NASA Technical Memorandum 89136

AN INTERACTIVE EDITOR FOR DEFINITION OF TOUCH-SENSITIVE ZONES FOR A GRAPHIC DISPLAY

Burt L. Monroe, III
Denise R. Jones

APRIL 1987

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

AN INTERACTIVE EDITOR FOR DEFINITION OF TOUCH-SENSITIVE ZONES FOR A GRAPHIC DISPLAY

Burt L. Monroe, III
Denise R. Jones
NASA Langley Research Center

SUMMARY

In the continuing effort to develop more efficient man-machine communications methods, touch displays have shown potential as straightforward input systems. The development of software necessary to handle such systems, however, can become tedious. In order to reduce the need for redundant programming, a touch editor has been developed which allows a programmer to interactively define touch-sensitive areas for a graphic display. The information produced during the editing process is written to a data file, which can be accessed easily when needed by an application program. This paper outlines the structure, logic, and use of the editor, as well as the hardware with which it is presently compatible.

INTRODUCTION

As research in efficient man-machine interfaces has expanded, touch input systems have become increasingly available. Although a wide variety of systems exist, the needed software is usually similar and often repetitive, requiring numerous lines of mundane code. Consequently, a "touch editor" has been developed to simplify the task of implementing touch screens as input devices for interactive graphic displays.

Without the touch editor, definition of touch-activated zones for a graphic display is difficult. The programmer must manually determine the touch coordinates which define the boundaries of the desired areas. Without visual feedback, this process can cause inaccurate or even illogical results. If touch areas are defined in an incorrect screen location, or if two or more touch areas are inadvertently defined so that they overlap, the software which utilizes them will not function as expected. The programmer has the additional task of storing the coordinate data for later use. This requires the definition and initialization of a data structure or file which, with several displays involved, can become large and unmanageable.

This time-consuming effort is reduced by the use of the touch editor. The editor allows the programmer to define touch areas interactively, using touch. The programmer can individually define each touch-sensitive zone by touching the display in one corner of the desired area, and then, before leaving the screen, moving to the opposite corner of the area. Coordinate information which defines the area is then written to a data file which can be easily accessed when needed.

This paper documents all aspects of the touch editor. The utilization of the editor and its present hardware and software compatibility are outlined to allow for direct use of the editor. Also, since incompatibility with a different host computer, graphics generator, or touch input system might prevent such direct use, the structure and logic of the editor are also discussed in order to simplify the process of transporting the touch editor to a different system.

TOUCH EDITOR

The touch editor is linked as a subroutine to the display-generating software. After answering questions concerning the format of the desired data file, the user defines each area individually by touch. The user touches the screen in one corner of the desired zone, and then moves to the opposite corner of the zone without leaving the screen. As this is done, a rectangle is drawn on the monitor showing the user the touch zone while it is being defined as shown in figure 1. The rectangular area will expand dynamically with the touch movement, in a manner similar to a rubber band, and stop when the user exits from the screen. When the area is defined to the programmer's satisfaction, the coordinate data are written to the file. The programmer can confirm visually that touch areas are of the appropriate size and in the correct display location. Additionally, diagnostic routines in the editor will alert the user when touch areas overlap and allow for correction of the error. The application program can then provide the appropriate response to a touch input by using the data file to easily determine which, if any, area has been pressed.

HARDWARE/SOFTWARE COMPATIBILITY

The touch editor is implemented in VAX-11 FORTRAN on a Digital Equipment Corporation VAX-11/780 minicomputer. Graphics routines are written in the Real-Time Animation Package (RAP), a high level graphics language used to control the Adage, Inc. Adage-3000 programmable display generator. RAP is similar to the C programming language and was developed by Research Triangle Institute (RTI) (ref. 1). Any VAX program which controls a graphic display by communicating with a RAP procedure can use the editor in its present implementation. This may provide limited applications due to incompatibility with other computers, graphics generators, or touch screens; therefore, the editor has been developed in a structured, well-documented manner to allow for straightforward transportation to other systems.

PROGRAM DIVISIONS

The touch editor has been developed in a modular fashion to provide logical structure and greater readability. The major divisions of the touch editor system are described below.

MAIN PROCEDURE

The subroutine "TOUCH-EDITOR" comprises the main procedure, the major phases of which are described below. This subroutine is located in the file "SUBFED.FOR." A listing of this file is presented in appendix A.

Initialization Phase.- In the initialization phase, communications channels are established with the graphics process and touch screen, commands necessary to establish the proper operating parameters of the touch screen are executed, and the user is allowed to define the format of the data file and to provide information concerning the touch-sensitive display.

The routines which establish communications channels with the graphics generator and the touch screen are external to the touch editor. The procedures to allow transferral of data to the graphics process were written in VAX-11 MACRO by RTI. These procedures are necessary to transfer data to the Adage-3000 processor from a VAX program. The routines used to initialize communications with the touch screen and establish operating parameters are screen-dependent. Therefore, they reside among the modules of the touch screen interface (See appendix B).

