JSC-13935
E80-10276
NASA CR-
160710

USER'S GUIDE FOR THE YIELD ESTIMATION SUBSYSTEM DATA MANAGEMENT SYSTEM (YESDAMS)

JOB ORDER 74-963

(E80-10276) USER'S GUIDE FOR THE YIELD ESTIMATION SUBSYSTEM DATA MANAGEMENT SYSTEM (YESDAMS) (Lockheed Electronics Co.) 51 p
HC A04/MF A01
CSCL 05A
G3/43

N80-30849
Unclas
00276

Prepared By
Lockheed Electronics Company, Inc.
Systems and Services Division
Houston, Texas

Contract NAS 9-15200

For
EARTH OBSERVATIONS DIVISION
SCIENCE AND APPLICATIONS DIRECTORATE

National Aeronautics and Space Administration
LYNDON B. JOHNSON SPACE CENTER
Houston, Texas

April 1978 LEC-12184

JSC- 13985

USER'S GUIDE
FOR THE
YIELD ESTIMATION SUBSYSTEM
DATA MANAGEMENT SYSTEM
(YESDAMS)

JOB ORDER 74-963

Prepared By
Roy Davenport
Mike Sestak

APPROVED BY

James A Wilkinson

for Philip L. Krumm, Acting Supervisor
Scientific Applications Section

Prepared By

Lockheed Electronics Company, Inc.

For

Earth Observations Division

Science and Applications Directorate

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
LYNDON B. JOHNSON SPACE CENTER
HOUSTON, TEXAS

April 1978

LEC- 12184

CONTENTS

Section Page

FIGURES

1. YESDAMS SYSTEM DESCRIPTION

The Yield Estimation Subsystem Data Management System (YESDAMS) was developed as a specialized data management system to solve the specific data handling problems of the YES. The basic functions of definition, replacement, addition, deletion, and listing were designed to address the unique requirements of the YES instead of a generalized data management system. This User's Guide has been developed to aid anyone who may want to develop a new Yield Data File or to update an existing file.

There are three applicable documents used as references by this User's Guide:

- AD 63-1347-4963-18 YES Data Management System
- LEC-11110 Detailed Design Specification for the Yield Estimation Subsystem Data Mangement System (YESDAMS).
- LEC-12186 "As-Built" Design Specification for YESDAMS.

The YESDAMS Data Base, according to specifications given in "LEC-11110", consists of four record types in addition to the data sets themselves. These are control, directory, descriptor and model definition records. The entire data base has one control record. Each data record will have associations with the control record and the other record types. Each of these record types will now be described in some detail. The structure of variables for each record type can be found in "LEC-11110".

1.1 CONTROL RECORD

The control record contains the file ID and passwords necessary to maintain the security of the file. It also contains the number identification, type, location on the data base, and length of

each record as well as the total number of records on the entire data base. Therefore, whenever part or all of a record is added to or removed from the data base, this master catalog must be updated.

1.2 DIRECTORY RECORDS

Every country represented on a data file will have a directory record. Each directory record consists of a series of elements giving information related to a particular block of data records. That information consists of an ID #, a level #, and three pointers to other directory elements which will be discussed in more detail later. It also contains the number, IDs, record location, and beginning location in the record of each descriptor, definition and data record related to the directory element.

1.3 DESCRIPTION RECORDS

Descriptor records contain a description of a data block, the number of variables, their format, units, length of the data block, etc as well as an identifier for the descriptor itself.

1.4 MODEL DEFINITION RECORDS

Model definitions describe how the data is to be manipulated for computational purposes in yield estimation.

1.5 DATA RECORDS

Data Records which represent one directory level are strung together. Each record contains a Block I.D., Physical Record location , and the displacement of the next record in the string, and the actual data formated by the related descriptor.

2. YESDAMS FUNCTION TYPES

Five basic functions are performed on each of these record types in the process of creating and maintaining the data base. These are: Define (DEF), Add (ADD), Replace (RPL), Delete (DEL) and List (LST).

2.1 DEFINE

The DEF function is used to create a new record. Initial values are set and space is set aside for the record on the data base.

2.2 ADD

The ADD function is used to include additional material in an already existing record. Added information must be set up to logically follow the already existing record.

2.3 REPLACE

With the replace function part of a record may be replaced if information therein is found to be erroneous or if circumstances have changed its validity.

2.4 DELETE

The delete function allows the removal from the data base of records or parts thereof no longer of use. This then frees dead record space for reuse.

2.5 LIST

The list function provides for printed review of the current contents of any portion of a record or an entire record or group of records.

3. SECTIONS AND PARAMETERS

Structures have been developed that are divided into several sections or sectors each of which consists of several parameters which are the actual variables relating to the data base and its management. For instance the control record has a security sector (SEC) consisting of a file ID# (FILEID), the number of passwords (NUMPASS), the passwords (PASSWORD(8)), a reserved area (RESERVED), the number of directories (NUMBDIR), the number of descriptors (NUMDES), the number of model definitions (NUMDEF), and the number of directory elements with data (NUMDAT). LEC-11110 contains a complete list of the sectors and parameters of each record as well as a short description of each. Many of these will be described individually as building and maintaing the data base are explained further.

4. JOB CONTROL LANGUAGE

The following is the JCL necessary to run the YESDAMS Data Management System as implemented on an IBM System 360/195 computer in Suitland Maryland... Input cards consists of function cards, sector cards, and "END;" delimiter cards.

4.1 FUNCTION CARD

This is the first card for any execution and contains the following information:

Col 1-3 Function Abreviation
Col 5-7 Subfunction Abreviation
Col 9-11 Section Abreviation
Col 13-80 Parameters

example; SC=(SEC,DTY), Numbers=(2);

Parameters are input by name or abreviation followed by an = sign and the actual value(s) inclosed in parenthesis. Multiple values for any parameter are separated by commas as well as each parameter. The subrange parameter uses colons as a separation delimiter between subrange sets. A semi-colon ends the parameter input. The order of appearance of the individual parameters does not matter since they are internally parsed in order as they appear on PARMFILE.

The parameter list may extend to more than one card in which case it is continued in column 13 of the continuation card and the previous card must end in a comma.

4.2 SECTION CARDS

Each section card consists of a three letter sector abreviation starting in column 9 and followed by a parameter list which follow the same rules as the Function Card parameters. For each function there may be different sectors which are necessary for a given subfunction. The same is true of the various parameters in each sector. The PARMFILE(see discussion on PARMFILE) contains

the sectors and parameters for each record and indicates whether each is required, optional or forbidden for use with each function. Also a certain order of use is necessary in some cases and this is also indicated. Certain sectors may be repeated but are limited to number.

4.3 "END;" DELIMITER CARDS

A card with "END;" starting in column 9 must follow the Function Card and each Sector Card type.

A card with "END;" starting in column 5 must follow the last Sector "END;" card to show the end of the subfunction sectors.

A card with "END;" starting in column 1 must follow the last mentioned card to show the end of the function.

5. INITIAL CREATION OF DATA RECORD TYPES

Before any other function can be used to alter or list a data type, that data type has to be defined and placed onto the data base. The define function for each subfunction (CTL, DIR, DES, DFN and DAT) will be discussed in the order shown.

5.1 DEFINE SUBFUNCTION CONTROL (CTL)

Since the first record to be set up for any data file must be the control record, the first execution card would have the following appearance:

```
DEF  CTL  CL1 DDNAME=(USA), NUMSECS=(6),
              SC=(SEC,DTY,DSY,DFY,DAY,REC);
          END,
```

Because all sectors are mandatory, the parameter DDNAME will be the only change to the initial card for other data file control records. All parameters for each sector are mandatory except the subrange parameter for SEC. Except that CL1 must appear on the function card, the other sectors can be placed in any order.

For the sector SEC which does not have a subrange the values are placed directly into the control structure on a one to one basis from the following example card:

```
SEC ID=(UNISTAT), NP=3, DW=(PHILLIPS, DVNPORT, LADVC,,,,),
    ND=(0),......;
END;
```

The other sectors are represented by a subrange parameter which should be listed first for reason of clarity. An example DTY sector follows:

```
DTY SR=(1:8,1:4), DI=(0), DL=(-1), DR=(0), DT=(0),......;
END;
```

For any single arrayed element the value shown will be placed into all array positions shown by the first pair of ranges; however, the second pair of ranges would represent the positions to be considered in a double subscripted array element.

