AN INTEGRATED SYSTEM FOR DOCUMENTATION MAINTENANCE, PROGRAM AND JOB SELECTION, AND MEMO PROCESSING

by

John A. Schick

Wm C. Brown Company, Publishers
Dubuque, Iowa

INTRODUCTION

One important requirement of any application system is the ability of the end user to conveniently obtain access to the various application programs. This is especially true when the end user lacks sophistication in the use of computer systems. User frustration and the possible loss of data integrity are the likely results of a system that requires novice users to directly run programs and stream jobs with cryptic file names qualified by group and account. One rudimentary solution is to use mnemonic names for program and job files which reside in the logon groups and accounts of the various users. This solution is less than completely satisfactory for several reasons:

i) Although mnemonic file names are by definition fairly easy to remember, users must still remember exact names.

ii) The eight character file name limit imposed by the operating system makes a consistent system of mnemonic names difficult to devise.

iii) To eliminate the need for group and account qualification, multiple copies of program and job files must be maintained if they are to be used from more than one group or account.

iv) Mnemonic names do not readily lend themselves to the classification of programs by system, the cross-referencing of data files to programs, and other similar aspects of program documentation important to the system maintenance function.

The Wm. C. Brown Company used mnemonic file names for about one year until the volume of systems and programs made the drawbacks of this technique evident. This paper describes the system developed to improve the friendliness of the program and job selection process and at the same time allow a systematic naming convention for all program, job, and data files. This system consists of a data base-driven process handling program (SELECTOR) which runs application programs and streams jobs for the end user, plus a set of documentation maintenance and reporting programs which maintain and document the data base used by SELECTOR as well as the actual source, USL, program, and documentation text files. Also discussed are two spinoffs of the original system: a job streaming utility (STREAMER) and a memo processing subsystem. Figure 1 shows a generalized system flowchart of the entire system.

The Program SELECTOR

SELECTOR is a process handling program written in recognition of the need for an interface between the end user and MPE. Through SELECTOR, users can run programs and stream jobs either by entering mnemonic names or by "X"-ing descriptions displayed according to a pre-defined network of menus. SELECTOR is the only program that need be run by most of our operators. Indeed, since UDC's became available they are prevented from running any other programs and do not even need to explicitly run SELECTOR because of the following UDC in effect for most users:

```
SELECTOR
OPTION LOGON,NOBREAK
CONTINUE
RUN SELECTOR.PUB.SYS
BYE
```

In several respects, SELECTOR simulates MPE. The program begins in character mode by displaying a double colon prompt. In response to this prompt, the user may enter any of the following:

i) any of a limited number of "safe" MPE commands which are carried out by means of the COMMAND intrinsic

ii) the mnemonic name (up to twenty characters) of a main program to be run or a job to be streamed

iii) the special mnemonic "HELP" which causes SELECTOR to enter block mode where menus of system and program descriptions are displayed for the user to choose from in "X-marks-the-spot" fashion

iv) the special mnemonics "BYE", "END", or "EXIT", all of which terminate SELECTOR.

SELECTOR employs a documentation data base named BASE03 to make the correspondence between the operator's input and the actual main program or job to be invoked. Figure 2 presents the schema layout of the most important data sets of BASE03. Each main program, subprogram, and job on our system is identified by an entry in the PROGRAM data set. The twenty character field MNEMONIC-NAME provides the means of associating mnemonic program names with actual internal file names. When an operator of SELECTOR enters a mnemonic name, the PROGRAM set entry containing that mnemonic name is retrieved and the corresponding file name is constructed from the fields PROG-ID, GROUP-NAME, and ACCOUNT-NAME. If the PROG-TYPE field contains "MAIN" (indicating a main program), the constructed file name is created and activated. If the PROG-TYPE field contains "JOB" (indicating a job file), the constructed file name is streamed.

