A technique for individual preassembly and linkage of input/output program sections, which reduces overall assembly time, is described.

Also discussed are two techniques used in generating channel programs for direct-access devices. One of the techniques is designed for random addressing of records, the other for indexed sequential addressing.

The developmental work that led to these techniques was heavily influenced by the objective of effectively minimizing the amount of main storage required for the input/output control functions in DOS/360, *the disk operating system for* SYSTEM/360 *configurations with intermediate amounts of main storage.*

Internal data management techniques for DOS/360

by D. H. Ricour and V. Mei

DOS/360, a disk operating system for intermediate SYSTEM/360 configurations, provides a set of data management routines[1] that are collectively called the Logical Input/Output Control System (LIOCS). The LIOCS permits a file to be organized either for random access or for random and sequential access, the former being referred to as the Direct Access Method (DAM) and the latter as the Indexed Sequential Access Method (ISAM). The IOCS routines can be used in creating and updating data files, thus relieving users of a substantial programming effort.

The purpose of this paper is to describe the linkage between tables and modules used in LIOCS, and the techniques by which the LIOCS routines generate channel programs.[2] Some of the differences between DOS/360 and the other two disk-oriented operating systems for SYSTEM/360, BOS/360, and OS/360, are remarked upon in passing.[3]

table and module linkages

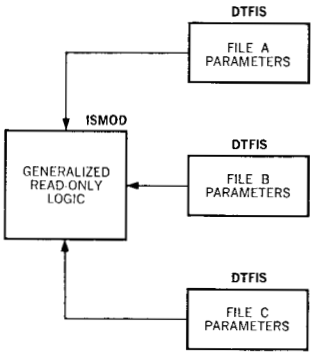
Declarative macroinstructions of the type used in defining a file (DTFDA for DAM files, DTFIS for ISAM files, etc.) permit the IOCS routines to generate tables of constants, called DTF (define the file) tables. Such tables (which are assembled internally but may also be assembled with the user's routine) specify record lengths, blocking factors, device identifications, processing modes, buffer addresses, and the like. The subroutines that actually perform deblocking, error recovery, buffer management, and I/O operations are called LIOCS modules.

Because these modules are ordinary subroutines and store no file parameters internally, several files of similar characteristics

can share a single module, as shown in Figure 1. However, the same module cannot be used asynchronously for several files, since it performs temporary register-save operations within the module, rather than within the file tables.

The user has the option of assembling logic modules by declarative macroinstructions within his problem program. Usually, however, a DOS/360 installation generates a collection of LIOCS modules while building a system-residence volume. These modules are normally retained on-line to facilitate access and maintenance.

Figure 1 Several files share a single logic module



Assembly time can be reduced by individually preassembling sections of the program. This makes it unnecessary to reassemble the entire program each time one or more of the sections are changed. Instead, only the changed sections need reassembly and are then linked with the other preassembled program portions.

Each generated DTFDA or DTFIS table contains the name of one LIOCS module, which is automatically generated from the macroinstruction DAMOD or ISMOD, respectively. Each module name is composed of a prefix, for example MOD, and a convenient number of letters. Thus, the complete module name could be MODxxx, in which the last three letters can be substituted and define a great variety of subfunctions. For instance, the first letter following the prefix could refer to the format of a record to be processed. Thus, MODFxx might indicate a module only for fixed-length records, and MODBxx a longer module processing both fixed-length as well as undefined-length records. Consequently, MODFxx is a subset of MODBxx.

If two DTF tables, DTFA and DTFB, are separately assembled, DTFA can—for example—call for a MODFxx module name, and DTFBxx for a MODBxx module name. When these two DTF tables are link-edited into executable form, the Linkage Editor considers the module names as external symbols in the usual sense: for each external symbol, the Linkage Editor searches in the relocatable library for the associated module, places it into main storage, and replaces the module name in the DTF table with the appropriate address of the module in main storage.