The remainder of the initialization process consists of user definition of the data file format. The user is asked to name the data file and to provide a file status. In effect, this will determine whether an old file will be extended or a new file created. The programmer is also asked to provide information concerning the display and the number of touch areas involved. A sample run of the touch editor is included in appendix C to illustrate this phase.

Area Definition Phase.- In this phase, touch information, initiated by the pressing of the screen by the user, is processed to define a touch-sensitive zone. The touch coordinates of the initial point of contact are used as a stationary corner of the touch area. As the user moves across the screen surface, the new coordinates define the opposite corner of the zone and are used to display the yellow rectangle. Visually, this has an effect similar to that of stretching a rubber band away from a fixed point. On exit from the screen, the area is defined. The user is allowed to redefine the area if it is unsatisfactory. Also provided is an opportunity to name each touch area, a useful option should the need arise to examine or alter the contents of the data file.

Several sections of fault-tolerant error-checking code are included. Due to differences in touch screen sampling rates and the scanning speed of the FORTRAN code, occasional errors in the receipt of touch information will occur. The editor will check for such errors, but will halt operation and restart only after several errors have occurred consecutively. Isolated transmission errors are ignored without affecting the editor operation. The filtering procedure, provided by the touch screen interface, is also called at this point. Without the filter, the rectangle will change with even slight variation of the touch coordinates, often caused by imperceptible movement of the touch or inaccuracies due to the analog nature of some touch screens.

This can have an unsettling visual effect, especially when the touch appears stationary to the user.

Data Storage Phase.- The editor allows the user to define a touch zone by pressing any corner of the rectangular area and moving the touch in any direction. As a result, the X and Y coordinate values which define the final corner may be greater than or less than those of the initial corner. Therefore, a sorting algorithm is implemented to insure that data records will be uniform. The data are then written to the requested file in a specified format to allow for straight-forward future access. A data file created with the editor is included in appendix C to illustrate this format.

Touch Screen Interface

All routines which handle touch screen communications have been developed as screen-dependent modules. They are, therefore, external to the main editor code. The routines which must be included in a touch screen interface in order for the editor to be utilized are explained below. Presently, interfaces have been developed for touch systems from MicroTouch Systems, Inc. and Carroll Touch Technology (refs. 2 to 4). Routines for other touch screens can be created through modification of one of these existing interfaces. Appendix B contains a listing of the file "MICROTCH.FOR," which contains the MicroTouch touch screen interface. The command procedure which must be run in order to establish the proper communications parameters (i.e., baud rate) with the MicroTouch screen is presented in Appendix D.

"INIT SCREEN."- This module is responsible for execution of commands necessary to establish the proper operating parameters of the touch screen. Some touch systems allow the user to request an operating mode which determines, for instance, whether data will be sent only once upon an initial touch of the screen or continuously until contact with the screen is broken. Data format and size of coordinate system are among other possible parameters that might require modification.

"INIT CHAN."- This routine is responsible for the initialization of a communications channel (RS-232) between the VAX-11-780 and the touch screen.

"GET TOUCH."- This is the main procedure of the touch screen interface. This module will check for information from the touch screen, interpret any data, and return touch coordinates with a status code. Touch coordinates will be integer values and fall within a range determined by the resolution of the touch screen's coordinate system. The status code will be one of the following characters:

"T" - Touch
"E" - Exit
"N" - No touch present
"F" - Data format error
"R" - Coordinate out of range

Touch data consists of coordinates which indicate the location of an actual touch of the screen. Exit status indicates that the accompanying coordinates were the location of the last detected touch, and that there is no longer a touch present. "No touch present" differs from exit data in that no touch data immediately precedes it. A data format error indicates that there was a transmission error, therefore, data may not be accurate and must be ignored. A coordinate out of range is also the result of a transmission error and must be ignored.

"FILTER."- The filtering routine is used by any procedure which requires touch coordinate data to differ by a certain preset amount before the change is recognized. The touch editor is an example of such a procedure.

"SNDMSG."- This module is responsible for communications to the touch screen. For those touch screens which can respond to commands, this routine will receive a character string as a parameter, concatenate it with the correct communications characters, and send it on the appropriate channel to the touch screen. For instance, in the case of the MicroTouch interface, this routine will accept the string "FH**," and send the string "<SOH>FH<CR>" to the screen. This causes the screen to report coordinate data in hexadecimal format. Some touch screens do not have the ability to recognize parameter changing commands, and would not be able to utilize this procedure.

"COORDMAP."- This subroutine will return a pair of graphic screen coordinates when sent a pair of touch screen coordinates. The routine consists solely of a pair of linear equations which map the coordinates directly. The exact form of these equations depends on the resolution and origin location of the particular touch screen.

Graphics Module

The purpose of this RAP routine is to define the data structures necessary to display a yellow, flashing rectangle. When the editor is executed, the routine becomes part of a continuous loop which resides in the Adage-3000 processor. As the screen is touched, the coordinate information is processed and downloaded to the RAP routine which causes the rectangle to be updated and rendered over the original display. This RAP routine is contained in the file "SUBGED.G" (Appendix E).

