CONF-820606--4

DATABASE AUTOMATION OF ACCELERATOR OPERATION

by

B. J. Casstevens

and

C. A. Ludemann

Oak Ridge National Laboratory*

Oak Ridge, Tennessee 37830

CONF-820606--4

DE82 017525

INTRODUCTION

The Oak Ridge Isochronous Cyclotron (ORIC) is a variable energy, multiparticle accelerator that produces beams of energetic heavy ions which are used as probes to study the structure of the atomic nucleus. Some 250 scientists from around the world use this multi-million-dollar accelerator at the Oak Ridge National Laboratory.

In order to accelerate and transmit a particular ion at a specified energy to an experimenter's apparatus, the electrical currents in up to 82 magnetic field producing coils must be established to accuracies of from 0.1 to 0.001 percent. Mechanical elements must also be positioned by means of motors or pneumatic drives. A mathematical model of this complex system provides a good approximation of operating parameters required to produce an ion beam. However, manual tuning of the system must be performed to optimize the beam quality. This procedure is followed to produce ion beams that have not been accelerated before. However, since the major portion of operation is with beams that have been produced previously, efficient operation can best be accomplished by using the operating parameters recorded for prior experiments--and not using the modeling approach.

*Operated by Union Carbide Corporation under contract W-7405-eng-26 with the U. S. Department of Energy.

The database system was implemented as an on-line query and retrieval system running at a priority lower than the cyclotron real-time software. It was designed for matching beams recorded in the database with beams specified for experiments. The database is relational and permits searching on ranges of any subset of the eleven beam categorizing attributes. A beam file selected from the database is transmitted to the cyclotron general control software which handles the automatic slewing of power supply currents and motor positions to the file values, thereby replicating the desired parameters.

OPERATING ENVIRONMENT

A portion of the ORIC facility is shown in Figure 1. The main magnet yoke is at the left of the photograph. Twenty-three power supplies, furnishing up to 5000 amperes of dc current, tailor the internal magnetic field to define the trajectory of the ions being accelerated. The resonator for the RF high voltage accelerating system can be seen above the operators. Electrostatic and magnetic elements extract the ions from the cyclotron into the chest-high evacuated beam line below the resonator. A series of bending magnets, the first of which is shown below the end of the resonator, deflect the beam through evacuated tubes to one of 20 experiment stations located throughout the 15,000 ft^2 facility.

The power supplies are controlled by reference voltages furnished by digital-to-analog converters (DAC's) connected to the ORIC control computer. A linear relationship, including slope and offset, exists between the digit impressed upon the DAC and the current generated by the power supply. Since power supplies can be switched to more than one magnetic element, the control software must keep track of the interconnections and guarantee that a supply cannot furnish more current than a load element can dissipate safely. This is accomplished by detailed tables describing both transducers (power supplies, motors, actuators, etc.) and load elements.

The global tables describing the hardware environment are accessible to all tasks of the real-time control software. Tasks execute operator communications entered via a keyboard or a manual control panel; monitor equipment operation by scanning analog- or synchro-to-digital converters; and provide system status information through CRT displays or audible alarms.

One task of particular importance for this discussion is SET. It provides for the controlled setting of any element (power supply, motor, etc.) to a desired value in a continuous and proper slewing manner. The operator may interact with SET via the keyboard and command a single element, or a generic group of elements, to be driven to a specified value. Furthermore, tasks can communicate with SET to affect the entire hardware environment. Set points can be supplied in files furnished by the mathematical modeling routine mentioned earlier, or by the database routines to be discussed in the following sections.

DATABASE DESIGN

This section describes the data structures and data management routines that, combined, define the ORIC database system. The database is relational, as it permits searching on ranges of any subset of the ion-beam attributes (keys). Sequential attribute processing in the file retrieval routine, rather than an extensive overhead structure, is used to implement the relational model. The database is small ($\approx$400 files), with simple data structures. Hence, even with the extra processing required by the relational model, response time is reasonable.