In "HELP" mode, SELECTOR uses the contents of the data sets SELECTORS and OPTIONS to determine the nature of the menu network. Each menu is defined by an entry in the SELECTORS set containing a forty character SELECTOR-TITLE field to be displayed at the top of the menu and a two character SELECTOR-CODE. The individual options within each menu are

defined by means of OPTIONS set entries containing the SELECTOR-CODE of the menu on which they are to appear. An option that leads to another menu contains the SELECTOR-CODE of the desired next menu in its NEXT-SELECTOR field. An option that runs a program or streams a job when it is selected is linked to the corresponding PROGRAM set entry via its PROG-ID field. A control set in BASE03 contains the SELECTOR-CODE of the first menu to be displayed whenever "HELP" mode is entered, as well as the name of the VIEW forms file containing the form on which the menus are to be displayed. Figure 3 shows the first menu that is displayed at our installation when SELECTOR is run in "HELP" mode.

The Documentation Maintenance Subsystem

The method used to name files can provide important documentation information to programmers and operations personnel. Since SELECTOR has eliminated any need to name files mnemonically, we have adopted a naming convention at the Wm. C. Brown Company that communicates the following information about each file:

i) the file type

ii) except for data bases, the system to which the file belongs

iii) in the case of data files, the program or job which makes principal use of the file.

Data bases, which frequently cross system boundaries, are identified by the four character code "BASE" followed by a two digit sequence number assigned sequentially from 01. Data bases designed before the adoption of this convention (such as the invoicing data base INVCS) still stubbornly retain their old names but all new data bases adhere to this standard.

All other files are identified by a four character system identification code followed by a one character file type code and a three digit sequence number. The following is a list of the various file type codes currently in use:

C = code of all kinds (source, USL, program, and documentation files)
D = data files not included in any of the other categories
F = forms files
J = job files
K = KSAM key files
L = KSAM data files
P = procedure documentation files
R = report files
S = system documentation files
T = miscellaneous text files
W = work files
Y = copylib files.

All the files associated with the installation of a program and the production of program documentation are assigned the same name within separate groups of a standard account (WCBLIB) via the type code C. For example all source files are found in the group SOURCE.WCBLIB and all text files containing operating instructions are contained in the group INSTRUCT.WCBLIB. When source, program, and USL files must all reside in the same group (e.g. in the programmer's logon group during program development), type codes U for USL files and O for program (object) files are used. Sequence numbers for the programs within each system are normally assigned from 010 in multiples of ten.

Data and job files are numbered to cross-reference them to programs. Data files are assigned the sequence number of the first (and frequently only) program that uses the file. Jobs are numbered according to the first program run in that job. The rare job that contains no run commands is assigned a sequence number in its own right.

This naming convention has served well as a program classification and cross-referencing aid. It also has allowed for a systematic process of program installation and documentation production.

As mentioned earlier, documentation on each main program, subprogram, and job is maintained in the data base BASE03. In addition to the header information in the PROGRAM set, information on data files and copylibs is maintained in the sets DATA-FILE and COPYLIB respectively. The sets PROG-PROG-XREF, PROG-FILE-XREF, and PROG-COPY-XREF contain the program, data file, and copylib cross-reference data that their names imply.

Besides the fixed-format information in BASE03, editor files are used to maintain the free-format aspects of program documentation. These files contain general program descriptions, revision histories, and operating instructions which reside in the DOCTEXT, FIXLIST, and INSTRUCT groups of the WCBLIB account.

A documentation maintenance program (DOCUC060) using VIEW, handles most of the aspects of maintaining BASE03, maintaining the program, USL, source, and text files in WCBLIB, and printing program documentation. As can be seen in figure 4, DOCUC060 offers eleven separate options:

i) Install a program into WCBLIB (optionally FCOPYing the source from any specified file) without also compiling all called or calling programs.

ii) Install a program including the compilation of all called or calling programs.

iii) Print program documentation on any specified device class or device number.

iv) Change a program's header information (in the PROGRAM data set).

v) Change a program's file usage information (in the DATA-FILE and PROG-FILE-XREF sets).

vi) Change a program's list of called programs (in the PROG-PROG-XREF set).

vii) Change a COBOL program's copylib usage information (in the COPYLIB and PROG-COPY-XREF sets).

viii) Add header, file usage, program usage, and copylib usage data for a new program.

ix) Delete all information for an obsolete program.

x) Change the name of an existing program.

xi) Invoke EDITOR to maintain the documentation text files in the DOCTEXT, FIXLIST, and INSTRUCT groups.

Figures 5 through 8 show the screens that are displayed during the process of adding documentation for a new program via DOCUC060. Figure 9 is a sample first page from program documentation produced by DOCUC060.