The basic rules for subrange input and non-subrange input will follow for other functions and subfunctions.

Most values in the control structure are initialized to zero since no other records yet exist but locations are initialized to -1 since location zero does exist and is used for the control record on the data file. Other sectors of define control follow:

```
DSY SR=(1:16), SI=(0), SL=(-1), SP=(0);
END;

DFY SR=(1:200), FI=(0), FL=(0), FP=(0);
END;

DAY SR=(1:456), TI=(0), TL=(-1),.....;
END;

REC SR=(1:456), RT=(0), RS=(0), RC=(-1);
END;
```

Notice that in the case of DAY that while the largest subscript allowable for some variables is only 200, in order to initialize all variables for their fullest extent, the largest possible upper bound is used. Care however must be taken when the lower bound is to be a large number. If values are to be put into high subranges of variables with ranges which exceed those of other variables in the same sector, such operations must only be performed on variables which have an appropriate extent or an error will result and execution of the job will be terminated.

5.2 DEFINE SUBFUNCTION DIRECTORY (DIR)

After the control record is defined the directories must be set up. A maximum of eight directories can be defined, one for each of eight countries. Every country's directory need not be set up before any descriptors or model definitions can be defined. But before descriptors or model definitions for a particular country can be defined, that country's directory must have been established. Each directory consists of up to 268 elements each of which has several sections of parameters. The first section establishes the identity of the section; this is the DID section. For instance, the first element would most likely

identify the country. The subrange would be 1:1, only element one is to be defined. The number of its level is NUMLEVEL=(1). Its level number(s) might be LEVELS=(3). The directory ID # is the last level number which is the only one in this case so DIRID=(3). The directory element name would be say, DIRNAME=(USA). No descriptors or model definitions have been created yet so NUMBDES=(0) and NUMBDEF=(0). Thus the first cards for defining a directory element including the function card would be:

```
DEF DIR DR1 DDNAME=(USA), ID=(UNITSTAT), PW=(LEDUC), NC=(7),
        SECTIONS=(DID,DSC,DTA,DFN,PTR,DTY,REC);
    END;

    DID SR = (1:1), NL = (1), LV = (3), RI = (3), DN = (USA),
        NS=(0), NF=(0);
    END;
```

The DID sector cards should all be put in together before a terminating column 9 END; card is put in and another sector is started. Some more sample DID sector cards for a state in the US Great Plains might be as follows:

```
DID SR=(2:2), NL=(2), LV=(3,1), RI=(1),
    DN=GREAT PLAINS), NS=(0), NF=(0);
DID SR=(3:3), NL=(3), LV=(3,18), RI=(8),
    DN=(COLORADO), NS=(0), NF=(0);
DID SR=(4:4), NL=(4), LV=(3,1,8,10), RI=(10),
    DN=(NORTHWESTERN), NS=(0), NF=(0);
DID SR=(5:5), NL=(4), LV=(3,1,8,20), RI=(20);
    DN=(NORTHERN), NS=(0), NF=(0);
DID SR=(6:6), NL=(4), LV=(3,1,8,60), RI=(60)
    DN=(WESTERN SLOPE), NS=(0), NF=(0);
    .
    .
    .
DID SR =(103:103), NL=(4), LV=(3,1,48,80), RI=(80), DN=(SOUTH TEXAS),
    NS =(0), NF=(0);
```

Level numbers on second level must be unique. On other levels they can repeat but must not do so when they have the same next higher level number. Two directory elements with all the same

level numbers except the last one are said to have the same parent. If the directory ID# of one is the next larger than the other with the same parent then the larger is said to be the brother of the smaller. Of the elements with the same parent, the one with the smallest directory ID # is said to be the child of that parent. The section which must be defined immediately after all DID sector cards are processed is the PTR or pointer section. In this sector, the PARENT, BROTHER and CHILD of each directory element is found. These are not specified as the level numbers are, but a card with null arguments is included to ensure that the sector is defined; the actual determinations are made from the level numbers by internal programming. That null card must contain the subrange over which DID cards have been made defining directory elements, and the three pointer parameters.

```
PTR SR=(1:103),PA=( ), BR=( ), CH=( );
END;
```

Next must come sectors DSC, DTA and DFN. Among the three they may be in any order, but all must follow the PTR sector. All give initial values to ID #s, record locations and record start pointers for records not defined yet so all should be initialized to zero or -1 for all directory elements defined as is appropriate. For instance:

```
DSC SR=(1:103), SI=(0), SL=(-1), SP=(0);
END;
```

Finally the information in the control record pertaining to the directory records must be updated. This is done by including cards for sectors DTY and REC in the DEF DIR run. The two can be in any order but the pair must be the last section cards in the run. For the DTY sector, the following must be specified: the directory ID # (the same as the directory ID# of the country level directory element), the directory name (an abbreviation of the country level directory element name), the number of records to be reserved for this directory (135 elements fill a record so if more than this is defined 2 records will be needed;

however, if less than 135 are defined now but it is projected that more will eventually be used, either 1 or 2 can be reserved at this time), and the total number of directory elements just defined for this directory.

The directory record location and total elements per directory record are mandatory parameters but should contain null arguments as they are internally determined. Thus for the above discussed directory the DTY sector card might be

```
DTY SR=(1:1,1:2), DI=(3), DC=(USA), DR=(2),
    DT=(103),DL=( ), DE=( );
END;
```

Record locations are assigned sequentially, that is the first defined goes to record 1, the second to record 2, etc. So this first directory will be assigned to record 1 and 2. If another directory is defined next it will be assigned to record 3; however, if a descriptor for the first directory is defined next it will be assigned to record 3 instead.

Sector REC contains a list of all the records of every type, their location and the amount of unused space left on each. This information must be calculated internally but to assure this is done, a REC section card with the three parameters with null arguments must be included.

```
REC RT=( ), RC=( ), RS=( );
END;
```

The RECTYPE is a number signifying the type of record by the following code:

1 = directory
2 = descriptor
3 = model definition
4 = data

RECLOCAT is the location number of the record containing the directory (and all other records). RECSPACE is the free space on the record calculated from the total space that can be used by that type of record. The total length reserved for each record is based on the length of the Control record or 13024 bytes. Since 135 Directory elements require only 12960 bytes, then at this point RECSPACE is zero for a directory record. No subrange is required for the REC parameters as values are placed in these arrays sequentially as well, or a spot must be filled in after a record has been deleted and this search is best program controlled.

5.3 DEFINE SUBFUNCTIONS DESCRIPTOR (DES) AND DEFINITION (DFN)

Next the descriptor and model definition records must be defined. They can be defined in any order but both must be defined before any data records.

5.3.1 DESCRIPTOR

Up to a total of 16 may be defined and each of these can be associated with any number of directory elements in any directory; however, it is most likely that one or two will be designed to cover all elements of a single country's directory. A descriptor record provides a format for the internal representation of the data as well as the format it will be output in and a description of the variable and units it represents. After the function card with its associated sectors which would look like the following,

```
DEF DES DS1 DD=(USA), ID=(UNITSTAY), PW=(PHILLIPS), NC=(6),
            CO=(3), SC=(PLM,IDS,COD,DSC,DSY,REC);
        END;
```

any of the three descriptor sectors may be begun. Only one PLM sector card is needed. This contains a unique descriptor ID#, the number of bytes in any data block associated with the descriptor, the number of data block to be described initially,

the total number of variables in those blocks (and therefore to be described), total number of units in the descriptor, total number of variable codes specified in this descriptor, and a reserved section of space. A sector card for this might then look like

```
PLM SI=(491), BS=(128), NB=(50), NV=(9), TS=(5),
    TC=(24), RV=( );
END;
```