Database structure

An ORIC database file contains all information on a beam which is necessary to reproduce it. This information includes: (1) ion-beam attributes, (2) interconnection assignments of the numerous power supplies and magnetic elements that steer the ion's path through the system, and (3) values that represent electrical currents of power supplies, positions of motor drive-trains, and status of mechanical actuators. The ion-beam attributes are the parameters which define or specify a particular beam. The remainder of the beam file describes the physical connections and transducer settings that will produce that beam. The ion-beam attributes thus serve as keys or relations upon which the database may be searched.

The beam attributes are grouped in a 10-word block called a file label, formatted as shown in Figure 2. To speed up database searching, the file labels of all beam files are grouped together at the start of the database in a directory. Word 10 of the file label contains pointers to the file body in the database. A brief explanation of each beam follows:

(1) Run Number - Uniquely identifies each chronological beam generated on ORIC. The alphabetic character denotes a series of runs made with the same ORIC equipment configuration.

(2) Experiment Number - Number assigned to the research experiment for which the beam is being generated.

(3) Ion Name - The atomic symbol of the ion being accelerated.

(4) Ion Mass - The atomic weight of the ions comprising the beam.

(5) Ion Charge - The number of unmatched protons, or stripped electrons, of the ions in the beam.

(6) Energy - The energy of the ions in million electron volts (MeV).

(7) RF Frequency - The frequency of the voltage accelerating the ions.

(8) and (9) E0 of Machine and Harmonic Number - Machine parameters that characterize the strength of the main magnetic field and mode of operation of the accelerating system which are calculated from the ion energy, mass, charge, and RF frequency.

(10) Date - The date when the run started.

(11) Target Station - The number of the beam line traversed by the beam from extraction from the ORIC to arrival at an experimental room.

From Figure 2 it is obvious that considerable data shifting and masking is necessary to compress the eleven ion-beam attributes into nine 16-bit label words. This was done to conserve system disc space (10 file labels can be stored in each 100-word directory sector). This data encryption subsequently mandated table-driven decoding of parameters in the database file and search routines, adding considerably to the database overhead and programming cost. Storing the beam attributes in ASCII in the file label would have been a far better design. The extra words needed probably would have been returned in reduced program size.

The ORIC database permits searching on ranges of any subset of the beam attributes. For example, all runs of $^{20}Ne^{8+}$ (ion Ne, mass 20, charge 8^+) with energies in the range 100-110 MeV could be scanned, or all beams with atomic mass 60 or greater that attained an energy of 100 MeV or more could be listed. All $^{14}N^{4+}$ beams channeled to Experiment

Room 14, or all beams run for Experiment B904 could be of interest. The versatility of the relational model is evident from these examples.

The format of a complete beam file is shown in Figure 3. The file consists of the following five blocks:

(1) Copy of file directory label - Beam attributes are duplicated in the file body to simplify hard-copy routines and verify file identification.

(2) Operator comment block - This block contains up to 255 characters of comments on the run. Irregular operating conditions or unusual beam observations are included here.

(3) Amplitude-sine harmonic table - This block contains coefficients of trigonometric relationships of ORIC harmonic coil currents, needed to properly shape the magnetic field for the accelerated ion.

(4) Physical quantities table - Mechanical positions or current readings not controlled by the computer but essential to successful beam generation are included here.

(5) Table of transducer assignments and settings - This is a table with a 6-word entry for each transducer under computer control. The entry identifies the transducer and its assigned load, if any. The flag-word contains a code for the transducer generic group (i.e., trimming coil, harmonic coil, beam line optics element, etc.,) which is used in formatting the beam run sheet. The transducer DVM reading and its setting in counts, from the global transducer table, are also included. For actuator transducers, a single bit in the counts field indicates on/off, inserted/withdrawn, or open/closed. This simple table of 6-word transducer entries, when sent to control routine SET, drives automated beam reproduction.

