1-1-1980

dMap A Graphics Package For dBase II.

Lauralynn S. Crocket

dMap

A Graphics Package For dBase II

by

Lauralynn S. Crocket

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Computer Science and Electrical Engineering

This research report is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

August 28, 1984
DATE

Professor D. J. Hillman
Advisor

Dr. E. Thompson
Chairman, Dept.
Computer Science and Electrical Engineering

ACKNOWLEDGEMENTS

I wish to thank my advisor, Donald Hillman, for his help in defining this project, John Wiginton for the use of his office and IBM XT, Andy Spitzer for his incite into interrupt handlers, Fletch for opening the lab at all hours of the day and night, and especially Takeo, for his prodding and support, without which this project would still not be completed.

Table of Contents

List of Figures

List of Tables

ABSTRACT

Discussion is presented on dMap, a graphics package that has been developed for use with dBase II, version 2.4. The package enables information from a user defined dBase II database file to be represented as datapoints on a map.

Specifications for the implementation of this package are discussed as well as an overview of what can be expected from the implementation of the package.

The primary focus of this paper however, is an analysis of the interfacing procedures. Discussed are the various interfacing options, and the problems associated with them. The option chosen, installs the graphics program as an interrupt handler, to be called by a user defined interrupt from an interfacing procedure.

1. Introduction

In the past several years, the memory capacity and processing speed of the microcomputer has been greatly improved. These improvements have made it practical to use the microcomputer for small to medium sized databases, at relatively low cost. At the same time, the graphics capabilities of many recent microcomputers have also been greatly enhanced. This makes it possible to provide an extensive man-machine interface which is not often feasible on a mainframe or dedicated system due to the high costs, such as graphics terminals and computation time.

Such graphic capabilities on microcomputers open new applications to database management systems by providing graphical interpretations of a database. In many cases, a graphical representation of certain data provides additional information. For example, a fire department maintains records of all calls received. If for each call, the following information is recorded; date, time, location, type of action, action taken, etc. it would then be possible to determine the number of false alarms for a given time period. For this time period, a map showing the location of each false alarm may show a significant clustering which would not be readily apparent by viewing the data as recorded.

Currently, there are available numerous software packages for

the microcomputer, ranging from file management systems to fairly complete database management systems. Unfortunately, few of these programs take full advantage of the available graphics except for drawing fixed formats such as bar graphs, pie charts, etc. For example, Lotus 123, a widely known package which presently integrates the use of graphs with its database management would not allow the user to implement their own graphic represention of a database for the aforementioned example. We feel there is a need for the development of programs for the microcomputer which will allow database management systems to make use of general graphics with user defined format.

The main project here is to explore the interfacing of a graphics program written in 'C' , using the HALO library of graphics routines, and dBase II from Ashton-Tate, one of the most widely known and used database management systems. The graphics capability will initially be limited to drawing a predefined map(s) and displaying data points also predefined on the map(s), representative of information selected from a user database. For simplicity, the main project will concentrate on the interface problems which are considered crucial in integrating the different programs.

The entire graphics package developed here is named dMap and hereafter shall be referred to by that name.

2. Overview of dMap

dBase II is a powerful database management system but in many ways it is not as extensive as several other systems, noteably Lotus 123 and Knowledgeman, both of which provide some graphics and spreadsheet capabilities. The result has been a plethora of programs which have been termed by Ashton-Tate as add-ons and add-ins. These programs attempt to increase the productivity of dBase II. As defined by Ashton-Tate, an add-on is a program which will use the information from a dBase II file but is executed from the system level. An add-in is a program which may be called from dBase II, and will return to dBase II upon completion. An add-in is necessarily more difficult to develop because of memory and stack considerations. It is also usually system dependant whereas an add-on need not be.

dMap is considered an add-in because it is activated upon execution of a dBase II command file. Fig. 2-1 illustrates a conceptual structure of dMap. In general, it can be divided into three major modules, each in a different environment.

