The Control Data® Star-100 file storage station

by G. S. CHRISTENSEN and P. D. JONES

Control Data Corporation
St. Paul, Minnesota

INTRODUCTION

Successful experience with the Control Data® 6000[1] and 7000[2] computer series has led to implementing improved concepts[3,4,5] of distributed computing in the STAR-100 computer system. In the STAR system different computing functions have been physically separated from one another. Each computing function is performed by an independent system unit which possesses its own processing logic and memory. Thus each is performed in its own right in an optimal manner.

STAR-100 computer[6] is a high speed processor capable of producing 100 million results (from a multiply operation, for instance) per second in its 4 or 8 million byte core memory. STAR itself cannot perform data input/output, this is performed by input/output units called stations which have channel interfaces to STAR. A station consists primarily of a mini-computer specially designed for data handling. The STAR design is thus simplified by not having to contain device interfaces; this modularity is important in the design of large computer systems.[7] Also the processor overhead of driving peripheral devices is relegated to the stations thus freeing STAR for additional user computation. Experience in several hundred Control Data® 6000 computer sites has shown it impossible to operate very high speed computers efficiently without distributing peripheral functions. As well as distributing the peripheral device drivers in STAR it has been found possible to perform system functions, such as file management, in the stations. So far 9 different STAR station types have been identified and built, these include: maintenance and monitoring, paging, storage, media (tape and disk), unit record, communication, display/edit, graphic and service. These contain the same basic hardware and software but vary at the device controller and system software interface level. The service station is a key station in that it manages the system resources and provides fan-out to the second level stations.

Operating system functions are thus distributed in a manner which closely follows the distribution of the hardware. The connecting links between the distributed operating system functions are controlled by a set of system messages and message handling is a key factor in efficient operation of the system.

The choice of where each operating function should be located is often self-evident, although a few functions are assumed to be movable from one element to another. Any final decision regarding function locations may depend on experience with particular work loads. In general each operating function is located closest to the resource being used and may be local or remote to the STAR processor. This provides modularity of both hardware and software and such advantages as:

- independence from other units, particularly in the areas of non-propagation of errors throughout the system and more immediate action on fault conditions.
- capability to be independently maintained.
- easier replacement of future new hardware or software parts.
- easier addition of new types of stations.

Figure 1 illustrates the layout of a large STAR system showing the connections between the various stations.

A STAR central processor with its immediate storage is simply another station within the system—a data processing station—and in no way does it have any extra authority. It does, however, have two stations

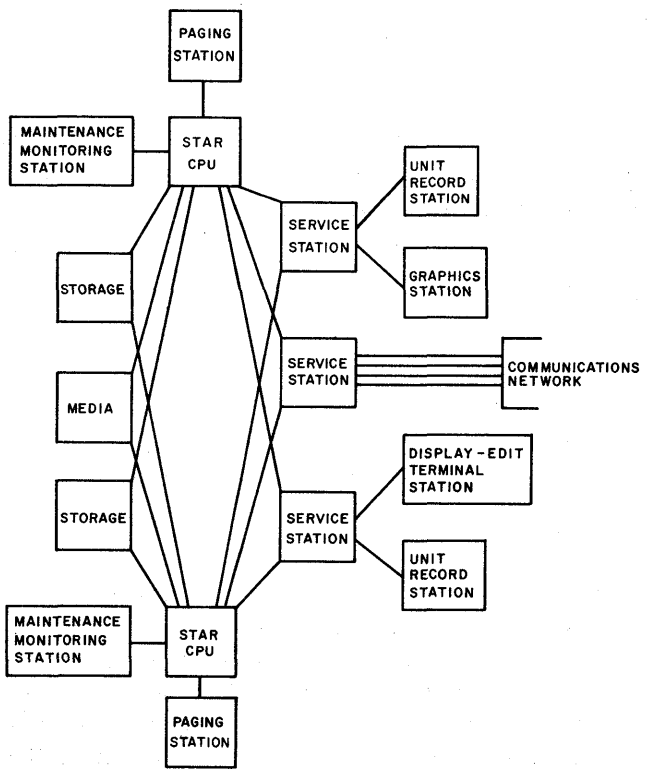


Figure 1—STAR system showing station connections

fairly intimately connected, the paging station and the maintenance/monitoring station. The paging station, under control of the hardware virtual page mechanism and the operating system, provides temporary storage for programs exceeding the available core space. The maintenance station, besides its functions of off-line and on-line fault diagnosis/repair and preventive checking, has the capacity to collect detailed information about STAR's performance.

