NATIONAL RADIO ASTRONOMY OBSERVATORY
SOCORRO, NEW MEXICO
VERY LARGE ARRAY PROGRAM

VLA COMPUTER MEMORANDUM NO. 147

TECHNICAL REPORT ON OCTOBER 1978 STATE OF THE INTERACTIVE MAP PROCESSING SYSTEM (IMPS)

J. M. Torson, R. M. Hjellming, D. L. Ehnebuske

October 1978

TABLE OF CONTENTS

NATIONAL RADIO ASTRONOMY OBSERVATORY
SOCORRO, NEW MEXICO
VERY LARGE ARRAY PROGRAM

VLA COMPUTER MEMORANDUM NO. 147

TECHNICAL REPORT ON OCTOBER 1978 STATE OF THE INTERACTIVE MAP PROCESSING SYSTEM (IMPS)

J. M. Torson, R. M. Hjellming, D. L. Ehnebuske

October 1978

1.0 INTRODUCTION

IMPS, the Interactive Map Processing System, is intended primarily to give the user the capability to interactively manipulate VLA radio map data. This includes display of the data in various ways, extraction of numerical information, and processing of the data in various ways to produce other data.

This memo is intended to describe the current state of IMPS. It is not intended to be a users manual for IMPS. Instead, it will describe the overall capabilities of the IMPS system. This will include some of the details of how the system is being implemented.

At the present time, all of the applications level software has been implemented by Jim Torson. Device drivers for the Comtal image display system and the Summagraphics data tablet have been worked on by Mike Duggan. Disk I/O routines, basic software for handling the connection between the PDP-11/40 and the DEC-10, and assistance with system software have been provided by Al Braun. Dave Ehnebuske is working on application specific data transfer between the DEC-10 and the PDP-11/40. Bob Hjellming has helped in planning how the system will look to the user and in deciding what functions will be implemented.

The listing of the IMPS software is about two inches thick (if you can measure software progress in inches). It is estimated that this is about 10,000 lines of code. Less than one-third of this is written

in assembly language. The rest is written in FORTRAN. Few additional routines will need to be written in assembly language.

2.0 GENERAL FEATURES OF IMPS

2.1 Growth of the System

The IMPS system is intended to grow and evolve with time. That is, it is being implemented in such a way that as the needs and desires of the users become more clearly defined, IMPS can be modified and extended to meet these needs. (See below for a description of how new functions are added to IMPS.)

Much of the initial implementation work on IMPS concentrated on producing packages of useful subroutines. A great deal of error checking has been built into these routines. This greatly aids in the development of the applications programs that actually implement the IMPS functions. When a new program is being tested, a coding error usually results in an error message that gives a clue as to what the problem is. This is unlike the frequent case of the FORTRAN program error which just causes a mysterious failure. A summary of the main IMPS subroutine packages is provided in the appendix.

2.2 IMPS Hardware and Operating System

IMPS runs on a Digital Equipment Corporation (DEC) PDP-11/40 with 64 k 16-bit words of core storage. Disk storage includes an RP06 disk pack which holds about 174 million bytes and two RK05 disk cartridges which hold about 2.5 million bytes each. Map images can be displayed on the Comtal image display system. The Comtal holds a single 256 x 256 image for display on a monochrome or color CRT display. Each pixel is stored in the Comtal refresh memory as an 8-bit value. Or, a pixel can be stored as a 7-bit value and the other bit can be used for a graphic overlay image.

Also included is a transfer function look-up table which contains 256 entries and a color look-up table which allows color encoding of the pixel values. Sixty-four different colors may be displayed at one time out of a possible 4096 colors. The Comtal also displays a computer controlled cursor that can be used for pointing to features in the displayed image. Interaction with the user is provided through a DEC VT-11 refreshed line drawing display, a Summagraphics data tablet and a keyboard. In addition, an ADDS alphanumeric display terminal serves as a console terminal on the system. Output devices include a Versatec electrostatic printer/plotter and a Dicomed D-47 film recorder. Connection to the DEC-10 is through a DA-28 high speed parallel interface. A teletype line can also be used as a connection to the DEC-10.

The operating system being used is RSX-11M.

2.3 Relationship Between IMPS System and Other VLA Computers

All of the IMPS applications software runs on the PDP-11/40. Map data will be obtained from the DEC-10 system and then stored on the IMPS disk. Operation of the IMPS system is then totally independent of any other computers. However, it is anticipated that in the future we will implement the capability for the IMPS user to talk to the DEC-10 system and run programs on that computer. Also, it will be possible for the IMPS user to send data to the array processor system for operations such as map cleaning.

2.4 IMPS Data Formats

The following is a brief description of the IMPS data base format. The appendix contains a detailed description of this format.

The IMPS system is of course able to handle VLA map data. However, IMPS is capable of handling any gridded two or three

dimensional data. Currently, there is some capability for reading in Westerbork maps and digitized optical images from the Kitt Peak IPPS system. The data is read from tape and converted to the proper format by programs that run in the DEC-10 system.

Each user of IMPS has a separate disk file which contains a catalog of that user's maps. Each record of the catalog file corresponds to one of the user's maps. The catalog record holds most of the header information for the map. This includes the source name and qualifier which identify the visibility data that were used in making the map. Also included is the map name, which is any unique name assigned by the user. This is used for identifying a map. Each map can also have a category specification. During an IMPS session, the user can restrict his operations to only maps that are in a given category. The catalog record also contains the information needed by the system to find the actual map data, which is stored as a separate disk file. A given user will be able to read the catalog file and map files of another user. However, he will not be able to make any modifications or deletions to those files.