DOCUC060 does not maintain the SELECTORS and OPTIONS data sets used by SELECTOR. A separate program to maintain these sets is on the drawing board. They are currently maintained through QUERY.

The Job Streaming Utility STREAMER

It should come as no great revelation that many of the functions of an application system are best performed in batch jobs. For the sake of operator convenience, the commands for all such jobs should reside in job files requiring no maintenance by the user. The MPE stream facility, however, has several fundamental limitations that precludes the use of standard job files. !JOB statements must be hard-coded with user and account names and passwords which represents a potential accounting and security problem and a tedious maintenance problem whenever passwords are changed. Also, the stream facility makes no provision for the dynamic entry of variable data into job streams at stream time. To overcome these limitations, a job streaming utility named STREAMER has been written to stream all jobs at our installation.

STREAMER reads a file that has been equated to the formal file designator JOBFILE ($STDIN by default), inserts the user's logon name, group, and/or account if any of these parameters are missing in the !JOB statement, inserts passwords if they are omitted, inserts data solicited from the operator in response to input commands in the job file, and then streams the job using the COMMAND intrinsic. STREAMER runs in privileged mode to extract passwords directly from the system directory. For the sake of system security, SELECTOR imposes several restraints to prevent or at least monitor one user streaming a job that names another user in its !JOB statement:

i) System manager jobs can only be streamed by users with system manager capability unless the job file contains the correct passwords.

ii) Account manager jobs can only be streamed by users with system manager capability or account manager capability in the same account unless the job file contains the correct passwords.

iii) System supervisor jobs can only be streamed by users with system manager or system supervisor capability unless the job file contains the correct passwords.

iv) A console message is produced and a comment line is inserted into the job stream identifying the operator if the operator's logon id does not match the job's user and account names.

To allow for entry of variable data into job streams, STREAMER provides for two pseudo-MPE commands !INPUT and !INPUTS. When STREAMER encounters an !INPUT command, it displays the rest of the command on $STDLIST as a prompt for the operator, reads the operator's reply from $STDIN and inserts the reply into the job stream in place of the !INPUT command itself. The !INPUTS command is processed the same way except that the prompt is repeated and replies inserted into the job stream until the operator gives a blank response.

STREAMER was designed to resemble as closely as possible the MPE STREAM command. It is most conveniently run by means of the following UDC:

```
STREAMER JOBFILE=$STDIN,REPLACE=0
FILE JOBFILE=!JOBFILE
RUN STREAMER.PUB.SYS;PARM=!REPLACE
RESET JOBFILE
```

Using this UDC, STREAMER is invoked exactly like the STREAM command with the one exception that the colon replacement character must be expressed numerically since it is communicated to STREAMER via the PARM= parameter of the RUN command.

STREAMER is used both stand-alone and from programs to stream jobs throughout our system. An SPL utility subprogram has been written to issue the file equation for JOBFILE and to create and activate STREAMER so that STREAMER can be invoked from application programs. It was decided to use an SPL interface routine to process-handle STREAMER rather than write a subprogram version of STREAMER so that calling programs would not need privileged mode capability but just process handling capability instead.

The Memo Processing Subsystem

An unanticipated spinoff of SELECTOR has been the development of a memo susbsystem to handle the processing of memos throughout the company. SELECTOR provides a common link between users by prompting the operator when he or she has been sent a memo from some other user. The memo subsystem consists of five separate programs:

i) a memo entry program that operates in unformatted block mode to accept memos from the user and append them to a transaction file

ii) an update job that runs constantly in background, waking up every five minutes to process the transaction file

iii) a character mode memo retrieval program

iv) a memo print program that is run periodically to print memos for those without terminals, memos that are a day old, and memos that have been flagged for hard copies by senders or receivers

v) a memo entry subprogram that allows application programs to send memos.

The update job places the fixed-format portion of each memo (sender name(s), receiver name(s), carbon receiver name(s), subject, date, and time) in a memo data base named BASE09. The variable-sized memo text is stored in its own ordinary disc file in a standard group and account (DATA.MEMO). The update job also puts the logon name of each receiver that has a terminal into the MEMOS data set of BASE03 as a signal for SELECTOR to prompt that user to run the memo retrieval program. A personnel data base (BASE06) contains the information necessary to make the correspondence between a user's real name as entered on memos and that user's logon id as returned by the WHO intrinsic.