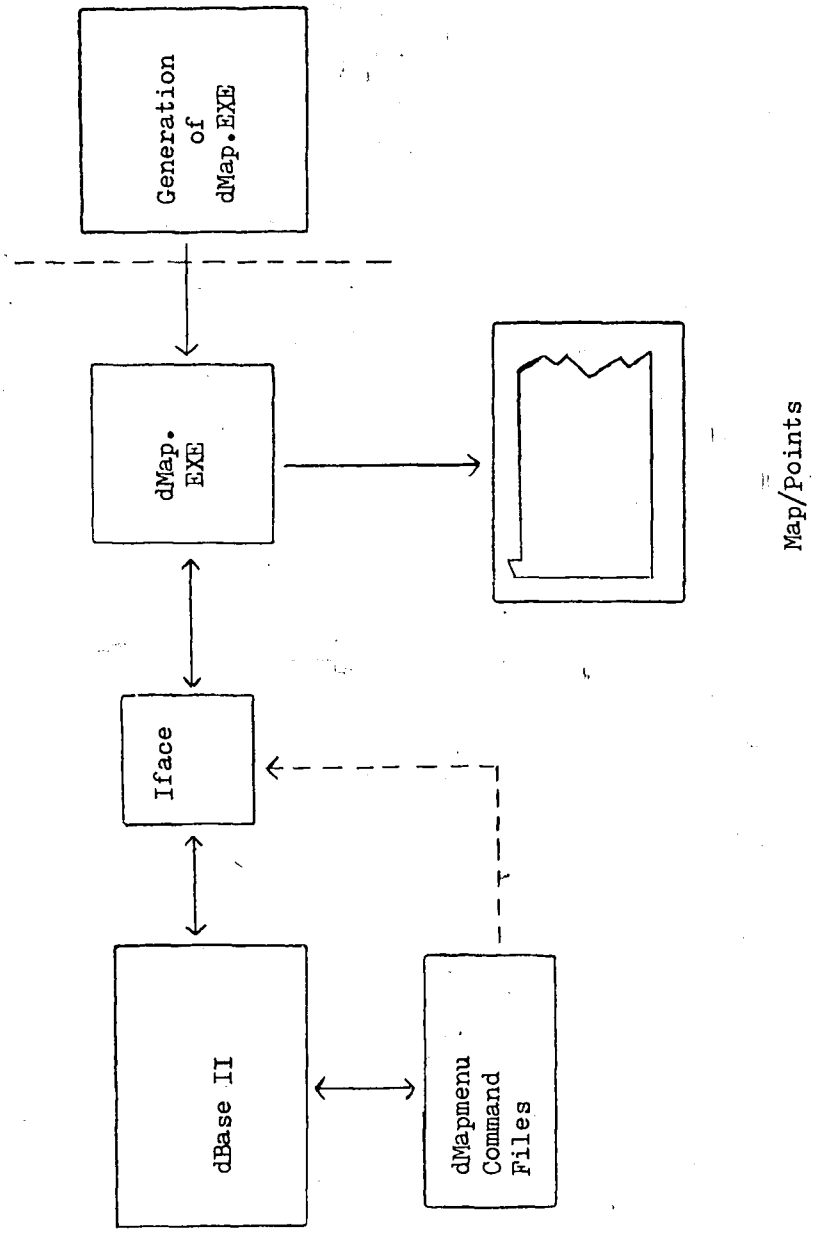

Figure 2-1: General concept of dMap

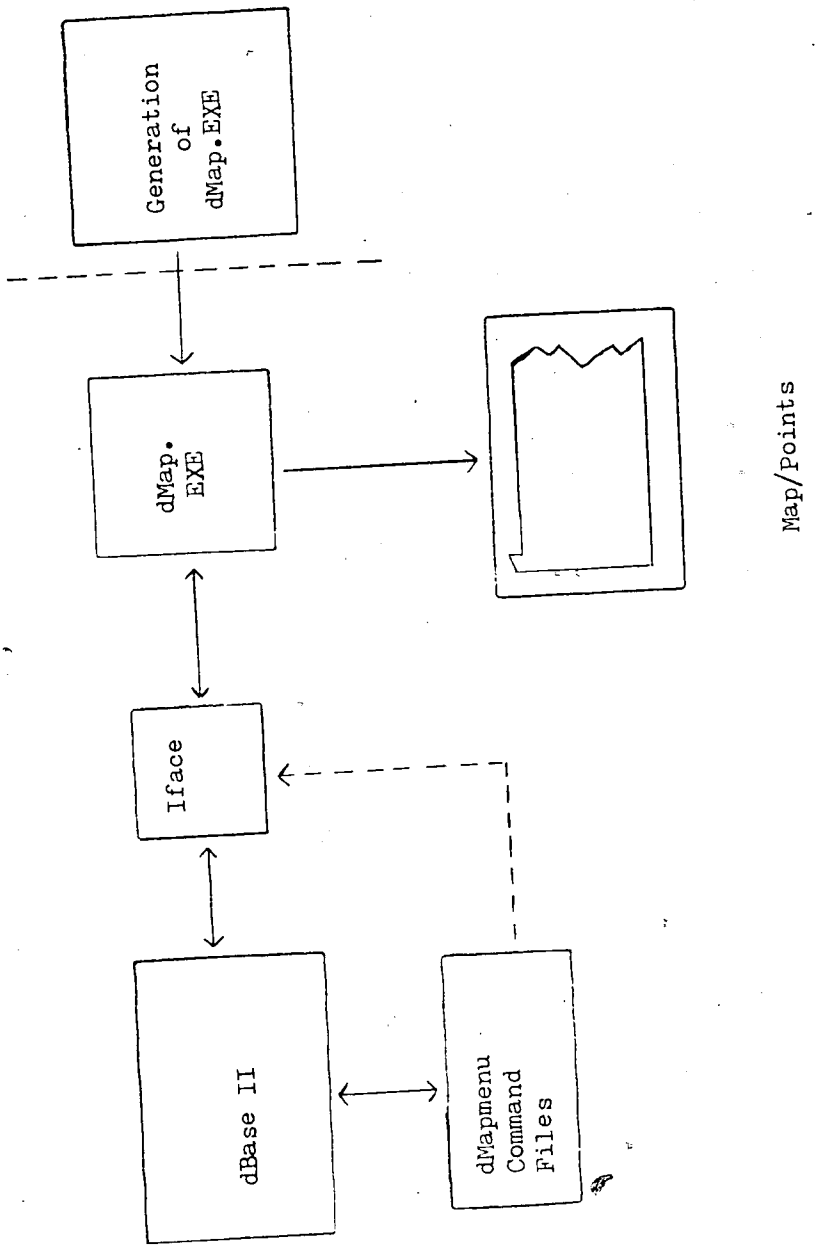

Figure 2-1: General concept of dMap

These three modules and their corresponding enviromments are named as follows:

1. dMap.Prg - dBase II environment hereafter referred to as dMapmenu
2. Iface - semi-dBase II environment
3. dMap.exe - DOS environment

The term environment is used imply a host system within which various programs or modules are loaded and executed.

2.1 dMapmenu

dMapmenu is a module consisting of a set of dBase II commands and is generated by the MODIFY[1] command in dBase II. The main function of this module is to create and edit records in database files specific to the dMap package. It is also responsible for reformatting these database files so that they can be read by dMap.exe in a non-dBase II environment. A detailed description of dMapmenu is given in chapter 3

2.2 Iface

The interface between the dBase II environment and dMap.exe, the graphics program is called Iface. It is an assembly language module that is generated online by one of the dMapmenu command files. A dBase II CALL command initiates execution of Iface. Iface

---

[1]For complete explanation and syntax of dBase II commands refer to the User Manual

then issues an interrupt to jump to dMap.exe. Upon return from dMap.exe, Iface restores the dBase II environment and returns to dMapmenu. A detailed description of this module is found in chapter 4.

2.3 dMap.exe

dMap.exe is a machine language module generated by the Lattice 'C' compiler and linked with a number of libraries and other object files. The main function of this module is to execute the graphics functions necessary to draw the map(s) and data points on the screen. The module must be loaded into memory prior to loading dBase II in order to co-exist with dBase II. Further explanation in terms of memory allocation for loading each program will be found in section 5-4. The generation of dMap.exe is described in section 5-3.

2.4 Data Files

In addition to the program modules, there exist two dBase II database files which are required by dMapmenu for processing data. The first file, Parametr.DBF contains information on the screen design for map drawing. Mappoint.DBF has the X and Y coordinates for map data points. These files, and their SDF[2] counterparts are fully described in section 6-1 along with file specifications. Also

---

[2]Standard Data Format

required is at least one data file containing X and Y coordinates for map drawing. The specifications for this type of file are found in section 6-2.

3. Module description - dMapmenu

3.1 Overview

dMapmenu consists of dBase II commands and is entirely within dBase II environment. The module can be divided into eight sub-modules as shown in fig. 3-1. The main functions of dMapmenu are:

1. File maintenance - Create and Edit dBase II database file Parametr.DBF

2. Record conversion - Modify dBase II database files Parametr.DBF and Mappoint.DBF so that their records are readable by dMap.exe as Ascii formatted strings in the non-dBase II environment.

3. Execution of dMap.exe - Generate the interface, Iface, and initiate a jump to Iface and then to dMap.exe.

An appropriate submodule is selected from the main menu submodule to execute the above functions.

3.2 File Maintenance

File maintenance under the dMapmenu environment is performed by the submodule SCRNMENU and its lower level submodules SCRNADD and SCRNUPD. SCRNADD and SCRNUPD add and modify, respectively, records in Parametr.DBF. Upon execution of these modules, a formatted input screen shows the entry points by indicating the record structure and waits for possible keyboard input. The programs return to the submenu SCRNMENU to await further action.

File maintenance to Mappoint.DBF must occur at the dBase II command level at this time because of the dependancy of its