The data for each separate map is stored as a separate disk file. IMPS is capable of handling spectral line maps. In this case, the different channels are considered part of the same map and are all stored in the same file. However, maps at different frequency bands are considered separate maps and are stored as separate files. The default storage format is for all the cells of the first channel map to be stored together in the file followed by all the cells of the second channel, etc. However, IMPS is set up to allow different storage formats in the future as needed.

The map cell values can be stored in four different formats: 32-bit real, 16-bit scaled integer, 8-bit pixel, or 1-bit graphic overlay. The first three are different ways to store real map

values. The IMPS routines convert values to the appropriate format as needed. For example, if a map is stored as 16-bit scaled integer values, the values are converted to 8-bit pixel values for loading into the Comtal image display. The fourth format (1-bit) is used for storing images that contain such things as grid lines and tic marks. (No graphic overlay data handling has been implemented yet.) The IMPS system recognizes a special map value called the "indefinite" value. This can be used for such things as dividing one map by another. If the divisor map cell value is zero or below some specified value, then the result of the division is considered to be undefined and the output map cell is set to the indefinite value. Indefinite values are always displayed as black. (We could easily allow the user to specify the desired gray scale level or color.) Indefinite values are also used as padding around a displayed map if it is smaller than the image display area.

IMPS can handle maps with any number of cells in either direction. The maps don't have to be square. The only restriction is that a line of the map must fit into the buffers in the programs. Currently, IMPS is set up to handle map lines which are up to 1024 elements long. When it becomes necessary to handle bigger maps, the software can be easily changed to allow for that.

In the future, the IMPS data base will include other files in addition to the catalog files and map files. These will hold header information for the individual spectral line channels, information on fitted sources, subtracted sources, and cleaned sources, and also information on the history of processing operations that have been done on the maps.

2.5 Accessing IMPS Data

A set of low level I/O routines has been implemented which

provides the capability to read and write an arbitrary length block of data at any arbitrary byte address in a disk file. These routines are implemented in assembly language but can be called from a FORTRAN program. These low level routines are called by a set of higher level routines which are the ones normally called by the applications programs to access the IMPS data base. These higher level routines are coded in FORTRAN. These routines are oriented towards reading or writing a line of a map at a time. All that the caller program needs to do is to ask for the desired line number in the desired channel number map. Determining where that data is located is automatically taken care of by the data base access routines.

2.6 User Interaction With IMPS

Instead of typing textual commands on the keyboard, most of the user control of IMPS occurs through use of three basic elements: moving the pen on the data tablet surface, closing the switch in the pen by pushing the tip of the pen down on the tablet, and typing a key (any key will do) on the keyboard. (The keyboard that is referred to here is the keyboard that sits in front of the VT-11 display. The keyboard on the console terminal is used by the IMPS user only for typing the single operating system command that starts up IMPS.) The following description of several different operations illustrates how these three elements form what can be considered a graphical control language. An important advantage of this type of control language is that it can be learned very easily by the new user.

The basic control of IMPS occurs through user selection of items out of lists of options called menus which are displayed on the VT-11 refreshed line drawing display. A multiple level tree structure of menus is used. When IMPS is started up, it displays the top level menu, which is a list of categories of

available functions. The user then points to the desired category by positioning a cursor on top of it. Movements of the pen on the data tablet causes the displayed cursor to move on the screen. The user then pushes down on the pen to indicate to IMPS that the cursor is pointing at the desired item. IMPS then intensifies the selected item on the screen and turns off the cursor. If the user is sure that that is the desired item, he then pushes down on the pen again. If he decides that he doesn't want that item, the user types a key on the keyboard (any key will do). In this case, IMPS will put the cursor back on the screen and allow the user to select another item. This illustrates a general convention in IMPS: pushing down on the pen indicates acknowledgement or a "yes" answer; typing a key on the keyboard indicates cancellation or a "no" answer. After the user selects a category of functions out of the top level menu, IMPS will display the corresponding second level menu. This may be a list of sub-categories, or it may be a list of functions. After a function has been executed, IMPS returns to the menu that initiated the function. If the user then wants to go to a different menu, he types a key on the keyboard. IMPS responds by "popping up" to the next higher level menu.

A menu display is also used when the user is to specify which map is to be loaded into the Comtal display or which map is to be the input for a processing operation. In this case, each menu item is a one line summary of identifying information for each map listed in the user's catalog file. (Actually, only those maps in the current selection category, if any, are shown.) If the user sees the desired map in the list before the list generation is complete, he may push down on the pen to terminate the list generation. Or, if he types a key on the keyboard, the list is canceled and he may then type in the map name of the desired map. Thus, he won't have to wait for the menu to be generated if

he remembers the name of the desired map. (It only takes a few seconds to fill the screen with a list of maps. Also, this can be speeded up if desired.)

If the user asks to load a map image into the Comtal display and the map is bigger than the screen size, then the user must specify what subsection of the map is to be loaded. In this case, IMPS will draw a rectangle on the VT-11 display which represents the input map. It will also draw a smaller square which represents the piece of the map to be loaded. Movements of the data tablet pen will then move the subsection square. IMPS will also dynamically display the center coordinates and the corner coordinates for the subsection square as it is being positioned. Thus, the user will be able to easily position the subsection at any desired coordinate and will also be able to visually see how that subsection relates to the entire map. The user pushes down on the pen when he has the subsection square positioned to the desired location. IMPS will then load this subsection into the Comtal image display.