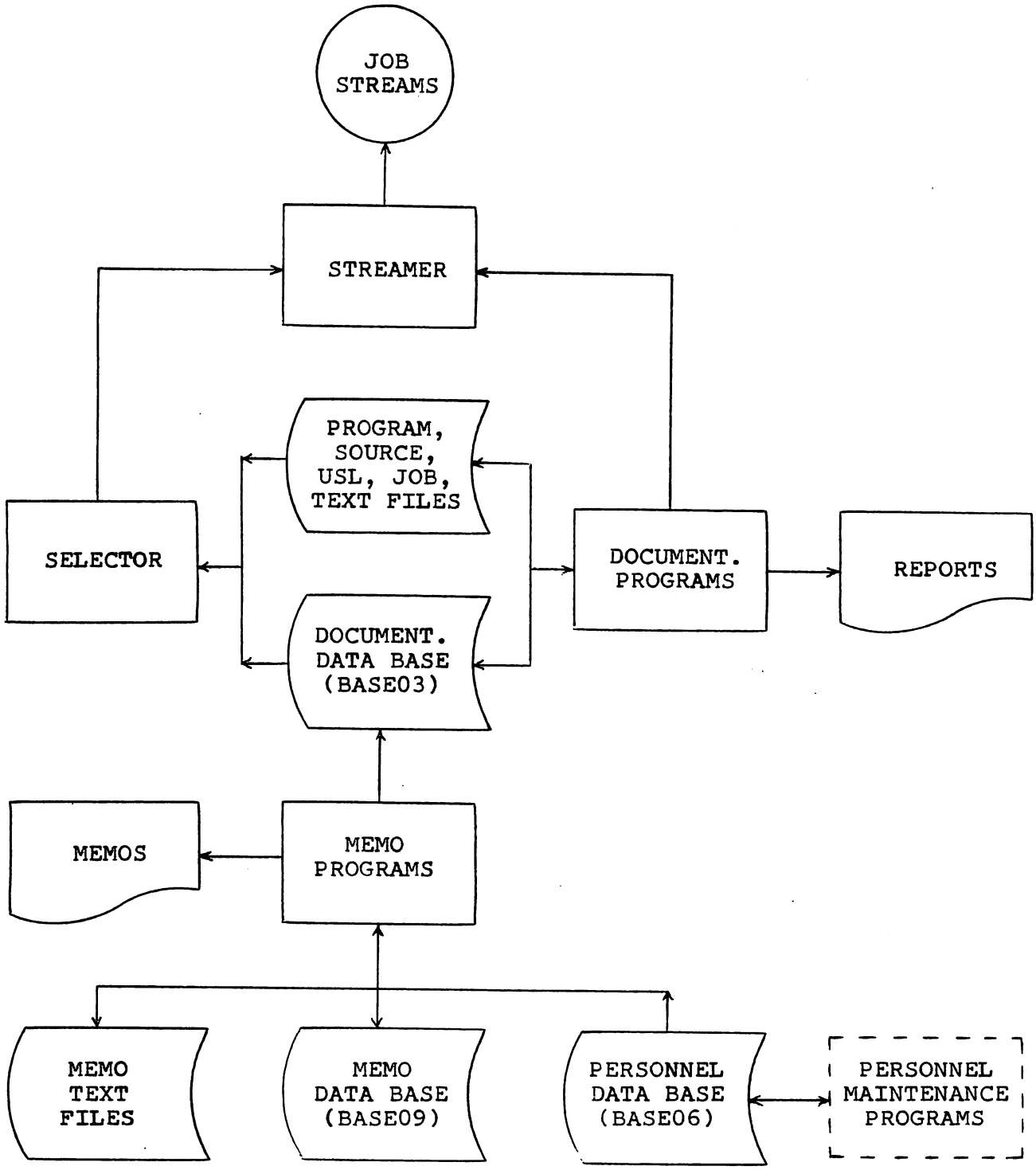

FIGURE 1 - GENERALIZED SYSTEM FLOWCHART

```
NAME:       PROGRAM, DETAIL (/1);
ENTRY:      PROG-ID (PROGRAM-INDEX),
            GROUP-NAME,
            ACCOUNT-NAME,
            PROG-TYPE,
            PROG-DESCRIPTION,
            MNEMONIC-NAME (MNEMONIC-INDEX),
            PROG-CLASS,
            SYSTEM,
            LANGUAGE,
            INSTALLED,
            REVISION,
            NEXT-REV-DUE,
            MAXDATA,
            CAPABILITY,
            MODE,
            ALLOCATE-FLAG,
            STATUS,
            STATUS-DATE;
CAPACITY:   25;

NAME:       DATA-FILE, MANUAL (/1);
ENTRY:      FILE-ID (1),
            GROUP-NAME,
            ACCOUNT-NAME,
            FILE-DESCRIPTION,
            FILE-TYPE;
CAPACITY:   50;

NAME:       COPYLIB, MANUAL (/1);
ENTRY:      COPY-ID (1),
            COPY-DESCRIPTION,
            COPY-TYPE;
CAPACITY:   25;

NAME:       PROG-PROG-XREF, DETAIL (/1);
ENTRY:      CALLING-PROG (PROGRAM-INDEX (CALLED-PROG)),
            CALLED-PROG (PROGRAM-INDEX (CALLING-PROG));
CAPACITY:   50;

NAME:       PROG-FILE-XREF, DETAIL (/1);
ENTRY:      PROG-ID (PROGRAM-INDEX (FILE-ID)),
            FILE-ID (DATA-FILE (PROG-ID)),
            ACCESS-MODE;
CAPACITY:   50;

NAME:       PROG-COPY-XREF, DETAIL (/1);
ENTRY:      PROG-ID (PROGRAM-INDEX (COPY-ID)),
            COPY-ID (COPYLIB (PROG-ID));
CAPACITY:   50;

NAME:       SELECTORS, MANUAL (/1);
ENTRY:      SELECTOR-CODE (1),
            SELECTOR-TITLE;
CAPACITY:   10;

NAME:       OPTIONS, DETAIL (/1);
ENTRY:      SELECTOR-CODE (SELECTORS (OPTION-SEQUENCE)),
            OPTION-SEQUENCE,
            NEXT-SELECTOR,
            PROG-ID;
CAPACITY:   50;
```