The data management function is performed by programs executed within the central processor. These functions include merge, sort, select, scan, append, extract and insert. The data manager in turn exploits the storage station via message commands. This paper describes the storage station which manages the storage and retrieval of working and archival files.

STATION HARDWARE

The hardware used to implement the distributed computing concept in STAR is designated as various classes of input/output stations. Each Star channel terminates at a station with a common interface. The station (Figure 2) consists of an SCU (Station Control Unit) and an SBU (Station Buffer Unit).

The SCU consists of a mini-computer, display/keyboard, small drum and channel interfaces which exist with power supplies, cooling fan and operator panel in one cabinet. The mini-computer has an instruction set which caters to bit and byte manipulation. It contains 8K (K=1024) 8-bit bytes, expandable to 16K of 1.1 microsecond core memory. There is a 200 nanosecond version of the same meory but the 1 MIP (million instructions per second) rate of the computer is adequate for most present applications. The drum has an average access time of 17 milliseconds and a capacity of approximatly 80,000 bytes. It is used as a store for program overlays and also as a refresh memory for the display console.

The mini-computer (or buffer controller) provides a single, parallel-block transfer channel with hardware control for high speed data transfer. Its maximum rate is one 16-bit word plus two parity bits per memory cycle, 1.1 microseconds. The buffer controller also provides up to 512 normal channel bits for lower speed data transfer and device and station control. These bits are organized into 16 input channels and 16 output channels with 16 bits in each channel. Their use is determined by the individual peripheral devices on the station. The normal channel bits of the buffer controller provide the primary mechanism for control of the other station elements and the attached devices. A direct

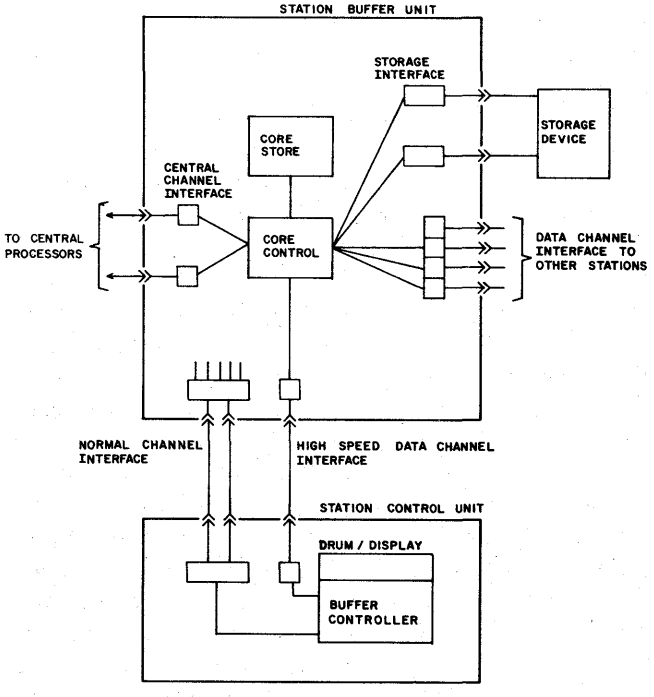


Figure 2—STAR station

interface of normal channel bits is provided between the SCU and the SBU (Figure 2).

The SBU consists of up to 64K bytes of memory organized in eight interleaved banks of 8K bytes each. Each bank has a memory cycle of 1.1 microseconds with a maximum bandwidth of 14 million bytes per second. Storage control logic provides for 12 independent channel accesses. The SBU is always associated with a controlling SCU. The general function of the SBU is to provide intermediate buffering of data, fan in/out from one STAR channel to many other station channels and working storage for the station. The interfaces to attached devices are contained in the SBU.

The following features of the SBU and its interfaces are important to its application and performance as a storage control mechanism.