When the user selects a function that modifies the transfer function for the image being displayed on the Comtal, the parameters that determine the transfer function are varied by moving the data tablet pen. Typing a key on the keyboard will then exit from the function and allow the user to select some other menu item.

2.7 Software Development Tools

The FORTRAN compiler that we are using compiles only one source file at a time. It has no provision for "including" the compilation of a second source file in the middle of the first source file. Since this is a very useful (perhaps essential) feature when building a large system, one of the first things that we implemented was a pre-processor that simulates the "include" feature. The pre-processor reads an input file and

writes out a temporary file which is the input file plus any "included" source files plugged in at the appropriate place. The FORTRAN compiler is then asked to compile this temporary file. The running of the pre-processor followed by the running of the FORTRAN compiler is easily accomplished by a single indirect command file in the RSX-11M system.

The files that get "included" are used to do such things as declare standard data structures, set up "common" areas, and initialize storage. For example, the file CATREC.DCL contains FORTRAN statements which define a common array which will hold a record from the catalog file. It also defines individual variables that correspond to the various items in the catalog record. Appropriate EQUIVALENCE statements then establish the correspondence between the named items and the locations in the catalog record block. Thus, a single "include" statement allows a program or subroutine to allocate storage for a catalog record and then refer to the items in the catalog record by name. The advantage of this occurs when the structure of the catalog record has to be changed. Instead of changing all of the routines which use the catalog record, we just change the CATREC.DCL file and then recompile all of the routines which use it. This is a simple matter since command files are set up which will do all the necessary compiling, etc. Typing the operating system command "@BUILDIMPS" will recompile and relink the entire IMPS system. (Typing this command is something that is done by the person who is maintaining the IMPS software. This is not something that the user of IMPS needs to worry about.)

Another important example of the use of "included" files is in setting up buffers to hold lines of map data. The files ML1.DCL, ML2.DCL, etc. each define a separate map line buffer. This includes an array to hold the data, but it also includes storage for several items of header information such as the number

of map elements currently in the buffer and the current data format of these elements. These header items are used by the I/O routines and the format conversion routines to keep track of things and to detect error conditions. These .DCL files also set up variables that allow the application routines to conveniently access the map elements as an array of the appropriate data type.

3.0 TASK ORGANIZATION AND CONTROL FLOW

IMPS is implemented as a number of separate "tasks" rather than as a single giant program. A "task" in the RSX-11M operating system is a single executable program, the result of running the Task Builder program. Currently, none of the tasks in the IMPS system make use of overlaying. (Although IMPS is composed of a number of separate tasks, as far as the user of IMPS is concerned, it looks like one single system.)

In addition to the tasks in the running IMPS system, there are several auxiliary tasks which initialize some things and which must be run prior to running IMPS. The GENMEN task generates the VT-11 display files which contain the menu items. These are written out into disk files. Thus, when IMPS needs to display menus, it reads the display files from the disk rather than having to regenerate them each time. The GENDIS task generates other display files that are used by the tasks which implement the various IMPS functions. Running GENMEN and GENDIS is something that is done by the person who is maintaining the IMPS software. These need to be run only when a change is made to the menus or other VT-11 images that are to be displayed.

To initialize the IMPS system after the operating system has been loaded, the operating system command "@INSIMPS" is typed on the console terminal. This executes a small command file which tells the operating system that the various IMPS tasks exist and then runs a little program which sets a value in a core-resident global common area that

indicates that the various IMPS parameters in global common need to get initialized. (The execution of this command file will soon be added to the automatic initialization that is done when the operating system is loaded.)

To start up the IMPS system, the user types "RUN IMPS" on the console terminal. This is the only command that the user ever needs to type to the operating system. (Also, this is the only typing that the user needs to do on the console terminal. Any typing that is required while running IMPS is done on the keyboard that sits in front of the VT-11 display.) This command causes the IMPS task to be run. This task looks to see if the global common area needs to be initialized. If so, it runs another task to do the actual initialization. Currently, there is only one IMPS task executing at any given time. (This won't necessarily be true in the future. See below.) The basic function of the IMPS task is to display menus and to figure out which function the user wants to execute. It then initiates the task that will perform the desired function and immediately exits. After that task is done performing its function, it initiates IMPS again and exits. Note that when the IMPS task passes execution control to another task it also passes use of the VT-11 display, the data tablet, and the keyboard to that task.

In some cases, the secondary task that is initiated by the IMPS task does not itself reinitiate the IMPS task. Instead, it passes control to a third task, which will then reinitiate IMPS when it finishes. In some cases this is done simply to break a large function into two smaller pieces. However, in other cases, this is done to break the function into two distinct phases: getting parameters from the user and actual execution of a processing operation. An example of this is the function which produces a smaller map from a bigger map by averaging groups of pixels together. The secondary task determines what input map is desired, what averaging factor is desired, and what name is desired for the output map. It then

passes this information to the task which actually does the averaging. The data is passed by using the facility provided by the operating system for passing 13-word blocks of data from one task to another. Note that this third task does not need to have any interaction with the user. In principal, the second task could initiate the third task and then return control to the IMPS task. The user could then continue to interact with IMPS while the third task produces the averaged map as a "background" task. In the near future, IMPS will be modified to allow initiation of this type of "background" processing operation. (This may require obtaining more core and/or obtaining a faster CPU to provide the needed CPU cycles.)

