lines (that is, 10 records). An editing program can use the line numbers to locate a particular line of text easily in the file.

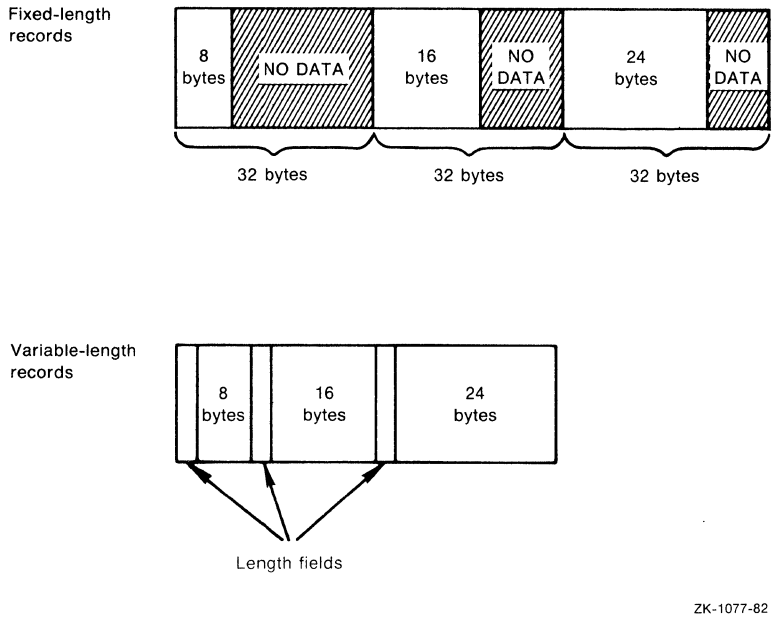

Figure 4: Fixed-Length and Variable-Length Record Formats

At file-creation time, you specify two sizes for the records in the file: the maximum size permitted for the variable-length portion and the fixed size of the control area.

As for variable-length records, RMS-11 keeps track of the length of each record by maintaining a length field for each record. Thus, in memory, a VFC record consists of a length field, a fixed-length control area, and variable-length data.

Figure 5 illustrates the VFC record format.

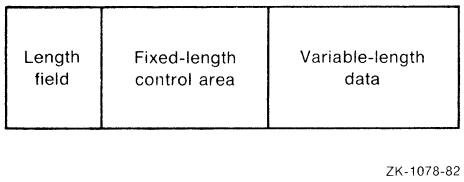

Figure 5: VFC Record Format

Stream Records

A stream record consists of a continuous series of ASCII characters delimited by a special character or sequence of characters called terminators: form feed, line feed, or vertical tab are all terminators.

For example, a text file might consist of a series of stream records terminated by carriage-return/line-feed characters pairs.

RMS-11 considers each record a series of bytes; the length of a stream record is determined by the position of the terminator.

Undefined Records

A file with undefined records either may have no record format or may contain records that are not in one of the four standard RMS-11 formats described in the previous sections.

RMS-11 considers a file with undefined records as a series of blocks; that is, it stores and retrieves data in units of 512-byte blocks. Your program must be able to interpret the contents of the blocks.

RMS-11's support of both undefined and stream records provides compatibility with non-RMS-11 files. This means that, with some limitations, RMS-11 can process files created under other file systems.

FILE ORGANIZATIONS

The arrangement of records within files directly affects access flexibility, or how quickly and easily RMS-11 can access those records. Your selection of file organization, therefore, should take access mode into consideration. For more information on access mode selection, see the Access Modes section and the File Structure Interdependencies section.

RMS-11 makes three file organizations available:

- Sequential
- Relative
- Indexed

Sequential Organization

In a sequential file, records normally are arranged within the file in the order in which they were written; that is, the first record written is the first record in the file, and the record written most recently is the last record in the file.

You can add records to a sequential file only at the logical end of the file. You can delete a record or series of records in a sequential file only by truncating the file from a specific point to the logical end of the file, which effectively deletes the record(s). You can, however, update a record -- retrieve it, change its contents, and rewrite it in the file -- as long as the length of the record has not changed.

Sequential file organization is supported for all record formats. In addition, sequential files allow shared access for reading data.

Figure 6 illustrates sequential file organization.

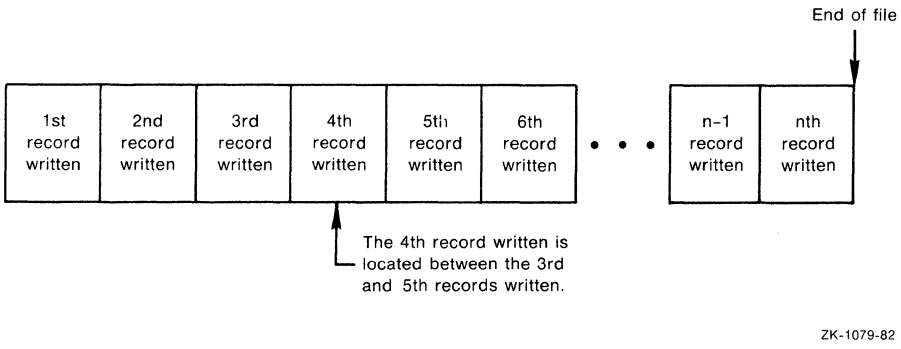

Figure 6: Sequential File Organization

Special Case - In a sequential file with fixed-length records, the records are numbered consecutively from 1 to n, where n is the number of records in the file. The record numbers are known as relative record numbers. In this case, records may not be arranged in the file in the order in which they were written. You can, for example, specify that relative record number 5 be written before relative record number 4. When relative record number 4 is written later, it will be inserted before relative record number 5. Fixed-length records in a sequential file can be retrieved by random access.