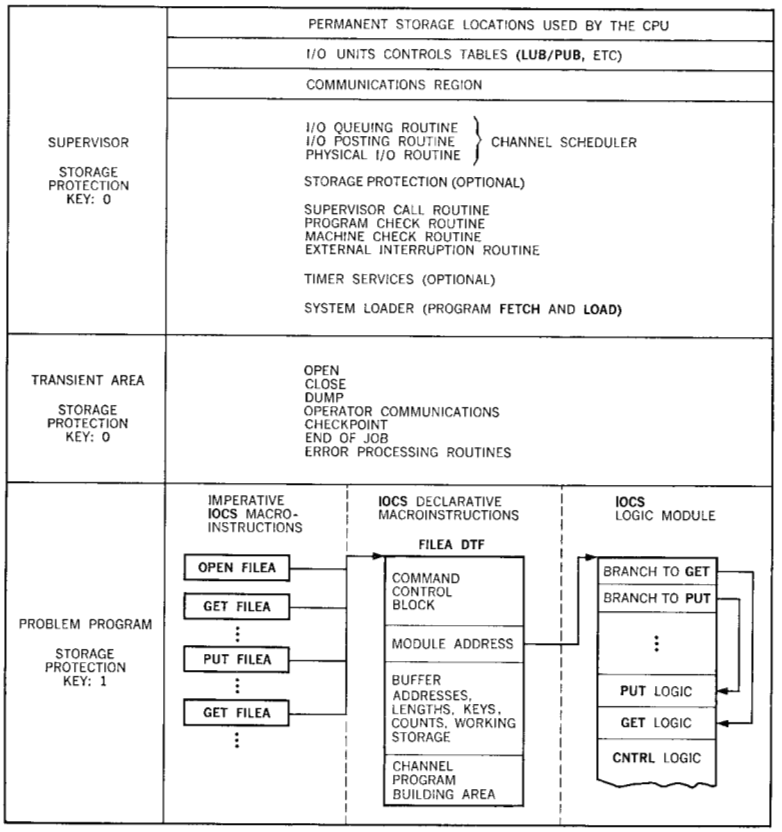
Since the Linkage Editor processes the external symbols in alphabetical sequence, the MODBxx name is resolved first. The associated MODBxx module in main storage contains an entry point for its subset MODFxx. Thus, another module fetch operation is eliminated, resulting in considerable main-storage savings.

In his application programs, each user generates machine-language linkages for OPEN, GET/PUT, READ/WRITE, and other IOCS imperative macroinstructions, as suggested by the schematic in Figure 2. In an assembler-language program, these must be coded explicitly; for COBOL, FORTRAN, PL/I, and RPG programs, the necessary linkages to file tables and IOCS modules can be system-generated.

I/O operations

SYSTEM/360 I/O operations are normally performed by channels and control units operating under control of a supervisor program. Each channel is an autonomous computing unit, capable of

Figure 2 Main-storage organization



asynchronously executing one or more I/O commands to the central processor. For most card and tape operations, at most two channel command words (CCW's) are required to specify the main-storage address, record length, control operations, etc. However, a direct-access-device operation requires significantly more CCW's, and the requisite channel program may consist of a string of four to about twenty-five CCW's. The operations start with a search for the cylinder, followed by a search for individual records on a track, based on key or count fields juxtaposed to these records. Often it is useful to chain two or more channel programs together, thus permitting one search operation to be followed automatically by several READ/WRITE operations—or even by another search.

program performance criteria

Different performance criteria are stressed by different systems, and each of the disk operating systems for SYSTEM/360 furnishes a selection of channel programs appropriate to its criteria. To conserve main storage, a DOS/360 channel program retains only a single key and a corresponding file-storage address in storage at a single moment. Moreover, DOS/360 allocates one

channel program per file because most intermediate system applications process a single transaction at a time.[4]

To conserve the amount of time required in servicing I/O interrupts, channel programs can be written to permit the channel and the device to operate completely autonomously (without attention from the central processor) until the pertinent record is retrieved or written. In general, DOS/360 management avoids this tactic. Such channel programs not only require more main storage than simple-retrieval channel programs, but they are also more difficult to generate. Moreover, SYSTEM/360 channels necessarily claim some main-storage cycles in performing comparisons on counts and keys and in fetching CCW's; this loss of storage cycles is called *channel interference*. Direct-access-device channel programs produce a significant amount of interference on medium-performance configurations, such as the SYSTEM/360 Model 30. Even on high-performance central processors, the interference is not negligible and can reduce processor throughput for fast direct-access devices such as megacycle drums.