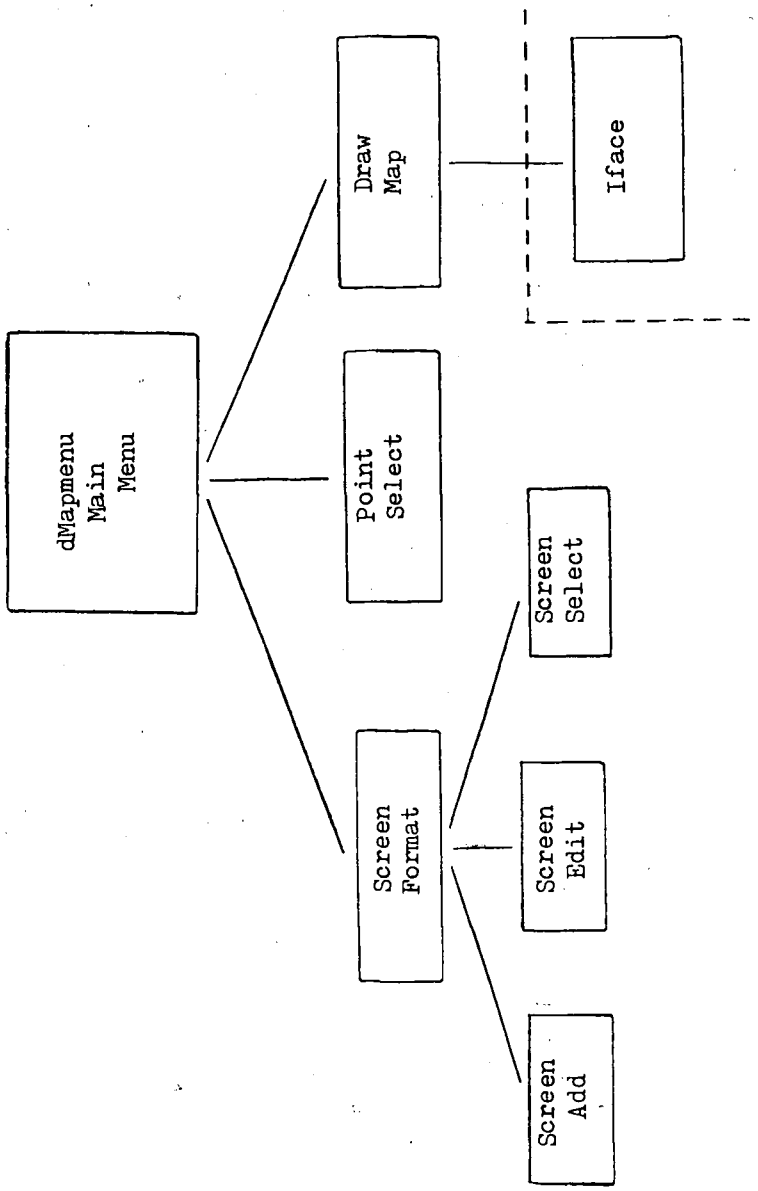

Figure 3-1: Command file structure

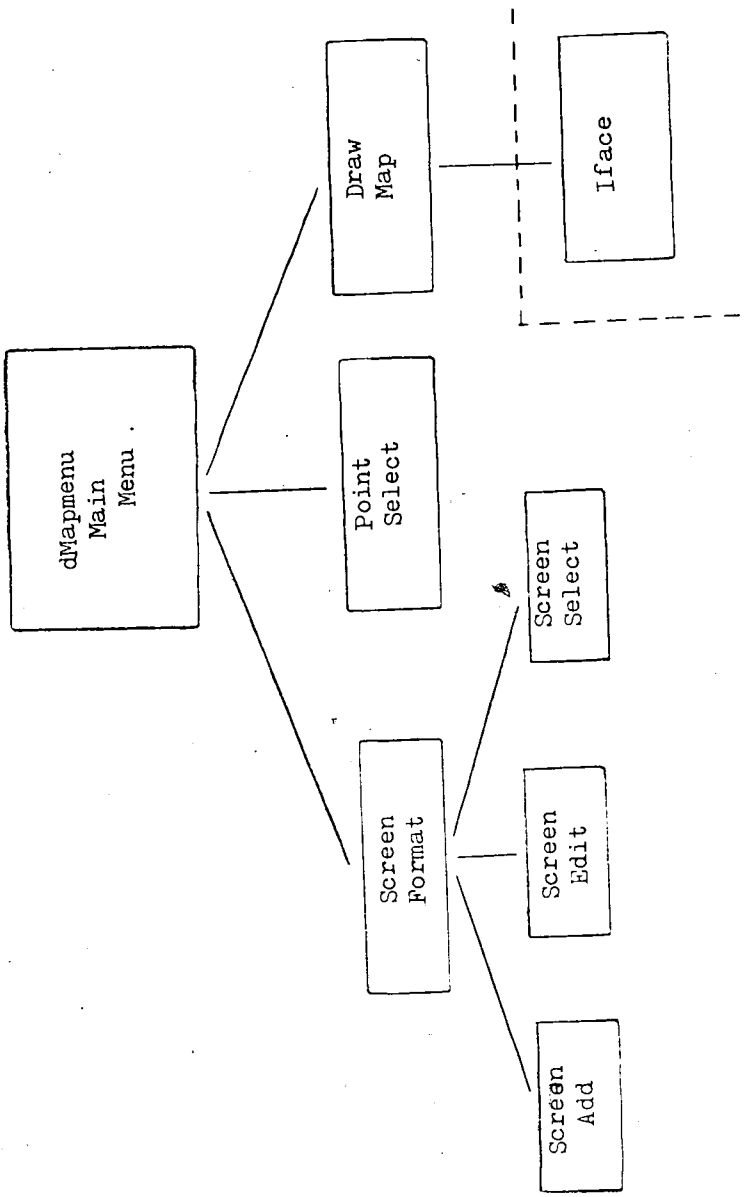

Figure 3-1: Command file structure

structure on the structure of the user's database.

The creation of the file structures for both Parametr.DBF and Mappoint.DBF must be done from the dBase II command level.

3.3 Record conversion

All the database files are transportable under the dBase II environment but must be modified if they are to be read under a non-dBase II environment. The main reason for this is that the file structure is included at the beginning of each .DBF file.

SCLTSCRN is used to select a single record from Parametr.DBF This record is copied to the file, Header.TXT as an ascii string of fixed field format similar to a Fortran data file. Header.txt may then be read by dMap.exe. Fig. 3-2 shows an original .DBF format for a record and the corresponding .TXT format.

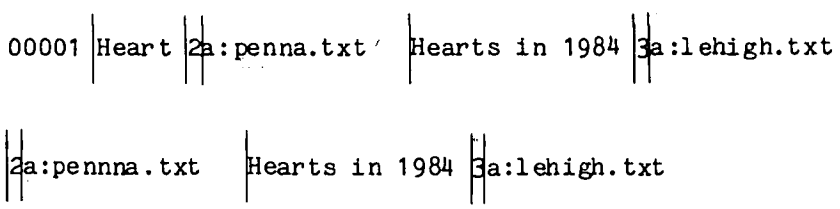

Figure 3-2: Fixed field conversion

DATASCLT uses information in a user's database file to select records from Mappoint.DBF which are then copied to Mppt.txt as ascii strings with a floating format, similar to a Pascal data file where data is separated by blanks. Fig. 3-3 is an example of floating field conversion.

Both SCRNSLCT and PTSLCT copy to the .TXT files only those fields in a database record that are actually used by dMap.exe.

00001	St.Lukes	75.567	40.123	1 3
00002	Easton	75.678	40.234	2 1

75.567␣40.123␣1␣3
75.678␣40.234␣2␣1

Figure 3-3: Floating field conversion

4. Module description - Iface

4.1 Overview

Iface is both generated and executed whenever the Display/Print Map option is selected from the dMapmenu, main menu. The command file, DRAWMAP is listed below and illustrates the necessary steps in the creation of Iface the interface with dMap.exe.

```
SET TALK OFF
POKE 57000,30,6,22,14,205,81,88,23,7,31,195
SET CALL TO  57000
CALL
```

4.2 Generation of Iface

Iface is created by poking the byte list shown above into memory starting at the offset location of 57000. In general, such self-generating programming should be avoided, but this process was found to be necessary in executing dMap since it is not possible to load Iface as an executable program in the dBase II environment. There are several reasons why this method was chosen and they are fully discussed in the technical section 7.1.

The byte list shown above consists of the decimal values of the machine instructions to be executed. The interface is created each time execution of dMap.exe is requested. This was done as a precaution, since it is unknown whether dBase II overlays at any

time the region of memory used by the interface

The execution of Iface is a two step process; setting the value of the CALL to the jump location and initiating the CALL.

5. Module description - dMap.exe

5.1 Overview

dMap is written in 'C' using functions from the HALO library to execute the graphics. The program draws one to four maps on the screen depending on parameters in a file created by dBase II. A single map will use the entire screen allowing for an optional title. Although, up to four maps may be presented on the screen simultaneously, only two map sizes are available, full screen or quarter screen. A half screen size is not used because of the image distortion due to the height width ratio. Fig. 5-1 shows the placement of multiple maps on the screen.

