SGP MEMO NO. 3A

ONE PERSON'S VIEW OF THE MAP DATA FORMAT PROBLEM

Version 2

Jim Torson

November 13, 1979

1. INTRODUCTION

On several occasions I have made the statement that I thought that the current effort on definition of the map data base format should start with the IMPS data format and proceed to improve on that by making [illegible]itions and changes. One reason for this was that the IMPS format is [illegible] result of a great deal of effort by a number of people. My hope was that the current effort could be directed towards improving upon the IMPS format rather than re-inventing the ideas that contributed to it. This seemed like a practical way to proceed since the IMPS format is fairly well defined in a document. However, as far as I can tell, for the most part people have basically started from scratch rather than building upon the work that was done a year and a half ago. I now suspect that one major reason for this is that the IMPS document falls far short of what is needed in such a document. Although the IMPS document describes the format that was finally arrived at, it contains no discussion of the advantages and disadvatages of the various alternatives that were considered. Thus, when some new people became involved, they had no choice but to repeat the work that led to the IMPS format.

I am firmly convinced that anything as complex as the map data base format cannot be defined by a committee. It is impossible to please everyone because different people often have conflicting ideas and priorities. (Also, careful thought sometimes shows that the various ideas of a single person are sometimes conflicting.) A major reason that the IMPS format finally got defined and implemented was that although we listened to and considered the ideas of a lot of people, the final decisions were made by a very small group of people.

Thus, in this document I will attempt to summarize all the pros and cons of the various alternatives that have been discussed. I am sure that some peoples' ideas will be omitted or distorted. That is precisely why it is important to write it all down. Also, I will state how I would decide each of the issues that are discussed. This is just my own personal opinion. It is in no way intended to represent the consensous of the various people who have participated in the discussions. Perhaps the person who makes the final decisions concerning this new data format will find it useful to work from this document by deleting my decisions and inserting his own. Perhaps not.

I have tried to organize this document by starting with the user's view of things and then proceeding to the implementation details. However, the various issues are of course interconnected. It is often difficult to discuss one issue without considering some others. Thus, the person who makes the final decisions will need to consider all of the issues together and arrive at a set of decisions that represent a coherent overall design.

2. SCOPE OF THE DATA FORMAT

The data base considered here is intended to handle VLA radio maps and their associated identifying information. This includes both two dimensional data arrays which are continuum maps and sets of two dimensional data arrays which are spectral line maps. Also, the data base should be able to handle other data arrays such as gridded UV data, westerbork radio maps, digitized optical images, and even an X-ray of your gall blader. However, the problem of handling ungridded UV data is not considered here.

3. GLOSSARY

Part of the difficulty of discussing the data base issues seems to be due to a lack of well defined terms. Thus, let me start with a few definitions of words that I will use. Some of them will hopefully be further clarified in the rest of the document.

MAP - a gridded data array together with its identifying information. The data array is considered to be three dimensional, with one of the dimensions representing frequency channels for spectral line maps. Continuum maps or things like optical images are considered to be the degenerate case of the frequency channel dimension having a length of one.

MAP HEADER RECORD or CATALOG RECORD - the identifying information that is associated with a map.

CATALOG FILE - a file ("file" is defined below) which contains the header records that are associated with a group of maps.

MAPNAME - one of the items of identifying information that is associated with a map. This is a completely arbitrary string of characters that is specified by the user. Since this string is stored in the header record associated with a map, it has a maximum allowable length.

MAP SPECIFICATION - the mechanism by which the data base allows a unique map to be specified and referenced by programs and users. The map specification might consist of a single string value such as a mapname value, or it might consist of the combination of several values of the same or different types.

MAP CATEGORY - one of the items of information in a map header. This is a completely arbitrary string of characters (with some maximum length) that is specified by the user. Unlike the mapname, this item is not a component of the map specification.

FILE - a bunch of bits that are stored on the computer's disk and which are considered to be a single thing by the operating system.

DIRECTORY FILE - a special file which contains identifying information for a group of files.

FILENAME - one of the items of identifying information that the operating system usually associates with a file. This is usually an arbitrary string of characters that is specified by the user. It usually has a maximum allowable length.

FILE SPECIFICATION- the mechanism by which the operating system allows a file to be uniquely specified and referenced by programs and users. Different operating systems have different conventions for what a file specification consists of. For example, on the DEC-10 a file specification consists of a specification of the disk structure and user area which contain the file plus a six character filename string value plus a three character string value called an extension. A file specification on the PDP-11 is similar except that the filename string may have nine characters and there is the addition of an integer value called the version.

FUNCTION or OPERATION - one of the things that the data processing system does.

FUNCTION NAME or OPERATION NAME - a string of characters that is used to identify a function or operation.

PROGRAM or TASK - the stuff which the operating system considers to be a single executable thing.

PROGRAM NAME or TASK NAME - a string of characters that is used to identify a function or and operation.

4. THE USER'S VIEW OF THINGS

"A map by any other name is still a map. ... or is it?"

4.1. DEFINITION OF "MAP"

DISCUSSION

It seems pretty clear that a good definition of the term "map" is that it is a set of identifying information together with the gridded data which is being identified. Of course the real question is how the header record is defined and how much data is defined to be associated with each header record.

One choice would be to have a different set of header information associated with each two dimensional data array. Thus, each different frequency channel of a single spectral line observation would be considered to be a different map. This would be convenient in that there is some header information such as location of the maximum value in the X-Y plane which is unique to each different frequency channel. However, if we do a transpose of a set of spectral line channels we get a set of two dimensional data arrays where the two dimensions are X (or Y) and frequency channel. Thus, we would end up with the number of maps, and hence header records, being the number of cells in the Y (or X) direction. Thus, we can no longer have each header record holding information for one of the frequency channels. Another problem with this definition would occur when the user asks for a summary list of all the maps that he has. It is doubtful that he would want a separate item listed for every different frequency channel. (Of course he may at some time want a listing of the information specific to each frequency channel, but that is another function.) Listing each frequency channel separately would be especially obnoxious if we are having the user select the data to be processed by pointing to the desired item in a summary listing. It seems clear that the different frequency channels should be considered to be all part of the same single map.