4.0 ADDING NEW FUNCTIONS TO IMPS

IMPS is being implemented in such a way that it is relatively easy to add new functions. In addition to adding new functions to the "standard" IMPS system, it will be possible for the astronomer/user to write a FORTRAN program that does some special "experimental" function and then have that function invoked through the IMPS system. (Appropriate documentation on the details of doing this will be provided in a separate document.)

4.1 Adding User Coded Functions to IMPS

In order to add his own function to IMPS, the user begins by creating the FORTRAN source file using the text editor on the RSX-11M system. Or, if desired, the source file can be created with the text editor on the DEC-10 system and then transferred over to the PDP-11 system. A two-line command file is also prepared. This contains operating system commands to run the FORTRAN pre-processor and then run the FORTRAN compiler. The user then executes the command file by typing an "@" followed by the name of the command file. Next, the user runs the Task Builder program (TKB). This produces an executable module in a

disk file. The user then types an Install command to the operating system. This informs the operating system that the user's executable task image file exists.

In order to execute the task, the user runs the standard IMPS system and selects the "Execute User Coded Task" function in the top level menu. IMPS then asks the user to type in the name of the desired function (the user may have more than one), and then runs the user's task. The user's function may be coded as a single task or it may be broken up into more than one task. For example, there may be one task which interacts with the user to obtain parameter values or desired option specifications. This task could then activate another task which actually executes the desired function.

4.2 Adding Functions to the "Standard" IMPS System

In order to add a function to the "standard" IMPS system, the GENMEN task is modified to include the new menu item at the desired place in the menu structure. The IMPS task is then modified so that it knows which task to activate when the new menu item is selected. If the new task needs a pre-generated VT-11 display file, then its generation should be added to the GENDIS task. Also, the compilation and task building of the new task must be added to the command files which build the entire standard IMPS system. And, the installation of the new task must be added to the command file which installs all of the standard IMPS tasks. The new task is coded exactly the same as if it were a user coded task rather than a standard IMPS task.

5.0 CURRENTLY IMPLEMENTED CAPABILITIES

The following functions are currently implemented:

Set Map Selection Category to be used.

Show Summary of Available Maps.

Initialize IMPS Parameters.

Load a Map Image into the Entire Comtal Display Screen -
if the map is smaller than the screen then the rest of the screen is loaded with the indefinite value. If the map is larger than the screen then the subsection specified by the user is loaded.

Clear Entire Screen.

Load Image into a Quadrant of the Screen -
handles any size map as with loading of entire screen.

Clear a Quadrant of the Screen.

Modify Image Display Parameters -
this includes all of the functions that were implemented in the old DATSEE program plus it adds some flexibility. Specifically, this includes "contrast sweep" modification of the transfer function, arbitrary three-segment transfer function modification, multiple contours, single contour, contrasting color encoding, and color encoding with a spectrum of colors.

Insert/Remove Linear Wedge in the Image -
this gives a visual display which shows how the image display is being modified.

Plot Horizontal Cross Section of Displayed Image.

Change Map Pixel Storage Format -
this actually creates a new map file and leaves the old one alone.

Decrease Number of Pixels in a Map by Averaging -
produces a new map file. (Only integer averaging factors are implemented.)

Increase Number of Pixels in a Map by Duplicating Pixels -
produces a new map file. (Only integer expansion factors are implemented.)

Execute User Coded Task.

6.0 CAPABILITIES TO BE IMPLEMENTED IN THE IMMEDIATE FUTURE

The remaining functions that are currently listed in the menus will be implemented soon. (The current menus are listed in the appendix.)

APPENDIX

A.1 CURRENT IMPS MENU STRUCTURE

The following shows the current IMPS menu structure. This is shown in an outline form rather than listing the pages as they appear when you run IMPS. For example, when you first start up IMPS you are presented with a menu of the main categories of functions. This corresponds to just the main items in the list below.

1. Utility Functions
    - 1.1 Set Map Category to be Used
    - 1.2 Show Summary of Available Maps
    - 1.3 Show All Header Information for a Map
    - 1.4 Change a Map's Category
    - 1.5 Delete a Map
    - 1.6 Save IMPS Parameters
    - 1.7 Restore IMPS Parameters
    - 1.8 Initialize IMPS Parameters
2. Image Loading Functions
    - 2.1 Load Image into Entire Screen
    - 2.2 Clear Entire Screen
    - 2.3 Load Image into a Quadrant of the Screen
    - 2.4 Clear a Quadrant of the Screen
    - 2.5 Load Two Images for Blink Comparison
    - 2.6 Scroll Load an Image
    - 2.7 Load Graphic Overlay Image into Entire Screen
    - 2.8 Clear Graphic Overlay in Entire Screen
    - 2.9 Load Graphic Overlay Image into Quadrant of Screen
    - 2.10 Clear Graphic Overlay in Quadrant of Screen
3. Image Display Modification Functions
    - 3.1 Modify Image Display Parameters

      Selecting this menu item gives you a display which includes a menu which lists the available types of

modification. It also shows you a plot of the current transfer function and a list of the current display status parameters.

3.2 Blink Compare Two Images

3.3 Blink Graphic Overlay

3.4 Insert/Remove Linear Wedge in Image

Selecting this item gives you the same display as item 3.1.