Database routines

The database software consists of routines that manipulate the database (file, query, retrieval, and edit functions) and routines to maintain the data base (back-up, compress, and dump functions). All database routines are on-line, running at a lower priority than the real-time control routines. The database routines are initiated via keyword commands entered at the operators' console.

Database manipulation routines

The following routines manipulate the ORIC database:

DDATTR -This routine performs all database reading and writing. All routines that access the database must do so via calls to DDATTR. DDATTR performs the following functions:

(1) DDATTR can dump a file to the database, entering its label in the database directory, and writing the entire file in the database proper. DDATTR references a database directory header to calculate pointers for performing this function.

(2) DDATTR can locate a file in the directory and transfer it to memory. In calling DDATTR to locate a file, a database routine must supply a min/max search block, and a search flag word (see Figure 4). The search flag word specifies which of the eleven beam attributes will be used in the search, while the min/max search block gives the range of each of the corresponding specified parameters. For example, to search for all NE beams run in September, 1981, a database task would set bits b_2 and b_9 of the search flag word to 1, and would define a 4-word min/max search block equivalent to NE,NE,09/01/81,09/30/81. The min/max values must be in the file label format of the corresponding attribute.

To perform the search, DDATTR sequentially scans each label in the database directory. To extract a specified attribute from the file label, DDATTR performs table-driven shifting and masking operations. It then compares the extracted attribute with the range of values given in the min/max search block. DDATTR searches for the first file that matches all the specified attributes, loads the file into memory, and returns. If no file in the directory matches all specified attributes, DDATTR flags an error. In calling DDATTR's locate option, a database routine can specify that the search start at the beginning of the directory, or continue from the last matched file. Thus, successive calls to DDATTR with the same min/max search block will return all database files matching the specified attributes.

(3) DDATTR can list or return successive database directory sectors, ten file labels per sector.

(4) Finally, DDATTR can redump or overwrite a file in the database. Here, DDATTR searches the directory for a label identical to the calling routine label, and then overwrites the corresponding file.

FILE - This database manipulation task is used to create a database file corresponding to the current cyclotron beam. A block diagram of FILE is shown in Figure 5. When initiated, FILE calls DEFINE to define the beam attributes and format them into a label block. DEFINE requests most file attributes from the operator, checks for legality, and performs shift and mask operations to encode it properly in the label block. DEFINE's processing is table-driven. The table describes each attribute in terms of alphabetic or numeric sub-attributes. Min/max values, relative label word, shift alignment, and extraction mask for each sub-attribute are included. Once all beam attributes have been defined, DEFINE calls DISPLAY to display a table of file attributes on the console. DEFINE will permit the operator to redefine attributes until he is satisfied, then DEFINE returns the formatted label to FILE.

FILE next calls FORMAT to format various data blocks into an ORIC beam file. FORMAT starts with the file label and then calls COMMENT to handle request and processing of the file comment block. FORMAT picks up the global amp/sine harmonic block then calls PHYS to handle specification of the physical quantities table. FORMAT builds the table of transducer assignments and settings from global tables. Finally, FORMAT formats the beam file from its constituent blocks and returns it to FILE.

FILE now calls DUMP to create a run sheet of the beam file. DUMP calls ENTRY to format the beam attributes in a single PRINT line, dumps each of the other beam file blocks, and returns to FILE. FILE calls DDATTR's dump option to enter the beam file in the database, and finally notifies the operator that the beam has been recorded.

FILE is the only database routine that performs the file or create function. All beam files are built and inserted in the database via FILE.