A second way we might define things would be to group the beam and the different Stokes parameter X-Y data arrays together and call that a single map. These things can be produced together in the map making operation, and this would ensure that they stay tightly associated. However, would this mean that we always have to eat up disk space for all these things even if the user didn't want or need them all? The bookkeeping would be more complicated if we grouped all these things together but yet allowed some to not be present in some cases. Also, what about the spectral line case? Do we really want a map to be a four nsional data array?

The other alternative of course is to say that the beam is one "map," the I Stokes parameter is another "map," etc. This makes it easier to give the user control over things like the existence or non-existence of the different items and their sizes. However, it makes them less tightly coupled so there is danger that you will end up doing things like cleaning an intensity map with the wrong beam.

At first it may seem that there is little distinction between the concept of a "map" and a "file." However, it is very important to understand the distinction. A file is the tool that the operating system provides for storing and accessing data on the disk. A map is a logical set of data which should be defined according to the needs of the user. There is not necessarily a one-to-one relationship between maps and files. For example, the implementation may be such that many maps are stored in a single file. Or, there may be only one map stored in one file. Or, the storage for one map may occupy more than one file. The physical storage of maps is discussed in another section.

DECISION

It seems quite clear to me that the different frequency channels of a spectral line observation should be considered to be all part of one map. I would prefer to define the beam and the different Stokes parameters to be different maps. I like the flexibility that this gives. er means can be used to associate the corresponding intensity map and m.

4.2. PERMITTED SIZES OF DIFFERENT MAPS

DISCUSSION

There are several restrictions that might be placed on the sizes of maps in the X and Y directions. Making these restrictions would simplify the implementation of the system, but relaxing the restrictions would make a more flexible system.

DECISION

Eliminating the restrictions really isn't very hard to do if you build it into the system from the start. I think the flexibility is well worth the effort. Specifically:

Maps do not have to be square.
NX and NY do not have to be powers of 2.
NX and NY do not have to be even numbers.

The maximum size of map that will be allowed is a different question. That depends upon how you access the data, which is discussed in another section. Also, there may be a maximum permitted value of NX*NY*NCHAN if the different spectral line channels are stored in the same file. This is also discussed in another section.

4.3. PERMITTED SIZES OF DIFFERENT CHANNEL MAPS WITHIN A SPECTRAL LINE MAP

DISCUSSION

We can consider allowing the different frequency channels to cover different areas of the sky and/or have different numbers of cells or different shapes. Allowing this could significantly reduce the amount of disk storage that is needed. However, implementing this capability would of course take more work. I haven't thought about it enough to know how much harder it would be, but I have a gut feeling that it would be quite a bit harder to implement. Also, the user interface would be harder. For example, how would the user specify the sizes and/or locations of each of the different frequency channels?

DECISION

I'm for taking the lazy way out on this by saying that all the different channels must be the same size and cover the same area of the sky. The complexity of implementing anything else sounds like a mess. Besides, it seems that you usually wouldn't save more than half the disk storage.

Of course the resolution depends upon the observing frequency. Is this effect important for the difference in frequency that occurs over the different frequency channels of a spectral line observation? Can we safely assume that this is all compensated for before the maps get anywhere near our map data base? This probably needs more thought.

It has been suggested that we could now put appropriate information in the map header to allow for the necessary bookkeeping. The plan would be to implement simple data access routines now which assume that all channels are the same size and cover the same area of the sky. Then, at some later time we could do the extra work of implementing the more general access routines to allow saving disk space. I haven't thought about it enough to be sure, but it seems that the user would need to know how things are being done. Thus, if you can't hide this design decision from the user, then you probably can't hide it from the application programs. Changing to the more general scheme later would not be a simple matter of changing the low-level data access routines. However, perhaps some very clever designing could make this work.

4.4. MAP SPECIFICATION

DISCUSSION

At any given time there will be many maps stored in the data base.

An operation that accesses map data must thus be able to figure out which map or maps are to be operated upon. This amounts to taking one or more values that are the components of the map specification and finding the header record (and thus the map data) which corresponds to the given specification. There are endless possibilities for defining the components of the map specification and how they are used to find the desired header record. The basic idea though is that we browse through the header records until we find one that has values in certain fields that are a satisfactory match with the values in the components of the map specification. Note that in this section we are only talking about the question of what values are used as components of the map specification. Given that we have decided upon a set of values, there are a variety of ways that the user can specify these values when he is using the system. The actual way that the user enters values for these components will be discussed in the next section.

For the sake of discussion, I will assume that one of the items of information that is stored in a map header is a string of characters called a mapname. This is an arbitrary string (of some maximum length) whose value can be specified by the user when the map is created or put into the data base. Note that there may seem to be little distinction between a mapname and a filename or between a map specification and a file specification. However, it is important to understand that these things may not be directly related. Filenames and file specifications are the tools provided by the operating system for selecting globs of data on the disk. Mapnames and map specifications should be defined according to the needs of the user. The restrictions on filenames and file specifications may or may not allow them to correspond exactly to mapnames and map specifications. This is discussed further in another section.