Graphics Communications Module

In the communications module, information is transferred to the graphics process in the Adage-3000, which runs in parallel to the main editing process in the VAX-11/780. The data, which were obtained from the touch screen, are mapped to screen coordinates and arranged in a FORTRAN array. This array has the same format as the RAP data structure which defines the rectangle. The information is passed directly to the RAP data structure, using an address obtained during the initialization process. This updated coordinate data is then used when rendering the rectangle on the monitor. Graphics

communications routines are within the routine "INIT_EDITOR," contained in the file "SUBFED.FOR" (Appendix A).

Diagnostics Module

Immediately following the complete definition of each touch area, excluding the first, the diagnostic module will examine the data for illogically defined touch areas. If any corner of one touch area lies within the boundaries of another area, the two overlap. This module will check for such errors and, when they occur, inform the user of the conflict and which zones are involved. At this time the programmer may elect to redefine an area. After redefinition, the new data record will replace the previous record. This process will continue until there are no more conflicting areas or the programmer elects to cease definition. Program listings of the routines "FILECHECK" and "REPLACE_RECORD," which comprise this module, are included in appendix A.

IMPLEMENTATION

The following section provides information concerning the preparation required to use the touch editor directly and the utilization of the data files created during the editing process.

Preparation of Touch Editor

This section assumes a knowledge of the procedure for interfacing a RAP program with a VAX host process, as this knowledge is necessary to develop a program which can use the editor directly. Further information concerning this procedure can be found in reference 5.

The touch editor is essentially a set of files containing subroutines to be linked to the user's display-generating software. These files must be included in BUILD (.BLD) files as appropriate. Also, the original display-generating procedures themselves must be modified to include calls to the editor subroutines.

The FORTRAN section of the editor must be linked to the main routine of the user's procedure by including the following files in the FORTRAN BUILD file. Note that the final file is the touch screen interface and that only one of the interfaces should be included:

```
subfed.for
carroll.for (OR microtch.for)
```

To link the RAP modules of the procedure, include the following file in the RAP BUILD file:

```
subged.g
```

The following line must be inserted into the user's FORTRAN procedure immediately following the transmission of data for the display:

```
CALL TOUCH_EDITOR
```

In order to call the graphics routine, two lines must be added to the user's RAP procedure. The "subged" routine must be declared external in the declaration section of the program and then must be called from within the infinite loop after the rendering of the display:

```
extern subged [];
   :
   :
while (1)
{
   :
   :
render(scene); /* DISPLAY GRAPHICS */
subged ();     /* DRAW RECTANGLE   */
   :
   :
}
```

The procedure must then be recompiled and linked using the MAKE command. The new executable file can then be run using the programmer's original command procedure. At this point, a data file can then be created as described above.

Utilization of Data Files

Utimately, the data file created during the editing process will be used to interpret touch inputs to the user's application program. A routine which compares touch input with coordinates from the data file has been developed to simplify this interpretation process. When this subroutine is passed data file coordinates and a pair of touch coordinates, the number of the touch area to which the input corresponds will be returned. A flag value of -1 will be returned if the touch input falls outside of all zones defined for the display.

During the initialization phase of the application program, the coordinate data should be read from the data file and stored in arrays. The routine "READ_TOUCH_DATA," which is located in Appendix F, accomplishes this. The application program must then use the routines "GET_TOUCH" and "COORDMAP" of the touch screen interface to receive input coordinate data from the screen. The interpretation routine, "GET_TOUCH_AREA," can then be called, allowing the application to provide the appropriate response. "GET_TOUCH_AREA" is located in the file "TCHAREA.FOR," a listing of which is provided in Appendix G.

CONCLUDING REMARKS

The editor has been used to define data files for several menu-driven displays. These data files have then been utilized to allow a user to operate the menus with touch input. The touch editor has provided a considerable decrease in the effort needed to integrate touch screens into the menu system, allowing researchers to devote time to the evaluation, rather than the implementation, of touch input. An effort has, therefore, been made to provide documentation concerning the editor in order to minimize the time which might be necessary to develop such an editor for other systems.