Each CCW has the format shown in Figure 3, and two or more CCW's are normally executed in sequence. A Transfer In Channel (TIC) CCW permits unconditional branching, and a No Operation (NOP) CCW is functionally redundant. TIC's and NOP's can be helpful in the manipulation of a single generalized program; with them, buffer addresses, key addresses, lengths, and file-index addresses can be substituted as needed. As each READ/WRITE operation is requested by the application program, TIC/NOP commands can be used to "gate" a particular sequence of CCW's. At the conclusion of this operation, the next operation can be accepted with its particular gating of TIC's and NOP's as shown in Table 1. Maximum use of this technique has been made in the OS/360 Data Control Block, which is the equivalent of DTF in DOS/360.

While one or more TIC or NOP commands are being sequentially interpreted by a channel, the channel claims main-storage cycles at such a high rate that little processing can proceed concurrently in other channels or in the central processor. Time-critical operations may be required on these other channels at the same instant that a sequence of TIC's and NOP's is being interpreted by the channel attached to the direct-access device. In such cases, there may be a non-zero probability of *overrun*—a loss of data on another channel during the transfer to or from the given device. Although the chance of overrun is normally a negligible factor in the design of channel programs for high-performance systems, it cannot be dismissed in the case of medium-performance processing units. Hence, BOS/360 and DOS/360 channel programs are invariably

Figure 3 Format of a CCW

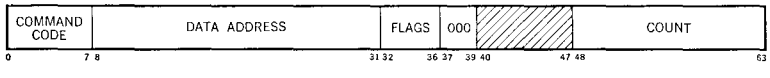


Table 1 Generating a tailored channel program

Generalized Channel Program			*Tailored Channel Program*		
	SEEK	CYLINDER1		SEEK	CYLINDER1
	SEARCH	KEY1		SEARCH	KEY1
	TIC	*–8		TIC	*–8
	READ	DATA1		READ	DATA1
GATE1	TIC/NOP		GATE1	TIC	GATE2+8
	SEARCH	KEY2		SEARCH	KEY2
	TIC	*–8		TIC	*–8
	READ	DATA2		READ	DATA2
				(END OF CHAIN)	
GATE2	TIC/NOP		GATE2	TIC/NOP	
	SEARCH	KEY3		SEARCH	KEY3
	TIC	*–8		TIC	*–8
	WRITE	DATA3		WRITE	DATA3
GATE3	TIC/NOP		GATE3	TIC	GATE1+8

"net," i.e., they avoid NOP commands and extraneous TIC commands. Net channel programs are particularly appropriate to Models 30 and 40 of SYSTEM/360, where the exposure to overrun is significantly greater than on the Models 50, 65, and up.

BOS/360 generates net channel programs at assembly time. Thus, file processing requires little manipulation of the channel programs other than substituting buffer addresses and keys as requested. However, an application needing more channel programs has a correspondingly larger main-storage requirement. On systems with small main storage, this requirement may force division of the application into several overlays, increasing both the programming task and the running time of the finished program.

DOS/360 generates net channel programs from the following elements: (1) a small number of generalized CCW's, (2) main-storage addresses of buffers and work areas, (3) file storage addresses of indices and record addresses, and (4) *channel-program descriptors* (table entries) interpreted by a *channel-program generator* which is a routine that starts with parameters and fashions a suitable channel program.

DOS/360 enqueues one I/O request at a time for each DTF table; therefore, storage sufficient for the largest channel program is required in a work area of each DTF table. This area is called BPA (Building Program Area). All imperative macroinstructions for the same file share this BPA so that each channel program overlays its predecessor. Whenever a record must be retrieved, the channel-program generator in the logic module is accessed one or more times during the READ/WRITE operation.