There are currently four components of a map specification on the DEC-10. The first is a mapname string value which is limited to six characters. This is a more or less arbitrary string of characters of the user's choosing. The second component of the map specification is an indication of the type of the map. This must be either map or beam; no other values are permitted. The third component is an indication of the Stokes parameter. This must be either I, Q, U or V. The fourth component is the band specification. This must be either 20cm, 6cm, 2cm, or 1.3cm. A map is uniquely specified by the combination of these four values. Thus, there may be more than one map associated with a given value of any one of these components. (Actually, there is another component, the user area, but we will ignore that for now.) Typically, a user will have several maps with the same mapname, e.g., "NGC40". Each different map may have a different Stokes specification. This is convenient because it helps to group together the different Stokes parameter maps for a given observation. Similarly, the scheme can be used to group together observations of the same source taken at different bands. However, since there is no distinction between a dirty map and a clean map, there is no way to associate a clean map with the dirty map from whence it came. The two are required to have different mapnames.

Of course a user may adopt his own private naming convention to help him remember which clean map corresponds to which dirty map. Another problem with this scheme is that it cannot nicely handle other kinds of data such as spectral index maps or polarization maps. The only way it will handle these is to enter them into the system with phony band and/or Stokes parameter specifications which don't really apply to the maps.

On the IMPS system a somewhat different scheme is used. There is only one component of the map specification: a 12-character mapname. By definition, only one map may have a given mapname value in its header. (Actually, the user area is another component of the unique map specification as with the DEC-10 scheme, but we will again ignore this for now.) The user can get a display of some of the other items of information in the map headers such as band and Stokes parameter, but these values do not participate in the map specification process. It is entirely up to the user to keep track of any needed associations between different maps. Of course the display of some of the header values in addition to the mapname helps him to remember what's what.

Another scheme that has been suggested is to have a map be uniquely specified by a number. This number would be chosen by the user when the map is produced or entered into the data base. If this number is treated as a string of characters, then this is really just the same as the IMPS scheme with the mapname composed of characters which just happen to be digits. If the number is treated as an actual integer value, then it is more like a sequence number. This subject is discussed in another section.

Yet another scheme that has been proposed would have a map be uniquely specified by a number which is generated by the computer. One variation of this would have the number be a unique more or less random number that is generated when an output map is created. This would be unsatisfactory because a later step in a sequence of batch operations would not be able to refer to a map produced earlier in the batch sequence. Another variation of this scheme would have the number be the order of the map in a summary listing of the available maps. In this case, the specification number would not "stick" to a map. This would also cause problems with batch runs, especially if the user is adding or deleting maps interactively while the batch run is in progress.

Yet another scheme that has been suggested is to use a set of data selection parameters in much the same way that the DEC-10 programs currently allow selection of visibility data. This amounts to using a relatively large number of values from the map header as components of the map specification, e.g., source name, date, time, band, Stokes, etc. In some ways, this might be convenient for the user because he could do the specification in a variety of ways and he would usually have to specify values for a relatively small set of components. For example, if he had only one map that was made on a certain date, he could select it by just specifying the date. (The other selection parameters would have default values which meant they would match any value in the header ord.) However, an important difference between this scheme and the ers that have been discussed is that the general selection criteria would not necessarily give a unique specification. For example, if only the source name were specified, the system would select the "first" map with that source name if there were more than one.

Yet another scheme that has been suggested is to use two more or less arbitrary strings to uniquely identify a map. The first string is a mapname which is used as described in the other schemes. The second string is an attempt to indicate the state of processing of the data. It is proposed that for output maps this second string will be invented by the processing operation if it has not been specified by the user. This invented string would typically be the name of the operation. (To make any sense at all, this would have to be the name of the processing operation or function. The name of the task or program would not due since a single task may carry out many different operations if they are implemented as overlays. Or, there may in some cases be an exact one-to-one relationship between operations and tasks. Or, in the other extreme, there may be an operation which is implemented as more than one task.) We would need to decide what should be done if the operation invents a string which doesn't result in a unique specification for the output map. (This problem probably isn't unique to this scheme.) In any event, since processing operations may be done in many different orders and since the user could invent his own value for the second string, a processing operation could probably rarely pick a useful default value for this second string for input maps. Thus, the user would probably most always have to explicitly specify both strings for input maps.

Another somewhat separate question is whether or not the user area should be a component of the map specification. If it isn't, then all of the maps belong to all of the users. If it is, then each user has his own set of maps which are completely separate from each other user's maps. We could then provide some protection so that a user can't easily delete somebody else's maps.

Is it possible to make sense out of all this? Perhaps not, but I will try. I think we need to consider what the desired goals of a map specification convention are. Perhaps much of the difference among the suggested conventions arises because different people want the specification convention to do different things. It seems to me that the following are the desired goals of the specification convention. These are listed in approximate order of importance, with the most important listed first.

1. It must be useable in both an interactive and a batch environment.
2. It should help the user avoid making erroneous map selections.
3. Its use should be convenient for the user.
4. Perhaps it should aid the user in keeping track of how different maps are related to each other.
5. Perhaps it should aid the processing operations in keeping

track of how different maps are related to each other.

It is certainly true that various different maps are often related to each other in a variety of ways. Should the system really keep track of these various relationships? Or, should the system just attempt to help the user keep track of these relationships? Or, should it be entirely up to the user to keep track of these things? This would certainly simplify the implementation of the system. However, I think it would be highly desirable for the system to do as much bookkeeping for the user as we can figure out how to do. Also, I think the system should try to help the user with any needed bookkeeping which is not handled automatically by the system. However, it may be that the map specification convention is not the best way to do this. Does anyone have any good ideas?

DECISION

The computer generated number schemes should be rejected because they would not work very well in a batch environment.

The user specified number scheme should be rejected because it would be error-prone and because it provides very little help in keeping track of the relationships amongst different maps. Surely it would be better to have the map specification be an arbitrary user-specified mapname string.