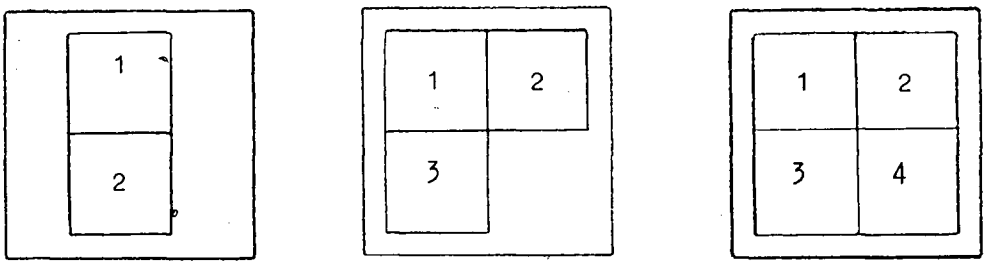

Figure 5-1: Multiple map screen layouts

A single, full screen map may also have a title of up to twentyfive characters. Unfortunately, HALO text functions do not provide a text size that is equivalent to that used in a regular text mode, so multiple map screens do not have titles.

The data points drawn on the maps also vary with the size of the maps drawn on the screen. On a single map, there are six allowable variations although more could easily be implemented. For each palette on the IBM PC there are three displayable colors, for each color a circle is drawn that may be filled or left unfilled. On the smaller maps, data points are represented by single pixels and therefore may be one of three colors.

5.2 Functional description of dMap.exe

The initial function of dmap.exe opens two files created by the dBase II portion of the package, dMapmenu. The first file opened and read is Header.txt, a parameter file containing the following information; number of maps to draw, and, for each map, the filename, the title and the color in which to draw the map. The second file, Mappoint.txt is opened but not read until a later point in the program.

For each map to be drawn, the mapfile specified in the header file is opened. The first read accepts the world coordinate values which are mapped to the area of the screen designated as the viewing area for that particular map. The location of a map on the screen is decided as a function of the total number of maps to be drawn and the number of the map currently waiting to be drawn. The map is drawn using a standard line drawing function.

Once the map is drawn, information is read from the mappoint file for each data point to be displayed. This information includes: X and Y world coordinates, mapnumber and colorcode. As previously mentioned, the world coordinates must be converted to device coordinates prior to display. The colorcode is a number from one to six, indicating the color and fill characteristics. Numbers one to three are open circles in colors one to three respectively, using the IBM-Halo color designations. Colorcodes four to six are filled circles of colors one to three, respectively. When more than one map is displayed on the screen, the data points are represented by single pixels and therefore the colorcodes are numbers one to three only.

A title can be drawn in the lower portion of the screen when a single map is drawn. The user does not have control over the characteristics of the text used to display this title.

A parameter passed from dMapmenu to dMap.exe via Iface will determine whether or not the screen display is sent to the printer prior to returning to dMapmenu. The map(s) will remain on the screen until a keyboard character is hit.

5.3 Generation of dMap.exe

Source code for dMap is written in 'C' and compiled by the Lattice 'C' compiler. As illustrated in fig. 5-2, two additional object files and three libraries must be linked with the main module, dMap.obj, for the creation of an executable program, dMap.exe.

The first object file in the linked program is dMapCS. The source code, dMapCS.asm is a modified version of CS.asm. CS.asm has been provided with Lattice 'C' to enable the user to make modifications, as we have done. Some form of this routine must be linked first because it provides the initialization as well as the entry and exit points for the program. Initialized are the static and dynamic data areas as well as the stack. A detailed explanation of the modified dMapCS will be given in the next chapter.

Main.obj is not usually part of the command line passed to the linker because it is part of the standard LCS library. The command link version of Main.obj takes precedence over the library version in the linking process. THe source code for this object file however is also provided with Lattice 'C' to allow for user modifications.

The first library linked is LCS which contains the standard 'C' functions. The second library, LCX, is the 'C' Food Smorgesbord,