FIGURE 2 - PARTIAL SCHEMA OF BASE03

SELECTOR.08 SUN, JUN 24, 1979, 12:10 PM

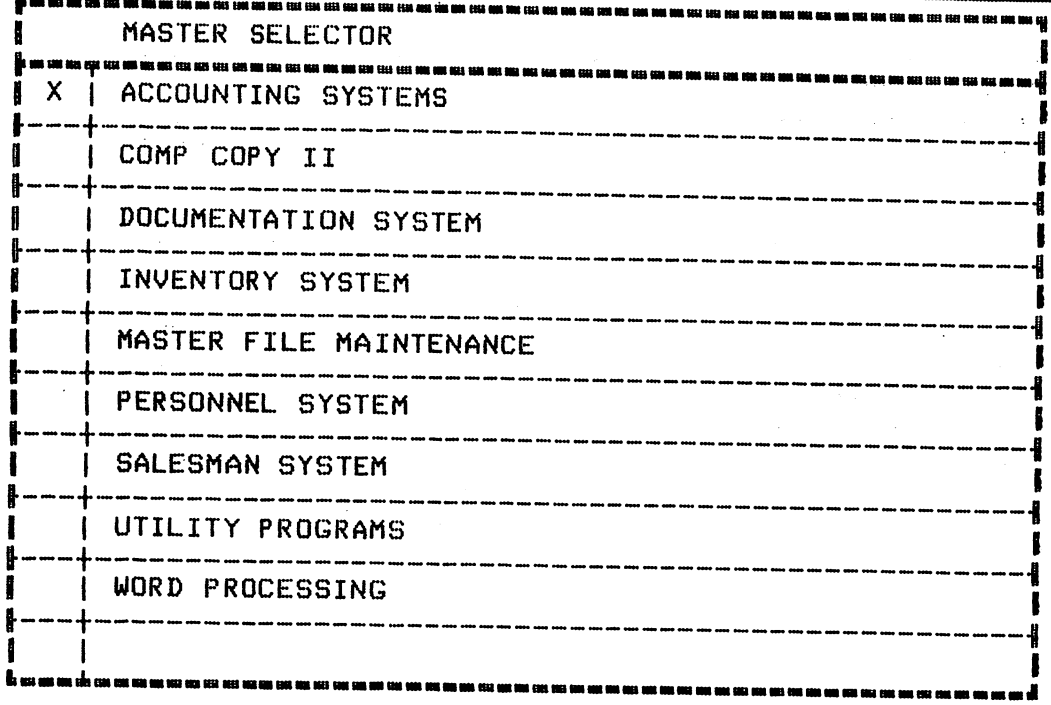

FIGURE 3 - SELECTOR MENU SCREEN

DOCUC060.00 DOCUMENTATION MAINTENANCE

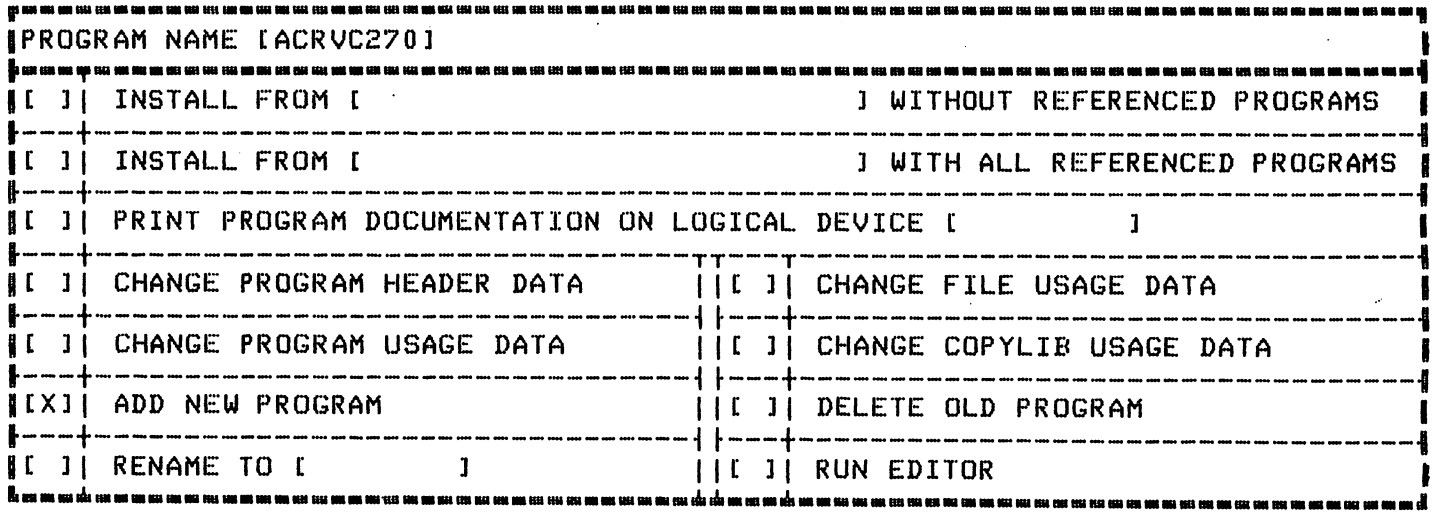

FIGURE 4 - DOCUC060 MENU SCREEN

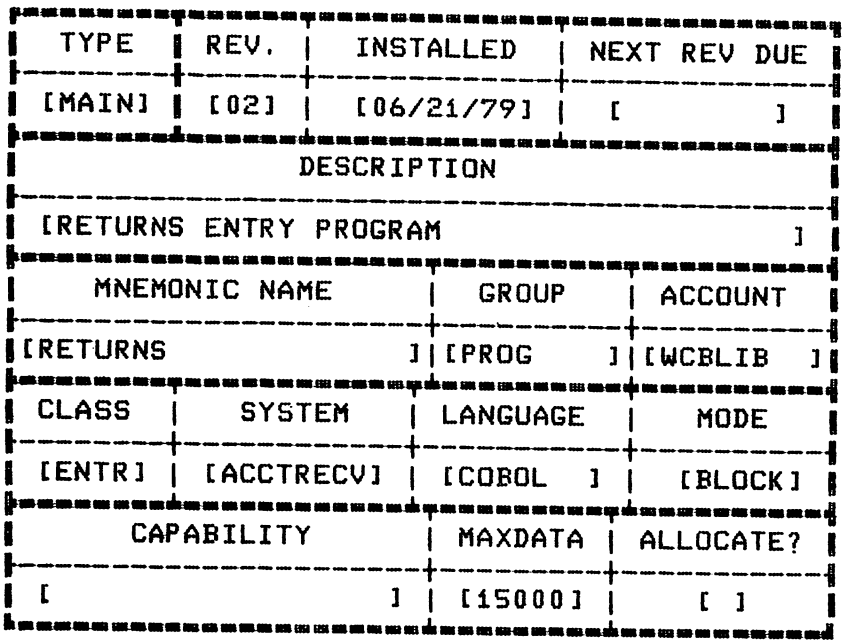

FIGURE 5 - DOCUC060 PROGRAM HEADER SCREEN

DOCUC060.00 FILE USAGE DATA Main program: ACRVC270

FILE	MODE	GROUP	ACCOUNT	DESCRIPTION	TYPE
ACRVF270	INPUT	FORMSLIB	WCBLIB	RETURNS ENTRY FORMS FILE	FAST
BASE02	i	DATA	DATABASE	BOOKS AND AUTHORS DATABASE	BASE
BASE0207	INPUT	DATA	DATABASE	BOOK (DETAIL)	DSET
INVCS	i	DATA	DATABASE	INVOICE DATABASE	BASE
INVCS15	I/O	DATA	DATABASE	CUSTOMER (DETAIL)	DSET
INVCS19	INPUT	DATA	DATABASE	DISCOUNT (DETAIL)	DSET
INVCS20	INPUT	DATA	DATABASE	DEAL (DETAIL)	DSET