For each variable code then an IDS sector card is needed. Units and variable codes with the same subscripts are not necessarily directly related, so each code need not have a corresponding unit or vice versa. In fact, there is a greater possible subscript extent for codes than units and therefore if codes are being specified in an area beyond the extent of the units' arrays, the units' parameters should not be included on the card. The code parameters on the other hand are always mandatory and would be set to zero and blanks in areas where the units continue beyond them. Part of a series of IDS cards which fits the preceeding PLM sector information is given below.

```
IDS SR=(1:1), CI=(491), CN=(YEAR BLOCK), CA=(YBLK)
    UI=(241), UN=(DEGREES (ELSIUS), UA=(C);
IDS SR=(2:2), CI=(61), CN=(YEAR), CA=(YR), UI=(201),
    UN=(MILLIMETERS), UA=(MM);
IDS SR=(3:3), CI=(90), CN=(POINTER), CA=(PTR), UI=(142),
    UN=(90 FAHRENHEIT),
IDS SR=(4:4), CI=(35), CN=(MEAN TEMPERATURE), CA=(AVGT),
    UI=(236), UN=(HECTARES), UA=(HEC);
IDS SR=(5:5),CI=(5), CN=(PRECIPITATION), CA=(PCP),
    UI=(228), UN=(QUINTALS), UA=(QU);
IDS SR=(6:6), CI=(46, CN=(DEGREE DAYS >), CA=(DD>),
    UI=(0), UN=( ), UA=( );
    :
```

```
IDS SR=(24:24), CI=(202), CN=(WINTER WHEAT), CA=(WW);
IDS SR=(25:56), CI=(0), CN=( ), CA=( );
END;
```

The COD sector describes each data item (subelement of each variable), its size, placement in the data block, the variable code corresponding to it, the unit code corresponding to it and the size of the output field it would be put into. Each variable must have a COD section card. It might be easiest to first zero all parameters rather than making sure unused ones are zeroed as one proceeds.

```
COD SR =(1:20, 1:31 ), CL=(0), UL=(0), EL=(0), BL=(0).....;
```

The result for a descriptor using information from the previous PLM and IDS sectors could be:

```
COD SR=(1:1), CL=(1), UL=(0), NE=(1), BL=(1), BA=(2),
    SE=(0), OF=(5), EL=(0);
COD SR=(2:2), CL=(2), UL=(0), NE=(1), BL=(3), BA=(2),
    SE=(0), OF=(5), EL=(5);

.
.
.

COD SR=(5:5), CL=(5), UL=(2), NE=(12), BL=(33), BA=(2),
    SE=(0),OF=(6), EL=(12,13,14,15,16,17,18,19,20,21,22,23);


.
.
.

COD SR=(9:9), CL=(9), UL=(6), NE=(4), BL=(113), BA=(4),
    SE=(0), OF=(12), EL=(24);
```

Next the Directory and Control record sectors pertaining to the directory must be updated. These are sector DSC in DIR and sectors DSY and REC in CTL which can be handled in any order. The descriptor is assigned a record location number in the first of these sectors processed, and this location is passed on to the other sectors as they are encountered.

Sector DSC cards indicate by level and level numbers which directory elements are to have data associated with this descriptor. The descriptor ID# record location and location of the first byte of this descriptor on that record are then entered in the DSC sector of the appropriate directory element. The location of the first byte is necessary because up to 4 descriptors can be on one descriptor record. A series of DSC cards for part of the directory previously given might be:

```
DSC NL=(2), LV=(3,1), SI=(491), SL=( ), SP=( );
:
DSC NL=(3), LV=(3,1,20), SI=(491), SL=( ), SP=( );
```

The DESLOCAT and DESDISPL parameters like most location parameters are best handled by programming rather than trying to keep track and assign each by hand.

In the DID sector of each directory element is a parameter NUMBDES (the number of descriptors associated with that element). This is also updated when a DSC card is processed.

The DSY sector of the control record contains a list of the ID # record location and first byte location on the record of each descriptor thus only one such sector card is needed per descriptor created. For the descriptor above this would be
DSY SI=(491), SL=( ), SP=( );

Again note the locations are determined internally. Also, sector SEC of the control record contains a parameter NUMDES, (the total number of descriptors) that is updated when a DSY card is processed.

Sector REC, the listing of all the records thus far created, must also be updated with the usual card of dummy parameters
REC RT=( ), RC=( ), RS=( );

In sector SEC of the control record is also a parameter NUMREC (the number of records defined); this is also updated when a REC sector card is processed.

5.3.2 DEFINITION

There is a separate model definition block for every district requireing a unique yield model. It contains the information needed to run the appropriate model for that district.
The define function card which follows:
DEF DFN DF1 DD=(USA), ID=(UNISTAT), PW=(LEDUC), NC=(16), CO=(3);
will represent the country given on this particular data file.
Note that the sector names are not given because all 16 sectors will be processed when a definition is created.

The sector MOD has no subrange input and contains the model definition number, model name, crop ID number, and counters for the number of inputs for the other definition sectors INV, ZON, WGH, HYR, RYR, XYR, XDX, TRN, YVR, YDA, and MDA. MOD should be run first and then the other sectors in any order. The update sectors DFY, DFN, and REC should then be input in any order. A description of the actual use of update sectors were discussed in descriptor. DFY relates to the definition storage in the Control Structure, DFN relates to directory entries, and REC is updated in the control structure.

The MOD input will appear as direct storage parameters as follows:

```
MOD NZ=(1)  MR=(3), NW=(4), HY=(1), RY=(1), XY=(1),
    NV=(5), NY=(1), NX=(9), DY=(0), DM=(0), ZX=(0),
    IB=(1), ZS=(4), CR=(201), MN=(USA-COLORADO),
    FI=(2001), RV=( );
```

For most other sectors in the definition structure all entries will be made one at a time using the subrange parameter to show array location. Sectors INV, YVR, YDA, MDA, and XDX will be discussed last because of special rules.

The sector ZON is actually representative of 3 parameters in the MOD sector. A zone is input first (represents a NZ entry).

```
ZON  SR=(1:1), NL=(3), LV=(3,1,8), LA=(39,0) NV=(0),
     VC=(0,0,0,0,0,0,0,0,);
```

and then the strata refering to the zone are input (represents NR entries)

```
ZON  SR=(2:2), NL=(4), LV=(3,1,8,20), LA=(39,5) NV=(5),
     VC=(5,35,40,101,103);
```

⋮

The MOD parameter ZS is the actual sum of the number of zones plus the number of strata. This sum should not excede 26. The parameter number of variables and number of levels should not exceed 8.

Other parameters in ZON are the latitude of the zone or strata and the actual levels and variables codes.

The inputs to WGH sector are up to 8 variables code numbers to be weighted, the number of weights per code (up to 20), and the actual weight values.

The actual defining of sectors HYR (Number of Historical Years), RYR (Number of Run Years), and XYR (Number of Normal Years) are indentical except the actual values input. Each contain the number of levels (up to 8) per set of levels (up to 6), and the number of year pairs (up to 8 beginning and ending years)

per set of levels. An example for HYR can represent all three sectors:

```
HYR SR=(1:1), NL=(3), LV=(3,1,8), NM=(1), BY=(1931), EY=(1977);
```

There can be up to 16 Truncation ID's designated in the TRN sector. To each is a related truncation abreviation, truncation name, and up to 24 variable code number assignments.

```
TRN SR=(1:1), TR=(713), TA=(TRN), TM=(TREND), NV=(3), VN=(1,2,3);
```

There can be up to 24 x-variables input using the XVR sector. Related to each are a variable operand, a variable deviation, and up to 3 each of codes of variables used to calculate the variable, variable subcodes, interger constants, variable locations, and floating constants.

```
XVR SR=(1:1), VI=(501), VV=(0,0,0), VS=(0,0,0), IK=(0,0,0),
    FK=(1.,0.,0.), VO=(1), DV=(0);
```

The sectors INV(raw variables) and YVR (Y-variable) are input by direct storage. INV has the parameters number of variables (up to 8) and the actual variable codes. Sector YVR represents one variable ID with a related variable operand and up to 3 each of codes of variables used to calculate variable, variable subcode, integer constant, and floating constant.

```
INV NV=(5), VC=(5,35,40,101,103);
END;
YVR VI=(104), VV=(103,101,0), VS=(202,202,0), IK=(0,0,0),
    FK=(0.,0.,0.), VO=(35);
END;
```

The sectors MDA (met data), YDA (yield data), and ZDX (index cards) are generally zeroed out when their information come from another source. For example:

```
ZDX SR=(1:6,1:8), NL=(0), LV=(0), UW=(0), LA=(0), UP=(0), LP=(0);
```

However, if values are inserted they are done so one entry at a time using subrange pair 1 like sector ZON and others.