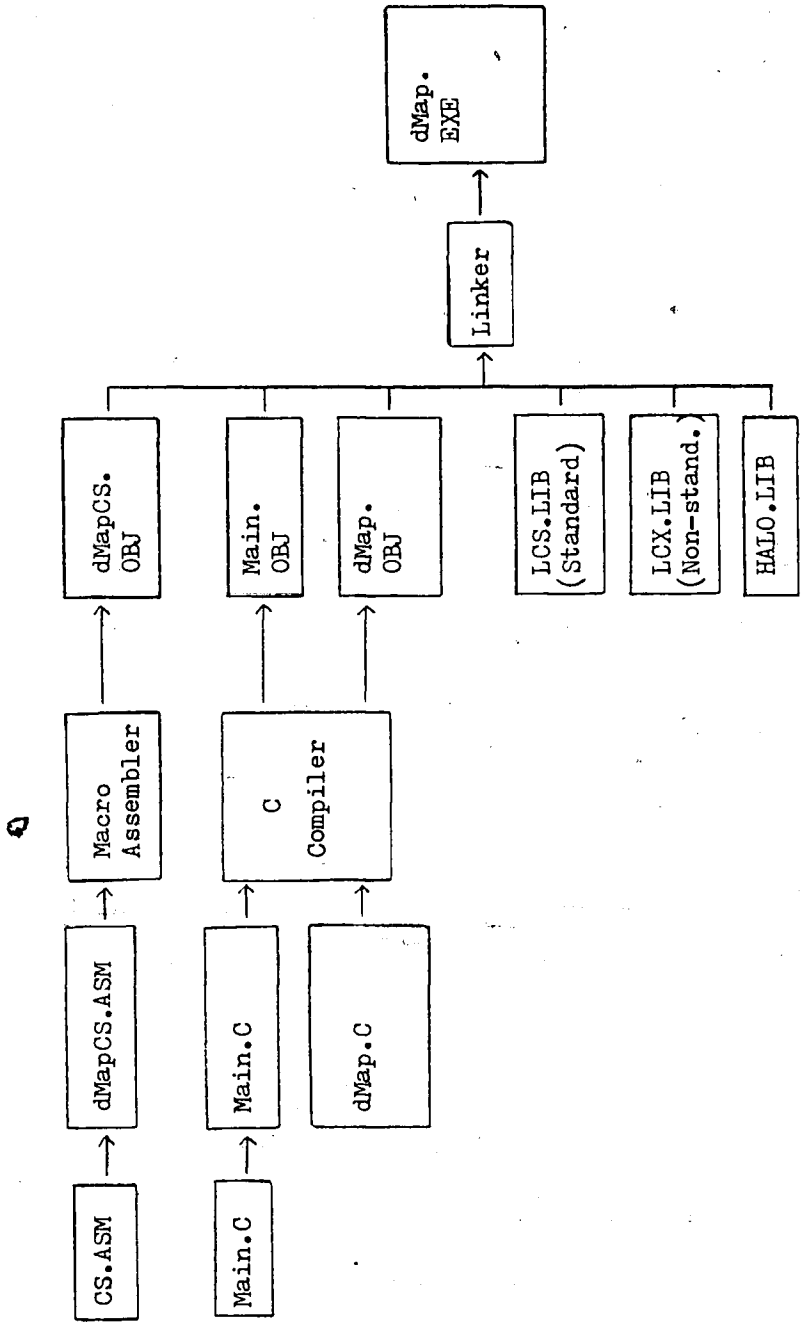

Figure 5-2: dMap.exe generation

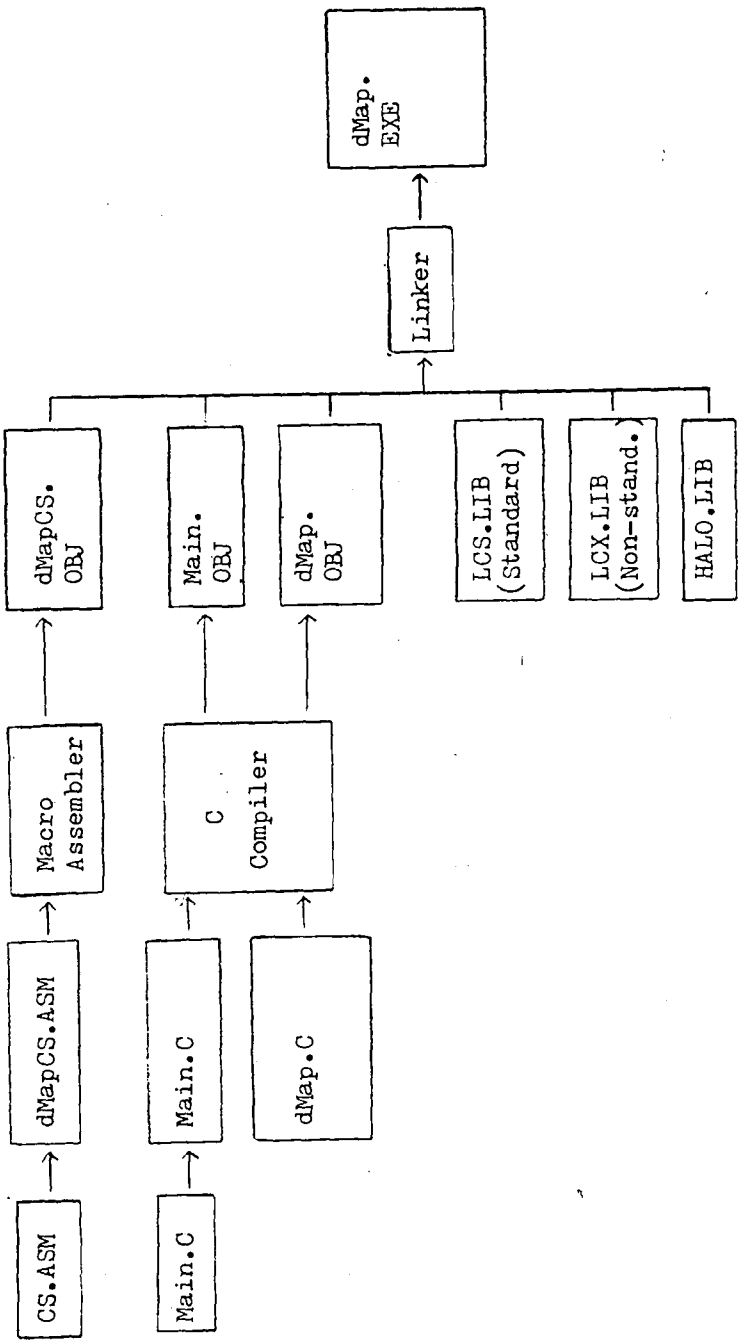

Figure 5-2: dMap.exe generation

and is comprised of low level nonstandard functions. The last library, Halo, contains the graphics functions necessary to create the maps.

5.4 Program execution order

Two programs must be executed prior to entering dBase II. Both programs upon execution will be made resident in DOS as interrupt handlers. The first, Graphics.com which will enable a PRTSCRN to print the graphics portion of the screen, increases the size of DOS by 737 bytes. The second program, dMap.exe provides the graphics functions necessary to draw maps on the screen. dMap.exe increase DOS by approximately 67k. The large size represents in addition to the code, a 64k data area reserved for use by dMap. Fig. 5-3 illustrates the sequence of memory allocation and usage by DOS, Graphics.com, dMap.exe and dBase II.

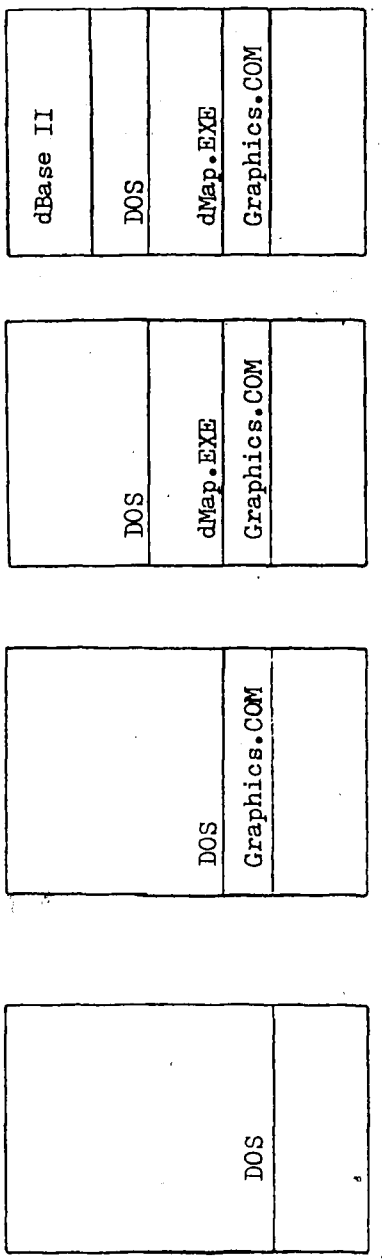

Figure 5-3: Transitive Memory allocation

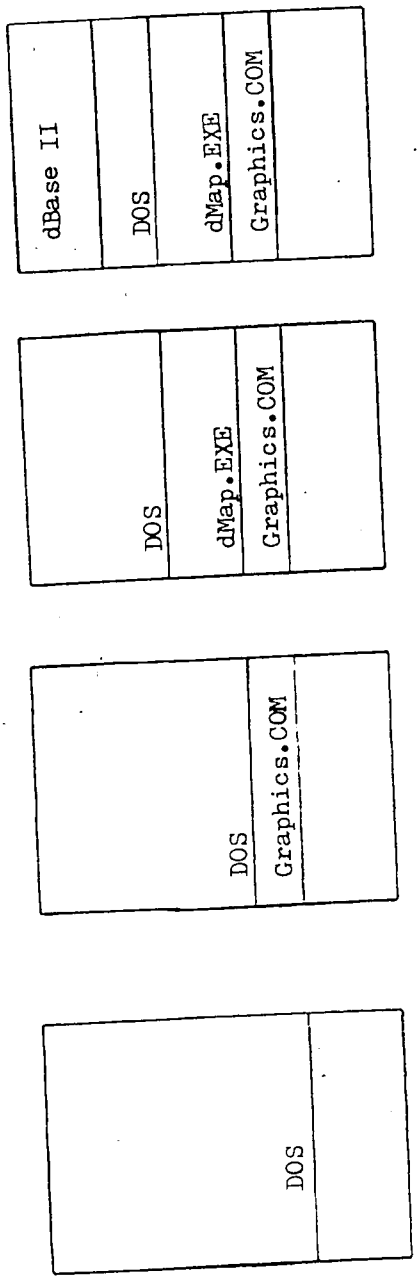

Figure 5-3: Transitive Memory allocation

6. Data files

6.1 dMap Database files - .DBF

6.1.1 Screen formatting - Parametr.DBF

Application programs have a tendency to perform tasks repetitively. Provisions are therefore made to eliminate as much of the repetitive work as possible. This includes designing report formats, input formats, and has been extended to the screenlayout for dMap's map drawing. The decision was made to maintain screen formats for map drawing in a database file, Parametr.DBF so that a standard format could be easily maintained. Each record contains the following information:

- screen name
- number of maps to be drawn on the screen
- names of the maps to be drawn on the screen
- titles of the maps to be drawn on the screen
- colors in which to draw the maps

Each screen is indexed on a key, the screen name, to facititate selection. A selected format is copied to Header.TXT, an SDF file to be read by the graphics program. File specifications for this parameter file may be found in Table 8-1 in the appendix.