4. Data Plotting Functions

4.1 Plot Horizontal Cross Section of Displayed Image

4.2 Plot Arbitrary Cross Section of Displayed Map

5. Data Processing Functions

5.1 Utility

5.1.1 Change Map Pixel Storage Format

5.1.2 Decrease Number of Pixels by Averaging

5.1.3 Select Subsection of a Map

5.1.4 Increase Number of Pixels by Duplicating Pixels

5.1.5 Increase Number of Pixels by Interpolating Pixels

5.2 Map Arithmetic

6. Execute User Coded Task

7. Exit from IMPS

A.2 DEC-10 TO IMPS DATA TRANSFER

A.2.1 Interim Data Transfer System

An interim data transfer system is currently implemented which allows us to get data into the IMPS data base. This system is not very convenient to use, but it was very easy to implement and it gives us some data to work with while the "final" data transfer system is being worked on. The interim system involves transferring the data to one of the disk cartridges under the old RT-11 DATSEE system. This uses the teletype line and is thus slow. Next, the RT-11 format file is transferred to the RSX-11M format disk using a standard program. And finally, a

program is run which adds this map file to the IMPS data base. This program obtains some of the header information by having the user answer questions.

A.2.2 Final Data Transfer System

Obviously, the interim data transfer system is not adequate for actual production use. A much faster and more complete system is currently under development. This new system is based on the processor-to-processor parallel link hardware called the DA-28 and on software written in-house. At the present time the lowest level software is implemented and runs (some of it is so new that it has yet to become settled).

The basic software provides the following capabilities:

1) Multi-leaved task-to-task communication of arbitrary data streams with automatic task initiation (at present, only in the PDP-11s). By multi-leaved we mean that multiple data transmission streams may be in progress at the same time. For example, a map file may be in the process of transmission from the DEC-10 to the PDP-11 at the same time that a print file is being spooled from the PDP-11 to the DEC-10.

2) A generalized queueing system which allows queueing of messages for specific tasks (at present only tasks in the DEC-10 queue requests).

Naturally, the software provides error checking and retry on data communications, necessary queue support functions and the like for the capabilities it implements.

The interprocessor data transfer system is intended to provide data transfer between the DEC-10 and all of the PDP-11s at the site. As such, its use by IMPS is only part of the work that it will be doing. Because the data transfer system provides

general task-to-task communication, implementation of IMPS specific communications tasks will be rather straightforward (similar function specific routines have already been implemented to allow general file copying, print spooling and so on). For the immediate future two functions seem urgent and easily implemented:

1) Retrieval of map data bases (in IMPS format) from either the DEC-10 or from the PDP-11/70 map maker and integration of the new maps into the IMPS map catalog system. When transferring DEC-10 format maps to IMPS, all necessary format conversion will be done by the DEC-10.
2) Queueing of data processing operations on maps that reside in the IMPS system for the DEC-10 or PDP-11/70 map maker and retrieval of the results.

Of these two, the easiest to implement and most important is the first.

As new needs are discovered, new IMPS specific data transfer capabilities may be built on the basic data transfer system.

A.3 IMPS SUBROUTINE PACKAGES

The following is a brief description of the main subroutine packages that have been implemented for IMPS. This is not a complete list of all the routines that are available. The following subroutine packages are implemented in FORTRAN unless indicated otherwise. All routines that are coded in assembly language are callable from FORTRAN programs.

A.3.1 IMPS I/O Routines

A.3.1.1 CATIO

This is a set of routines for reading and writing the catalog files. Included are routines for opening and

closing catalog files and reading and writing a given catalog record.. There is also a routine which reads catalog records and searches for the next valid record that describes a map in the given category. Also included is a routine that finds the record number of an available slot in the catalog file and a routine which checks to see if a given map name would be unique.

A.3.1.2 MAPIO

This is a set of routines for reading and writing the map data files. First there is a routine which invents a name to be used for a new map file. (Map data are stored in files whose names are generated by IMPS.) Also included are routines for opening and closing map files and for reading and writing lines of map data. There is also a routine which waits for a data transfer to be completed. (We have implemented the capability to initiate a read or write operation and then continue to do some useful computing while the transfer is in progress.) Also included are routines for converting map cell data formats and for handling scaling into Comtal pixel values.

A.3.1.3 CVFMT

This is a single assembly language routine that actually does the conversion of map cell data formats.

A.3.1.4 TBPRIM

This is a set of assembly language routines that provide the basic operations involving the data tablet.

A.3.1.5 RCPRIM

This is a set of assembly language routines that

provide the basic operations involving the Comtal display.

A.3.1.6 KBPRIM

This is a set of assembly language routines that provide the basic operations involving the keyboard.

A.3.2 IMPS Utility Routines

A.3.2.1 CATLST

This is a set of routines that generate people-readable lists of the coded information in catalog file records.

A.3.2.2 GETSSB

This is a routine which gets a map subsection specification from the user. This is done by using the VT-11 display and the data tablet.

A.3.2.3 GETMRN

This is a routine which gets the catalog record number that describes a map that the user desires to be used as an input map. This is done by displaying a summary list of available maps on the VT-11 and allowing the user to point to the desired map.

A.3.2.4 LBUTN

This is a generalized light button routine. This routine allows a calling routine or program to conveniently display a list of options on the VT-11 and determine which one the user wants to select.

A.3.2.5 RELTAB

This is a set of routines that provide use of the

data tablet at a higher level than that provided by the TBPRIM package. These routines implement use of relative rather than absolute coordinates from the data tablet.

A.3.2.6 STRSUB

This is a package of assembly language routines which provide string handling facilities. A string is defined as a sequence of consecutive character bytes followed by a null character. Strings are stored in LOGICAL*1 arrays. Implemented functions include determining the length of a string, copying strings, concatenation, picking substrings, comparing strings, padding with blanks, striping off blanks, reading a string typed on the console terminal, and outputting a string on the console terminal.