- The high bandwidth allows simultaneous transfer of a number of storage devices into the SBU. The CDC 844 disk pack, for instance, has a transfer rate of approximately 1 million bytes per second compared with the SBU bandwidth of 14 million bytes per second.
- Device control operations such as connecting, addressing, and status are accomplished directly from the SCU over the buffer controller normal channel to the SBU device interfaces. This provides direct, detailed control of the devices.
- Actual data transfer between a storage device and SBU takes place automatically under control of the SBU device interface hardware. This frees the SCU during SBU data transfers.
- The SCU can directly access STAR storage via normal channel bits and the SBU interface. This mode is advantageous for message transfer and queue control.
- The SBU device interfaces are capable of stacking (queueing) functions and data transfer specifications. This allows maximum performance of the devices while relieving the SCU of having to intervene during brief, critical events such as crossing of intersector gaps.
- The SBU device interfaces have the capability of chaining SBU memory areas creating a contiguous data stream to a storage device from several SBU memory areas. This is used to automatically assemble and disassemble sync pattern and header information with the data block.
- All data is stored in fixed length blocks of 4096 bytes.

Storage station software

Tasks are communicated to the storage station via system messages. Each message selects a specific task and is handled by an SCU routine referred to as a task overlay. The task overlay contains the control code necessary to accomplish the task by calling various station subroutines and device drivers.

Associated with each device attached to a station is a device software driver in the SCU. This is a specialized routine which actually drives the devices through the SBU hardware interfaces. The other station routines communicate with the drivers through a driver parameter table and a driver-maintained status table. One status table exists for each device.

In addition to the device drivers other station subroutines are associated with station resource management and utility functions. Examples of these are:

- Rent buffer space in SCU
- Rent block in SBU
- Transfer SBU/SCU data
- Transfer CPU data
- Hash file name

Each station contains a standard program referred to as the nucleus or monitor. It contains a set of simple diagnostic routines known as quick-look diagnostics, a system autoload program, driver programs for the microdrum and for the keyboard associated with the character display, programs to manage the microdrum overlay mechanism, and the main control and organizational program.

The SCU microdrum holds a copy of all station software. The SCU operates under one of four different systems. These systems are allocated as follows:

1. Microdrum loader system
2. Run system (normal case)
3. Diagnostic system
4. Off-line system

The system is selected at start-up of SCU programs. The selection of a system causes linking of all routines associated with the system via scanner and overlay tables. When running, a given system contains the operating portion of the nucleus (the system selection and set-up routines are discarded to be called again from the microdrum for a new autoload) and specified routines fixed in core. The remaining routines are called when required from the microdrum. Calling a routine is accomplished through an overlay table which contains the core address of the called routine or the address of a routine which reads it into a core area available for temporary overlay and buffers. All routines associated with a system are thus directly accessible yet only the

most active routines reside dynamically in the SCU core memory.

The scanner is the idle loop of the nucleus. The primary purpose of the scanner is to map normal channel data signals to overlay programs based on priority and logical selection, thereby providing a low overhead mechanism for handling asynchronous external events. The external events (such as channel flags, microdrum busy, or input read signals) are presented to the scanner program via one or more normal channels. Associated with each channel are two logical selection words, the ENABLE mask, and the STATE mask. The channel data is exclusive or'ed with the state mask in order to select the appropriate signal polarities, and then matched against the enable mask. Any bits that are now set represent selected channel events in the desired state. These bits are scanned from left to right and the first bit found set is used to enter the overlay program associated with that bit. If all bits are zero, the scanner moves on to the next channel and repeats the procedure. One or more memory words are used to initiate internal events via the scanner. In this case, the memory words rather than the channels represent the raw input to the scanner. In a typical station, the scanner cycles through two normal channels and two memory words.

A detailed error handling and maintenance system is provided in the stations. Abnormal conditions in the operation of a device cause the device driver to exit to an associated error handling routine. This routine handles retries and error logging. It operates in conjunction with a device monitor routine which is used to set the parameters for a device, such as number of retries, turning device off to system, and breakpointing in the driver. A maintenance information system provides an English translation of the driver parameter tables and the device status tables on the SCU display and provides operator access to control the device operation via the device monitor.

Included in the maintenance system is the capability to run diagnostics and utilities associated with a device. These tests are controlled using the device driver, parameter table, and status tables and may be run in conjunction with the system operation on other station devices.