6.1.2 Data point selection - Mappoint.DBF

Selection of the data points to be displayed on a map(s) is not difficult but does require the user to be familiar with the FOR <expression>. For each map to be displayed in a screen format, the user is allowed to access information from one database file. A valid For <expression> is entered from the keyboard which will copy the NAME field of the desired records to a work file. Examples of valid FOR <expressions>:

```
FOR Type_of_surgery = 'Heart'
FOR T_o_s = 'heart' and No_of_operation > 15
```

A second FOR <exp> is used to select records in the Mappoint.DBF file which will be copied to Mppt.TXT.

```
FOR PTNAME = s.NAME
```

As illustrated above the fields Ptname(Mappoint.DBF) and Name(User.DBF) must be of the same type and width, or the test for equality can not be performed.

Two additional fields, mapnumber and colorcode are appended to a Mappoint working file so that information that generated by PTSLCT can be passed to the graphics program. For example, each user defined FOR <expession> generates a separate colorcode. The information that is actually passed to the graphics program includes the X and Y world coordinates, a mapnumber and a colorcode. Tables

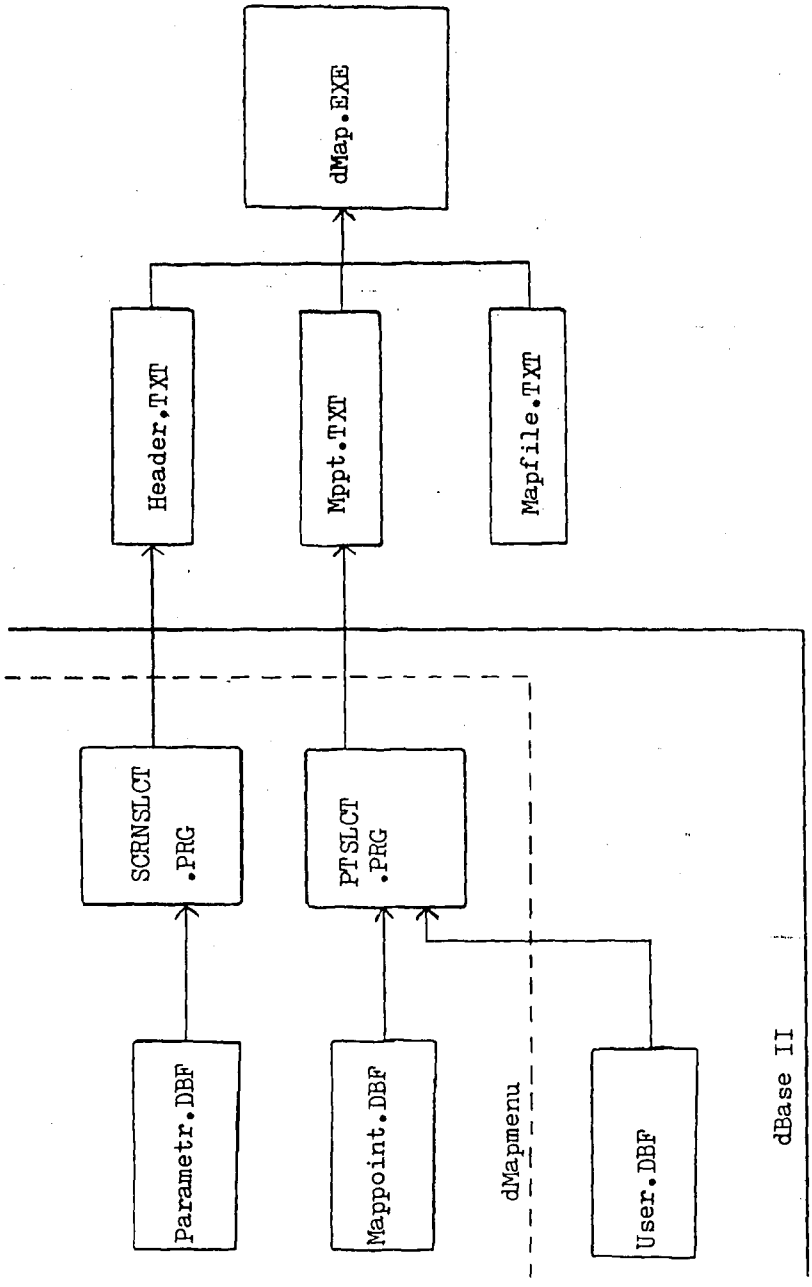

Figure 6-1: Data file environments

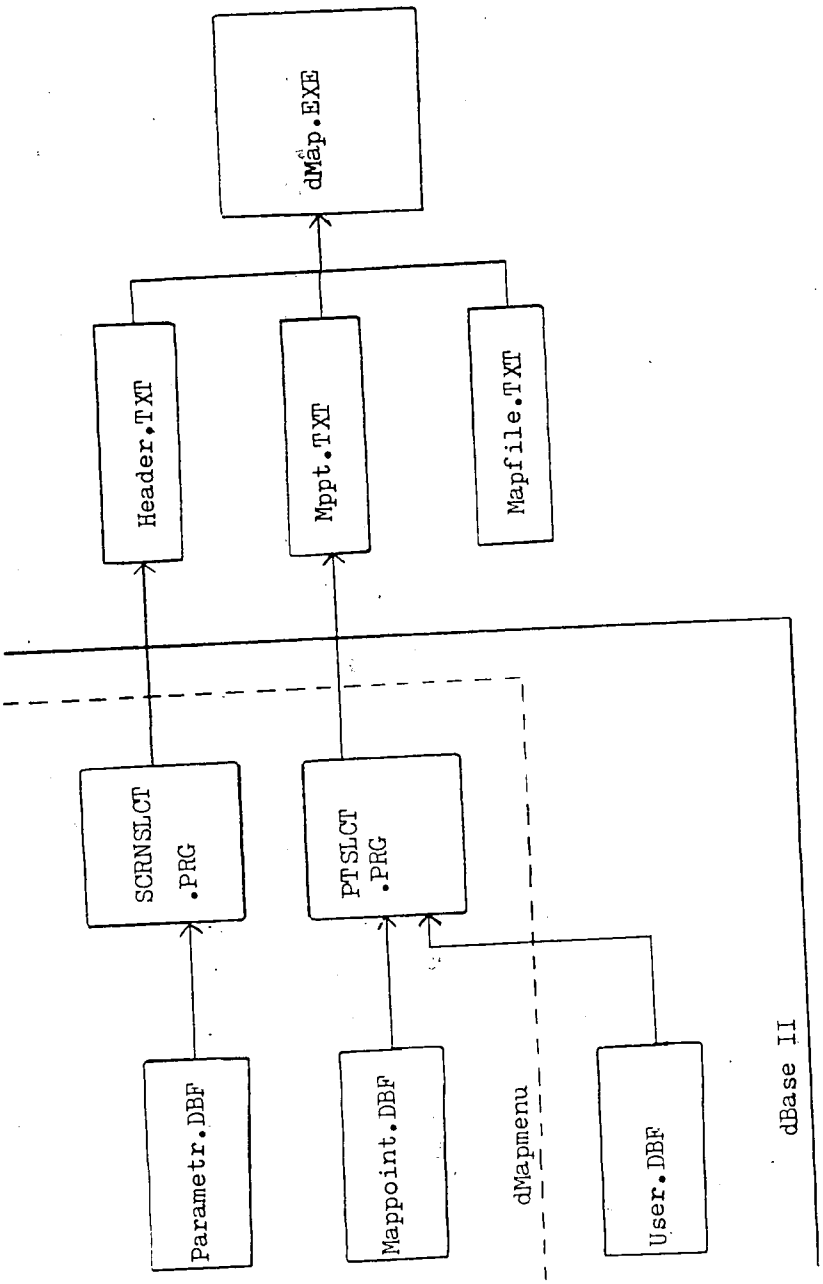

Figure 6-1: Data file environments

8-2, 8-3 in the appendix define structures for possible Mappoint.DBF and User.DBF.