In addition, RMS-11 supports the use of unit-record devices (such as terminals and line printers) by treating them as sequential files. That is, your program can write to and read from a unit-record device in much the same manner as it would to a sequential file.

Relative Organization

A relative file consists of a series of cells, or fixed-length units of storage. Each record within a relative file is stored in a cell; however, not every cell need contain a record. The cells are numbered consecutively from 1 to n, where n is the number of cells in the file. As for sequential files with fixed-length records, the cell numbers are known as relative record numbers.

To randomly access a record in a cell, your program must keep track of the relative record number. One way to do so is to associate a value within the record (for example, an order number) with the relative record number.

Because records are stored by relative record number, they need not be arranged within the file in the same order in which they were written. For example, order number 4 may be completed and its record written to the file before order number 3. Thus, order number 4 will be stored in cell 4, and cell 3 will remain empty until order number 3 has been completed and its record written to the file.

Figure 7 illustrates relative file organization.

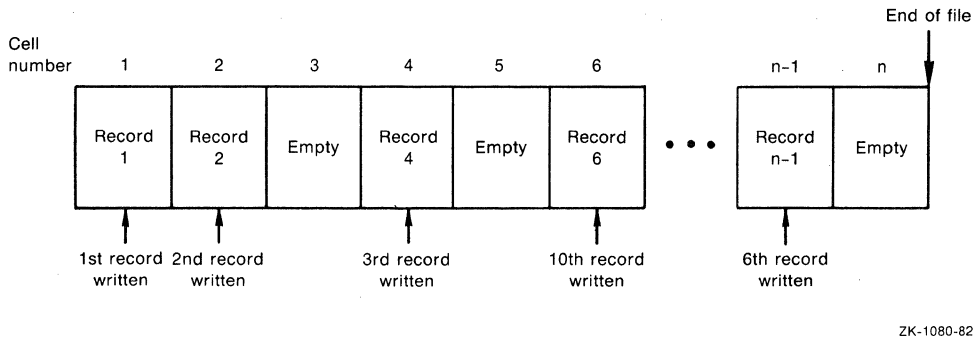

Figure 7: Relative File Organization

The records in a relative file can be fixed length, variable length, or VFC. Since there can be only one record per cell, RMS-11 determines the cell size based on the maximum record size that you specify. This means that with variable-length and VFC records there may be unneeded bytes in a cell (much like fixed-length records in which there may be unneeded bytes).

Unlike sequential files, relative files permit the following capabilities:

- Random access by relative record number, regardless of record format
- Record deletion
- Access sharing for both reading and writing data

Indexed Organization

In an indexed file, records are arranged in ascending order by key. A key is a data field within the record that RMS-11 uses to determine the order in which to access the records in the file. This is the only case in which RMS-11 interprets data within records; thus, a record can be identified by its contents, instead of by its position within the file.

When you create an indexed file, you must define one field of the record as the primary key. A key is defined by its location within the record, its data type, and its length. When a record is stored in that file, RMS-11 inserts the record in order based on the value that it finds in the primary key field; that is, RMS-11 inserts the record after a record with a lower or equal value in the primary key field, and before a record with a higher value in the primary key field.

You can optionally define other fields of a record as alternate keys. These keys specify alternate access orders for the retrieved records; they do not affect the order in which the records are arranged within the file.

The data type of values in key fields can be byte (character) string, signed integer, unsigned binary number, or packed decimal number.

For each data field defined as a primary or alternate key, RMS-11 constructs an index. A primary index contains the values in the primary key fields, and an alternate index contains the values in alternate key fields.

An indexed file is a heirarchical, or tree, structure consisting of levels of records. The highest level(s) are the index records. The data records comprise the lowest level. An indexed file consists of at least two levels of records: index and data. Each index record contains the highest key value in a group of records at the next lower level plus a pointer to that group of records.

Thus, each key value provides a logical access path to locate a specific record or set of records in a file. The indexes also allow your program to retrieve your records in a specific order. RMS-11 stores the indexes in the file itself.

For example, to maintain employee files in alphabetical order by employee name, you could create an indexed file and specify the employee-name field of the record as the primary key. RMS-11 would construct a primary index consisting of employee names, in alphabetical order, and insert the records in the file in that order.

Figure 8 illustrates an indexed file with only a primary key defined; note that the data records contain other fields, including address and badge number.

In many cases, employee records may need to be retrieved in different orders; for example, the payroll may be processed by badge number rather than by employee name. Thus, you might want to define the badge-number field as an alternate key for the file.

Figure 9 illustrates an indexed file with an alternate key defined.