INVCS25	I/O	DATA	DATABASE	COMPANY-CONTROL (DETAIL)	DSET
INVCS26	I/O	DATA	DATABASE	CUSTOMER-INFO (DETAIL)	DSET
INVCS29	OUTPUT	DATA	DATABASE	RETURNS (DETAIL)	DSET
INVCS34	OUTPUT	DATA	DATABASE	RETURN-ITEMS (DETAIL)	DSET
INVCS35	OUTPUT	DATA	DATABASE	RECEIVED-FROM (DETAIL)	DSET

FIGURE 6 - DOCUC060 FILE USAGE SCREEN

DOCUC060.00 PROGRAM USAGE DATA Main program: ACRVC270

PROGRAM	DESCRIPTION
IMAGE	*** PROGRAM NOT DEFINED ***
INVCC250	RETRIEVES OR VERIFIES CITY,STATE,ZIP
INVTC050	UPDATE BOOK QUANTITIES
UTILC050	CHANGE MULTIPLE IMBEDDED SPACES TO ONE
UTILC161	CALL WHO INTRINSIC FROM COBOL
UTILC190	GET PARAMETER NUMBER
UTILC250	RIGHT-JUSTIFY ZERO-FILL ROUTINE
UTILC340	PERFORMANCE LOGGING SUBROUTINE
UTILC380	DATABASE ERROR BOMBOUT ROUTINE
UTILC390	VIEW ERROR BOMBOUT ROUTINE