APPENDIX A

```
*****************************************************************
*                                                               *
*                          TOUCH EDITOR                         *
*                                                               *
*****************************************************************
*                                                               *
*    Filename:  SUBFED.FOR                                      *
*                                                               *
*       This subroutine allows a programmer to interactively    *
*    define touch areas for a graphic display.  These touch areas *
*    will be used to interpret touch inputs to an application   *
*    program.                                                   *
*       The subroutine will ask the user for the name of a file *
*    where data obtained from the editing process is to be      *
*    written.  If the file is "new", it will be created.  If it *
*    is "old", the editor will write data at the end of the file, *
*    maintaining any data that were already there.  Note that if *
*    the programmer wants to use the name of a file already in  *
*    existence, and does not wish to keep any of the present    *
*    data, the file should be declared 'NEW'.                   *
*       The editor will also ask the programmer to assign a     *
*    number to the display being edited.  This should be '1'    *
*    unless the data are going to an old file where there are   *
*    data from previous displays.  The programmer will also     *
*    be prompted for the number of touch areas to be defined    *
*    for the display.                                           *
*       For each touch area, the programmer defines a box       *
*    which surrounds the area to be chosen.  This is done by    *
*    pressing the screen at one corner of the desired box, and  *
*    then moving to the opposite corner of the box, without     *
*    leaving the screen surface.  The touch should not be       *
*    removed from the screen until the area is as desired.  The *
*    program will then ask if the area is correctly defined.  If *
*    so, the programmer is asked to provide a name for the area *
*    (for readability of the data).  A data record is then      *
*    entered in the file.  Each data record contains the display *
*    number, the touch area number, the lowest X value, highest X, *
*    lowest Y, highest Y, and the area name.  The touch zone is *
*    defined by the X and Y values and identified by the other  *
*    data.                                                      *
*                                                               *
*****************************************************************
*                                                               *
*    VARIABLE DICTIONARY                                        *
*    -------------------                                        *
*                                                               *
*  AREANM      - Name given to a touch area by the user. Used for *
*                identification of area within the data file.   *
*  AREAS       - Total number of touch areas within the display. *
*  BY          - Bottom Y.  Largest Y coordinate which identifies *
*                the area.                                      *
```

```
*  CONFIRM     - A character variable input by the user to indicate *
*                satisfactory completion of the touch area.         *
*  CORAR       - Contains the number of the touch area being        *
*                corrected.                                         *
*  CORRECTING  - Logical flag to allow definition loop to be        *
*                executed while correcting an overlapping area.     *
*  DISPLAY     - Input by user to indentify the present graphic     *
*                display.                                           *
*  DRAWENABLE  - A flag used by the RAP program to indicate         *
*                that a rectangle has been defined and it should be *
*                drawn.                                             *
*  EDITOR_BLK  - Array which holds all information used by the RAP  *
*                process to draw a rectangle.  EDITOR_BLK(1) holds  *
*                the DRAWENABLE flag.  The remainder of the array   *
*                is similar to the data structure CORNERS within    *
*                the RAP process.  This data structure defines the  *
*                four points which are to be connected to form the  *
*                rectangle.                                         *
*  ENDX        - X coordinate of the changing corner of the touch   *
*                area.                                              *
*  ENDY        - Y coordinate of the changing corner of the touch   *
*                area.                                              *
*  ERROR_CNT   - Counter used to allow for fault-tolerance.  This   *
*                must exceed a preset amount to cause a restart.    *
*  FILENM      - Contains the name of the data file.                *
*  FILEST      - File status.  Input by user to indicate whether    *
*                the file is old or new.                            *
*  FLNMEND     - File name end.  Integer which indicates the length *
*                of the user input filenm to allow for .DAT         *
*                extension.                                         *