5.4 DEFINE SUBFUNCTION DATA (DAT)

The define data function serves two basic purposes. It reserves an appropriate amount of space on the data base for a praticular data set and it also sets initial values of that data set.

The record space is reserved by two means. The Control Record is updated to indicate the use of the amount of space designated in the appropriate descriptor (this will be described more with the appropriate sectors). Each data block is also given a block ID# and pointers to indicate the position on the data record of each succeeding block. Block ID# are years in the case of monthly data and year-month concatenations in the case of daily data (i.e. 1931 or 193610). The pointer on the first data block of a data set consists of a record pointer and a displacement pointer indicating the record position of the start of the next data block in the data set. The last block will have pointers of -1 and 0 respectively. The block ID and pointers will occupy the first three bytes of each data block and descriptors should be built in accordance with this.

The card set up for defining a data set begins as before with a function card

```
DEF DAT DA1 DD=(USA),FD=(UNITSTAT),PW=(PHILLIPS),CO=(3),SI=(491),
            NC=(4),SC=(BLK,DTA,DAY,REC);
        END;
```

The descriptor indicated in the DA1 sector is to be used in formulating the set up of each data block on this data set and determining allocation of space on the data base.

The first sector to be processed is the BLK sector. This contains information on which variable codes in the descriptor are to be given initial value. There is no specific structure associated with the set up of the data block as such since each descriptor would impose different requirements; however, a structure (DHOLD) was designed to hold the various parameters needed to set up each block and data set. A set of BLK cards would be similar to the following

```
BLK SR=(1:1,1:12),BC=(35),NE=(12),IV=(-9999);
BLK SR=(2:2,1:12),BC=(5),NE=(12),IV=(-9999);
BLK SR=(3:3,1:12),BC=(40),NE=(12),IV=(-9999);
BLK SR=(4:4),BC=(101),NE=(4),IV=(-9999,-9999,-9999,-9999);
BLK SR=(5:5,1:4),BC=(102),NE=(4),IV=(-9999);
BLK SR=(6:6,1:4),BC=(103),NE=(4),IV=(-9999);
END:
```

The number of BLK cards should correspond to the number of variables in the applicable descriptor minus the two for the block ID# and the pointers.

The variable code BC= parameter is the same as the variable code on the descriptor. These BLK cards need not be in the same order as the variable codes on the descriptor, but this is probably adviseable for ease of interpretation. The number of elements for each variable should also correspond to the descriptor. The initial value would generally be -9999 the same value which indicates missing data in a completed data block.

The DTA sector is next. The purpose of this sector is to indicate the directory element to which the data set will correspond and ascertain that the given descriptor is indeed associated with that directory element. Also a check is made to make sure no data set has previously been assigned to this directory element and descriptor. Thus DTA cards would appear as follows:

```
DTA NL=(4),LV=(3,1,8,60),TL=( ),TH=( );
DTA NL=(4),LV=(3,1,8,20),TL=( ),TH=( );
END;
```

Since only initial values are to be placed on the data base, any number of directory elements associated with a particular descriptor can be defined at once provided they are all on the same country's directory.

There must be as many DAY sector cards as DTA sector cards and the two sets must be in order so they correspond one to one. The processing of the sector DAY cards causes the space for each data set to be allocated on the data base, the control record to be updated to indicate this, the pointers block IDs and initial values to be arranged to form each data block, the blocks to be arranged into data records, and the data records to be written to the data base itself. DAY sector cards would appear as follows.

```
DAY TI=(3010860),TD=(491),BS=(128),TB=(50),TU=(0),
    TN=( ),TP=( ),TL=( ),TH=( ),TO=( ),TZ=(0),TF=(1931),
    TX=( );
END;
```

The DATID parameter should be a concatenation of the level numbers of the directory element to which this data set is to correspond. The DATDESID, DATBLKSZ and DATBKALC parameters should be the same as the ID's, block size and total number of blocks specified in the descriptor to be used. DATBKUSE is initialized to zero and upon completion of the define run will indicate how many of the blocks allocated by the descriptor have been set up with ID#s, pointers and initial values on the data base. Blocks are not set up for those which would have block IDs greater than the current date (year for monthly data, year-month for daily data).

There is no subrange parameter for this function and sector. The data set ID and related parameters are simply placed in the first available position of the next available data record.
To be available, the record must have at least enough free space to accomodate one third of the data blocks.
Then the next unused position of the data location array in sector DAY is found and the subscript of this location is placed in the DATPOINT parameter. Parameters DATLOCAT and DATDISPL represent the record number and location on the record of the first byte of this data set. DATNOREC contains the number of records to be used by this data set. DATRBALC and DATRBUSE are the number of the allocated and used data blocks from the data set which are on the data record given in DATLOCAT. The DHTFBKID parameter contains the block ID of the first block on each data record at the position of DATLOCAT and DATDISPL. Succeeding block IDs are then produced as increments from this value. If another record is needed to handle the full number of data blocks allocated then another record with free space is found and another position in the DATLOCAT array is found. This array position is placed in the previous DATNXARR position. Then the free space on the old record is updated. Finally, the corresponding new DATLOCAT, DATDISPL, DATRBALC, DATRBUSE, DATFBKID, parameters are set. If another new record is needed the process begins again. If no more records are needed, the final DATNXARR is set to -1. An error will result if the defining chain of records exceeds five as this will exceed the capacity of arrays needed to process these positions.

When the REC sector card is processed all data records just used are set to RECTYPE of 4 and RECLOCAT and RECSPACE updating is checked. Thus the format of this card is again

```
REC RT=( ),RC=( ),RS=( );
END;
```

This must be the last sector processed for Define Data.

6. DATA BASE MAINTAINANCE

As new information becomes available it will become necessary to add to, replace or in some cases even delete old information from the data base. Seperate functions have been set up for each of these processes for each of the five subfunctions (record types). In many cases not all functions apply to a subfunction, or use of some functions is very restricted, therefore each function-subfunction combination will be described seperately.

6.1 ADD CONTROL

In the case of the control record, only passwords in sector SEC may be added. To do this the subrange where the new passwords are to be added and the total number of passwords after the addition must be supplied. These are mandatory and the only allowed parameters in the only allowed sector so a complete ADD CTL run would look like the following

```
ADD CTL CL2 DD=(USA), FILEID=(UNITSTAT),PW=(LEDUC),
            NC=(1),SC=(SEC);
        END;
        SEC SR=(4:6),PW=(CCEA,THOMAS,HANSEN),NP=(6);
        END;
```

6.2 REPLACE CONTROL

If for any reason a parameter in the control record should become invalid the replace function can be used to return it to a correct value. The replace function can also be used to replace one password with another without having to seperately delete the old then add the new password. Some samples of replacing control information are given below.

```
RPL CTL CL2 DD=(USA),ID=(UNITSTAT),PW=(CCEA),NC=(5),
            SC=(SEC,DTY,DSY,DAY,REC);
        END;
        SEC ID=(US OF A),NS=(10),NF=(12);
        SEC SR=(4:4),PW=(JOHNNY);
        END;
```

```
        DTY SR=(1:1),DC=(UNSTAM);
        END;
        DSY SR=(14:14),SI=(481);
        END;
        DAY SR=(52:52),SI=(83);
        END;
```

6.3 DELETE CONTROL

If some values of a subscript parameter are no longer needed then they can be set to their initial values by the delete function. Some single valued parameters can also be deleted but this is less likely to be necessary. Descriptors cannot be deleted directly but only by removing the control information for that descriptor. Some examples of deleting control information would then be.

```
DEL CTL CL2 DD=(USA),ID=(US  OF A), PW=(HANSEN),NC=(4),
            SC=(SEC,DSY,DFY,REC);
        END;
        DSY SR=(12:12),SI=(479);
        END;
        SEC SR=(4:4),PW=(CCEA),NP=(5);
        END;
        DFY SR=(25:25),FI=(265);
        END;
        REC SR=(77:77),RT=(3),RC=(77),RS=(0);
        END;
```