The non-unique identification property of the general selection parameters scheme sounds likely to confuse the user. I think it would be error-prone if the selection parameters required to specify a given map vary depending upon which maps are in the data base at a given time. We might consider defining some relatively large set of parameters that would uniquely identify a map and then requiring that the user always provide values for these items. However, the user would probably find it very obnoxious to have to specify a large set of values for a map specification. I would think that five parameters would be too many to be required.

The IMPS scheme of having a single mapname uniquely identify a map has the disadvantage of making the user think up a unique name for each map, even for two maps which are obviously closely related such as a map and the corresponding beam.

The multi-component scheme that is currently used on the DEC-1Ø seems to me to have the potential for working well. Of course the DEC-1Ø scheme in its current form is obviously inadequate. I would suggest that there be three components to the map specification:

1. mapname
2. data type
3. map type

The mapname would be an arbitrary string that can be specified by the user. The six character mapname in the current DEC-1Ø system is too ort. We need enough characters to allow the user to use the mapname r some of his own private bookkeeping without having things be too cryptic. (Of course if the user wants to be cryptic, he is free to make his names as short as he wants. He can even use mapnames that are strings of numbers if that turns him on.) The mapname should not be too long because we must leave enough character positions for a full mapname in the summary line that is generated when we list the available maps. There's just so many characters that can be displayed on a single line, so the more we use up with the mapname, the fewer that will be left for additional useful information. IMPS uses a twelve character mapname, which seems to be about the right length, but perhaps a few more or a few less would be better.

The data type would be an expansion of the map/beam specification that is used on the DEC-10. IMPS currently keeps track of this information even though it isn't a component of the map specification. The values that this item may have in the current IMPS definition are:

1=map (further defined by MAPTYP),
2=beam (further defined by MAPTYP),
3=model (further defined by MAPTYP & MODTYP),
4=clean residual (further defined by MAPTYP & MODTYP)
5=COVER map - this indicates the UV coverage. The meaning of this depends upon the value of WTTYPE in the Mapping Parameters Section. If WTTYPE=1 (natural weighting) then each value in this COVER map gives the number of visibility measurements in that cell of the UV plane. If WTTYPE=2 (uniform weighting) then each value is either 1 or 0 indicating whether or not the UV cell has any visibility measurement.
6=AMPLITUDE map - each value in this map gives the amplitude of the visibility function for a cell in the UV plane. (Note that the WTTYPE value specifies the type of weighting used in calculating this amplitude value.)
7=PHASE map - each cell gives the phase of the visibility function. This is analogous to the AMPLITUDE map.
8=GRID file - this contains basically the same information as the corresponding AMPLITUDE and PHASE maps. However, the GRID file contains the real and imaginary values in a special format. For

example, if the size of an AMPLITUDE map is NxN, then the corresponding GRID file size will be (2N)x(N/2). Each row is 2N long because it contains N pairs of numbers which are the real and imaginary parts of the complex visibility. The real part is the first number in the pair. The GRID file contains N/2 lines because it is only necessary to store half of the UV plane. One additional thing about the data format is that within each line, the left and right halves of the line are swapped, i.e., the (N/2)'th pair is stored at the beginning of the line.

9=GRID WEIGHT file - this is similar in format to the GRID file. The values in this file give the weighting that applies to the UV data in the corresponding GRID file. These weighting values are a combination of the weight values from the visibility data records, the type of weighting used in calculating the map (natural or uniform), and the type of taper used in calculating the map. The imaginary parts of these values are all zeros.

10=contour map 1-bit overlay

11=grid line or tic mark 1-bit overlay

12=phase closure data

Note that the third value specifies that the data is a model rather than a map. The intention here was to consider a cleaned map to be a model that hopefully closely approximates the real source. I have no doubt that this list of legal data type values could be improved upon. For example, the IMPS header has another item which specifies which type of model we have. Perhaps it would be better to just list the various different model types as different data type values. Another improvement would be the inclusion of a defined value which means that the data is of some weird type which is not any of the other defined values.

The map type component of the map specification would be an expansion of the Stokes parameter specification that is currently used on the DEC-10. IMPS currently keeps track of this information even thougn it isn't a component of the map specification. The values that this item may have in the current IMPS definition are:

1=I

```
                    2=Q
                    3=U
                    4=V
                    5=FORMALI
                    6=A channel only
                    7=B channel only
                    8=C channel only
                    9=D channel only
                    10=P, i.e., sqrt(Q*Q + U*U)
                    11=m, i.e., P/I (this is sometimes
                      called p)
                    12=psi, i.e., 0.5*arctan(U/Q)
                      (this is sometimes called chi)
                    13=spectral index, i.e.,
                      log(I1/I2) / log(Nu2/Nu1)
                    14=some arithmetic function of
                      one or more other maps
```

I have no doubt that this list of legal map type values could also be improved upon. For example, there should probably be a value which means that the map type is of some weird type which is not any of the other defined values. Also, perhaps there should be a value which explicitly identifies the map as being an optical image.

We might consider adding the band designation as a fourth component of the map specification. This would help in grouping the spectral index map with the different bands from whence it came. However, adding a urth component might make the scheme too cumbersome to be worth the fort.

I don't particularily favor having a second arbitrary string as a component of the map specification. It is certainly useful for the user to be able to know the name of the operation which produced each map in the data base. Perhaps this should be one of the items of information which is included in the one-line summary of the map header information that is displayed when the user is to point to the map that he wants. Perhaps it would be useful for this field to be long enough (or for the operation names to be short enough) so that the names of the last two or three operations could be indicated. However, I'm not convinced that it is really useful for this item to be a component of the map specification. But if that's what would make the users happy, so be it.