*  HOLDX       - Contains the previous value of ENDX.  Used to      *
*                allow for error-checking and filtering.            *
*  HOLDY       - Contains the previous value of ENDY.               *
*  I           - Loop counter. Contains the number of the current   *
*                area being edited.                                 *
*  LX          - Left X.  Smallest X coordinate which identifies    *
*                the area.                                          *
*  NORMAL      - Logical flag to cause definition loop to be        *
*                executed once for each area.                       *
*  RX          - Right X.  Largest X coordinate which identifies    *
*                the area.                                          *
*  SAMEBOX     - Logical flag used in definition loop to indicate   *
*                continuing editing of same touch area.             *
*  SAME_CNT    - Counter used to indicate that a touch zone has not *
*                changed in a number of consecutive scans.          *
*  STARTED     - Logical flag used to indicate that initial contact *
*                has been detected.                                 *
*  STARTX      - X coordinate of the initial corner of the touch    *
*                zone.                                              *
*  STARTY      - Y coordinate of the initial corner of the touch    *
*                zone.                                              *
```

```
*  STATUS      - Character returned by touch screen interface which *
*                has one of the following values:                   *
*                       'T' - Touch                                 *
*                       'E' - Exit                                  *
*                       'N' - No touch present                      *
*                       'F' - Format error                          *
*                       'R' - Range error                           *
*  TY          - Top Y.  Smallest Y coordinate which identifies the *
*                area.                                              *
*  X           - X coordinate of a touch, returned by touch screen  *
*                interface.                                         *
*  Y           - Y coordinate of a touch.                           *
*                                                                   *
*********************************************************************

        SUBROUTINE TOUCH_EDITOR

        INTEGER EDITOR_BLK(18)
        INTEGER STARTX,STARTY,ENDX,ENDY,LX,RX,TY,BY,X,Y
        INTEGER FLNMEND,DISPLAY,AREAS,DRAWENABLE
        INTEGER ERROR_CNT,SAME_CNT,HOLDX,HOLDY,CORAR
        CHARACTER STATUS*1,CONFIRM*1,AREANM*20,FILENM*20
        CHARACTER*1 FILEST
        LOGICAL SAMEBOX,STARTED,NORMAL,CORRECTING

        INCLUDE 'ACSMODELS:SYSTEM.CMN' ! Common block containing LUSYM

        COMMON /EDIT_INFO/  EDITOR_BLK
        COMMON /FILE_INFO/  AREAS,I,CORRECTING,FILENM
        COMMON /REPL_INFO/  DISPLAY,CORAR,LX,RX,TY,BY,AREANM

        DATA EDITOR_BLK /18*0/

C********
C
C  The initialization phase begins here.  IKOPEN and IKBSEI are
C  MACRO routines which are necessary to establish communications
C  with the Adage-3000.  INIT_EDITOR determines the address of the
C  data structure which will receive the coordinate data that
C  defines the rectangle.  INIT_CHAN establishes a communication
C  channel with the touch screen.  INIT_SCREEN establishes desired
C  operating parameters of the touch screen.
C
C********

        LUSYM=1
        CALL IKOPEN()
        CALL IKBSEI("1520)
        CALL INIT_EDITOR
        CALL INIT_CHAN
        CALL INIT_SCREEN
```

```
C********
C
C  The programmer is asked to enter the name of the data file.
C  NOTE: The editor will automatically give the file an extension
C  of '.DAT' so no extension should be provided.
C
C********

        TYPE *, 'ENTER NAME OF DATA FILE.'
        READ (5,10) (FILENM)
10      FORMAT(A20)
        DO 13 K=1,19
            IF (FILENM(K:K).NE.' '.AND.FILENM(K+1:K+1).EQ.' ')
     1          FLNMEND=K+1
13      CONTINUE
        FILENM(FLNMEND:FLNMEND+3)='.DAT'

C********
C
C  The programmer is asked to provide a status for the file which is
C  then opened accordingly.
C
C********

        DO WHILE(FILEST.NE.'O'.AND.FILEST.NE.'N')
            TYPE *, 'IS THIS AN OLD OR NEW FILE? (O/N)'
            READ (5,20) (FILEST)
20          FORMAT(A1)
        END DO

        IF (FILEST.EQ.'N') OPEN(UNIT=7,FILE=FILENM,STATUS='NEW')
        IF (FILEST.EQ.'O') OPEN(UNIT=7,FILE=FILENM,STATUS='OLD',
     1                          ACCESS='APPEND')

C********
C
C  The programmer is asked to provide information concerning the
C  display.  Some instructions are also given at this point.
C
C********

        TYPE *, 'PLEASE ASSIGN A NUMBER TO THIS TOUCH DISPLAY.'
        ACCEPT *, DISPLAY

        TYPE *, ' '
        TYPE *, 'HOW MANY TOUCH AREAS FOR THIS DISPLAY?'
        ACCEPT *, AREAS

        TYPE *, 'TOUCH SCREEN AT ONE CORNER OF TOUCH AREA.'
        TYPE *, 'MOVE FINGER TO OPPOSITE CORNER OF AREA.'
        TYPE *, 'WHEN AREA IS SATISFACTORY, REMOVE FINGER.'

        NORMAL=.TRUE.
        CORRECTING=.FALSE.
```

```
C********
C
C  AREA DEFINITION LOOP
C  --------------------
C
C  The logic to process the touch information in order to define
C  the touch-sensitive zones is located within.  This loop will
C  operate once for each touch area (NORMAL=.TRUE.), unless an
C  overlap error is detected and areas must be redefined