6.2 Map file creation

The method of creating map points is to decimalize the longitude and latitude of the area to be drawn. This creates a standard by which to interchange the points of various maps. Longitude is represented as X and latitude is represented as Y. The accuracy of decimalizing is not entirely satisfactory if maps of large areas are to be drawn. There is however, no calculation provided for the change in size of the basic units of degrees, minutes, seconds on a global basis. The effect however, is minimal on a level of state, county, city, etc. If the exact location is known, in degrees, minutes, seconds the conversion is:

degree + minutes/100 + seconds/1000

Included in the mapfile as the first line must be two sets of X,Y coordinates representing the lower left and upper right corners of the window to the world coordinate system in which to draw the map. The second and subsequent lines of the mapfile are X and Y coordinates of the endpoints of lines in the map. A sample mapfile is given below.

```
76.250 40.400 75.235 40.905
75.900 40.555
75.487 40.413
75.425 40.475
75.900 40.555
```

The data points are in a floating field format, separated by at least one blank.

7. Technical Discussion

7.1 Interfacing Options

When deciding to create an Add-in graphics program for dBase II , there were two interfacing options available.

- Interrupt function 31, KEEP
- Interrupt function 4B, EXEC

The first option, KEEP, makes the program a resident part of DOS as an interrupt handler. Execution of the program occurs when a user defined interrupt is issued from another program, in this case the dBase II interface, Iface. The second option loads and executes a program, returning to the origin of the interrupt call. The second method was decided against for several reasons.

- Whenever a program is executed using EXEC, it is necessary to load the program from disk. This is an unnecessary time overhead if there is frequent travel between the two programs.
- All available memory is allocated to a program when it is loaded. Prior to issuing the EXEC function, memory allocated to the calling program, in this case dBase II, must be released for use by the EXEC function. We are not sure of the effect this has on dBase II when control is returned to it. As there is no clear indication of the location of dBase II's stack and it may possible that some portion of dMap overlays the dBase II stack area and destroys its contents.
- Due to the unavailability of a suitable assembler, the assembly language interface, Iface must be hand-assembled. It is significantly larger than the seven byte sequence required by the first method.

7.2 Technical Discussion - Assembly Language Interface

The assembly language code that is poked into memory by dBase II is a short seven byte sequence. The three basic actions in order are:

1. issue the interrupt 51 to give control to the graphics program

```
INT 51
```

2. Restore the segment registers, DS and ES

```
PUSH SS
PUSH SS
POP  DS
POP  ES
```

3. Return to dBase II

```
RET
```

The registers, SS and CS are restored to the values initially passed from dBase II to the assembly language interface by dMap and interrupt return respectively. The address in the interrupt vector table is the entry point into the graphics driver program of the initial procedure of dMap. This is illustrated in Fig. 7-1.

7.3 Technical Discussion of Dbase II

Ashton-Tate has provided the means to load and execute an assembly language program within dBase II. The easiest method, as recommended by Ashton-Tate, loading the program requires an object file in Intel Hex Format that is loaded directly into memory using the LOAD command. This method was rather impractical for us because

the assembler currently available to us, MASM does not create an object file of the Intel Hex Format. The alternative method is assembling the program either by hand or by using the assemble command available with the IBM debugger, DEBUG. If this method is used, after the program is assembled, it is necessary to convert the hex values to decimal values. The decimal values are then poked into memory using the dBase II POKE command. Execution is initiated in both methods via the CALL command.

Information provided by Ashton-Tate indicated that an assembly language program could be poked into memory starting at offset 56832d (DE00h). No size limit was specified, but information gathered using the debugger indicates the 64k boundary of dBase II is probably the limit.

Using the debugger, it appears that dBase II creates a special user stack, setting the stack pointer at D0BBh. On the stack at the time control is passed to the assembly language program is the offset but not the segment of the return address. It was recommended that no registers be saved by the user's program, we found however that this recommendation did not include the segment registers. It was probably assumed by Ashton-Tate that the user's program would have no need to change the segment registers which would also explain why only the offset was initially placed on the stack. When command is passed to the user's program, by dBase II,

each segment register contains the code segment value and the BX register contains the offset address of the parameter string, if the optional memory variable is used in the CALL <memvar> command.

A problem was encountered when we attempted within a command file to poke a value into memory without first setting dBase II's TALK parameter to off. A subsequent PEEK revealed that the memory location in question had a value of null. There is no documentation of this problem. This was not an inconvenience as one would normally expect to have the TALK parameter off within a command file.

7.4 Technical Discussion - dMapCS

The standard Lattice 'C' provides an assembly language module which is the first item linked with the user's program in the creation of an executable program. This module provides both the entry to and the exit from the user's C program. It is also reponsible for creating the stack, defining the dynamic and static data areas, and processing the command line. In our application, this module has been changed to allow the dMap.exe, to stay resident in memory as an interrupt handler.. Further references to this modified module shall be as dMapCS. Fig. 7-1 illustrates the change in program flow resulting from modifications made to the original file, CS.ASM.

7.4.1 Terminate and stay resident

The first procedure in dMapCS, Vect, performs several tasks, the most important is setting of Interrupt 51 to the offset of the second procedure in dMapCS, C, which is the entry point to the graphics program. Vect then calculates the number of paragraphs which must be retained when the program terminates. This number includes the number of paragraphs of code and that of data. Exit to DOS is via interrupt function 31, KEEP rather than the normal interrupt 20. At one time this procedure also saved the DS segment register as an interrupt until we discovered that DOS maintains the value of DGROUP, which is the address required by DS. This point should be noted if this program is interrupted by anything other