I think that the user area should also be a component of the map specification. Each user should have his own set of maps which is completely separate from all the other users' maps.

4.5. HOW THE USER SUPPLIES A MAP SPECIFICATION

DISCUSSION

In the current DEC-10 system, the four components of the map specification are entered by having the user type four separate commands: MAPNAME, STOKES, BAND, and TYPEMAP. This is consistant with the rest of the standard DEC-10 programs since it follows the convention that one command is used to specify each parameter value. (Well, actually, the MAPNAME command can also specify the user area, which is really a fifth component of the map specification.)

Another way that this could be done would be to have a single command which allows the user to type in a string of characters which specifies all the components of the map specification. we could define some syntax for the form of such a string. For example, we might say that the different component values are separated by periods. Thus, the following single command:

```
        MAPSPEC    NGC40.I.6cm.BEAM
```

would specify the same information as the following four commands in the current DEC-10 system:

```
        MAPNAME    NGC40
        STOKES     I
        BAND       6cm
        TYPEMAP    BEAM
```

We could of course define some other syntax. For example, we could say that the last three items are to be specified by typing three character codes after a period. Thus, we might have the command:

```
        MAPSPEC    NGC40.BI6
```

Of course this looks suspiciously like a DEC-10 file specification rather than a DEC-10 map specification. But in fact this is how a map specification is entered some of the time on the DEC-10, e.g., when a map is to be deleted or copied. This mixing of conventions for how a map specification is entered probably leads to some user confusion. (At least it has certainly led some people to think that a map specification is exactly the same as a file specification, which does not necessarily have to be true.)

If we adopt the convention that each component of a map specification is entered by typing a separate command, then for a given operation the number of commands involved in map specification will be the number of maps to be specified times the number of components per map specification. This might require an undesirably large number of commands. On the other hand, if we have the various components of of a map specification all entered by a single command, then the number of commands involved in map specification is just the number of maps to be specified.

If several components of the map specification are to be entered by typing a single command, then several questions are raised. Must the user always specify all the components? For example, if the user has only 6cm maps in his area, must he still always explicitly specify 6cm? If he is permitted to omit the 6cm, does he type

```
        NGC40.I..BEAM
```

or will

```
        NGC40.I.BEAM
```

do the job? Another question is whether or not the items must always be specified in the same order. Is

```
        NGC40.BEAM.6cm.I
```

just as good as

```
        NGC40.I.6cm.BEAM     ?
```

Whenever the user is typing strings of characters to specify values which really aren't string values, there is the question of what he is required to type. Perhaps it would be desirable to allow the user to type only enough characters to uniquely specify the name of the desired value rather than requiring him to always spell out the name in full. For example, is

```
        NGC40.I.6.B
```

just as good as

```
        NGC40.I.6cm.BEAM     ?
```

Also, are the names of these values defined to be just upper case characters, or may the user type in any mix of upper and lower case characters?

Rather than having the system just sit there waiting for the user to type one or more commands to enter a map specification, the system could ask questions and wait for the user to answer the questions. However, for text interaction, this is contrary to the basic philosophy that seems to be generally agreed upon. (At least we agree on something.) However, it has been suggested that for map specification it might be desirable to show the user a list of available maps and allow him to somehow indicate which one is desired. Of course this wouldn't work too well in a batch environment, but it might be useful when the user is running the system interactively. This technique also would not work very well if the list of available maps is very long. Allowing the user to divide his maps up into various categories would help to keep this list short enough. (This subject is discussed in another section.) If the system has a graphics terminal, then the user can indicate the desired map by "pointing" to it in the generated list. This is the way IMPS works. If the system only has a text terminal, then the items in the list might be numbered and the user could indicate which map is desired by typing in its number.

IMPS actually uses a mix of techniques for allowing the user to enter map specifications. In addition to pointing to the desired map in a list, the user may interrupt the list generation by typing a key on the keyboard. He then has the opportunity to type a string of characters which gives the map specification. Currently this is just the twelve character mapname. For output maps the desired map can't be pointed to because it doesn't exist yet. Thus, the user is just asked to type in the desired mapname for the output map.

If we are using a multi-component map specification and it is being used to allow the computer to keep track of some of the relationships among maps, then we need to consider the question of automatic computer generation of default values for components that are not specified by the user. One way that we could handle this would be to say that any component that is not specified by the user has a certain default value and that this value is always the same. For example, assume that the components of a map specification are mapname, data type and map type as described in the previous section. We might make the definition that the default map type is I in all situations. This would certainly be easy to explain to the user and easy for him to remember. However, there may be an advantage to having the default value vary to suit the situation. For example, consider the operation which produces a polarization map (map type = P). There are two input maps and one output map. For text oriented interaction with the user, the user might need to specify only a single mapname and a single data type. The desired mapnames for all three maps would be assumed to be the same as the given mapname. Similarily, the data types for all three maps would be assumed to be the given data type. The map types would default to Q and U for the two input maps and it would default to P for the output map. Thus, instead of specifying nine items of information (three components for each of three map specifications), the user would only need to specify two items of information. Of course the user should be able to specify all nine items of information if he really wants to. We probably should allow the user to request that this calculation (P = sqrt(Q*Q + U*U)) be carried out for two input I maps instead of an input Q map and U map. However, should we still assume that the default output map type is P? Or should we default to "some other weird thing" in this case? If the user explicitly specifies the output map type, should we allow him to specify any of the defined map type values? Note that if we allow these sorts of things to be components of the map specifiction and if we allow the user to supply values for these things which override the default computer generated values, then we are really allowing the user to have control over more than just the identification of maps for purposes of selecting which map is to be processed. We are also giving him control over some of the header information which determines how some of the processing operations interpret the data.