SEARCH - This database manipulation task is used to search for, and optionally load, a database beam file. A block diagram of SEARCH is shown in Figure 5. SEARCH begins by calling DEFINE to define the file search attributes. The operation of DEFINE here is nearly identical to its operation in FILE. The only differences are: (1) Any subset of the beam attributes may be defined, (2) A range, rather than a specific value is defined for each attribute; and (3) A search flag word, and min/max search block, rather than a file label block, is returned. Again, the attribute processing is table-driven, with numerous shift and mask operations. DEFINE permits redefinition of search attributes and calls DISPLAY to display the attributes in a console table.

SEARCH next performs successive calls to DDATTR's load option using DEFINE's search data structures. Each call returns either a matched file, or an error indication. For each file, SEARCH calls ENTRY to format a line of beam attributes on the console. SEARCH then asks the operator if he wishes to refine his search, set up the cyclotron to run from a file, or exit. If the operator wishes to search some more, SEARCH cycles; if he wishes to exit, SEARCH exits; if he wishes to load a beam file, SEARCH requests the run number. SEARCH builds a search flag word with the run number bit set, and a min/max search block with min run number = max run number = requested run number. SEARCH calls DDATTR's load option with this data. The returned beam file is then passed to a control routine SET, which slews the cyclotron transducers to the corresponding filed settings. Except for manual adjustment of a few transducers not yet under computer control, the cyclotron is set up to rerun the beam.

SEARCH is the only database routine which performs query and retrieval functions. These functions do not exist independently but are embedded in a special routine to set up the cyclotron from a beam file. The ORIC database was designed to meet the demands of an application--computer control of an accelerator--and independent search and retrieval functions, while possible, have not been found necessary.

EDIT - This database manipulation routine is used to edit or modify any of the file label parameters. As these beam parameters are the database keys or relations, modifying them corresponds to changing the interrelationships of files in the database. Figure 5 shows a block diagram of EDIT. EDIT requests the run number of the file to be edited, builds corresponding search data structures, and calls DDATTR's load option to locate the beam file in the database. EDIT then passes the file label to LABEL, which handles redefinition of the file label block. LABEL calls DISPLAY to format the file beam attributes into a console table. LABEL then enters a loop requesting parameter changes. LABEL's parameter processing is exactly the same as that of DEFINE. LABEL shifts and masks the redefined attributes into the file label block, and calls DISPLAY to redisplay the attribute table. When the file label is correct, LABEL returns it to EDIT. EDIT then calls DDATTR's redump option to overwrite the file in the database.

EDIT is the primary database edit routine. All database keys or relations may be changed interactively, using EDIT. On-line editing of file body data is not supported. As this data is a record of the physical beam setup at the time of the run, it is not usually subject to change. Twice in four years we have adjusted quantities in the file body to reflect new ORIC equipment configurations. This editing was performed on all files in the database using an off-line routine.

Data maintenance routines

Database maintenance functions are grouped together in a single routine called TRAN. A block diagram of TRAN is shown in Figure 6. TRAN performs the following data base maintenance operations:

(1) List database directory on printer. LSTDIR calls DDATTR's list option for successive database directory sectors. Each of the ten file labels in the sector is then formatted by ENTRY into a single printer line.

(2) Print run sheet for file in directory. PRINT requests run number from the operator, builds run number search data structures, and then calls DDATTR's load option to locate the beam file in the database. PRINT uses FILE's DUMP routine to produce a run sheet for the file.

(3) Write magnetic tape copy of database. BACKUP writes a tape directory label as the first tape record. BACKUP then retrieves sequential data-base files by performing null calls to DDATTR's load option. MAGRW's write option writes each file out to tape in 100-word records (corresponds to a disc sector), terminated by an end-of-file. MAGERR handles tape error detection and recovery.

(4) Transfer files from magnetic tape to database. TTOD requests the start and stop file run numbers, checks for a legal tape label, and calls MAGRW's read option to return successive beam files from the tape. TTOD scans the run number of each file until the start file is identified. Thereafter, TTOD calls DDATTR's dump option to enter the tape files in the database directory. When the stop file has been encountered and entered, the transfer cycle terminates.