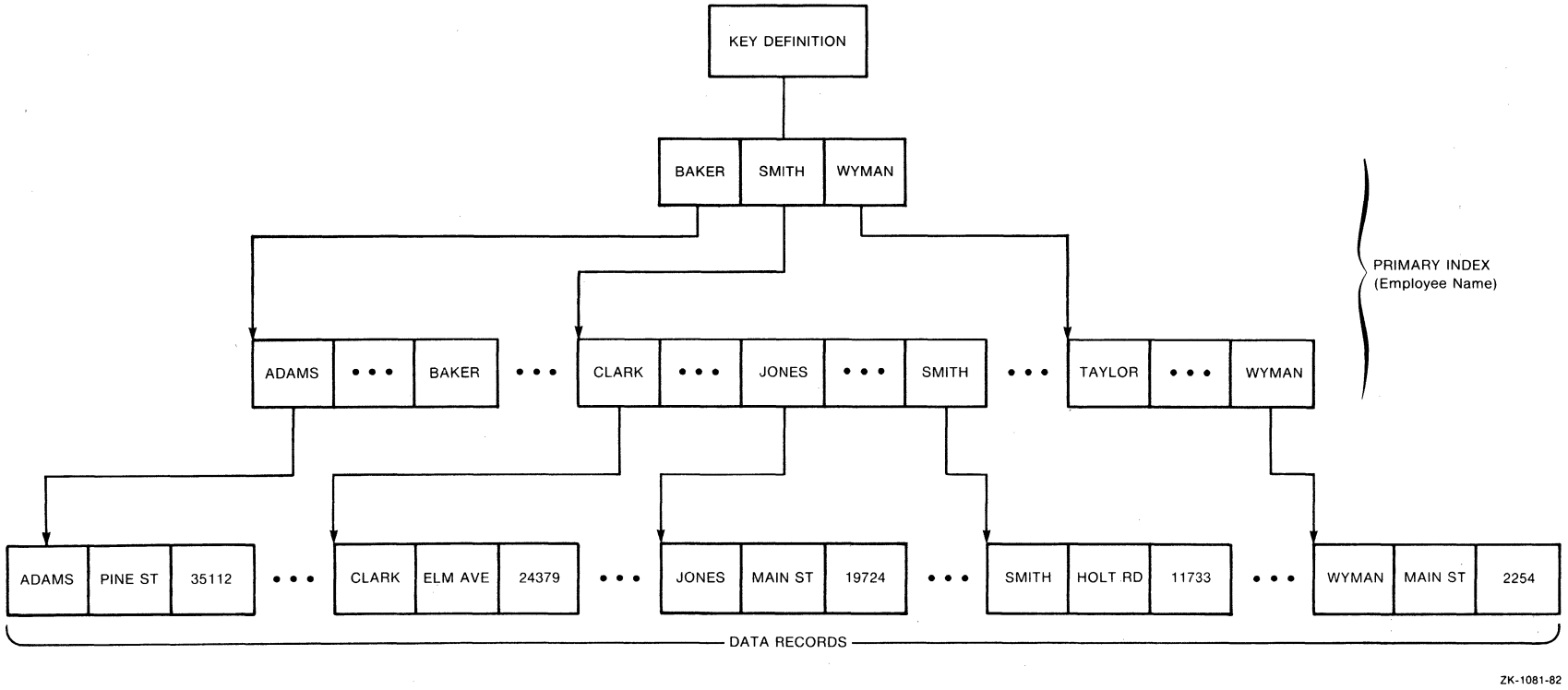

Figure 8: Indexed File Organization -- Single Key

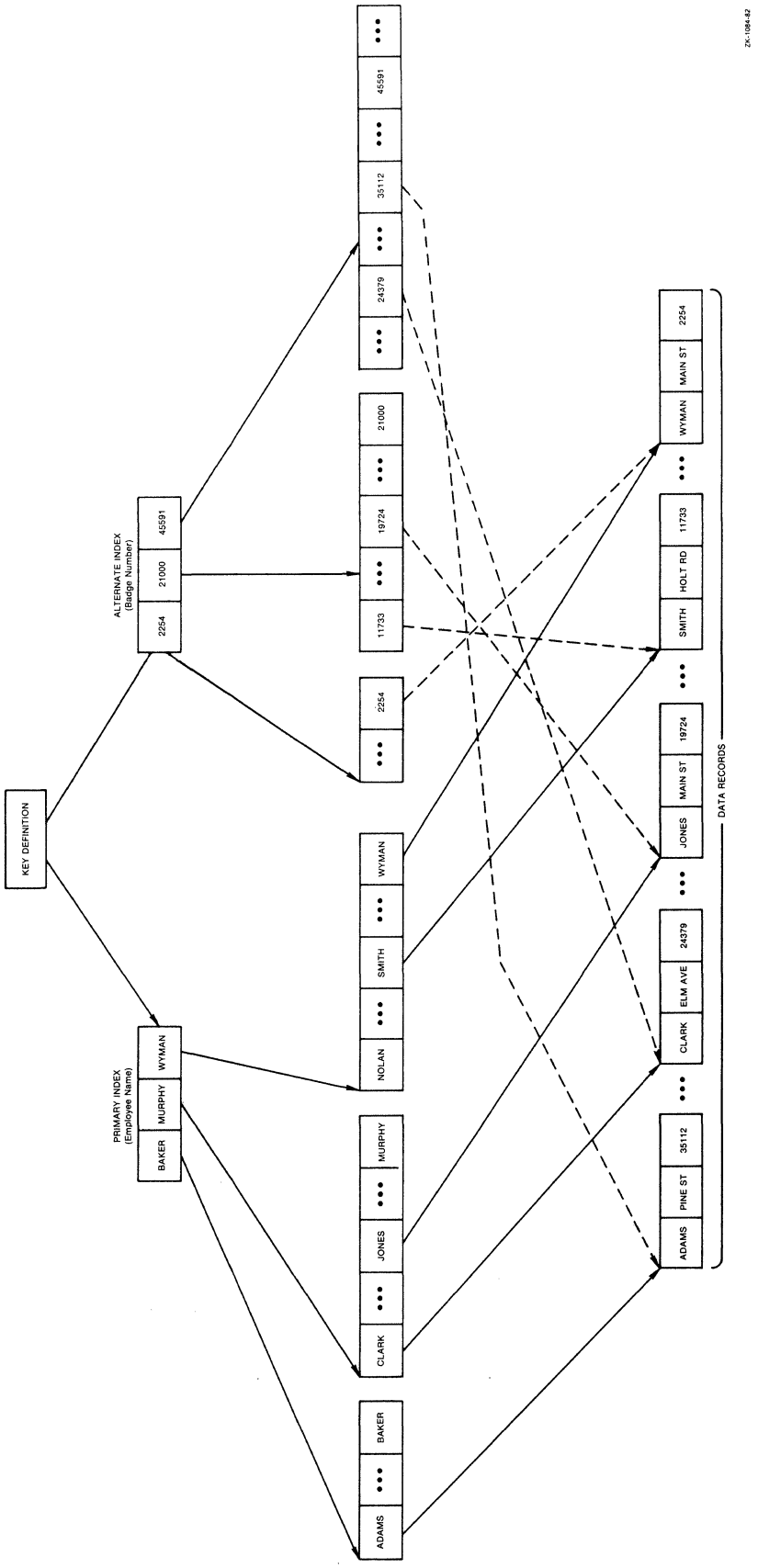

Figure 9: Indexed File Organization -- Multiple Keys

You can optionally specify three key value characteristics for individual keys:

1. Duplicate values for primary and/or alternate keys.

   This means that more than one record in the file can have the same value for the key.

   In the employee file example, duplicate key values could be allowed for employee name but not for badge number.

   By default, RMS-11 does not allow duplicate values for primary keys but does allow duplicate values for alternate keys.

2. Changeable values for alternate keys.

   This means that the data in the key field can be modified. For example, if employee address was also an alternate key in the employee file, key values would be allowed to change in case the employee moves.

   If the field does change, RMS-11 updates the appropriate index to reflect the new key value.

   By default, RMS-11 allows alternate key values to be changed. Note, however, that alternate key values can change only if duplicate key values are also allowed. Primary key values cannot be changed.

3. Null values for alternate keys.

   This means that the data in the key field can be null, that is, have no value. It is possible, for example, that employee information may be incomplete when the record is put into the file, and that one of the incomplete fields is a key.

   RMS-11 makes no alternate index entry for such a record; if the record is subsequently modified to include the information, RMS-11 makes an entry in the appropriate index. Note that primary key values cannot be null, nor can a null value later be changed unless the key was specified to be changeable.

   By default, RMS-11 treats any value in the key field as a true key value.

When keys are character strings, you can also specify that keys are segmented; that is, the keys consist of separate fields of the record. For example, you can define a key in an employee file as consisting of both the employee-name and badge-number fields. RMS-11 will concatenate the segments to form the key.

Indexed files permit fast sequential access (in key order) and random access by key value. Indexed files allow only fixed- and variable-length record formats.

ACCESS MODES

The access modes are the methods that RMS-11 uses to store or retrieve the contents of files. The contents of files can be either records or blocks.

The record access modes are:

- Sequential
- Random by record file address (RFA)
- Random by key

The block access modes are:

- Sequential