If the user area is to be a component of the map specification, how should this value be entered by the user? On the DEC-10 the user area can optionally be specified by the MAPNAME command by enclosing the desired project-programmer number within square brackets after the six character mapname value. In IMPS, the user is asked to type in his user number (which is the same as his DEC-10 programmer number) when he starts up the system. IMPS contains a function which allows the user to change the user area to which he is "connected." When he is connected to some other user's area, he may look at the maps, but he is not permitted to produce any new output maps or to delete any maps. Thus, this scheme does not allow the user to do such things as produce a spectral index map from a 6cm map in his own area and a 20cm map in some other area.

Another question concerning how the user enters a map specification involves the use of "wild-card" specifications. Many of the data selection commands in the DEC-10 pre-synthesis programs allow the specification of an asterisk (*) as the data selection value. This value is defined to mean "all." For example, the command

```
        SOURCES *
```

means that all sources are to be selected. Should such a convention be permitted for map specifications? The answer to this question really involves more than just the map specification. It also involves questions of how the user controls the invocation of processing operations and how we code the implementation of these processing operations. For example, consider the clean operation which requires the specification of the input map to be cleaned. If a map specification has the three components described above, mapname, data type and map type, we might allow one or more of these values to be *, meaning "all." Thus, if mapname = *, data type = map, and map type = I, then we are specifying that ALL of the dirty I maps are to be cleaned. This might involve several separate executions of the clean operation, each operating on a different input map. This would mean that the clean operation would have to be coded so that it understands that a * specification for one or more of the components of a map specification means that it is to do some internal looping. (I vaguely recall that this sort of internal looping was a design feature of CANDID.)

If we do not allow the "wild-card" value in a map specification, then each operation just needs to be coded so that it does its job on one input map or one set of input maps if multiple inputs are needed for each execution. However, in this case the user might still be able to easily ask the system to clean all dirty I maps. For example, in the post processing system, he might be able to define a POPS verb which contains a loop wherein each time around the loop we call the "clean a single map" function with a different map specification. The use of sequence numbers might help with constructing these loops and with selecting a different map each time through the loop. Sequence numbers are discussed in another section. It is not clear whether or not such a capability could be added to IMPS.

As an alternative to the "wild-card" specification meaning "all" we might consided a value which means "any one." This would select just one map, but the selection might not be unique.

DECISION

The technique of displaying a summary list of maps and allowing the user to point to the desired one has worked reasonably well in IMPS. I think its use should be retained in IMPS.

For the situation where the user is entering a map specification by typing on a keyboard, I think that the user should be allowed to omit entering values for some of the components of the map specification. In this case, the system will pick default values for unspecified components that are appropriate to the particular situation.

The user should always be allowed to enter values for all the components of the map specification, even if the values for output maps are not what the system would consider sensible. This allows the user to get himself into a mess, but I don't think this is a big problem since it should be easier for the user to let the system generate these values "properly," and thus most users will do that.

The scheme of entering the user area specification when the system is started up or when a special function is invoked limits flexibility, but seems adequate to me. However, I'm not opposed to allowing entering a user area for each map specification if the users think they really need it.

I don't think that a "wild-card" "all" or "any one" concept should be used. A possible exception to this rule might be the case of a program which gives a summary list of maps. Perhaps it would be useful to ask for a list of all I maps, or a list of all maps with a given mapname. It might also be nice to allow wild-card characters within such a mapname specification.

I am somewhat undecided on the question of whether there should be one command to specify all the components of a map specification or one command per component. However, I think I would favor one command for all the components. This would help keep the number of commands down and would probably be less confusing for the user in cases where more than one map specification needs to be made. Also, it could work in IMPS much the same way things are done now. It would certainly be nice if the user only had to type as many characters as necessary to uniquely identify the names of the component values. Also, it would be nice if the components could be specified in any order. Perhaps we could define the syntax so that any separator such as space, comma, tab, or period could be used between values. Perhaps it would make sense to require that the mapname is entered first, with the others supplied in any order.

4.6. USE OF SEQUENCE NUMBERS

DISCUSSION

There may be situations where the map specification scheme described above becomes very cumbersome. For example, consider the case where the user has made snapshot observations of fifty different locations on the sky just to see if anything is there. If he just wants to turn the DICOMED loose on making fifty pictures, it would be pretty obnoxious to have to type in fifty sequences of commands to initiate the DICOMED output function for fifty different maps. Perhaps it would be nice to be able to somehow number the various maps with sequence numbers and then tell the system to do something with the maps which have sequence numbers 1 through 50, or 10 through 20, or whatever, without having to give the explicit map specifications for each of these maps.

It's not at all clear how this would fit in with the map specification scheme. Perhaps each map could be assigned a sequence number, either automatically by the system or explicitly by the user. Then, we could consider the sequence number value (an integer value) to be an alternate way to uniquely specify a map.

Perhaps the same effect could be had without complicating the map specification convention. For example, we could teach POPS to deal with an array whose values are map specifications rather than just simple numbers. If we had a convenient way to fill in the values of this array, could then just pick map specifications out of the array within a POPS p.

DECISION

I'm not at all sure how necessary this feature is. And, if it is necessary, I'm not sure how it could best be implemented. Here's a place where somebody could make a real improvement to the IMPS data structure definition.