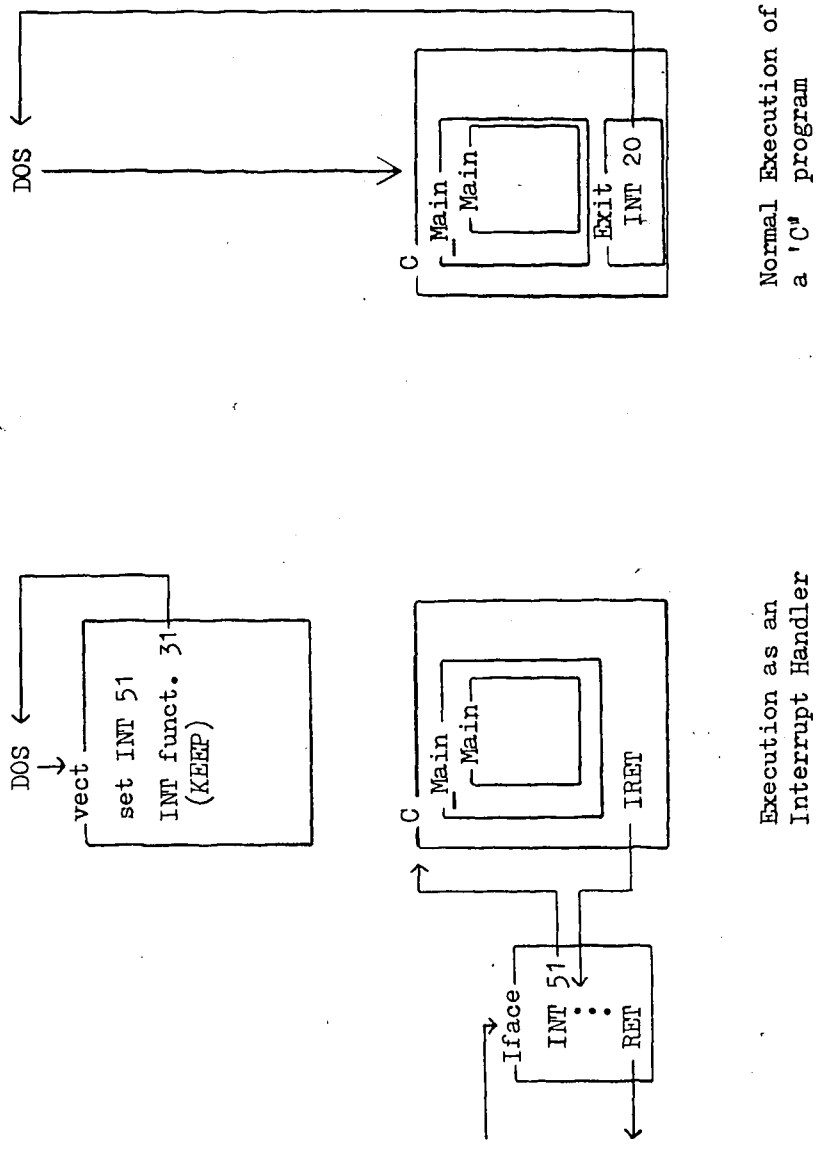

Figure 7-1: Execution environment for dMap.exe

Figure 7-1: Execution environment for dMap.exe

than dBase II, in particular, another .EXE program.

7.4.2 Modifications to CS.asm

The entry point to dMap.exe from interrupt 51 is made to procedure C. The first step is the restoration of the DS segment register, after which the SS and SP registers of the calling program are saved. It was first thought to save them on the stack but it was thought to be safer to store them as variables.

As few changes as possible were made to the rest of the procedure. The next change set a fixed date area of 64k, and set the stack pointer to the top of this area. Originally, we tried to set a data area of 16k but found that this was not large enough. The Halo library routines alone require 18k. We did not try any smaller numbers.

The last changes were the restoration of the registers, SS and SP belonging to the calling program and the issuing of an IRET (interrupt return). The interrupt return restores the CS, IP and flags before returning to the calling program.

7.5 Modificaton to Main.C

A single change was made to the library function _MAIN which is called from dMapCS and in turn calls MAIN, in the user's program. _Main upon return from MAIN called EXIT(0) , this is call was elimimated and control returned to dMapCS where the IRET is executed.

7.6 Add-on option

When the dMap.obj is linked is a standard configuration, it will act as a normal standalone program. This is a very important feature since it enabled most of the testing of dMap to be done independantly of other programs. dMap will handle any ASCII data files that are properly set up and hence may be used with any database management system that can create standard ASCII files. File specifications for all files may be found in the appendix.

8. Conclusions

8.1 Results

The development of the dMap Package has resulted in a viable interfacing technique that is not limited to creating interfaced between dBase II and C programs. The dBase II interface, Iface, may be used with any program that can be installed as an interrupt handler as long as proper restoration of the stack is made by the interrupt handler. Conversely, dMapCS allows and C program to be used as and interrupt handler. Programs that can issue an interrupt, either directly or through a small interfacing procedure, as was necessary with dBase II, may interface with a C program. Again, proper restoration of registers is necessary.

The other aspect of the package was the development of the procedures necessary for the generation of the maps. Neither of the programs created on either side of the interface, dMap.prg or dMap.C have been developed to their fullest potential. Both are awkward to use and somewhat limited in scope.

8.2 Further work

At this time, there is no facility for interactive design of the maps to be drawn. It is an item which would best be done through the use of a lightpen, joystick or mouse, none of which were available for this project. The same may be said for the creation

of the points on the maps themselves. In this case, a lightpen is preferable. Cursor control using the keyboard is not satisfactory for use with graphics where the desired point is a pixel and not a character as in an editor. Halo and 'C' both provide capabilities for using a lightpen and the matter should be persued to complete the project.

Another area to be worked on is the presentation of more than one defined map, in the same viewing area, for example; the maps of Lehigh and Northampton counties within the state of Pennsylvania.

I. Appendix

I.1 Hardware configuration

- IBM XT
- 512k Memory
- IBM Color Graphics Board
- IBM Color Monitor
- IBM Fixed Disk
- IBM 5 1/4 Inch Disk Drive

I.2 Software Requirements

- PC DOS version 2.00
- dBase II version 2.4
- Lattice 'C' compiler
- C Food Smorgesbord
- HALO Library of Graphic Routines
- MASM Assembler
- DEBUG.com Software debugger
- Personal Editor

File Name: Parametr.DBF

Name	Type	Width
Scrnname	Character	15
Format	Numeric	1
Map1nam	Character	14
Map1title	Character	25
Map1col	Numeric	1
Map2nam	Character	14
Map2title	Character	25
Map2col	Numeric	1
Map3nam	Character	14
Map3title	Character	25
Map3col	Numeric	1
Map4nam	Character	14
Map4title	Character	25
Map4col	Numeric	1

Table 8-1: Parametr.DBF file structure

File Name: Mappoint.DBF

Name	Type	Width	Dec
Ptname	Character	15	
Xcoor	Numeric	8	3
Ycoor	Numeric	8	3

Table 8-2: Mappoint.DBF file structure

File Name: User.DBF

Name	Type	Width
NAME	Character	15
Typesurgery	Character	15
Surgery_nos.	Numeric	4
Beds	Numeric	4
OR_s	Numeric	2

Table 8-3: Possible user's database file structure

References

Books

1. Dinerstein, Nelson T. dBase II For the Programmer, A How to do it Book. Glenview, Ill.: Scott, Foresman and Co. 1984.

2. Foley, J.D., and VanDam, A. Fundamentals of Interactive Computer Graphics. Reading, Mass.: Addison-Wesley Publishing Co., 1982.

3. Green, Adam B. dBase II User's Guide. Arlington, Mass.: SoftwareBanc, INC., 1984.

4. Kernighan, B.W., and Ritchie, D.W. The C Programming Language. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1978.

5. Kochan, Stephen G. Programming in C. Hasbrouck Heights, New Jersey: Hayden Book Co., Inc., 1982.

6. Morse, Stephen P. The 8086/8088 Primer, An Introduction to Their Architecture, System Design, and Programming. 2nd ed. Rochelle Park, New Jersey: Hayden Book Co., Inc., 1982.

7. Prague, C.N., and Hammitt, J.G. Programming with dBase II. Blue Ridge Summitt, Penna.: Tab Books, Inc., 1984.

Manuals

8. The C-Food Smorgasbord. New York, New York: Lifeboat Associates, 1982.

9. HALO. New York, New York: Lifeboat Associates, 1982.

10. Lattice 8086/8088 C Compiler.
New York, New York: Lifeboat Associates, 1982.

Unpublished Manuels

11. dBase II Assembly-Language Relational Database Management System. Ashton-Tate, 1982.

12. Disk Operating System for the IBM Personal Computer. version 2.00. Microsoft, Inc., 1983.

13. Macro Assembler for the IBM Personal Computer. International Business Machines Corp., 1981.

14. Technical Reference. International Business Machines Corp., 1983.

Articles

15. Graham, James D. "Extenders: Pushing dBase II to the Limit". PC Magazine, Feb 7 (1984), pp. 167-170.

16. Webster, Robin. "Three Easy Add-Ons for dBase II". PC Magazine. Feb 7 (1984), pp. 147-158.

VITA

Place of Birth:	Manchester, Connecticut
Date of Birth:	June 22, 1954
Education:	Lafayette College, Easton, Pa. 1976 A.B. majoring in Biology
Experience:	1983-1984 Pentamation Enterprises, Inc. Systems Programmer
	1983 Lehigh University Consultant - Computing Center
	1977-1982 Lehigh University Research Technician - CMES