C  (CORRECTING=.TRUE.).  Several error-checking sections of code
C  are contained in this loop.  Note, however, that these are in
C  most cases fault-tolerant, requiring several consecutive
C  errors to cause a restart.  The definition of a touch area
C  is obtained by designating the initial touch coordinates as one
C  corner of the area (STARTX and STARTY).  Subsequent touches,
C  until exit from the screen, are used to provide the coordinates
C  of the opposite corner of the area (ENDX and ENDY).
C
C********

        DO 100 I=1,AREAS
            TYPE *, 'DEFINE AREA #',I

            DO WHILE(NORMAL.OR.CORRECTING)
                IF(CORRECTING) THEN
                    TYPE *, 'CORRECT AREA #',CORAR
                ENDIF
                SAMEBOX=.TRUE.
                STARTED=.FALSE.
                ERROR_CNT=0
                SAME_CNT=0
                HOLDX=-1
                HOLDY=-1

                DO WHILE(SAMEBOX)
C********
C
C  Obtain coordinates of touch input.
C
C********

30                  CALL GET_TOUCH(X,Y,STATUS)

                    IF (.NOT.STARTED.AND.STATUS.EQ.'N') GOTO 30
                    IF (.NOT.STARTED.AND.STATUS.NE.'T'.AND.
     1                  STATUS.NE.'N') THEN
                        TYPE*,'AN ERROR(1) HAS OCCURRED. PLEASE'
                        TYPE *, 'ATTEMPT AREA DEFINITION AGAIN.'
                    ELSE IF (.NOT.STARTED) THEN
                        STARTED=.TRUE.
                        STARTX=X
                        STARTY=Y
```

```
C********
C
C  Set DRAWENABLE ( EDITOR_BLK(1) ).  Drawenable is a logical flag to
C  indicate that a rectangle has been defined and should be rendered
C  to the display.
C
C********
                        EDITOR_BLK(1)=1

                        ENDX=STARTX
                        ENDY=STARTY
                        HOLDX=ENDX
                        HOLDY=ENDY
                    ELSE IF (STARTED) THEN
                        IF (STATUS.EQ.'R'.OR.STATUS.EQ.'F') THEN
                            ERROR_CNT=ERROR_CNT+1
                            IF (ERROR_CNT.GT.10) THEN
                                TYPE *, 'AN ERROR(2) HAS OCCURRED. PLEASE'
                                TYPE *, 'ATTEMPT AREA DEFINITION AGAIN.'
                                STARTED=.FALSE.
                            ENDIF
                        ENDIF
                        IF (STATUS.EQ.'T') THEN
                            ENDX=X
                            ENDY=Y
                        ENDIF
                        IF (ENDX.EQ.HOLDX.AND.ENDY.EQ.HOLDY
     1                      .AND.STATUS.NE.'E') THEN
                                SAME_CNT=SAME_CNT+1
                                IF (SAME_CNT.GT.10) THEN
                                    WRITE (*,*) 'BOX LOST.  TRY AGAIN'
                                    STARTED=.FALSE.
                                ENDIF
                        ELSE
                            SAME_CNT=0
                        ENDIF

C********
C
C  FILTER is a screen-dependent routine which requires touch coordinates
C  to change by a certain preset amount before a new rectangle will
C  be drawn.
C
C********

                        CALL FILTER(X,Y,HOLDX,HOLDY)

                        HOLDX=ENDX
                        HOLDY=ENDY
                    ENDIF
```

```
C********
C
C  Map touch screen coordinates to graphic display coordinates.
C  Draw the rectangular area onto the display.
C
C********

                    CALL COORDMAP(STARTX,STARTY,LX,TY)
                    CALL COORDMAP(ENDX,ENDY,RX,BY)
                    CALL DRAW_BOX(LX,TY,RX,BY)

C********
C
C  If no exit has occurred, continue to get touch data and update
C  touch zone.
C
C  On exit, the touch zone is defined.  Check for user error.
C
C********

                    IF (STATUS.EQ.'E') THEN
                        CONFIRM='A'
                        DO WHILE (CONFIRM.NE.'Y'.AND.CONFIRM.NE.'N')
                            TYPE *, 'AREA IS DEFINED. SATISFACTORY? (Y/N)'
                            READ(5,70) (CONFIRM)
70                          FORMAT(A1)
                            IF(CONFIRM.EQ.'N') THEN
                                TYPE *, 'PLEASE ATTEMPT DEFINITION AGAIN.'
                                STARTED=.FALSE.
                            ENDIF
                           IF(CONFIRM.EQ.'Y') SAMEBOX=.FALSE.
                         END DO
                    ENDIF
                END DO

                TYPE *, 'ENTER LOGICAL NAME OF TOUCH AREA.'
                READ(5,80) (AREANM)
80              FORMAT(A20)

C********
C
C  Sort X and Y data into low and high values to insure uniformity
C  of data records.  LX = Left X; RX = Right X; TY = Top Y; BY =
C  Bottom Y.  All directions are according to the display coordinate
C  system with the origin (0,0) in the upper-left corner.
C
C********
```

```
                IF(LX.GE.RX) THEN
                    IHOLD=RX
                    RX=LX
                    LX=IHOLD
                ENDIF
                IF(TY.GE.BY) THEN
                    IHOLD=BY
                    BY=TY
                    TY=IHOLD
                ENDIF

C********
C
C  Store data record and return if necessary.
C
C********

            IF (.NOT.CORRECTING) THEN
                WRITE(7,90) (DISPLAY,I,LX,RX,TY,BY,AREANM)
90              FORMAT(I2,1X,I2,1X,I4,1X,I4,1X,I4,1X,I4,1X,A20)
                NORMAL=.FALSE.
            ELSE
                CLOSE(UNIT=7)
                CALL REPLACE_RECORD(FILENM)
                OPEN(UNIT=7,FILE=FILENM,STATUS='OLD',ACCESS='APPEND')