(5) Delete a file from directory. DELETE requests the run number of the file to be deleted and then calls DDATTR's load option with run number search data structures. DELETE then sets the DEAD bit of word 1 of the file label (see Figure 2), and calls DDATTR's redump option to rewrite the file in the database. The file has been effectively

deleted, as all DDATTR options will skip a file whose DEAD bit is set.
(6) Revive a file in directory. REVIVE is an alternate entry point in DELETE which clears a file's DEAD bit, thereby marking it active.
(7) Rename a FILE in directory. RENAME requests the run number of the file to be renamed and then calls DDATTR's load option with run number search data structures. RENAME overwrites word 0 of the file label (see Figure 2) with the new run number and calls DDATTR's redump option to rewrite the file in the database.
(8) Compress database. COMPRESS first writes a blank database directory header on a scratch disc partition. COMPRESS then retrieves sequential database files by performing null calls to DDATTR's load option. (In the process all files marked DEAD will be skipped.) COMPRESS calls DDATTR's dump option with each successive file, and a pointer to the scratch disc partition, not the database. When all active database files have been entered in the scratch disc partition, COMPRESS uses DISCRW to overwrite the database, sector by sector.
(9) List magnetic tape files. LSTTPE checks for a legal tape label and calls MAGRW's read option to return successive beam files on tape. The beam files are sent to ENTRY which formats each file's beam attributes in a single line on the printer.

TRAN performs the ORIC database maintenance functions (backup, delete, compress, list). These maintenance functions, together with the previous record, modify, and retrieval functions, exhibit all of the characteristic data-handling operations of complex databases.

OPERATION OF THE DATABASE SYSTEM

Prior to the implementation of this system, 4 to 8 hours were spent to produce a beam for an experiment. This involved locating a beam record in the ORIC log books, manually adjusting all elements to the hand recorded values, and fine tuning to maximize the beam--each step prone to human error. Now the operations crew can rapidly search through the nearly 400 run files for the beam they desire, automatically set up the cyclotron's components to the recorded operating values in about 5 minutes (depending on excursions and permissible slewing rates), and have beam to the experimenters in half an hour. The time saved means not only an increase in productive research time but a cost savings as well. The ORIC consumes 75-100 megawatts of electrical power a day ($3000-4000 at $0.04 per kilowatt-hour).

FIGURE CAPTIONS

Figure 1. Photograph of the Oak Ridge Isochronous Cyclotron.

figure 2. Format of beam database directory label.

Figure 3. Format of database beam files.

Figure 4. Database search structures.

Figure 5. Block diagram of database manipulation functions.

Figure 6. Block diagram of database maintenance functions.