A.3.2.7 FAUTIL

This is a package of assembly language routines which help FORTRAN do some arithmetic things. Included are routines for converting single precision integers to double precision integers, routines for doing double precision integer arithmetic, etc.

A.4 DATA FORMAT FOR IMPS MAP STORAGE (Version 10, 9-24-78, JMT)

A.4.1 General Organization

The map data is stored in several different types of disk files. For some maps, some of these types of files are absent. The different types are:

Catalog - list of available maps, includes most of the important map header information

Map Data - the actual map data array

Spectral Line Header - additional header information pertaining to the different frequency channels

for a spectral line map

Fitted Sources - list of parameters for the sources that have been fitted

Subtracted Sources - list of parameters for the sources that have been subtracted

Cleaned Sources - list of parameters for cleaned sources

History - information about how the map was created

For each user, there is a file named <pn>.CAT which contains the catalog of available maps belonging to that user. (<pn> is the user's programmer number on the DEC-10 system.) There is one record in the catalog file for each different map. Note that a "map" as defined here may be a spectral line map. That is, a single "map" contains all of the data for all of the different frequency channels. One of the things listed for each map is a map file name. The various other types of files pertaining to a given map will have the same file name but different extensions.

A.4.2 Format of Catalog File

In the following descriptions, the capitalized name listed under "Item" is the name of the variable that holds the item in the FORTRAN programs. Also, the letter following the length of the item indicates the format of the stored item. I indicates an integer, F indicates a floating point, and C indicates a character format. Note that for character items the length given is the length of the field for storing the string. This includes a character space for the null character that terminates the string. Thus, the allowable length of the string is one less than the length here. In the following lists, items whose names are marked with a * are those items that must have proper values in order for the data to get accessed properly. (The * is not part of the variable name of course.) The following is the format for each entry in the catalog file:

Word No.	Byte No.	Length (bytes)		Item	
				***** MAP IDENTIFICATION SECTION *****	
0	0	13	C	*MAPNAM	Map name - must be unique for each map (supplied by user)
	13	1		(unused)	
7	14	13	C	*MAPCAT	Map category (supplied by user) - used to group maps together in any way desired by the user
	27	1		(unused)	
14	28	9	C	SOURCE	Source name (8 characters)
	37	1		(unused)	
19	38	2	I	QUAL	Source Qualifier (integer)
20	40	2	I	BAND	(1=1.3 cm, 2=2 cm, 3=6 cm, 4=20 cm)
21	42	2	I	DATTYP	Type of data in the file 1=map (further defined by MAPTYP) 2=beam (further defined by MAPTYP) 3=model (further defined by MAPTYP and MODTYP) 4=clean residual (further defined by MAPTYP and MODTYP) 5=UV coverage - indicates which cells have data 6=UV coverage - number of measurements in each cell 7=UV coverage - amplitude of visibility in each cell 8=contour map 1-bit overlay 9=grid line or tic mark 1-bit overlay 10=phase closure data
22	44	2	I	MAPTYP	Type of map data 1=I 2=Q 3=U 4=V 5=P, i.e., sqrt(Q*Q + V*V) 6=p, i.e., P/I 7=psi 8=spectral index 9=some arithmetic function of one or more other maps

Word No.	Byte No.	Length (bytes)	Item	
23	46	2 I	MODTYP	Model type 1=clean - keep residuals 2=clean - don't keep residuals 3=maximum entropy
24	48	4 F	MAPDAT	Date this map was created
26	52	4 F	MAPTIM	Time this map was created
28	56	2 I	*MAPNX	Number of map cells in X direction
29	58	2 I	*MAPNY	Number of map cells in Y direction
30	60	20	(unused)	

***** AUXILIARY MAP IDENTIFICATION SECTION *****

40	80	81 C	MLABEL	Any descriptive label supplied by the user
	161	1	(unused)	
81	162	4 F	OBSDAT	Date of observation (if more than one date is involved, this will be the first date)
83	166	2 I	MOBSDT	Multiple observation date indicator 0=just one date 1=multiple dates
84	168	4 F	EPOCH	The date applicable to CNRAE, CNDECE, PHRAE, and PHDECE
86	172	8 F	CNRAE	Center cell RA at EPOCH - stored in radians as a double precision floating point number. ("Center" cell is the upper right cell of the four in the center of the map.)
90	180	8 F	CNDECE	Center cell DEC at EPOCH
94	188	8 F	CNRAO	Center cell RA at observation date
98	196	8 F	CNDECO	Center cell DEC at observation date

Word No.	Byte No.	Length (bytes)	Item	
				The following four parameters are the coordinates of the phase tracking center in the data that was used in making the map:
102	204	8 F	MPHRAE	Map phase tracking center RA at EPOCH
106	212	8 F	MPHDCE	Map phase tracking center DEC at EPOCH
110	220	8 F	MPHRAO	Map phase tracking center RA at observation date
114	228	8 F	MPHDCO	Map phase tracking center DEC at observation date
				The following four parameters are the coordinators of the phase tracking center in the original data produced by the synchronous system. This is the same as the coordinates of where the antennas were pointing:
118	236	8 F	OPHRAE	Original phase tracking center RA at EPOCH
122	244	8 F	OPHDCE	Original phase tracking center DEC at EPOCH
126	252	8 F	OPHRAO	Original phase tracking center RA at observation date
130	260	8 F	OPHDCO	Original phase tracking center DEC at observation date
134	268	4 F	CELLDX	Cell size (radians) in X direction
136	272	4 F	CELLDY	Cell size (radians) in Y direction
138	276	4 F	MAPROT	Map rotation angle (radians of clockwise rotation)
140	280	13 C	XLABEL	X coordinate label
	293	1	(unused)	
147	294	13 C	YLABEL	Y coordinate label
	307	1	(unused)	
154	308	4 F	BW	Bandwidth
156	312	4 F	DFREQ	Frequency difference between adjacent channels (minimum difference if they are not all the same)