- Random by virtual block number (VBN)

In the sequential access modes, RMS-11 stores or retrieves records or blocks consecutively, one after another.

Sequential access is efficient when you need to read all the records or blocks in a file. For example, you would use sequential record access mode with an employee file to perform weekly payroll processing because you would always want to access all of the employee records.

In the random access modes, RMS-11 stores or retrieves records or blocks directly, using an identifier unique to the required record or block. This means that your program, not the sequence of records or blocks within the file, establishes the order in which records or blocks are processed.

Random access is efficient when you need to "jump around" in a file, rather than access all the records or blocks one after another. For example, if your program needs to process only the records for employees who receive sick pay during a payroll period, you would use a random record access mode to access only those records, by employee name.

Record Access Modes

For **sequential access**, record storage or retrieval begins at a point in the file and continues with consecutive records through the file. Your program issues a series of requests to RMS-11 to successively retrieve the next record in the file.

Sequential access does not allow you to backspace through a file; you must reopen the file, or rewind to the first record in the file, and begin again at the first record.

For **random by RFA access**, RMS-11 uses the record file address (RFA) as an identifier to gain direct access to a specific record in a file, without a successive search of all the records that precede it. The RFA is a unique record identifier that RMS-11 establishes for every record that it writes to a file.

When RMS-11 stores a record in a file, it establishes the RFA for that record and returns the RFA information to your program. Your program can then use the RFA to retrieve the record.

Note that because only RMS-11 can establish the RFA, you cannot store a record by RFA (that is, you cannot specify an RFA for the record).

RFA access can be used with files of any organization and record format; it is the fastest way to read a record randomly in an indexed file.

For **random by key access**, your program specifies an identifier that allows RMS-11 to gain direct access to a specific record without a successive search of all the records that precede it. For sequential files with fixed-length records or for relative files, the identifier is a relative record number (RRN). For indexed files, the identifier is a key value.