6.4 ADD DIRECTORY

Adding to a directory is much like defining one. What is added are further directory elements. These, though they are added using consecutive subranges starting at the end of the previously defined list of directory elements, can be parts (higher level partitions) of areas already on the data base, or they can be of

entirely new areas. Thus the directory of the U.S. described previously stopped at element 103. Element 104 could be a new subdivision of say KANSAS whose previous elements were in the teens, or it could start a whole new area not even in the Great Plains region. Since these are new elements, DID section cards would look the same as for the DEF function. Thus a section of ADD DIR cards would begin

```
ADD DIR DR2 DD=(USA),ID=(US OF A),PW=(PHILLIPS),CO=(3),NC=(7),
          SC=(DID,PTR,DSC,DTA,DFN,DTY,REC);
      END;
      DID SR=(104:104),NL=(2),LV=(3,2),QI=(2),     ;
          DN=(NORTH CENTRAL),NS=(0),NF=(0);
      DID SR=(105:105),NL=(3),LV=(3,2,17),RI=(17),
          DN=(ILLINOIS),NS=(0),NF=(0);
      DID SR=(106:106),NL=(4),LV=(3,2,17,10)RI=(10),
          DN=(NORTHWESTERN),NS=(0),NF=(0);
      .
      .
      .
      DID SR=(154:154),NL=(4),LV=(3,2,34,90),RI=(90),
          DN=(SOUTHEASTERN),NS=(0),NF=(0);
      END;
```

The pointer card must include the entire range of the directory in the subrange parameter, not just that of the new values since the presence of the new elements may change some of the old pointers. Thus the PTR section card for this add group would be

```
PTR SR=(1:154),PA=( ),BR=( ),CH=( );
```

This will force a re-evaluation of the PARENT, BROTHER and CHILD of every directory element.

Since these are new directory elements, the DSC, DTA and DFN sectors must be initialized. If the descriptors and model definitions for these already exist, then the parameters will have to be changed from their initial values by use of the replace function. The initialization sector cards would be as follows

```
DSC SR=(104:154),SI=(0,0,0),SL=(-1,-1,-1), SP=(0,0,0);
END;
DTA SR=(104:154,1:3),TL=(-1),TH=(0);
END;
DFN SR=(104:154,1:6),FI=(0),FL=(-1),FP=(0);
END;
```

The sectors in the control record pertaining to the directory must now be updated. Section DTY and REC would require the following cards to accomplish this.

```
DTY SR=(1:1,1:2),DI=(3),DC=(USA),DR=( ),DT=( ),DE=( );
END;
REC RT=( ),RS=( ),RC=( );
END;
```

Due to the programming which keeps track of the following parameters, no value should appear within the parenthesis for these in the ADD function

```
DIRNOREC(DR),DIRTOTDE(DT),DIRLOCAT(DL),DIRNUMDE(DE);
```

As in define the DSC, DFN and DTA cards can be any order, but must be together and after the PTR card. The DTY and REC cards must then be next in any order.

6.5 REPLACE DIRECTORY

Sometimes boudaries between geographical areas which define directory elements change. In these and other similar cases, directory ID#s and pointers must also be changed. In the case of changing directory ID#s, DID sector cards are used, identifying the element by level and level numbers rather than subrange. Any of the DID parameters to be changed in that element can then follow, but after the DID cards have been processed, the total list of levels, level numbers and directory ID#s must make a coherent pattern as they did when first defined.

In some cases when large numbers of directory ID #s are changed it may be advantageous to trace the pointer variable changes and insert them individually. Even in this case, however, a card specifying the entire directory range must be included to assure no indirect associations are missed. In almost all cases it is best to leave the sorting of pointers to the built in programming even when it is helpful to the person running the system to determine some of the changes by hand. In any case some DID and PTR replace cards follow

```
RPL DIR DR2 DP=(USA),ID=(US OF A),PW=(DAVNPORT),C0=(3),
            NC=(5),SC(DID,PTR,DSC,DFN,DTA);
      END;
      DID NL=(4),LV=(3,1,8,60),DN=(CENTRAL);
      DID NL=(4),LV=(3,1,20,10),RI=(11),NS=(1);
      END;
      PTR NL=(3),LV=(3,1,20),CH=(11),PA=(2),BR=(20);
      PTR NL=( ),LV=( ),PA=( ),BR=( ),CH=( );
      END;
```

The null arguments are needed for the NUMLEVEL and LEVELS parameters as well as PARENT, BROTHER and CHILD because all are required parameters but all elements rather than one specific element are to be evaluated and that is what this card signifies. This must be the last PTR sector card in the replace function.

When directory ID#s are changed as described above, descriptors, model definitions and data blocks must be changed as well. Also, when directory elements are added if descriptors which have already been created are to be assigned to them, this must be done through the replace function, both in directory and descriptor. These three sectors can be any order but must be after the DID and PTR sectors.

Subrange or level and level numbers can be used to specify directory elements for these sectors since making the changes would require familiarity with the old directory and thus both identifiers would probably be known. Examples of such sector cards are given below.

```
DSC SR=(1:1),NL=(4),LV=(3,1,46,60),SI=(410);
END;
DTA SR=(45:45,1:1),TH=(3220);
END;
DFN SF=(1:1),NL=(3),LV=(3,1,20),FI=(918);
END;
```

When only one subrange appears, it refers to the subrange of the parameter values and level and level numbers must be used to specify the directory element. When two subranges are present, the first specifies the directory element and the second the range of parameter values.

6.6 DELETE DIRECTORY

For the same reasons directory elements might need to be replaced, some might also be deleted. Since the only reasonable deletion is an entire element, all that needs be specified is the identity of that element. This is done with a DLE sector card which simply specifies the level and level numbers of an element to be deleted. If a deleted element has a child and other higher level related elements, these too must be deleted. Since this is

so, rather than risk error in the data base operator tracing all relationships, this is done automatically. Thus only the deletions at the lowest level which are not related need be specified explicitly. Some DLE sector cards follow

```
DEL DIR DR2 DD=(USA),ID=(US OF A),PW=(PHILLIPS),CO=(3),
            NC=(4),SC=(DLE,PTR,DTY,REC);
        END;
        DLE NL=(3),LV=(3,1,8);
        DLE NL=(4),LV=(3,1,46,90);
        DLE NL=(4),LV=(3,1,20,30);
        END;
```

The first card will result in the removal from the directory not only of element 3, but also of elements 4-9. The remaining elements will then be compressed to fill consecutive directory element positions. Thus if there were 154 element to start with filling positions 1 to 154 now there will be 145 elements in positions 1-145. This makes it necessary that whenever elements are deleted, the pointers for the entire directory be re-evaluated.

The range for this should be the new total number of directory elements; but, if this is not exactly known because of checking out higher level related elements, the original total minus the number of DLE cards used will be sufficient. Thus for the directory change previously described one could have a PTR section card such as

```
PTR SR=(1:151),PA=( ),BR=( ),CH=( );
END;
```

This, of course makes it necessary to update the control record sectors related to the directory. In sector DTY the number of record will need to be changed if enough elements are deleted and there were over 135 to start with. The total number of directory elements and number of elements per record must be

updated. If a record is dropped then its directory record location will have to be re-initialized. The subrange for these will depend on the initial not the final range to be considered. In any case the parameters used should all have null arguments, resulting in the following

```
DTY SR=(1:1,1:2),DT=( ),DE=( );
END;
```

In case a record is dropped or in any case to update the freespace on the appropriate record, sector REC of control must be processed. As with all other cases of updating this sector all three parameters are mandatory, but with null arguments.

```
REC RT=( ),RC=( ),RS=( );
END;
```

REC and DTY must be the last two sectors but between the two they can appear in either order.

6.7 REPLACE DESCRIPTOR

Of the data base updating functions replace is the only one allowed for descriptors. New descriptors are added by defining. To delete an entire descriptor is not possible as there is likely to be no occasion for such, but only the deletion of its use by a particular directory element which can be accomplished by the replace function of the directory and control subfunctions.

However, the descriptor is specifically a data descriptor and if it should be desireable to describe a slightly different data set with an existing descriptor, this can be done by modifying that descriptor by the replace function.