4.7. MAP CATEGORY

DISCUSSION

In the IMPS system, one of the items of information in the map header is a twelve character string called the map category. This is an arbitrary string which can be set or changed by the user at any time. Although this value is not a component of the map specification, it does aid the user in entering the desired map specification. While the user is running the system, he may set a twelve character string value called the current map selection category (perhaps a poor choice of terms). Then, whenever the system needs a map specification from the user, it generates a list which contains only those maps whose map category matches the current selection category. (Currently, the list also includes maps which have their map category set to the null string.) If the current map selection category is set to the null string then all the maps will be listed. If the user interrupts the generation of the list by typing a key on the keyboard, he is given the opportunity of typing in the mapname of the desired map rather than pointing to an item in the list. In this case the map with the specified mapname is selected even if its map category does not match the current map selection category.

DECISION

In systems where the user enters a map specification by pointing to an item in a list of available maps, it is very useful for the user to be able to restrict the list generation to those maps of current interest. The user would find it very obnoxious to have to wait for the system to generate several pages of listing before getting to the desired map.

Thus, I think that the use of a map category should be part of the data base design. However, a map's category value should not be a component of the map specification.

4.8. SPECIFICATION OF THE SUBSECTION OF A MAP TO BE PROCESSED

DISCUSSION

In some cases an operation is to use only part of the data in a map rather than the entire map data array. For example, you must specify which frequency channel of a spectral line map is to be loaded into the image display system. In some cases it will be necessary to specify subsections of the data in each of the three dimensions.

One way that this could be handled would be to have a "data to be processed specification" rather than a "map specification." This would involve adding some sort of three dimensional bounds specification to the map specification that has been described above. However, as with allowing "wild-card" components in the map specification, allowing a subsection specification as a part of the map specification would effect the way we code the implementation of operations. (I have a vague recollection that that is one feature that was in the CANDID design.)

Another way to handle subsection specification would be to consider it to be part of the parameters to a processing operation. That is, instead of having a map specification together with a subsection specification be a single complex parameter to an operation, we would break it up into several simpler parameters. Thus, those operations which process an entire map would just have a map specification as a parameter. Those operations which process a subsection of a map would have some additional parameters which specify the desired subsection.

DECISION

I think that the subsection specification should definitely be considered to be part of the parameters to an operation rather than part of the map specification. This simplifies the coding of the operations which operate on entire maps. However, it also allows you to implement operations which have complex subsection requirements. You just end up with as many parameters as are needed to specify the desired complexity.

5. BOOKKEEPING IMPLEMENTATION - HEADER RECORDS

5.1. LOCATION OF THE HEADER RECORDS - CATALOG FILES

DISCUSSION

Now we finally begin considering how to actually implement the map data base. The basic problem is that we have to store and retrieve multiple maps, where a map is a two or three dimensional data array together with a set of header information.

First we note that the basic tools that we have available for implementing the map data base consist of the disk files and their associated file specification conventions and access mechanisms. These tools are provided by the operating system. (This of course assumes that we all agree that we should use the operating system that is available for the computer.) The basic tools provided by the operating system may or may not be modified by the Fortran compiler and run-time system. This modification may result in either extensions to or restrictions on the basic capabilities provided by the operating system.

One way to implement the map data base would be to have each map stored in a separate disk file. This is the way things are currently done on the DEC-10. The header record is stored in the beginning of the file. This is followed by the actual map data. This approach offers the advantage that each map is nicely self-contained in exactly one disk file. Thus, the standard file manipulation utility programs in the system can be directly used for some map manipulation operations, e.g., copying maps or deleting maps.

In the DEC-10 map data base, there is a very close relationship between map specifications and file specifications. The user area of the map specification corresponds exactly to the UFD (user file directory) in which files are contained. The mapname corresponds exactly to the filename. The Stokes, band, and typemap components of the map specification are encoded as character codes in the file extension component of the file specification. This offers the advantage that the standard system utility program which lists a summary of available files can also be used for listing a summary of available maps. Of course since the Stokes, band and typemap values of a map are encoded in the three character file extension, we end up with a somewhat cryptic listing.

A disadvantage of this scheme is that the mapname is limited to a length of six characters since the operating system limits a filename to six characters. Another disadvantage of this scheme occurs when we want to make a summary listing of the available maps which contains more information than that encoded in the filename and file extension. In this case we must implement a function which opens each file and reads in the header information. This could be a problem if we want to allow the user to enter a map specification by pointing to the desired item in a nerated list. The time needed to open and read a separate file for ch map could make the list generation so slow that it would not be acceptable from the user's point of view.

Another basic disadvantage of this scheme is that it is so dependent upon the tools provided by a specific operating system. Transporting the data base design to another computer with a different operating system might be difficult or impossible. For example, the RSX11-M operating system on the PDP-11 provides a file system which is very similar to that on the DEC-10. However, due to a restriction on project-programmer numbers (which are called group and member numbers on the PDP-11), dividing the maps up into different user areas cannot be handled the same way on the PDP-11.

The implementation approach currently used in the IMPS system is an attempt to provide more flexibility and to also provide a system which can be more easily transported to some other computer or operating system. In this implementation, the data for each map is stored in a separate file. However, the header records are not stored in the same file as the map data. Instead, all of the header records for all of the maps which belong to a single user are stored in a single file called a catalog file. The filename of this catalog file is a string of characters which looks like the user number. The extension (which is called the file type in PDP-11 terminology) for this file is always ".CAT". The filenames for the files which contain the map data are unique strings of characters which are generated by a subroutine. The er has no control over these file names. (In fact, the user has no eed to even know of their existence.) The extension for map data files is always ".MP". (".MAP" is not used because that extension is used for memory allocation listings that are output by the task builder.) One of the items of information contained in the header record (also called catalog record) is the filename of the file which contains the actual map data.

The IMPS scheme has the advantage that the map specification convention is totally separate from the file specification convention that is required by the operating system. Also, in order to produce a summary list of the available maps, it is only necessary to open and read one file, i.e., the catalog file. This allows the summary to be generated much more quickly than would be the case if many different files had to be opened and read. Also, each user has a set of maps which are completely separate from the maps belonging to other users even though all the map files and all the catalog files are stored in the same UFD (user file directory) on the disk.