To randomly read a record in an indexed file, your program specifies a key value and the index (primary, first alternate, and so on). RMS-11 searches the index to locate the record with the specific key value. For byte (character) string keys, your program can also specify the number of characters on which the match is to be made; this is called generic match.

In addition to specifying a key value or RRN, you can also specify match criteria:

- An exact match on the key value or RRN, that is, only the record with the specified key value or RRN
- A greater-than match on the key value or RRN, that is, the next record after the record with the specified key value or RRN
- A greater-than-or-equal-to match on the key value or RRN, that is, either 1) the record with the specified key value or, if there is no such record, the record with the next higher key value; or 2) the record in the cell with the specified RRN or, if that cell is empty, the first record in a cell after the specified RRN

For example, in an employee file the second alternate key might be the social security number. If you want to read the first record that contains a social security number beginning with 175, you would specify key number 2 (the index), key value 175, and that the match is to be an exact match on the first three characters of the key.

Block Access Modes

For **sequential access** to blocks, RMS-11 stores or retrieves data as a consecutive series of blocks. Your program issues a series of requests to RMS-11 to successively access the next block(s) in the file.

For **random by VBN access**, your program specifies the virtual block number (VBN) as the identifier of the first block to be accessed. RMS-11 uses the VBN to gain direct access to the specified block(s) without a successive search of all the blocks that precede it.

Access Mode Switching

RMS-11 allows your programs to change access modes -- either among the three record access modes or between the two block access modes -- at any time during file processing, as long as the file organization allows the access modes selected.

However, you must make the basic selection of either record access or block access when you open the file. You can switch between record and block access only by closing and reopening the file.

Access mode switching is most useful when a random access mode is used to retrieve the first record or block of a series, and then sequential access mode is used to retrieve subsequent records or blocks of that series.

For example, if an indexed employee file uses the department code as one of the alternate keys, your program could print a report on all the employees in a specific department. It would do this by using key record access mode to locate the first record whose department field matches the specified key and then switch to sequential record access mode to consecutively retrieve all the remaining records of the employees in that department.

FILE STRUCTURE INTERDEPENDENCIES

As the previous sections indicate, your selection of record format, file organization, and access mode(s) cannot be made independently of each other. For example, if you want to use VFC records, you cannot choose indexed file organization; and if your records are undefined, you must use a block access mode.

File Organization and Access

You can use sequential record access mode with any RMS-11 file organization. The organization of the file determines the order in which the records are stored and retrieved.

You can use random by RFA record access mode and VBN access mode with any file organization. For random by RFA record access mode, your program must save the RFAs established by RMS-11 when it stores the records in the file.

Random by key record access mode also can be used with any file organization. For sequential files, however, the record format must be fixed length.

You can use sequential block access mode with a file of any organization, including non-RMS-11 files.

The block access modes may improve access time because RMS-11 ignores file structure and record characteristics. Your program, however, must be able to interpret the contents of the blocks.

Sequential Access to Sequential Files. If you use sequential record access mode with a file of sequential organization, your program must open the file and read through the records one after the other beginning with the first record. Even though the record you require may be near the end of the file, your program must read every record that precedes it. In addition, records can be added sequentially only at the end of a sequential file.

Sequential Access to Relative Files. If you use sequential record access mode with a relative file, RMS-11 will search for successive record cells, ignoring cells that are empty. For example, if a relative file contains records in cells 1, 2, 4, and 8, and all other cells are empty, RMS-11 will check each cell from 1 through 8 but will read only RRNs 1, 2, 4, and 8, in that order.

To retrieve a specific record in a relative file, using sequential access, your program first must retrieve all the records in all the preceding cells.

To insert a record sequentially in a relative file, RMS-11 puts the record in the next cell after the cell just accessed (the cell with an RRN one number higher than the current cell), if that cell is empty. If the cell contains a record, RMS-11 will return an error code.

Sequential Access to Indexed Files. If you use sequential record access mode to retrieve records in an indexed file, you must specify a key for RMS-11 to use to establish the order in which it will read the records sequentially. The key you specify can be the primary key or any alternate key. You can use sequential record access mode to retrieve records in the order represented by any index for the file.

RMS-11 will retrieve the records in ascending order by key value; that is, if you specify the employee-name key, all employee records will be read beginning with the A's.

To store, or insert, a record sequentially in an indexed file, RMS-11 uses the primary key value to insert the record in order in the file, and updates the primary and alternate indexes according to the key fields within the record.

In a series of sequential insertions, RMS-11 verifies that the primary key of each new record has a value equal to or greater than the primary key of the previously inserted record.

Random Access to Sequential Files. If you use random by key record access mode with a sequential file, the record format must be fixed length. Your program specifies the RRN of a record in the file and the match criterion, and RMS-11 stores or retrieves that record.

When you are reading a record from a file, RMS-11 will retrieve that record or return an error code if the specified record position lies beyond the end of the file.

Note that RMS-11 does not initialize sequential files. Thus, if you attempt to read a record from a sequential file with fixed-length records to which records have been randomly written, you may get "garbage" data. That is, if you specify an RRN that is not beyond the end of the file but at which no record has been stored, RMS-11 will return whatever data happens to be at that location.

When you are writing records to a file and specify an RRN, RMS-11 will store the record in that RRN's position in the file regardless of whether a record with that RRN already exists. If the specified record position lies beyond the end of the file, RMS-11 will extend the file and store the record at the designated RRN.