                CORRECTING=.FALSE.
            ENDIF
            CLOSE(UNIT=7)
            CALL FILECHECK(DISPLAY,CORAR)
            OPEN(UNIT=7,FILE=FILENM,STATUS='OLD',ACCESS='APPEND')
            END DO
        NORMAL=.TRUE.

100     CONTINUE

        CLOSE (UNIT=7)

        STOP
        END

**********************************************************************
*                                                                    *
*       DRAW_BOX sends data needed by the RAP program to draw        *
*    the correct rectangle on the screen.  EDITOR_BLK is identical   *
*    to the data structure CORNERS contained in the RAP code.        *
*                                                                    *
**********************************************************************

        SUBROUTINE DRAW_BOX(AX1,AY1,AX2,AY2)

        INTEGER AX1,AY1,AX2,AY2
        INTEGER EDITOR_BLK(18)
```

```
      COMMON /EDIT_INFO/ EDITOR_BLK

      EDITOR_BLK(2)=17
      EDITOR_BLK(3)=AX1
      EDITOR_BLK(4)=AY1
      EDITOR_BLK(7)=AX1
      EDITOR_BLK(8)=AY2
      EDITOR_BLK(11)=AX2
      EDITOR_BLK(12)=AY2
      EDITOR_BLK(15)=AX2
      EDITOR_BLK(16)=AY1

      CALL SEND_INFO

      RETURN
      END

*****************************************************************
*                                                               *
*       FILECHECK is called immediately after the completion of *
*    each touch area.  It will read from the data file and check *
*    each pair of touch areas in the present display for overlap.*
*    If any two zones are in conflict, it will name the areas and*
*    allow for redefinition.  NOTE: Only one area may be named for*
*    redefinition at this time.  However, after redefinition, this*
*    routine will again check the file and allow another area to *
*    be redefined if there are still errors.                     *
*                                                               *
*****************************************************************

      SUBROUTINE FILECHECK(DISPLAY,CORAR)

      INTEGER AREAS,DISPLAY,CORAR
      INTEGER HDISP,HAREA,HLX,HRX,HTY,HBY
      INTEGER CDISP,CAREA,CLX,CRX,CTY,CBY
      CHARACTER REDEFINE*1,FILENM*20
      LOGICAL CORRECTING

      COMMON /FILE_INFO/ AREAS,I,CORRECTING,FILENM

      HDISP=0
      HAREA=0

      DO 300 JA=1,I-1

          OPEN(UNIT=7,FILE=FILENM,READONLY,STATUS='OLD')
          DO WHILE(HDISP.NE.DISPLAY.OR.HAREA.NE.JA)
              READ(7,200)(HDISP,HAREA,HLX,HRX,HTY,HBY)
200           FORMAT(I2,1X,I2,1X,I4,1X,I4,1X,I4,1X,I4,1X)
          END DO
```

```
            DO 250 JB=JA+1,I
                READ(7,200) (CDISP,CAREA,CLX,CRX,CTY,CBY)

C********
C
C  This condition, if found true, indicates that the corner of
C  one area lies on or within the boundaries of another area.
C
C********

                IF((((CLX.GE.HLX.AND.CLX.LE.HRX).OR.
     1               (CRX.GE.HLX.AND.CRX.LE.HRX)).AND.
     1              ((CTY.GE.HTY.AND.CTY.LE.HBY).OR.
     1               (CBY.GE.HTY.AND.CBY.LE.HBY))).OR.
     1             (((HLX.GE.CLX.AND.HLX.LE.CRX).OR.
     1               (HRX.GE.CLX.AND.HRX.LE.CRX)).AND.
     1              ((HTY.GE.CTY.AND.HTY.LE.CBY).OR.
     1               (HBY.GE.CTY.AND.HTY.LE.CBY)))) THEN

                    WRITE(*,*) 'AN OVERLAP ERROR HAS OCCURRED.'
                    WRITE(*,*) 'CONFLICTING AREAS: ',JA,', ',JB
                    CORRECTING=.TRUE.

                ENDIF
250         CONTINUE
            CLOSE(UNIT=7)
300     CONTINUE

        IF (CORRECTING) THEN
            REDEFINE='A'
            DO WHILE (REDEFINE.NE.'Y'.AND.REDEFINE.NE.'N')
                WRITE(*,*) 'DO YOU WISH TO REDEFINE AN AREA (Y/N)?'
                READ (6,350) REDEFINE
350             FORMAT(A1)
            END DO

            IF (REDEFINE.EQ.'N') THEN
                CORRECTING=.FALSE.
            ELSE
                WRITE(*,*) 'ENTER NUMBER OF AREA TO BE REDEFINED.'
                READ (5,*) CORAR
            ENDIF

        ENDIF

        CLOSE(UNIT=7)
        RETURN
        END
```

```
*****************************************************************
*                                                               *
*       REPLACE_RECORD will replace a data record which has been *
*    redefined due to an overlap error.  The original data file is *
*    copied to a storage file (STORE.DAT).  This is then written *
*    back to the original data file, line by line.  The new record *
*    is written to the file in its appropriate place.           *
*                                                               *
*****************************************************************

      SUBROUTINE REPLACE_RECORD(FILENM)

      INTEGER DISPLAY,CORAR,LX,RX,TY,BY
      INTEGER RDISP,RAREA,RLX,RRX,RTY,RBY
      CHARACTER*20 AREANM,RAREANM,FILENM

      COMMON /REPL_INFO/  DISPLAY,CORAR,LX,RX,TY,BY,AREANM

      OPEN(UNIT=8,FILE='STORE.DAT',STATUS='NEW')
      OPEN(UNIT=7,FILE=FILENM,READONLY,STATUS='OLD')

      DO 100 KA=1,1000

          READ(7,50,END=110) (RDISP,RAREA,RLX,RRX,RTY,RBY,RAREANM)
50        FORMAT(I2,1X,I2,1X,I4,1X,I4,1X,I4,1X,I4,1X,A20)

          WRITE(6,50) (RDISP,RAREA,RLX,RRX,RTY,RBY,RAREANM)

100   CONTINUE
110   CONTINUE

      CLOSE(UNIT=7)
      OPEN(UNIT=7,FILE=FILENM,STATUS='NEW')
      CLOSE(UNIT=8)
      OPEN(UNIT=8,FILE='STORE.DAT',READONLY,STATUS='OLD')

      DO 200 KB=1,1000

          READ(8,50,END=210) (RDISP,RAREA,RLX,RRX,RTY,RBY,RAREANM)