The three descriptor sectors can be in any order after the function card. All parameters of all sectors are optional, except in the IDS and COD sectors where the subrange parameter is needed to identify which array element is intended. If a descriptor ID # DESID is changed one must remember to change the new value in the directory and control sectors where it is to be used. A sample RPL DES run follows.

```
RPL DES DS2 DD=(USA),ID=(US OF A),PW=(LEDUC),SC=(4 91),
            NC=(3),SC=(PLM,IDS,COD);
        END;
        IDS SR=(1:1),CA=(YRBK),UI=(242);
        END;
        PLM NB=(52);
        END;
        COD SR=(4:4),OF=(5);
        END;
```

6.8 ADD DEFINITION

Sectors MOD, DFY, DFN, and REC are not applicable to this function and sectors INV and YVR follow special rules because there are no double subscripted parameters in either. Variable codes can be added to the INV sector with or without a subrange and the value entered for number of variables should be the number of variables to be added. The program adjusts the old total by this amount. Since there is no counter parameter in YVR, use a subrange to place a new value in all arrayed elements. Example for the above sectors follow;

```
ADD DFN DF2 DD=(USA).....;
        END;
        INV VC=(101,103),NV=(2);
        END;
        YVR SR=(3:3),VV=(101),VS=(202),IK=(0),FK=(0);
        END;
```

For all other sectors (ZON, WGH, HYR, RYR, XYR, TRN, XVR, ZDX, MDA, and YDA) the function ADD has a double purpose. An entire sector entry can be made or values can be added to arrayed parameters already defined. Two examples follow to show the difference:

```
ADD DFN DF2 DD=(USA)....
        END;
        ZON SR=(5:5),NL=(4),LV=(3,1,8,40),LA=(39,0).....;
        END;
        TRN SR=(5:5,10:10),NV=(1),VN=(10);
        END;
```

Note to add an entire entry (ZON) only a single subrange is required and that it should follow sequentially behind the last ZON entry defined. The TRN example represents the addition of a value to an existing entry. Notice that there are two subranges and that the counter variable (in the case NV) represents the value of the additional elements to add.

6.9 REPLACE DEFINITION

For all parameters in MOD and all non subscripted parameters in the other definition structure sectors a direct replacement with the input value is possible without a subrange. Otherwise parameter values input replace the values in an arrayed parameter designated by the subrange (single or double subscripted). Three examples follow:

```
RPL DFN DF2 DD=(USA).....;
        MOD XY=(2),NR=(4);
        END;
        ZON SR=(4:4),LA=(38.0);
        END;
        XYR SR=(1:1,1:2),NM=(2),BY=(1932,1955);
        END;
```

As noted before the MOD sector has no subranges, and the ZON sector shows a direct replacement of the fourth element in LATITUDE ARRAY. The third example (XYR) reflects that the first value of a single subscripted array NUMBER of Year Pairs is replaced by the parameter NM value and that the first two values of a double subscripted array BEGYEAR are replaced by the parameter BY values.

6.10 DELETE DEFINITION

Sector MOD has no deleteable parameters, and for sector INV and YVR which have no double subscripted elements there are special rules as there was in the ADD function. The variable codes in INV can be deleted by subrange or parameter specification. For YVR all single subscripted elements will have values deleted within the ranges specified.

All other definition sectors follow two deletion options. If a single subrange is specified, all elements are deleted for the ranges specified (entire sector entry). To delete unique values from any double subscripted element requires a double subrange input. Examples follow:

```
DEL DFN DF2 DD=(USA).....;
        END;
        INV NV=(1), VC='(103);
        END;
        ZON SR=(4:4)
        END;
        WGH SR=(4:4,3:3),WN=(1);
        END;
```

Things of note. To delete an entire entry requires no parameter input other than the subrange (ZON). To delete unique values (WGH) which have a related element counter requires a counter parameter input. This input is the actual number of values being

deleted which will be internally subtracted. The program also shifts all entries lying outside a deleted entry to avoid blank entries between actual values.

6.11 REPLACE DATA

Replace data is used under two circumstances. First it is used to replace initial data values with "good" data values. Secondly it is used to replace incorrect or missing values in already established data blocks. Two types of input are needed to accomplish this: function and sector cards and either data cards or card images on a magnetic disk. Function and sector cards control the setup of the new data blocks while data cards contain the actual replacement data values. The replace function card is similar to that for defining data.

```
RPL DAT DA1 DD=(USA),ID=(UNITSTAT),PW=(PHILLIPS),CO=(3),
            SI=(491),NC=(3),SC=(BLK,FMT,DAD);
```

The BLK sector cards perform the same purpose in replace as define, setting up the variables to be used in each data block, with the additional FM parameter indicating which of the FMT sector cards applies to the variable. The EC parameter also indicates the subelements, if there are any, which will be updated. A set of sample BLK cards for a RPL DAT run follows

```
BLK SR=(1:1),BC(61),NE=(1),EC=(0),FM=(1);
BLK SR=(2:2),BC=(35),NE=(12),FM=(3),
    EC=(701,702,703,704,705,706,707,708,709,710,711,712);
BLK SR=(3:3),BC=(5), NE=(12),FM=(3),
    EC=(701,702,703,704,705,706,707,708,709,710,711,712);
BLK SR=(4:4),BC(40),NE=(1),EC=(701),FM=(4);
BLK SR=(5:5),BC=(101),NE=(1),EC=(202),FM=(5);
BLK SR=(6:6),BC=(102),NE=(1),EC=(202),FM=(5);
BLK SR=(7:7),BC=(103),NE=(1),EC=(202),FM=(5);
END;
```

The first BLK must have the variable code for the block ID. This is so the first data card of each data block can easily be identified.

The FMT sector cards describe the format of data on the data value cards. As with BLK cards the subrange parameter values are for defining the order of the cards only. The NA parameter indicates the number of values on the data card. The LC parameter indicates the first column of each value. The SZ parameter indicates the length of each value. The AT parameter indicates the data type of each value (n=integer and F=fixed decimal).

The FM parameter of the BLK cards refers to one of these FMT cards by the subrange number. The program is set up to use FMT 1 to parse from each data cards the Block ID and Variable Code. FMT 2 is used to parse the level numbers from each data card. Starting at FMT 3 all other formats are used to parse the actual data values for the parsed variable code.

The DAT sector card provides information on how the data is to be input and what directory element it is to be associated with. The IN parameter gives the input medium (CARD or DISK). The DD parameter is the input ddname to be used i.e. SYSIN for cards, your own for disk. The BL parameter is the record length. This will be a maximum of 160 for card input and disk input (Equivalent to 2 card images). The EF indicates the end of file symbol. This occurs after the data cards for the last block to be replaced for the directory element indicated on the DAD card. Any character or combination may be used. Finally the level numbers and number of levels of the directory element for which the data is to be replaced should be included on the DAD card. These last two are the only mandatory parameters. The others are optional. If they do not occur on the first DAD card the default values are
IN=(CARD),DD=(SYSIN),BL=(80),EF=(STOP).