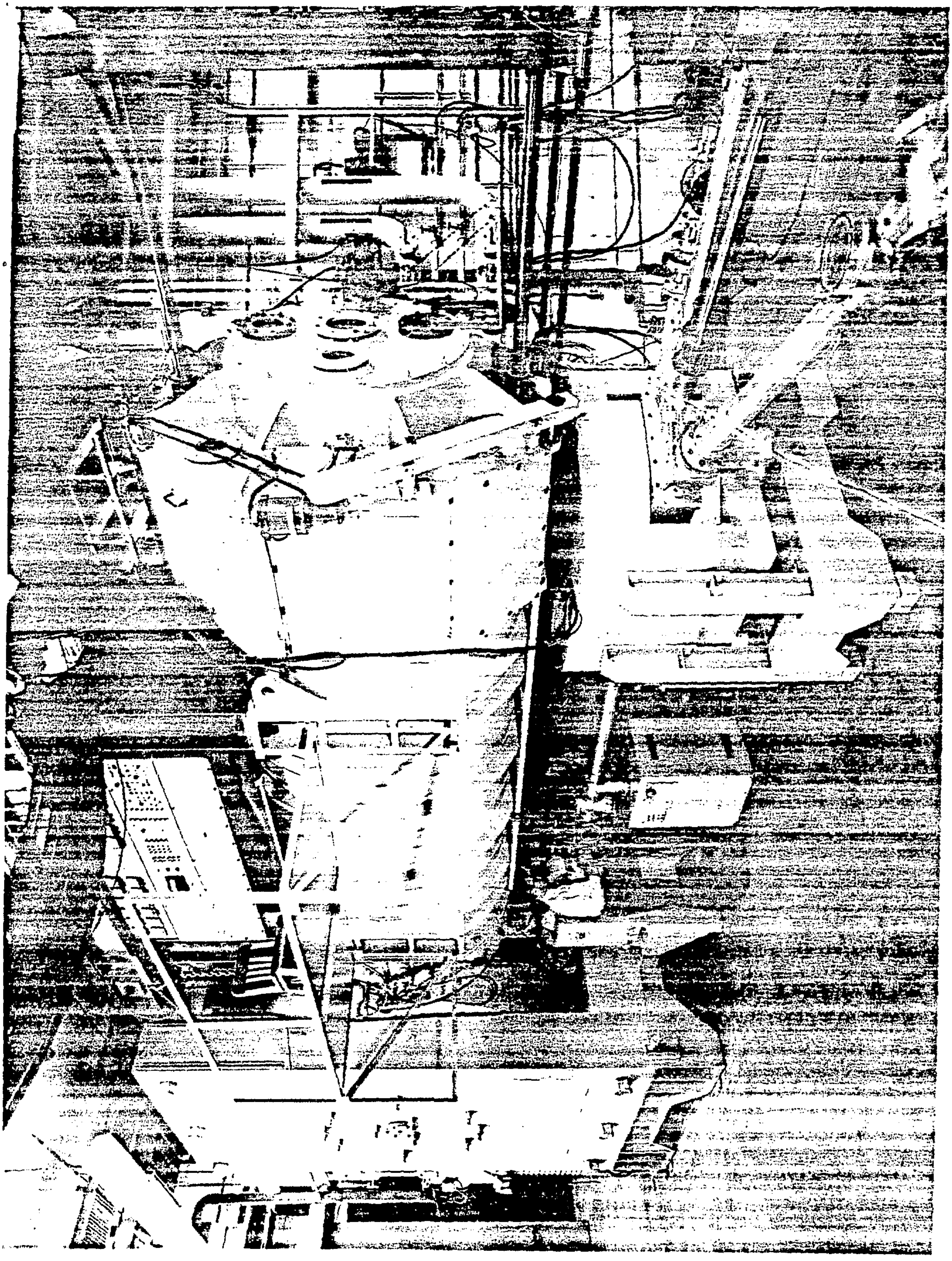


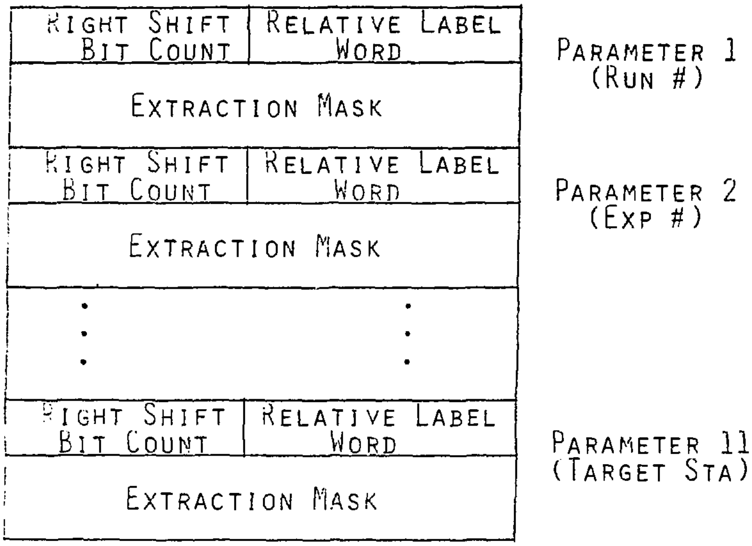
FIGURE 2. FORMAT OF BEAM DATABASE DIRECTORY LABEL