In BOS/360 and DOS/360, the problem program can specify up to six different READ/WRITE imperative macroinstrunctions per

DAM channel programs

file. Each of these instructions can perform a basic function (e.g., retrieve a record by key and update it) and several additional subfunctions (e.g., returning the address of the next record or verifying the updating). Thus, there are several possible channel programs for each imperative macroinstruction and a total of more than fifty for the six possible imperative macroinstructions, requiring the use of sixty different channel control words. With BOS/360, all these channel programs are cataloged in the macro library; for each imperative macroinstruction, the correct channel program is selected at assembly time by the DTF macroinstruction statements and then entered in the DTF table. Since only one of these six channel programs in the DTS table is used at any given time, DOS/360 uses only one channel program at any time, generating it from tables at I/O time.

The sixty different CCW's required for the six READ/WRITE macroinstructions of DAM can be readily generated from eleven basic CCW's, changing only the command bytes and/or flag bits. Of these eleven CCW's, five are required for initial loading of the file; six are required for normal file maintenance processing. TIC CCW's are generated directly from storage addresses.

To each desired CCW corresponds an eight-bit byte as follows: Field a_1 (1 bit) determines the command byte, Field a_2 (4 bits) selects one of the eleven basic CCW's, Field a_3 (3 bits) further defines the command byte and inverts the flag bits of the basic CCW. If both a_1 and a_3 are zero, the desired CCW is a TIC. In this case, a_2 determines the required relative address, $*\pm 8n$. For DTFDA, channel-program descriptors (which are strings of single bytes in this case) are generated into each table at program-assembly time, depending upon which imperative macroinstructions are used to access the file.

Since the first CCW in each DOS/360 channel program must be a Seek command (to support the file-protect feature and to permit asynchronous access to several files on the same device), the Seek CCW is assembled at the top of the BPA and never modified. As each channel program is requested, the channel-program generator moves an appropriate CCW into the BPA, then performs certain logical operations to alter the command and flag bits. The same procedure is repeated with each successive CCW until the generator detects the absence of command chaining and stops the generating process. The generator is about 100 bytes long and requires two milliseconds to build an average channel program in a SYSTEM/360 Model 30. The generating operation, which is performed during the cylinder access time, slightly increases the load on the computer, but does not affect the average access time to a track; this is very important for a random-access file.

An example demonstrates the DAM channel program generator on the following task: read a certain keyed record on a cylinder specified by SEEKADR into IOAREA1 (according to the key at KEYARG) and return the corresponding track-record identifier

Table 2 Typical channel program

Command Byte	*Operands (per CCW format)*	*Comments*
SEEK	SEEKADR+1, CC, 6	First byte of SEEKADR is the volume identifier
SRCHHA	SEEKADR+3, CC, 4	Locate the home address
TIC	*–8	
RDID(M/T)	IDLOC, CC, 5	Read CCHHR for this record
SRCHKE(M/T)	KEYARG, CC, key-length	Search for the desired key
TIC	*–16	
RDDATA	IOAREA1,, data-length	Read its data field

Table 3 Basic CCW's

a_2	*Command*	*Operands*
0000	SRCHIDE	SEEKADR+3, CC, 5
0001	SRCHKE	KEYARG, CC, key-length
0010	RDDATA	IOAREA, CC, data-length
0011	RDID	IDLOC, CC, 5
0100	SRCHHA	SEEKADR+3, CC, R
0101	RDKD	IOAREA1, CC, (key+data)-length

(CCHHR) to the location specified by IDLOC. This technique permits the user to update the record using its physical address, rather than requiring a second scan of the cylinder. The search operation is multitrack (M/T); a second operand of "CC" denotes "command chaining." The desired channel program is shown in Table 2. The corresponding channel-program descriptor, descriptor interpretation rules, and basic CCW's are as follows:

Descriptor. X'A7189A881014', one pair of characters per CCW. Thus, X'A7' corresponds to the search-home-address (SRCHHA) CCW. (Recall that the SEEK CCW is never explicitly built.)