UTILC420	CUSTOMER NUMBER RETRIEVAL SUBPROGRAM
VIEW	*** PROGRAM NOT DEFINED ***

FIGURE 7 - DOCUC060 PROGRAM USAGE SCREEN

DOCUC060.00 COPYLIB USAGE DATA Main program: ACRVC270

COPYLIB	DESCRIPTION	TYPE
BASE0200	DATABASE BASE02	WORK
BASE0207	DATA SET BOOK	WORK
INVCS000	DATABASE INVCS	WORK
INVCS015	DATA SET CUSTOMER	WORK
INVCS019	DATA SET DISCOUNT	WORK
INVCS020	DATA SET DEAL	WORK
INVCS025	WORKING STORAGE FOR DATA SET RETURNS	WORK
INVCS026	DATA SET CUSTOMER-INFO	WORK
INVCS029	DATA SET RETURNS	WORK
INVCS034	DATA SET RETURN-ITEMS	WORK
INVCS035	DATA SET RECEIVED-FROM	WORK
INVTY050	UPDATE VALUES PARAMETERS	WORK
UTILY161	CALL WHO 'WHO' INTRINSIC PARAMETERS	WORK
VIEWREAD	VIEW READ ROUTINE	PROC
VIEWWORK	VIEW WORK AREA	WORK
WSDBWORK	COMMON DATABASE WORKING STORAGE	WORK