WORD				Description
0	6 BIT ASCII LETTER	10 BIT NUMERIC FIELD		RUN # (A000-Z999)
1	DEAD / 7 BIT CHARGE	8 BIT MASS		ION CHARGE ION MASS
2	TWO CHARACTER ASCII ION NAME			ION NAME (AL-ZR)
3	16 BIT POSITIVE ENERGY SCALED BY 10			ENERGY
4	16 BIT POSITIVE FREQUENCY SCALED BY 3			RF FREQUENCY
5	16 BIT POSITIVE E0 SCALED BY 10			E0 OF MACHINE
6	7 BIT YEAR	4 BIT MONTH	5 BIT DAY	DATE
7	6 BIT TARG NUMBER	7 BIT TARG LETTER	3 BIT HARMONIC	TARGET STATION HARMONIC NUMBER
8	6 BIT ASCII LETTER	10 BIT NUMERIC FIELD		EXPERIMENT #
9	12 BIT START SECTOR	4 BIT # SECTORS		DATABASE POINTERS

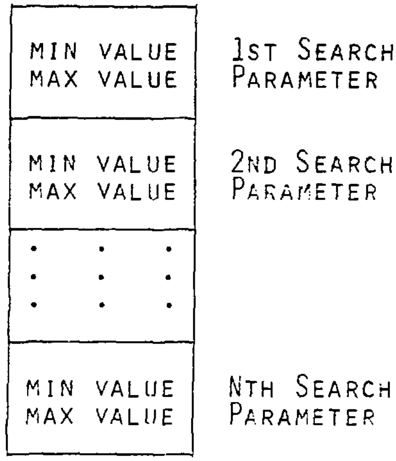
Figure 3. Format of database beam files

Word		Length
0-9	Copy of Beam Directory Label	10 words
10-138	Operator Comment Block (255 characters permitted)	128 words
139-168	AMP/SINE Floating Point Harmonic Table	30 words
169-218	Physical Quantities Block	50 words
219-999	Power Supply # | Load #	$\leq$ 675 words
	PS Flag Word	1st Power Supply Entry Block
	Present Count (DPI)	
	DVM Reading (FP)	
	: : : :	
	Power Supply # | Load #	
	PS Flag Word	Nth Power Supply Entry Block
	Present Count (DPI)	
	DVM Reading (FP)	
	F F F F	Trailer Words
	F F F F	

FIGURE 4. DATABASE SEARCH STRUCTURES



(A) BEAM PARAMETER EXTRACTION TABLE



(B) MIN/MAX SEARCH BLOCK

B_0 B_1 B_2 B_3 $\cdots$ B_{14} B_{15}

B_i = 1 (i+1)TH PARAMETER SPECIFIED IN SEARCH

B_i = 0 (i+1)TH PARAMETER <u>NOT</u> SPECIFIED IN SEARCH

B_0 CORRESPONDS TO 1ST LABEL PARAMETER (RUN #).

B_1 CORRESPONDS TO 2ND LABEL PARAMETER (EXPERIMENT #).

B_2 CORRESPONDS TO 3RD LABEL PARAMETER (ION NAME).

⋮

B_9 CORRESPONDS TO 10TH LABEL PARAMETER (DATE).

B_{10} CORRESPONDS TO 11TH LABEL PARAMETER (TARGET STATION).