If you know the RFA, you can use random by RFA record access mode to read sequential files. This is the only random record access mode in which your program can read sequential files of variable-length or VFC record format.

Random Access to Relative Files. In random by key record access mode, your program specifies the relative record number of a record's cell in the file. RMS-11 then stores or retrieves the record in that cell.

When you are reading records from a file and specify an exact match on an RRN, RMS-11 will retrieve the record in that cell or return an error code if the cell is empty.

When you are writing records to a file, RMS-11 will store the record in the cell if the cell is empty or return an error code if the cell contains a record. Only on an UPDATE record operation (see Chapter 4) will RMS-11 write a record into a cell that already contains a record.

If you know the RFA, you can use random by RFA record access mode to read relative files.

Random Access to Indexed Files. In random by key record access mode, your program supplies the key value of a record in the file. RMS-11 then retrieves the record associated with that key value.

If you know the RFA, you can use random by RFA record access mode to read indexed files. RFA access is faster than key access because it bypasses the searching of the index, but your program must keep track of the RFAs.

Summary

Table 1 summarizes the RMS-11 file structure interdependencies.

Table 1: File Structure Interdependencies

Device	File Organization	Record Access Mode	Record Format
Disk	Sequential	Sequential, Random by RFA	Fixed, Variable, VFC, Stream
		Random by key	Fixed
	Relative	Sequential, Random by key, Random by RFA	Fixed, Variable, VFC
	Indexed	Sequential, Random by key, Random by RFA	Fixed, Variable

NOTE: The block access modes can be used to access files of any organization and any record format. A block access mode must be used to access a file with undefined record format.

FILE DESIGN

To create a file that will allow your program to store, retrieve, and process your data records readily and efficiently, you must specify to RMS-11 the file and record attributes. The attributes consist of the device, file organization, and record format, plus additional values that describe the characteristics of your files and records.

You can design files with the attributes you require and create them by using the CREATE operation routine (see Chapter 4, Operations).

RMS-11 requires this information to access your data. The information is stored in the file directory and, for relative and indexed files, in the file prologue, and is passed to RMS-11 as required for the file and record operations. The prologue consists of the first blocks of a relative or indexed file and is used to store specific relative or indexed file attributes that cannot be stored in the file directory.

Table 2 lists the RMS-11 file and record attributes.

Table 2: File and Record Attributes

Attribute	Explanation
File Attributes:	
File specification	System node, device, account, file name, extension, and version
File organization	Sequential, relative, or indexed
Allocation	Initial size of the file, in blocks
Extension	The number of blocks to be added to the file each time its size must be increased
Bucket size[1]	For relative and indexed files, the number of blocks per bucket
Contiguity	Whether disk space is to be allocated to the file in continuous, adjacent logical blocks
Location	Where the file is to be physically placed on the disk
Protection	The type of access different categories of user may have
Maximum record number	For relative files, the maximum number of records the file can contain

(continued on next page)

1. See the File Processing section in Chapter 4 for more information.

Table 2: File and Record Attributes (Cont.)

Attribute	Explanation
Record Attributes:	
Record format	Fixed length, variable length, VFC, stream, or undefined
Record size	The size of the record in bytes: for variable-length and stream records, record size is a maximum length; for fixed-length records, record size is the exact length of each record; for VFC records, record size is the maximum allowable length of the variable portion
Control field size	For VFC records, the size in bytes of the fixed-length portion of the record
Block spanning[1]	For sequential files, the ability for records to span physical block boundaries
Record-output handling	Carriage control: how a record is to be treated when written to a terminal or line printer
Indexed File Key Attributes:	
Key number	The number of the key
Key position	The location of the key within the record
Key size	The length of the key, in bytes
Key data type	Byte (character) string, signed integer, unsigned binary number, or packed decimal number
Key name	The name for the key (optional)
Duplicate key values	The ability to have more than one record in the file with the same primary or alternate key value

(continued on next page)

1. See the File Processing section in Chapter 4 for more information.

Table 2: File and Record Attributes (Cont.)

Attribute	Explanation
Indexed File Key Attributes (Cont.):	
Changeable key values	The ability to modify alternate key values
Null key values	The ability to have null alternate key values
Segmented keys	The ability to use separate fields of the record as the key value
Bucket fill size[1]	The level to which buckets are to be loaded with records

1. See the File Processing section in Chapter 4 for more information.

CHAPTER 4

OPERATIONS

The RMS-11 routines operate either on a file as a whole or on a record or block within a file. The operation routines allow your programs to add, retrieve, modify, and delete files, records, or blocks.

At the file processing level, RMS-11 makes requests of the file processor to access files, by means of the directory and file operations.

At the record processing level, RMS-11 stream and record operations provide access to individual records.

At the block processing level, RMS-11 block operations provide access to the individual virtual blocks of a file, regardless of file organization and record format.

FILE PROCESSING

The RMS-11 directory and file operations are the interface between your program and your data files.

The directory operations affect only file specification entries in directories. RMS-11 operations construct and use file specification strings and file identifiers, including wildcard file specification strings. The directory operations are:

- ENTER -- places a file specification in a directory
- REMOVE -- deletes a file specification from a directory
- RENAME -- replaces an existing file specification with a new one
- PARSE -- returns file specification information to your program
- SEARCH -- examines one or more directories for a specified file and returns the file specification and location in a form that can be used by other directory operations or by file operations

The file operations provide access to files as whole entities (that is, they do not provide access to records within files). These operations are:

- CREATE -- creates a new file with the attributes you specify and opens it for processing
- OPEN -- makes an existing file available for processing
- CLOSE -- terminates access to a file
- ERASE -- deletes a file and removes its directory entry, if one is specified
- EXTEND -- increases the allocated size of an open file
- DISPLAY -- returns file information to your program

I/O Buffers and Data Transfer

For each file that is open for processing, RMS-11 requires that you provide at least one I/O buffer for its internal use. RMS-11 either transfers the data from disk to an I/O buffer, or from an I/O buffer to disk.