FILE SYSTEM

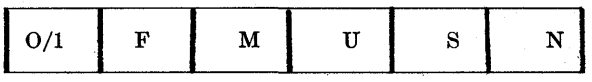
The file system described here exists totally within the storage station and is independent of any particular processor station, network configuration or storage device type. Creation, maintenance, recovery, access, security, storage layout, accountancy data, and performance statistics are all managed within the station.

The station file system is implemented as a set of task overlays. Each overlay is associated with a specific system message and provides the coordination necessary to accomplish the system task using the station device drivers and subroutines. Each message has a separate overlay to process it. If the message occurs frequently, the overlay remains in SCU core; otherwise, it is called in from the microdrum when it is needed.

Active file index

All the file messages are listed in the Storage Station Messages section. A file is simply a collection of stored bits, which has a descriptor and can be operated on by a set of file functions. No file function is processed until the file is first opened, and the last file function must always be a close function. In the open message, identification of a file is by file name (File Name Section). For other messages, identification of a particular file is by its active file index, the index of the file entry in the active file table (Figure 3). The file index is assigned by the storage station and returned to STAR in response to the open message. The advantage of this arrangement is that the majority of file messages use a 16-bit identifier rather than a variable length string of characters which could be quite long. By maintaining active-file information in core storage, access validation and transformation between logical (file page) and physical block locations is normally accomplished with negligible overhead and without introduction of superfluous input-output operations.

The size of each active file table entry is 8 characters (Figure 3). Initially, one SBU block of 4096 characters is devoted to the active file table, allowing 512 open files at any one time. This can be easily expanded if required. If the file is noncontiguous, read/write of file pages which are not in the first contiguous section require an access to the storage map in the file descriptor. One could trade the number of open files allowed for fewer open files with each entry containing the map of more than one file section.



O/1 = free/used flag
F = description pointer
M = access mode
U = unit number
S = starting address of file on device
N = number of blocks contiguous to S

Figure 3—Active file table entry format

File descriptor (catalog entry)

Each file has a descriptor which describes the file as seen by the system. The descriptor (Figure 4) consists of 8 sections: Header, characteristics, name, storage map, access map, activity map, and two free sections reserved for later use.

The set of descriptors for those files occupying a particular storage unit is itself part of a file and may be processed like any other file; it is called the descriptor file or catalog. This catalog may or may not be on the same storage media as the files it describes. Normally, removable media contain their own catalog files, but these may be copied elsewhere on mounting.

The size of an individual descriptor is variable in modules of 256 bytes up to a maximum of 4096 bytes. Initially, just one module (256 bytes) is used for each descriptor.

As an example the Control Data® 844 disk pack at present has the following layout.

Blocks 0, 1	Pack Label	Pack Catalog File
Blocks 2, 3	Free Storage Map	
Blocks 4 through 67	Descriptor Modules (1024)	
Blocks 68 through 23,027	Data Files	

To facilitate processing in the SCU, the descriptor proper is kept reasonably small, but the sections can have pointers to overflow areas and these may be of any length. The space allocated for the catalog is also variable. Initially 64 blocks of 4096 characters are used providing 1024 files per storage unit.

The allocation of a descriptor module to a newly created file is done either by the use of a free space map for the modules or by a hashing algorithm. To locate a file descriptor, the file name is hashed to locate a bucket in a hash table which contains entries of file names and pointers to their descriptors. This hash table is re-created (say at autoload) so that the system is not tied to any one hashing algorithm. The hash table may itself become quite long and is kept on the storage unit with the files or some associated storage device. An alternate implementation simply hashes directly to the descriptor module. If the file name does not match the name in that module, a search is made of the surrounding modules in that block. It is to be emphasized that normally the descriptor is only referenced on the open and close functions. All read/write file pages reference the active file table which is in SBU core.