Descriptor interpretation rules

1. If $a_1 = a_3 = 0$, generate a TIC with address field $*+8(a_2-4)$.
2. If $a_1 = 0$ and $a_3 \neq 0$, invert the following flag bits of the a_2th basic CCW, as the latter is moved into the generated channel program:

a_3 = 1xx	Reset the Command Chain bit
a_3 = x1x	Set the Suppress Length Indication bit
a_3 = xx1	Set the Skip bit

3. If $a_1 = 1$ and $a_3 =$ xx0, turn on the Multiple-Track bit of the command byte of the a_2th basic CCW, as the latter is moved into the generated channel program.
4. If $a_1 = 1$ and $a_3 =$ xx1, change a Read command byte to Write, as the a_2th basic CCW is moved into the generated channel program.
5. If $a_1 = 1$ and $a_3 = 111$, move the a_2th basic CCW without change into the generated channel program. Basic CCW's are given in Table 3.

The first two CCW's are generated as follows:

SRCHHA CCW. Since $a_1=1$, $a_2=0100$, and $a_3=111$, X'A7' means "move the fourth basic CCW without change" (Rule 5).

TIC *−8. Since $a_1=0$, $a_2=0011$, and $a_3=000$, X'18' means "generate a TIC instruction whose relative address is 8 (3–4)" (Rule 1).

DAM storage requirements

Consider two different DTFDA's, one using four imperative macroinstructions or about 30 CCW's (average), and one using the six possible imperative macroinstructions or about 60 CCW's (maximum). We now compare two of the three techniques for direct-access programming: the Assembled Channel Program Technique using BOS/360, and the Generated Channel Program Technique using DOS/360. For applications requiring only one average DTFDA table, the two approaches are equivalent:

Assembled Channel Program Technique:	
DTF table T_{A1}	240 bytes
30 CCW's	
Generated Channel Program Technique:	
DTF table T_{G1}	140 bytes
6 basic CCW's, 48 bytes	
BPA of 8 CCW's, 64 bytes	
4 information strings, 28 bytes	
Channel program generator	100 bytes
	240 bytes

If three different DTFDA tables of average size are used in the same application, approximately 200 bytes are saved:

Assembled Channel Program Technique:	
DTF tables	
3 T_{A1}	720 bytes
Generated Channel Program Technique:	
DTF tables	
3 T_{G1}	420 bytes
Channel program generator	100 bytes
	520 bytes

For an application using maximum DTFDA tables, considerable storage savings result, particularly if the application uses several files simultaneously:

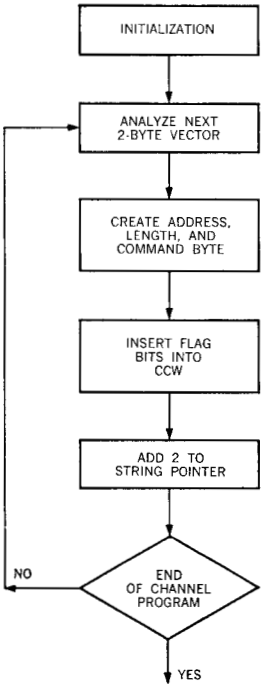
Assembled Channel Program Technique:	
DTF tables	
3 T_{A2} (60 ccw's each)	1,440 bytes
Generated Channel Program Technique:	
DTF tables	
3 T_{G2} (240 bytes each)	720 bytes
Channel program generator	100 bytes
	820 bytes

ISAM channel programs

The Indexed Sequential File Management System for DOS/360 permits processing of disk records in any order by control information stored in a set of indices. The imperative macroinstructions—GET, PUT, READ, WRITE, and WAITF—allow the user to load the file and also to add, read, and update individual records randomly, and retrieve them sequentially.

The generation of channel programs into DTFIS tables is significantly different from that for DTFDA tables. There are more imperative macroinstructions and distinct channel programs than in DTFDA, several of which are used to perform each GET/PUT or READ/WRITE function. If all the channel programs were entered into the DTF table at assembly time, this would amount to more than 30 channel programs requiring up to 15 CCW's each, aggregating 2000 bytes, although not all channel programs are required for each application. However, DOS/360 uses a channel-program generator with DTFIS. This generator is more generalized, slower, and somewhat larger than the generator for DTFDA; however, it has more flexibility and may reduce the size of the DTF tables because the basic CCW list is not required.