Data in sequential files is transferred in blocks. The user can define whether records in files can cross block boundaries. If you specify block spanning when you create a file, records may continue across block boundaries. This means that they can be stored most efficiently within a file.

When records are restricted by block boundaries, they must be less than or equal to 512 bytes. Unneeded bytes may remain after each record to the end of that block if the next record will not fit within those bytes.

Data in relative and indexed files is transferred in buckets. A bucket consists of one or more blocks that RMS-11 treats as a unit. The bucket size, or number of blocks in a bucket, is user specified when a file is created, and cannot be changed unless the file is redesigned. Although records may span block boundaries, they cannot span bucket boundaries.

For indexed files, you can also specify a bucket fill size. This file attribute directs RMS-11 to load records into buckets only to the particular level -- number of bytes -- specified, leaving free space within each bucket. If a large number of random insertions are to be made to the file, filling buckets only partially can improve processing time.

Buckets are an RMS-11 concept; thus, when RMS-11 requests an I/O operation for a file, it directs the file control processor to move a bucket by specifying the VBN for the first block in the bucket and the size of the bucket in bytes.

Access Sharing

The degree to which other users can access files is determined by the protection code and by access sharing.

The protection code is a file attribute (see Table 2) that describes the types of access granted to different users or potential users of the file. This attribute is defined for the file at the time the file is created, and is checked by the file control processor to determine whether the user of an accessing program has the right to process or delete the file. The protection code can also be changed when the file is closed.

In many cases, more than one program must access the same file at the same time. RMS-11, therefore, permits access sharing, which is controlled by the accessing programs at the time a file is opened for processing.

Each program to open a file supplies two items of information:

1. The types of operations it will perform on records
2. The types of record operations it will allow other accessing programs to perform

The file control processor will check the file's protection code to ensure that the intended operations are allowed. If so, the file control processor will then check that the operations requested and the operations allowed are compatible with the operations being performed and with the operations allowed by all current accessing programs (if any) of the file. To share access to the file, accessing programs must not only meet the protection criteria but must also specify compatible access sharing.

To protect data in a file in which access sharing is allowed, RMS-11 provides bucket locking. When a set of programs that are concurrently accessing a relative or indexed file allows access sharing for write operations, each bucket moved from disk is locked against access by other programs until RMS-11 unlocks it, usually after completion of the operation, or at the start of the next operation on the stream.

Access sharing also depends on file organization. All files can be shared for reading by any number of programs, but sequential files cannot be shared for writing. Relative and indexed files can be shared for writing by multiple programs.

RECORD PROCESSING

Once file access is established, the RMS-11 stream and record operations are the interface between your program and the records your program requires.

An access stream is a path between your program and the records in a file (or blocks in a file; see the Block Processing section).

The stream operations are:

- CONNECT -- initiates an access stream
- DISCONNECT -- terminates a stream

- FLUSH -- writes the contents of I/O buffers (modified records) to the file
- FREE -- releases control of the record or block most recently accessed by the stream
- REWIND -- resets stream context to the beginning of the file (to the first record in the file)
- WAIT -- suspends processing until an outstanding asynchronous operation is completed

The record operations process records within files. These operations are of three types: locate, read, and write. Locate and read operations do not affect the data contents of the records; for example, a FIND operation merely returns the position of a record in a file. Write operations, on the other hand, may result in altered data; for example, an UPDATE operation replaces the contents of an existing record with new data.

The record operations are:

- FIND -- reads a record from a file to an I/O buffer and sets the current-record stream context to that record
- GET -- reads a record from a file to an I/O buffer and then to a user buffer, and sets the current-record stream context to that record
- PUT -- writes a record from a user buffer to an I/O buffer and then to a file
- UPDATE -- transfers a modified record from a user buffer to an I/O buffer and then to a a file, overwriting the previous version of the record in the file
- DELETE -- removes an existing record from a relative or indexed file
- TRUNCATE -- effectively deletes all records in a sequential file from the current record through the logical end of the file

Record Context

Each record operation processes one record at a time along the path established by the access stream. RMS-11 keeps track of the stream's position in the file so that it knows which record is the target of the operation. The stream's position is called its context, which is either the current record or the next record, depending on the operation. The stream's position changes at the completion of the operation.

In general, the current record is the record that was the target of the just-completed operation (the record operated upon), and the next record is the record that follows the current record.

For example, when you use a CONNECT stream operation to establish an access stream to a file opened by means of the OPEN or CREATE file operation, the current-record context is undefined (no record operation has been performed), and the next-record context is the first record in the file.

If the file is a sequential file and you use a sequential-access FIND record operation, the target will be the next record, which was established by the CONNECT operation as the first record in the file. After the FIND operation is completed, the current record will be the found record (the first record in the file), and the next record will be the record after the found record (the second record in the file).