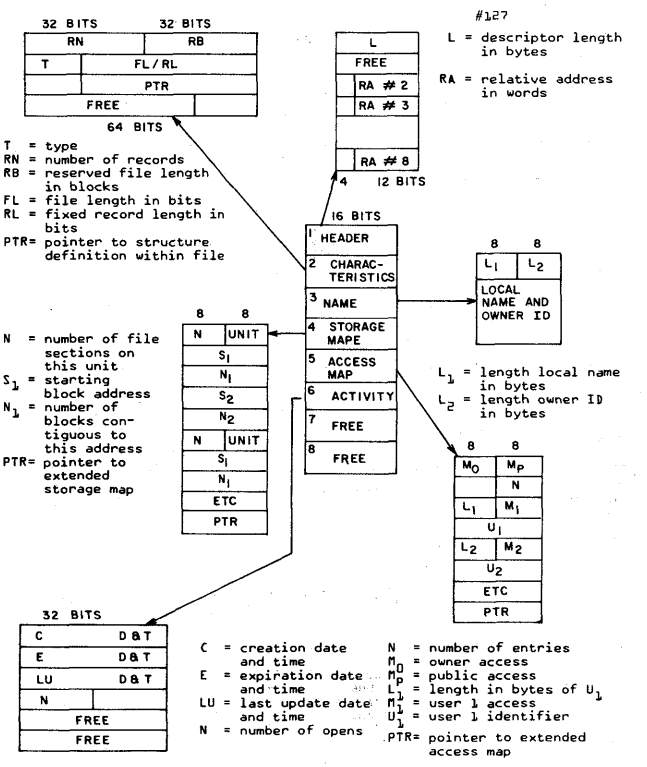


Figure 4—File descriptor format

Storage map section

The storage map (Figure 4) allows for a storage system to be divided into 256 units, each with a capacity of 65,536 blocks (2^{28} bytes: approximately 268 million). A variation on this scheme is being implemented which has 32-bit field lengths for block addresses and number of blocks contiguous to an address. This will cater for larger storage systems with capacity up to 2^{32} (approximately 4 billion) blocks or 2^{44} (16 trillion) bytes.

Characteristics section

The characteristic section of the descriptor is shown in Figure 4; the different file types are undefined (0), ASCII coded delimited (1), AS CII coded fixed (2), binary STAR (3), binary fixed (4), foreign delimited (5), foreign fixed (6), virtual memory (7), drop (8), labeled (9), multiple volume (10), incomplete (11), temporary/permanent (12), input (13), and output (14).

Types 1 through 6 categorize file types according to their internal coding. The exact definition is not important but it should be noted that types 3 through 6

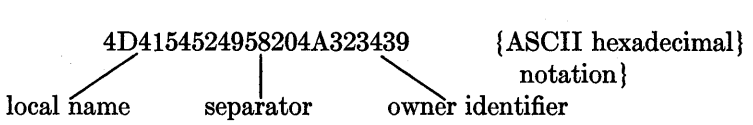
have an associated record map which describes the record structure of the file. A virtual memory file has a virtual address associated with each file page. The drop file is similar to the virtual memory file, it is a frozen image of an executing job which has been suspended for some reason together with the virtual address list and current program status information. The labeled file is one that has a label (somewhat similar to the file descriptor) within the file. These last three types use a pointer address to locate the relevant structural information within the file. The multiple volume/unit file is one that is spread over a number of storage units; yet, it is logically one file. An incomplete file is one upon which, although incomplete, processing begins; such is the case when processing begins after only a portion of tape is spooled onto a disk. No doubt other file types will be added, but these provide sufficient categorization for the present.

File name section

Perhaps the most important thing about a file is its name. It is that which identifies it uniquely and which must be used to open the file before it can be processed. The name consists of two parts, a local name followed by an owner identifier. Each part consists of a variable length string of characters (the ASCII alphanumeric set plus −$#). The parts are separated by the ASCII space character. Certain characters are reserved for special use within file names: *, /, ., &, |, and ?. The period character . for instance, is used to indicate some hierarchical structure within the name.

The file system is not normally concerned with the internal structure of either the local name or owner identifier, who gave this name or identifier, or where it came from. Essentially the name is used to locate the descriptor.

Example of File Name



Storage layout section

The storage layout of a file varies with the particular storage device but the goal in each case is the same, that is, to organize file storage in a manner which does not deter high-performance of expected access requests. A large block of data, stored as 128 consecutive physical blocks on a Control Data® 817 disk requires a little over a tenth of a second for transferring its half million bytes; stored differently, its transfer could take up to 10 seconds. The allocation and layout of a file are governed by a RENT/STORE routine which can be replaced or modified in order to implement more elaborate policies. This routine normally tries to allocate the desired number of blocks in a contiguous fashion; if this is not possible it will allocate the total space on as few large sections as possible.