Word No.	Byte No.	Length (bytes)	Item	
158	316	4 F	DVEL	Velocity difference between adjacent channels (minimum difference if they are not all the same)
160	320	4 F	SUBFLX	Total subtracted flux (Jy)
162	324	4 F	BEAMPA	Position angle of fitted beam
164	328	4 F	BHPMAJ	Half-power width along major axis of fitted beam
166	332	4 F	BHPMIN	Half-power width along minor axis of fitted beam
168	336	2 I	DIFBEM	Different beam flag 0=different beams not used, i.e., BEAMPA, BHPMAJ and BHPMIN used for all cleaning 1=different beams used for cleaning, see CLNPA, CHPMAJ and CHPMIN in Cleaned Sources File
169	338	4 F	CLNFLX	Total cleaned flux - sum of fluxes of cleaned components (Jy)
171	342	20	(unused)	

***** MAPPING PARAMETERS SECTION *****

181	362	2 I	WTTYPE	Type of weighting of data 1="natural" (cell sum) 2="uniform" (cell average)
182	364	2 I	TAPER	Type of taper function used 1=Gaussian 2=linear
183	366	4 F	TAPWDX	Taper parameter in X direction (sigma for Gaussian taper, dist. to zero for linear taper)
185	370	4 F	TAPWDY	Taper parameter in Y direction

Word No.	Byte No.	Length (bytes)	Item	
187	374	2 I	CONVLV	Convolving function 1=boxcar (II function) 2=Gaussian
188	376	4 F	CNVWDX	Convolution parameter in X direction (width for boxcar convolution, sigma for Gaussian convolution)
190	380	4 F	CNVWDY	Convolution parameter in Y direction
192	384	2 I	UNGRID	Indication of whether or not effects of gridding were undone 0=no 1=yes
193	386	20	(unused)	

***** SPECTRAL LINE CHANNEL INFORMATION SECTION *****

203	406	2 I	VELREF	Velocity reference frame: 1=LSR? 2=???
204	408	4 F	CHFREQ	Frequency of first channel
206	412	4 F	CHVEL	Velocity of first channel
208	416	2 I	CHANFL	Flag for first channel
209	418	2 I	CHANWT	Weighting factor for first channel
210	420	4 F	*IMAX	Maximum strength in first channel (Jy)
212	424	2 I	IMAXX	X location (pixel coordinate) of maximum in first channel
213	426	2 I	IMAXY	Y location (pixel coordinate) of maximum in first channel
214	428	4 F	*IMIN	Minimum strength in first channel (Jy)
216	432	2 I	IMINX	X location (pixel coordinate) of minimum in first channel

Word No.	Byte No.	Length (bytes)	Item	
217	434	2 I	IMINY	Y location (pixel coordinate) of minimum in first channel NOTE: if NCHANS is greater than 1 there will be a file with extension .CHN which contains the various items for each channel (see below)
218	436	2 I	*MAPSCL	Binary scale factor for first channel map
219	438	4 F	*PSCALE	Pixel scale factor (for scaling pixel values into map values)
221	442	4 F	*PTRANS	Pixel translation amount (for scaling pixel values into map values)
223	446	20	(unused)	

***** MAP ACCESS INFORMATION SECTION *****

233	466	2 I	*MDBVER	Map data base version (This is set by the I/O routines and should not be changed by the user program.)
234	468	10 C	*MAPFIL	Name of file that contains the map data. The name of the file will be a number, e.g., 000001, 000002, etc. The user will not have to worry about these names. They will be generated by the programs.
239	478	2 I	BEAPTR	Beam pointer - If this catalog entry is for a map (DATTYP=0), the beam pointer is the number of the catalog entry for the corresponding beam. If there is no beam, this will contain zero.
240	480	2 I	*NCHANS	Number of frequency channels contained in this map

Word No.	Byte No.	Length (bytes)	Item	
241	482	2 I	NFITS	Number of fitted sources NOTE: if NFITS is greater than zero there will be a file with extension .FSR which contains the information about the fitted sources (see below)
242	484	2 I	NSUBS	Number of subtracted sources NOTE: if NSUBS is greater than zero there will be a file with extension .SSR which contains the information about the subtracted sources (see below)
243	486	2 I	NCLNS	Number of cleaned sources NOTE: if NCLNS is greater than zero there will be a file with extension .CSR which contains the information about the cleaned sources (see below)
244	488	2 I	*CELFMT	Cell data format 1=1-bit graphic overlay 2=8-bit pixel 3=16-bit integer (scaled by MAPSCL) 4=32-bit floating point
245	490	2 I	*MAPORG	Map data organization parameter 1=bottom line (left pixel first) of first channel map is first. After complete first channel map comes the complete second channel map, etc. 2=???
246	492	20	(unused)	

The catalog file can be thought of as a FORTRAN unformatted direct access file. That is, it contains the binary data only. There are no extra words for block lengths or anything like that. However, the first word of the first record contains the number

of actual catalog records in the file. This number does not include this header record. The actual catalog records start with the second record in the file. Also, the record count includes all of the space for catalog records in the file. That is, it doesn't matter whether or not some of the records at the end of the file currently contain valid catalog records.