Cards containing the data values to be replaced in the data set for each directory element follow the DAD sector card for that directory element. As explained in the FMT section each data card also contains the Block ID, Variable Code, and level numbers. A sample set of FMT, DAD, and data cards follow:

```
        END;
        FMT SR=(1:1),NA=(2),LC=(4,8),SZ=(4,3),AT=(N,N);
        FMT SR=(2:2),NA=(5),LC=(71,73,75,77,79),SZ=(2,2,2,2,2),AT=(N,N,N,N,N);
        FMT SR=(3:3),NA=(12),LC=(11,16,21,26,31,36,41,46,51,56,61,66),
            SZ=(5,5,5,5,5,5,5,5,5,5,5,5),AT=(N,N,N,N,N,N,N,N,N,N,N,N);
        FMT SR=(4:4),NA=(1),LC=(31),SZ=(5),AT=(F);
        FMT SR=(5:5),NA=(1),LC=(11),SZ=(10),AT=(N);
        END;
        DAD IN=(CARD),DD=(SYSIN),RL=(80),EF=(/*),NL=(4),LV=(3,1,8,20),VF=(8);
1931  35   -35    -7    12    58   124   267   359   222   194   113    36   -18 3 1 B20 0
1931   5    31     5    76   115   175    75   104   245    12    10    86    63 3 1 B20 0
1931  40                          3.4                                           3 1 B20 0
1931101    7504576                                                            3 1 B20 0
1931102    6755043                                                            3 1 B20 0
1931103   47456789                                                            3 1 B20 0
1951  35   -10    -7    12    58   100   267   359   222   194   113    36   -18 3 1 B20 0
1951   5    31    10    76   115   175    75   104   245    40    10    86    63 3 1 B20 0
1951  40                          4.4                                           3 1 B20 0
1951101    8504576                                                            3 1 B20 0
1951102    9755043                                                            3 1 B20 0
1951103   60456789                                                            3 1 B20 0
/*
        DAD LV=(3,1,8,60),NL=(4);
1951  35   -35    10    12    58   124   267   359   222   200   113    36   -18 3 1 B60 0
1951   5    31     5    90   115   175    75   104   245    12    50    86    63 3 1 B60 0
1951  40                          5.4                                           3 1 B60 0
1951101    4504576                                                            3 1 B60 0
1951102    7055043                                                            3 1 B60 0
1951103   20456789                                                            3 1 B60 0
STOP
        DAD LV=(3,1,8,70),NL=(4);
1951  40                          5.4                                           3 1 B70 0
1951  35   -35    -7    24    58   124   267   359   222   200   113    36   -18 3 1 B70 0
1951   5    31     5    76    95   175    75   104   245    12    30    86    63 3 1 B70 0
1951102    5055043                                                            3 1 B70 0
1951101    7004576                                                            3 1 B70 0
1951103   60456789                                                            3 1 B70 0
STOP
        END;
    END;
END;
```

6.12 ADD DATA

The input cards and their setup are identical for add and replace data. The only difference is in the purpose and results of the add function.

The ADD function is used to place data on a data set in blocks with block IDs beyond those which were set up in the define function. This would occur when more data is avaialble than the descriptor was set up to allocate blocks for, or when data for new years or months becomes available after the date of the define run which date is the last block ID set up whether or not more blocks are allocated. ADD is also used after the blocks allocated by the descriptor are finally all set up by previous uses of the ADD function. In the first and third cases above new blocks and space for them must be allocated. This is done in increments of 10 blocks regardless of the number allocated in the descriptor. In all other specifications, however, the descriptor is followed exactly. The necessary updating of the DAY and REC sectors in the Control Record are accomplished during processing of the DAT sector card information, thus no seperate sector cards are needed for this purpose.

7. LIST FUNCTION

The user will find occassion to print out certain information on any record type. A special sector ALL has been inserted for all LIST subfunctions to allow the possibility of outputting all parameters of each sector for that subfunction in one execution.

7.1 LIST CONTROL, DESCRIPTOR, AND DEFINITION

The list function for these three subfunctions are actually the same except for the sectors to output. The following example shows the function card on ALL sector for subfunction Control

```
LST CTL CL2 DD=(USA),ID=(491),PS=(LADUC),NC=(1),SC=(ALL);
        END;
        ALL SR=(1:212,1:2);
        END;
```

All elements for all sectors will be listed. Non subscripted elements are automatically output but subscripted elements require the subrange parameter. All single and double subscripted elements within the ranges shown will be printed.

Individual sectors to be listed require the subrange and the actual elements to be listed from the sector. An example from the definition sector ZON follows:

```
ZON SR=(1:4,1:5),NL=( ),LV=( );
```

7.2 LIST DIRECTORY

There are four listing options for directory which include two individual sector options and two ALL options. Examples follow:

```
DID SR=(1:22),NL=( ),LV=( )
DID NL=(3),LV=(3,1,31),DN=( ),NS=( );
END;
ALL SR= (0:0,1:6),NL=(3),LV=(3,1,31);
ALL SR= (1:105,1:6);
```

For the first DID example the parameters input will be output for all directory entries within the requested subrange. The program internally locates the directory entry using the LV parameter of the second DID example and outputs the parameters requested for that directory entry only. The program locates the directory entry in the same manner for the first ALL option, and outputs all parameters for all sectors within the second set of subranges. For the second ALL example all parameters of all sectors for directory entries 1 through 105 will be listed within the limits of the second set of subranges.

7.3 LIST DATA

For the data that is to be listed the directory entry level has to be input on the function card along with the country and descriptor ID.

```
LST DAT DA2 CO=(3),SI=(491),LV=(3,18,20).....;
        END;
        ALL SR=(1:50,1:12);
        END;
        BLK SR=(1:50,1:12),BC=(35,5,40);
        BLK SR=(1:50),BC=(61,101,102,103),EC=(202);
        END;
```

After the proper data is located for the indicated level, the user has three options available. In all three cases the first set of subranges (1:50) indicates the actual range of data blocks to output. The second set of subranges (1:12) for the ALL option indicates that for each data block specified that all variable code subelements within this range will be output. The first BLK option is similar to the ALL option except that only those variable codes entered by parameter BC will be output. BLK option two allows output of specified subelements of desired variable codes by the use of parameter inputs BC and EC.

8. PARMFILE

The parmfile was set up to allow the program to retrieve only the sector and parameter information needed for the requested function. An entire listing is available to the user, but only examples of each record type will be shown in this guide. Record locations 0 and 1 on the file are index records used only by the program to locate the proper records needed for existing function (presently only one of the index records contains information). Record locations 2-6 contains subfunction information in the respective order of CTL, DIR, DES, DFN, DAT. Then record locations 7-63 contain section parameter information for the individual sections. Sections are grouped according to the subfunctions mentioned above. All the information from this file is used by the program, but the users familiarity with its structure is necessary to aid in errorless input.

8.1 SUBFUNCTION RECORDS

As mentioned there is one entry for each subfunction, but the discussion of CTL (figure 8.1) will be representative of this type record. There is a subscripted SECNAME entry for each sector possibly needed by the functions DEFINE, REPLACE, ADD, DELETE, and LIST.

The double subscripted entries OPT, ORDER, and MAXTIME use the function numbers (1,2,3,4,5 respectively to earlier mentioned functions) as the first subscript and SECNAME subscript as the second subscript.

8.1.1 OPT ENTRIES

An OPT entry can be 0, 1 or 2. The following table shows their respective meanings:

Entry	Meaning
0	Indicated sector is optional for indicted function
1	Indicated sector is mandatory for indicated function
2	Indicated sector is not used for indicated function

The example S.OPT(1,1)=1 indicates that the sector CL1 is mandatory for defining a CONTROL entry.

8.1.2 ORDER ENTRIES

The ORDER entries indicates the order that the mandatory and/or optional sectors for a given function-subfunction combination must appear. For a non-used sector the entry will be 0, but other entries will indicate priority with 1 being top priority of order. If there is a repeat priority, say 2, then either sector may come first after the priority 1's. An example from figure 8.1 follows for DEF CTL.

```
S.ORDER(1,1)=1
S.ORDER(1,2)=0
S.ORDER(1,3)=2
S.ORDER(1,4)=2
```

Sector CL1 has priority and sector CL2 is not used. Remaining sectors (SEC, DFC....) follow in any order.

8.1.3 MAXTIME ENTRIES

The MAXTIME entries designate the actual number of times that a mandatory and/or optional sector for a given function-subfunction combination can be repeated. For example, S.MAXTIME(2,7)=200 indicates that the sector DFY can be repeated up to 200 times for a REPLACE CONTROL execution.

8.2 SECTION RECORDS

The parameters input by any one section follow certain rules for which the user should be aware of. Figure 8.2 shows the parmfile

section entry for CTL REC. There is a subscripted PARM entry for each parameter name and a PABV entry for the related parameter name abreviation.

The PLEN and TYPEFLAG entries are not of concern to the users, but are only for program option checks. The MAXNO and MINNO entries relate to each individual parameter the maximum and minimum number of values allowed. The OPTN is similar to the OPT entries of the subfunction records except that the entries 0, 1, and 2 relate to the optional, mandatory and unused parameters of the given section for each function. As mentioned before the order of input of parameters on the sector card is open because the program internally checks for all mandatory parameters and stores all values in a structured format.