The map of the disk file is a vector. Each element of the vector is a storage location and a number indicating how many blocks are contiguous to the location. As many contiguous sections as possible are represented in the descriptor proper and the rest are kept in an overflow area.

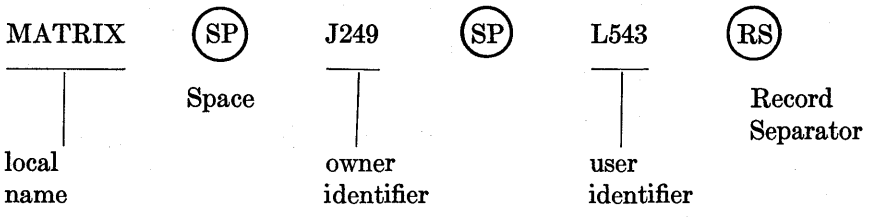
Access security section

Every time an OPEN operation is requested through a storage station, the access rights of the user are checked against the access map in the file descriptor. The open function has an owner identifier and a user identifier catenated to the local file name and terminates with the ASCII record separator code. If the user and the owner are the same person, the user identifier may be omitted. If the access is not permitted, an invalid access response is returned to the message sender. For the other file messages, validity of the operation is checked against the mode stored with the file entry in the active file table. Initial file access mode is one of four:

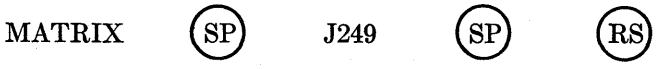
- Cannot delete
- Cannot alter access modes
- Cannot write
- Cannot read

The access modes for the different users are set or modified by access mode messages to the station. The default option on creation of a file is that the owner has open access and the public has no access. The file system again is not concerned with the internal structure of the user identifier; it is simply a variable length string of characters, and in fact, could be an agreed upon group name rather than an individual user identifier.

Example of File Identifier



If the user is the owner, then this can reduce to



Multiple stations

A typical STAR installation might include two STAR processors supported by a number of storage stations each having different storage devices attached. Such a system exists and is in experimental operation. It is possible for a user to specify on opening a file its location; if this is not done STAR sends messages to all storage stations listed in its directory. The station where the file exists opens the file and makes the appropriate response which STAR keeps till the file is closed. The other stations return a "not found" response.

At present on "create and open" the user must specify the storage station where the file is to be created but need not specify the device on that station unless he wished to do so. If a file of the same name already exists on the station it will be deleted if it is a "temporary" file and the new one will be created; otherwise, if it is a "permanent" file an "already exists" response will be returned to STAR. Files may be shared between different users and two STAR processors providing they are open for read only access. The station has no difficulty returning responses and data to the correct STAR processor since it is identified by its zip-code in the message header.

File system extensions

The basic file system can be extended to provide specific features. The basic file system and these extensions are expected to provide a very complete, standalone storage system.

- Automatic mounting—(packs cartridges, tapes, etc.)—Standard ASCII labels, automatic allocation of drives, and the mounting and dismounting with label validation.
- Multivolume files—Allowing a file to spread itself over a number of units.
- Archival file directory—One archival file directory for all files, on-line and off-line.
- Structured file name and owner/user identifiers—Structured names and identifiers linking files of a given class into a more complex access mechanism.
- Shared access security—Extended access mode conditions.
- File editions—Allow the user to specify file edition numbers or default to the latest edition.
- Accounting and performance statistics—Recording of station accounting and usage statistics.
- Experiment with distributing certain data management functions, which are now performed in STAR, to the stations.

STORAGE STATION MESSAGES

The following list gives messages which can be processed by storage stations. The underlined parameters are returned with the response.