A.4.3 Format of Map Data File

The extension of the map file is .MP (even if it contains a beam or u,v coverage map).

The map data is stored in rows, with the bottom (south) row first. The elements within a row are stored left to right (east to west). All of the data for the first spectral line channel is stored together, followed by all the data for the second channel, etc. {This is the "normal" storage organization. There may be different values of the map organization parameter (MAPORG) corresponding to different organizations.}

The map data file can be thought of as a FORTRAN unformatted direct access file. There are no header words in the actual map data file. (All of the needed header information is contained in the appropriate catalog record.)

A.4.4 Format of Spectral Line Header File

The extension of the spectral line header file is .SLH. For a given map, this file will be present only if NCHANS in the header part of the map data file is greater than one. Each record in the spectral line header file contains information about one spectral channel. There are thus NCHANS records in this file. The format of each record is:

Word No.	Byte No.	Length (bytes)		Item	
0	0	4	F	CHFREQ	Frequency of this channel
2	4	4	F	CHVEL	Velocity of the channel
4	8	2	I	CHANFL	Flag for this channel
5	10	2	I	CHANWT	Weighting factor for this channel
6	12	4	F	*IMAX	Maximum strength in this channel (Jy)
8	16	2	I	IMAXX	X location (pixel coordinate) of maximum in this channel
9	18	2	I	IMAXY	Y location (pixel coordinate) of maximum in this channel
10	20	4	F	*IMIN	Minimum strength in this channel (Jy)
12	24	2	I	IMINX	X location (pixel coordinate) of minimum in this channel
13	26	2	I	IMINY	Y location (pixel coordinate) of minimum in this channel
14	28	2	I	*MAPSCL	Binary scale factor for this channel map
15	30	4	F	*PSCALE	Pixel scale factor (for scaling pixel values into map values)
17	34	4	F	*PTRANS	Pixel translation amount (for scaling pixel values into map values)

The spectral line header file can be thought of as a FORTRAN unformatted direct access file.

A.4.5 Format of Fitted Sources File

The extension of the fitted sources file is .FSR. For a given map, this file will be present only if NFITS in the header part of the map data file is greater than zero. Each record in the fitted sources file contains information about one fitted source. There are thus NFITS records in this file. The format of each record is:

Word No.	Byte No.	Length (bytes)	Item	
0	0	2 I	FITTYP	Type of fitted source 1=Gaussian 2=???
1	2	4 F	IFIT	Peak strength of fitted source (Jy)
3	6	2 I	FITX	X coordinate of fitted source (pixel coordinate)
4	8	2 I	FITY	Y coordinate of fitted source (pixel coordinate)
5	10	4 F	FITPA	Position angle of fitted source
7	14	4 F	FHPMAJ	Half-power width along major axis
9	18	4 F	FHPMIN	Half-power width along minor axis
11	22	4 F	EIFIT	Error in peak strength of fitted source
13	26	4 F	EFITX	Error in X coordinate of fitted source
15	30	4 F	EFITY	Error in Y coordinate of fitted source
17	34	4 F	EFITPA	Error in position angle of fitted source
19	38	4 F	EFHPMJ	Error in half-power width along major axis
21	42	4 F	EFHPMN	Error in half-power width along minor axis

The fitted sources file can be thought of as a FORTRAN unformatted direct access file.

A.4.6 Format of Subtracted Sources File

The extension of the subtracted sources file is .SSR. For a given map, this file will be present only if NSUBS in the header part of the map data file is greater than zero. Each record in the subtracted sources file contains information about one subtracted source. There are thus NSUBS records in this file. The format of each reocrd is:

Word No.	Byte No.	Length (bytes)	Item	
0	0	2 I	SUBTYP	Type of subtracted source 1=Gaussian 2=???
1	2	4 F	ISUB	Peak strength of subtracted source (Jy)
3	6	2 I	SUBX	X coordinate of subtracted source (pixel coordinate)
4	8	2 I	SUBY	Y coordinate of subtracted source (pixel coordinate)
5	10	4 F	SUBPA	Position angle of subtracted source
7	14	4 F	SHPMAJ	Half-power width along major axis
9	18	4 F	SHPMIN	Half-power width along minor axis

The subtracted sources file can be thought of as a FORTRAN unformatted direct access file.

A.4.7 Format of Cleaned Sources File

The extension of the subtracted sources file is .CSR. For a given map, this file will be present only if NCLNS in the header part of the map data file is greater than zero. Each record in the cleaned sources file contains information about one cleaned source. There are thus NCLNS records in this file. The format of each record is:

Word No.	Byte No.	Length (bytes)	Item	
0	0	2 I	CLNTYP	Type of cleaned source
1	2	4 F	ICLN	Peak strength of cleaned source (Jy)
3	6	2 I	CLNX	X coordinate of cleaned source (pixel coordinate)

Word No.	Byte No.	Length (bytes)	Item	
4	8	2 I	CLNY	Y coordinate of cleaned source (pixel coordinate)
5	10	4 F	CLNPA	Position angle of cleaned source
7	14	4 F	CHPMAJ	Half-power width along major axis
9	18	4 F	CHPMIN	Half-power width along minor axis

The cleaned sources file can be thought of as a FORTRAN unformatted direct access file.

A.4.8 Format of History File

The extension of the history file is .HIS. This file is a text file that describes the history of how the map was created.