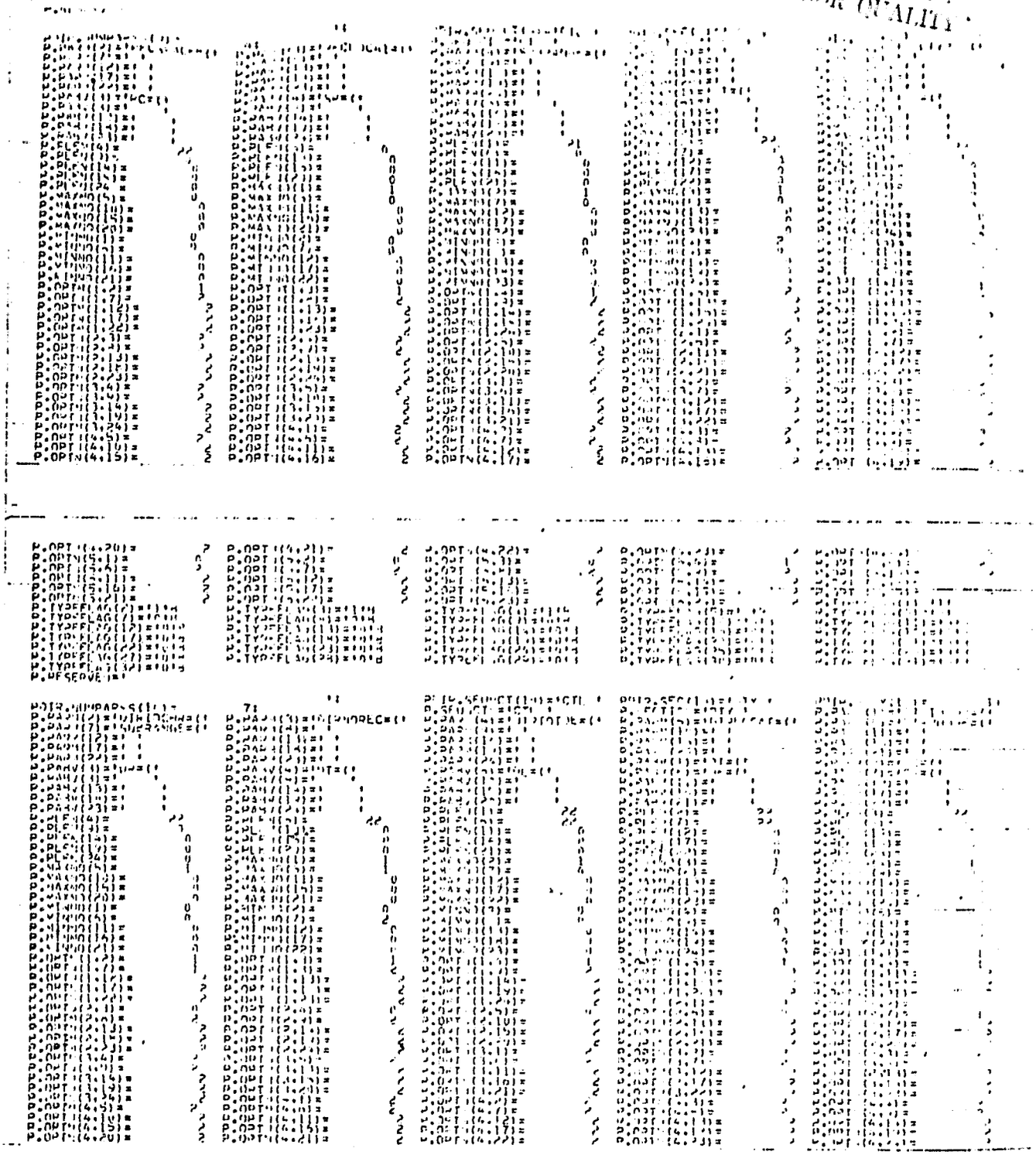
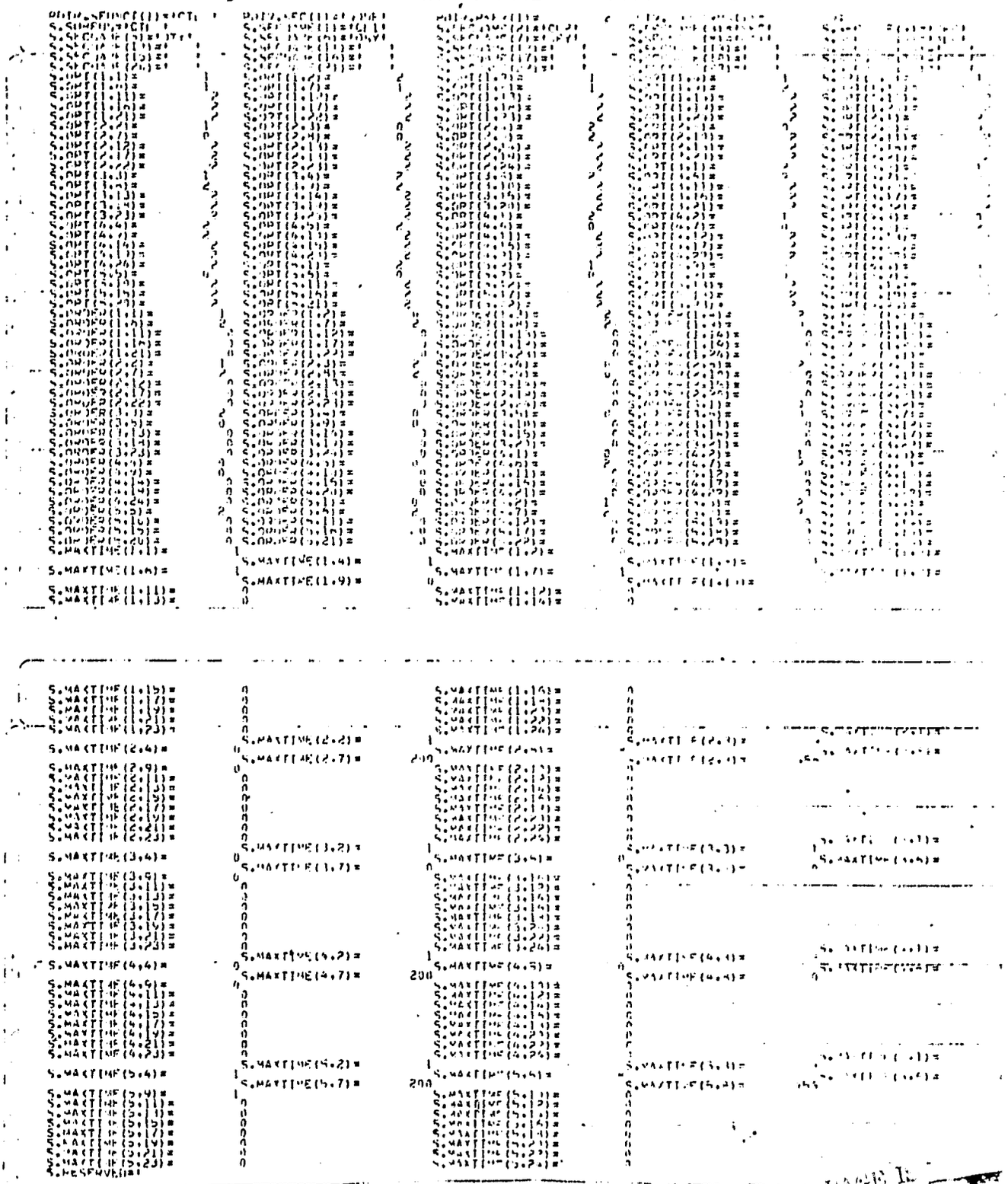


Figure 8.2 Parmfile Control REC Section

Figure 8.1 Parmfile Control Subfunction Record.



9. Error File

The error file is not user controlled but is a file containing all I/O errors that the program may recognize. The user should realize that most errors are considered fatal to avoid destruction of data files. For any one function, the program will detect all parameter format errors if there are no other type errors.

10. PROGRAM EFFICIENCY

The entire program package requires a large core size during execution, but if clock time is not important there is more flexibility by not overlaying the programs. Full program advantages follow:

1. Functions can be run back to back on the same data file in the same execution deck. On each function card the DDNAME value should match the DDNAME on the file assignment card.
2. Any number of data files can be input in one execution deck with advantage 1 applying to each file. Each file will require an assignment card.

Depending on time, number of changes, etc., it is up to the user to decide on method of execution.