FIGURE 8 - DOCUC060 COPYLIB USAGE SCREEN

```
ACRVC270  RETURNS ENTRY PROGRAM                          07/17/79   ACRVC270   1

TYPE:      MAIN                  CLASS:    ENTR          REV.: 02
NAME:      RETURNS               SYSTEM:   ACCTRECV      DATE: 06/21/79
GROUP:     PROG                  MODE:     BLOCK         NEXT:   /  /
ACCOUNT:   WCBLIB                CAP.:
LANGUAGE:  COBOL                 MAXDATA:  15000         ALLOCATE? NO
```

----------FILES USED----------

NAME	GROUP	ACCOUNT	MODE	DESCRIPTION
ACRVF270	FORMSLIB	WCBLIB	INPUT	RETURNS ENTRY FORMS FILE
BASE02	DATA	DATABASE	i	BOOKS AND AUTHORS DATABASE
BASE0207	DATA	DATABASE	INPUT	BOOK (DETAIL)
INVCS	DATA	DATABASE	i	INVOICE DATABASE
INVCS15	DATA	DATABASE	I/O	CUSTOMER (DETAIL)
INVCS19	DATA	DATABASE	INPUT	DISCOUNT (DETAIL)
INVCS20	DATA	DATABASE	INPUT	DEAL (DETAIL)
INVCS25	DATA	DATABASE	I/O	COMPANY-CONTROL (DETAIL)
INVCS26	DATA	DATABASE	I/O	CUSTOMER-INFO (DETAIL)
INVCS29	DATA	DATABASE	OUTPUT	RETURNS (DETAIL)
INVCS34	DATA	DATABASE	OUTPUT	RETURN-ITEMS (DETAIL)
INVCS35	DATA	DATABASE	OUTPUT	RECEIVED-FROM (DETAIL)

--------COPYLIBS USED--------

NAME	TYPE	DESCRIPTION
BASE0200	WORK	DATABASE BASE02
BASE0207	WORK	DATA SET BOOK
INVCS000	WORK	DATABASE INVCS
INVCS015	WORK	DATA SET CUSTOMER
INVCS019	WORK	DATA SET DISCOUNT
INVCS020	WORK	DATA SET DEAL
INVCS025	WORK	WORKING STORAGE FOR DATA SET RETURNS
INVCS026	WORK	DATA SET CUSTOMER-INFO
INVCS029	WORK	DATA SET RETURNS
INVCS034	WORK	DATA SET RETURN-ITEMS
INVCS035	WORK	DATA SET RECEIVED-FROM
INVTY050	WORK	UPDATE VALUES PARAMETERS
UTILY161	WORK	CALL WHO 'WHO' INTRINSIC PARAMETERS
VIEWREAD	PROC	VIEW READ ROUTINE
VIEWWORK	WORK	VIEW WORK AREA
WSDBWORK	WORK	COMMON DATABASE WORKING STORAGE

-------PROGRAMS CALLED-------

NAME	LANGUAGE	DESCRIPTION
IMAGE		PROGRAM NOT DEFINED
INVCC250	COBOL	RETRIEVES OR VERIFIES CITY,STATE,ZIP
INVTC050	COBOL	UPDATE BOOK QUANTITIES
UTILC050	SPL	CHANGE MULTIPLE IMBEDDED SPACES TO ONE
UTILC161	SPL	CALL WHO INTRINSIC FROM COBOL
UTILC190	SPL	GET PARAMETER NUMBER
UTILC250	SPL	RIGHT-JUSTIFY ZERO-FILL ROUTINE
UTILC340	SPL	PERFORMANCE LOGGING SUBROUTINE
UTILC380	SPL	DATABASE ERROR BOMBOUT ROUTINE
UTILC390	SPL	VIEW ERROR BOMBOUT ROUTINE
UTILC420	COBOL	CUSTOMER NUMBER RETRIEVAL SUBPROGRAM
VIEW		PROGRAM NOT DEFINED

FIGURE 9 - SAMPLE PROGRAM DOCUMENTATION