It has been suggested that we could store a copy of the header record in each map file in addition to the catalog file. An advantage to this would be that each map file would again be totally self-contained. If we have a map file it would be possible to determine which map file it is. If the pointer from the record in the catalog file got destroyed or if the entire catalog file got destroyed, then we would be able to recover. It has been suggested that having a duplicate of the catalog record in the map file would simplify debugging. If there are aspects of this other than what was just mentioned, I don't know what they are. It has also been suggested that it is simpler to lock files (restrict use of a file to one person at a time) if the header and map data are in the same file. This subject is discussed in another section.

A disadvantage of duplicating the catalog record in the map file is that there would be a possibility for the two copies of the catalog record to get updated differently, or perhaps the updating of one copy would get forgotten. Thus, any application program which updated the catalog record or which produced a new catalog record would have to be sure to update both copies properly. This might be a problem because there may be times when a catalog record exists in the catalog file, but the corresponding map file does not yet exist. It seems like there is really little need for the catalog record in the map file for recovery purposes. In a year and a half of running IMPS, there has not been a single case of a catalog file getting destroyed. If hardware or operating system software failure were to destroy a few files, it would be much more likely that the map files themselves would get destroyed since there would be at least ten times as many map files as catalog files. If a failure destroyed many files, then you probably would want to just wipe the disk clean and start over anyway.

It is interesting to note that there is a great deal of similarity between the file systems provided by many operating systems and the map data base system that we are building on top of the file system. For example, on the DEC-10 there is a special file called a directory file which is analogous to our catalog file. Each user has his own directory file. Each record in a directory file contains identifying information for one of the user's data files. It also contains a pointer to the actual data file which is used to access the data. When the user asks the system to give him a summary list of files in his area, what actually happens is that a program reads the directory file and types out a summary of its contents. On the DEC-10 the header information for the data files is split up between the directory file and the data files. The records in the directory file actually contain only the filename and extension for the file. Other information, such as the date and time the file was written, is stored with the data file. When the user asks for a list of his files, he may ask for a "fast" listing which gives only the filenames and extensions for the files. This is fast because it is only necessary to read the directory file. A full listing giving dates and times runs much slower because each individual file must be accessed. Some people consider this splitting of the header information to be a design error in the DEC-10 file system.

DECISION

A catalog file should be used because it allows us to define the map specification conventions according to the needs of the user rather than be restricted to the file specification convention that is required by

[illegible]

the operating system. Each user area should be implemented as a separate catalog file. This helps keep the size of the catalog files down to a reasonalbe size. The catalog record should not be duplicated in the map data file. To do so would buy you very little, but would cost a fair amount in effort needed in the implementation of a large number of application functions.

5.2. HANDLING VARIABLE LENGTH HEADER INFORMATION

DISCUSSION

There are some items of header information which may be repeated a variable number of times for different maps. For example, each channel of a spectral line map has some header information associated with it, e.g., frequency of the channel. One way to handle this would be to just define a fixed size large catalog record and just fill in as many values as were needed in each particular case. However, there may be some items which may be repeated a large number of times such as information about individual clean components. This approach would require a catalog record that would be just too big.

Another approach would be to define a variable length catalog record. Each record would have several parts of variable length. In any specific catalog record, each part would only be as large as it needed to be in that case. However, this would mean that the bookkeeping would be complicated just to access the header information. This would especially be true if we wanted to change the number of items in one of the variable length parts, e.g., in the case where we wanted to go back and do another hundred clean iterations on a map.

A third approach to handling variable length header information would be to define a basic fixed length catalog record. Each variable part would be put in a separate file. Thus, there could be a file which contains the header information that is needed for the various different frequency channels. Each record in this file would contain the information pertaining to one frequency channel. The size of this file could be as large or as small as needed.

DECISION

Each of the variable length parts of a map's header information should be put in a separate file. This greatly simplifies the accessing of this information since all the records in each file are of a fixed size. Also, this helps keep the size of the main catalog record down to the point where we may be able to squeeze it into a single disk block (sector).

The current IMPS data base uses this technique. However, there is a problem. The IMPS scheme defines a file for the spectral line channel information and another file for the cleaned sources information.

However, if we are to allow cleaning of spectral line maps, it appears that we need a cleaned sources file for each channel of the map instead of just one for the entire map.

5.3. CRITERIA FOR PUTTING ITEMS IN THE HEADER RECORD

5.4. HISTORY FILES

5.5. UNITS USED FOR VALUES IN THE HEADER RECORDS

5.6. STORAGE OF NON-NUMERIC VALUES

5.7. LAYOUT OF THE HEADER RECORDS

6. MAP DATA STORAGE IMPLEMENTATION

6.1. LOCATION OF THE MAP DATA ARRAYS

6.2. PHYSICAL STORAGE OF SPECTRAL LINE CHANNEL MAPS

6.3. PHYSICAL ORDER OF MAP DATA CELLS WITHIN THE MAP DATA FILE

6.4. INDEXING OF THE MAP DATA CELLS

6.5. PACKING THE MAP DATA INTO PHYSICAL STORAGE

6.6. FORMAT OF MAP DATA CELL VALUES

6.7. INDEFINITE MAP DATA CELL VALUE

6.8. DATA BASE ACCESS ROUTINES

7. MISCELLANEOUS TOPICS

7.1. CONTROL OF DATA ACCESS BY MULTIPLE CONCURRENT PROGRAMS

7.2. DUMPING MAPS TO TAPE FOR BACKUP OR INTERCHANGE

TABLE OF CONTENTS