Function	Parameters
	File Messages
Create and open file	*F*, M, M_o, M_p characteristics, name and user ID
Open File	F, M, *characteristics*, name and user ID
Close File	F, characteristics
Close and delete file (temporary and permanent)	F
Close and delete temporary file	F
Keep file	F
Set file characteristics	F, characteristics
Set file length	RB
Is file open	F, *characteristics*, name
Read file pages	F, N, S, B
Write file pages	F, N, S, B
Read file descriptor	F, B

Function	Parameters
Write access list entry	F, M, user access key
Modify owner and public access	F, M, user access key
Mount (tape, pack, cartridge) label L	L
	Test Messages
Read N blocks from storage unit	B, N, S
Write N blocks from storage unit	B, N, S
Read N blocks from storage unit with header	B, N, S, Header
Write N blocks from storage unit with header	B, N, S, Header
Storage unit status	N^1, N^2, N^3, N^4

Legend for Parameters

F = active file index (given by storage station)

M = access mode (used on open)
- bit 0 set means cannot delete
- bit 1 set means cannot alter access modes
- bit 2 set means cannot write
- bit 3 set means cannot read

M_o, M_p = access modes of owner and public, respectively (used on creation)

N = number of blocks/file pages to be transferred

S = starting file page number (starts with zero)

B = core block number, if bit 0 set B = SBU address

User ID = user access identifier, variable length string of characters which ends with the record separator character.

N_1 = total number of blocks
N_2 = number of disabled blocks
N_3 = number of active blocks
N_4 = number of free blocks

L = label on pack, cartridge, tape

RB = Length of file in blocks

Message header format

Preceding each set of message parameters is a header which has the following format.

RESPONSE CODE	MESSAGE LENGTH	PRIVATE USE OF SENDER	PRIVATE USE OF SENDER
PRIVATE USE OF SENDER	TO ZIPCODE	FROM ZIPCODE	MESSAGE FUNCTION CODE
16	16	16	16 BITS

Details of the message formats are not significant here, except to mention that it is valuable to limit the number of different formats used and to ensure field lengths are large enough to cater for future storage devices. The format is important, however, in respect that once it is established and used by a number of routines even small modifications to it can have widespread effects and are often time consuming and difficult to checkout.

CONCLUSIONS

The storage and file functions of a general-purpose computing system have been identified and separated to operate outside and in parallel with the central processor in a stand-alone, local or remote, storage station. This station forms part of an overall plan to distribute specific functions associated with general-purpose computing into separate computing elements or stations. The same basic hardware and software is used in all these stations to lower manufacturing costs by high volume production. The features and performance of this station have worked out well on delivered and in house systems using drums, large disks and disk packs for archival and working store on both large and small computers. The main reason for success has been the clear identification of the basic file and message functions required and a careful implementation of these functions, utilizing both hardware and software techniques on a standard STAR peripheral station. Although designed to meet the needs of the STAR-100 processing unit the storage station is well suited to be used with any processor which matches its channel and message protocol; it is also relatively independent of storage device type and system configuration.

ACKNOWLEDGMENTS

This work was performed in the Advanced Design Laboratory of Control Data Corporation in St. Paul, Minnesota. The head of this laboratory and chief designer of the CDC STAR-100 and STAR-1B Computer

Systems is J. E. Thornton. The success of the project is mainly due to his leadership and support, together with the hard work over a number of years of the following people in the Advanced Design Laboratory's peripheral group—N. G. Horning, W. C. Hohn, D. J. Humphrey, L. H. Schiebe, E. V. Urness, D. A. Van Hatten, C. L. Berkey, D. C. McCullough and R. A. Sandness.

REFERENCES

1 J E THORNTON
Design of a computer—The Control Data 6600
Scott Foresman 1970

2 T H ELROD
The CDC 7600 and Scope 76
Datamation April 1970 Vol 16 No 4 pp 80-85

3 J E THORNTON
System design and implementation
Proceedings of Third Australian Computer Conference 1966 pp 90-102

4 P D JONES C J PURCELL
Economics and resource parallelism in large scale computing systems
Proceedings of Fourth Australian Computer Conference 1969 pp 241-244

5 P D JONES N R LINCOLN J E THORNTON
Whither computer architecture
Proceedings of IFIP Congress 1971 pp TA4/162-TA4/167

6 W R GRAHAM
The parallel and pipeline computers
Datamation April 1970 Vol 16 No 4 pp 68-71

7 D J WHEELER
Assessing the complexity of computer systems
Proceedings of IFIP Congress 1971 pp I/164-I/168