Figure 4 Flowchart of channel program generator for ISAM

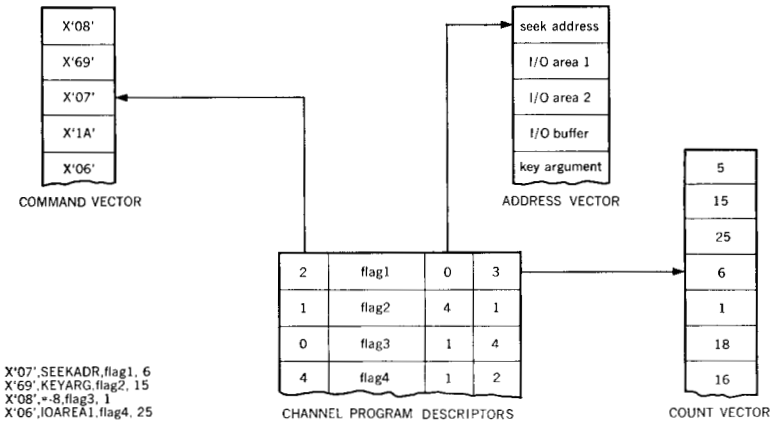


To each desired CCW corresponds a 2-byte string: the first of four 4-bit fields selects the command byte, the second the flag field, the third the address field, and the fourth the count field. At program assembly time, the storage addresses for buffers, keys, etc. are ordered into one vector of 4-byte entries, the relevant count fields are ordered into another vector of 2-byte entries, and the relevant command bytes are ordered into a third vector of 1-byte entries. Channel programs are generated from elementary fields in DTFIS, rather than from selected preassembled CCW's, as shown in the general flowchart for the channel-program generator of Figure 4.

Figure 5 shows the generation of a typical channel program using the command-byte vector, address vector, count vector, and channel-program descriptor string.

The internal data management techniques for DOS/360 aim at economizing the use of main storage. Overall assembly time

Figure 5 Typical solution for ISAM



summary comment

is reduced by individual preassembly of program sections, which makes it unnecessary to reassemble the entire program if some portions are changed. Only the changed sections are reassembled and then linked with the unchanged ones.

Different techniques for generating channel programs are appropriate to the direct-access and indexed-sequential file organizations in DOS/360. One technique selects preassembled CCW's and modifies selected fields, the other generates CCW's entirely from their constituent fields. Both of these techniques require more processing time per I/O request than required by BOS/360 or OS/360, although much of this time can be overlapped with seek time for random I/O requests.

BOS/360 preassembles channel programs that contain no extraneous TIC or NOP CCW's. This minimizes both the processing time per I/O request and the probability of overrun, but would require more main storage if several files in the same program were used. OS/360 preassembles generalized channel programs, which are gated by TIC/NOP commands as appropriate to each I/O request. DOS/360, however, generates net channel programs as needed. This method conserves main storage, a particularly important consideration for applications requiring several files on direct-access storage; the more files in use, the more main storage is saved.

ACKNOWLEDGMENT

The authors are greatly indebted to Dr. D. N. Freeman for his helpful suggestions during the preparation of the paper.

FOOTNOTES

1. The general data management concepts for DOS/360 (and TOS/360, a similar tape operating system) are outlined by A. R. Cenfetelli in "Data management concepts for DOS/360 and TOS/360," *IBM Systems Journal* **6**, No. 1, 22–37 (1967).

2. A similar technique is being developed for the DOS/360 Basic Teleprocessing Access Method (BTAM).
3. The operating systems for IBM SYSTEM/360 configurations are the Basic Operating System BOS/360 for small configurations with at least 8K bytes of main storage, the Disk and Tape Operating Systems DOS/360 and TOS/360 for intermediate configurations with at least 16K bytes of main storage, and the Operating System OS/360 for intermediate and large configurations with at least 32K bytes of main storage.
4. An alternative method for conserving main storage allocates a block of main storage, possibly several thousand bytes, to a partial or complete index for the file, substantially increasing retrieval speeds. It is also permissible to allocate several channel programs to a file and to process the file by one or more of the following queuing disciplines: (1) First-In First Out, which reduces interrupt-servicing overhead; (2) nearest cylinder or cylinder sweep, which reduces the average seek time per transaction; and (3) priority class, which permits servicing of urgent requests out of sequence.