If you then use a GET record operation, the target will be the current record, which was established by the FIND operation as the first record in the file. After the GET operation is completed, the current record will be the retrieved record (the first record in the file), and the next record will be the record after the retrieved record (the second record in the file).

To modify the record, you can use the UPDATE record operation. Its target will be the current record, which was established by the GET operation as the first record in the file. After the UPDATE operation is completed, the current record will be undefined (it was overwritten), and the next record will be the record that follows the overwritten record.

After a DISCONNECT stream operation, there is no current-record or next-record context.

Although only one record can be processed at a time by means of a stream, more than one stream can be connected to a relative or indexed file. For example, you can open an indexed file and, with one stream, use the primary index to access records randomly and, with a second stream, use an alternate index to access records sequentially.

Record Access Modes

For the FIND, GET, and PUT record operations, your program specifies record access mode, which determines which record is the target of the operation.

A sequential-access FIND operation can be used to position to records from a file of any organization. The target of a sequential-access FIND operation is the next record, as established by a previous CONNECT or REWIND stream operation or a FIND or GET record operation.

Repeated sequential-access FIND operations may be time-consuming for files with a large number of records. For relative and indexed files and sequential files with fixed-length records, you can use a key-access FIND operation, the target of which is the record whose RRN or key value matches the one that you specify for the operation.

If you know the RFA for a disk file of any organization, you can also use an RFA-access FIND operation, the target of which is the record whose RFA matches the one that you specify for the operation.

Exactly the same dependencies exist for GET operations, which are the same as FIND operations except that records are moved to the I/O buffer and then on to the user buffer. In addition, the target of a sequential-access GET operation depends on whether it was immediately preceded by a FIND operation. If so, the target is the current record; otherwise, the target is the next record.

A sequential-access PUT operation can be used to write records to a file of any organization. The target of a sequential-access PUT operation depends on the file organization: for sequential files, the target is the end of the file and for relative files, the target is the next record. For indexed files, RMS-11 inserts the record in primary key order and updates the indexes.

The target of a key-access PUT operation for a relative file or for a sequential file with fixed-length records is the cell specified by the RRN. For an indexed file, there is no target; RMS-11 inserts the record and updates the indexes.

Note that you cannot specify RFA access for PUT operations. For more information, see the Access Modes section in Chapter 3.

Record Transfer Modes

Your program can use either of two record transfer modes to perform record operations: move mode or locate mode.

In both modes, a user buffer, established by your program, is required in addition to the I/O buffer. For read operations in move mode, RMS-11 first transfers the data into the I/O buffer, then copies the record into the user buffer for processing by the program.

For write operations in move mode, your program builds or modifies the record in the user buffer. Then, upon a write request, RMS-11 copies the record into the I/O buffer before writing it to the file.

In locate mode, on the other hand, your program can, under some circumstances, access the record in the I/O buffer directly. This may reduce the amount of data movement, thereby reducing processing time.

BLOCK PROCESSING

Your program can bypass RMS-11 file and record structures and use block access to process 512-byte blocks one after the other. Your program identifies the starting VBN of the file and the number of bytes to be accessed.

When you use block access, RMS-11 regards the file as a series of blocks, rather than as a series of records. RMS-11 keeps track of the stream's position, or block context, which is either readable block or writable block, depending on the operation.

Block access requires minimum time and space for processing; however, your program must be able to interpret the contents of the blocks.

The block access operations are:

- READ -- gets blocks from a file
- WRITE -- puts blocks in a file

Block Access Modes

For the READ and WRITE operations, your program specifies block access mode -- sequential or random by VBN -- which determines which block is the target of the operation. Upon completion of an operation, RMS-11 resets the readable-block and writable-block contexts.

MACROS AND SYMBOLS

The RMS-11 operation routines are fully accessible only from a MACRO-11 program. High-level languages restrict your options for some operations; see your particular language documentation for information on its support of RMS-11.

The interface between a MACRO-11 program and the RMS-11 operation routines is defined by the RMS-11 macros and symbols. Your program uses the RMS-11 macros to call the RMS-11 operation routines.

Information is passed between the calling program and an RMS-11 operation routine by means of control blocks, which consist of structured series of data fields that contain the information. The information passed includes the attributes that your program supplies to RMS-11 for file and record access and the returned values that RMS-11 supplies to your program as a result of an operation. RMS-11 macros also allow your program to declare and manipulate the control blocks.

The fields of a control block are referenced by symbols that name the locations of fields within control blocks (field-offset symbols) and the codes and bit masks used within the fields (code and mask symbols).

The RMS-11 control blocks are:

- Area allocation (ALL) block -- contains information about a file area
- File date (DAT) block -- contains file creation and revision dates
- File access block (FAB) -- contains information about the file, including device, file attributes, access sharing, record format, record blocking, record-output handling
- File key (KEY) block -- contains index and key information for an indexed file
- File name (NAM) block -- contains file specification information
- File protection (PRO) block -- contains file owner and protection information
- Record access block (RAB) -- contains information about a stream or record and record access mode
- File summary (SUM) block -- contains file and area information for an indexed file

In addition to calling RMS-11 operations and declaring and manipulating control blocks, RMS-11 macros allow your program to:

- Declare and manipulate memory space (pools)
- Declare RMS-11 facilities that are required for certain operations based on file organization
- Extract from a macro library definitions for RMS-11 macros and symbols

INDEX

PRO/RMS-11: An Introduction
AA-P098A-TK

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or Country

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

BSSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03061

Do Not Tear - Fold Here