(C) SEARCH FLAG WORD

FIGURE 5. BLOCK DIAGRAM OF DATABASE MANIPULATION FUNCTIONS

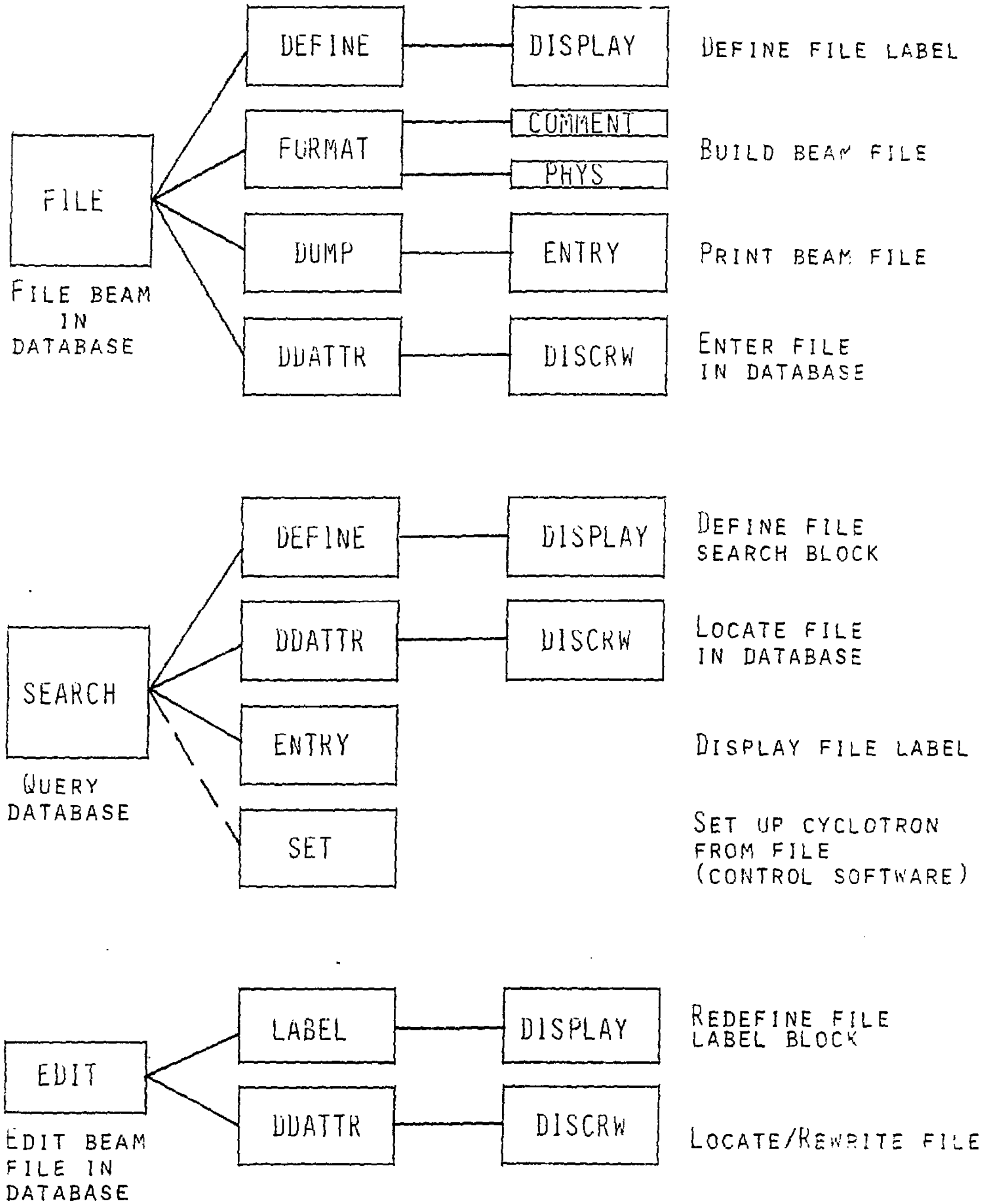


FIGURE 6. BLOCK DIAGRAM OF DATABASE MAINTENANCE FUNCTIONS