          IF(RDISP.NE.DISPLAY.OR.RAREA.NE.CORAR)THEN
              WRITE(7,50)(RDISP,RAREA,RLX,RRX,RTY,RBY,RAPEANM)
          ELSE
              WRITE(7,50)(DISPLAY,CORAR,LX,RX,TY,BY,AREANM)
          ENDIF

200   CONTINUE
210   CONTINUE

      CLOSE(UNIT=7)
      CLOSE(UNIT=8)

      RETURN
      END
```

```
*****************************************************************
*                                                               *
*       INIT_EDITOR finds the address of the variable 'drawenable'.*
*    This is the address where the rectangle information contained *
*    in 'EDITOR_BLK' will be sent so that the graphics procedure  *
*    can update the touch area visually.                          *
*       SEND_INFO actually writes the block of data to the address.*
*                                                               *
*****************************************************************

        SUBROUTINE INIT_EDITOR

        INTEGER*4 EDIT_DATA
        INTEGER*4 GGADDR
        INTEGER EDITOR_BLK(18)

        COMMON /EDIT_INFO/ EDITOR_BLK

        EDIT_DATA = GGADDR( 1, '$drawenable$', IERR)

        RETURN

        ENTRY SEND_INFO

        CALL IKBWR( 16 , EDIT_DATA, 18, EDITOR_BLK )

        RETURN

        END
```

```
*****************************************************************
*                                                               *
*                      TOUCH SCREEN INTERFACE                   *
*                           MICROTOUCH                          *
*                                                               *
*****************************************************************
*    Filename: MICROTCH.FOR                                     *
*                                                               *
*       These routines handle all communications with the       *
*    MicroTouch Systems, Inc. touch screen.  Before utilizing   *
*    these routines, the command procedure 'microtch.com' must be *
*    run (@microtch).  This will allocate the touch screen and set *
*    the correct communications parameters.  Be sure that the   *
*    VAX port in the command procedure is correct as this will  *
*    affect all touch screen operations.                        *
*                                                               *
*****************************************************************
*                                                               *
*       INIT_SCREEN will set the MicroTouch screen to hexadecimal *
*    format.  This is recommended since  it provides for a more *
*    direct mapping correlation to typical display coordinate   *
*    systems.  The hexadecimal format provides a 1024x1024 coor- *
*    dinate system, whereas the default decimal setting provides *
*    a 1000x1000 system.  Most display coordinate systems are   *
*    512x512 or 1024x1024.                                      *
*                                                               *
*****************************************************************

       SUBROUTINE INIT_SCREEN

       CHARACTER HEXMD*4

       HEXMD='FH**'
       CALL SNDMSG(HEXMD)

       RETURN
       END

*****************************************************************
*                                                               *
*       INIT_CHAN will establish a channel for touch screen     *
*    communications.  This is accomplisned using the SYS$ASSIGN *
*    system function.  The function is passed the port of the   *
*    desired operation (touch$screen) and returns the channel   *
*    (ICHAN) which is used by all routines which read from or   *
*    write to the touch screen.                                 *
*                                                               *
*****************************************************************
```

```
        SUBROUTINE INIT_CHAN

        INTEGER*4 SYS$ASSIGN
        CHARACTER*12 TERMINAL

        COMMON /CHANNEL/ ICHAN
        COMMON /TERMPORT/ TERMINAL

C********
C
C  Terminal is given the value of the touch screen port which is
C  contained in 'touch$screen'.  This is defined in 'microtch.com'.
C
C********

        TERMINAL='touch$screen'

        JERRFLG=SYS$ASSIGN(%DESCR(TERMINAL),ICHAN,,)
        IF(.NOT.JERRFLG) THEN
            WRITE(*,*) 'ASSIGN FAILURE'
            WRITE(*,*) 'ERROR CODE ',JERRFLG
        ENDIF

        RETURN
        END

*************************************************************************
*                                                                       *
*       GET_TOUCH is the main routine of the touch screen               *
*    interface.  When called, this subroutine will look for data        *
*    from the touch screen and return X and Y coordinate data as        *
*    well as a status code.  Each coordinate datum will be an           *
*    integer value from 0 to 1023.  The status code will be one of      *
*    the following characters:                                          *
*                                                                       *
*                        'T' - Touch                                    *
*                        'E' - Exit                                     *
*                        'N' - No touch present                         *
*                        'F' - Format error                             *
*                        'R' - Range error                              *
*                                                                       *
*    Touch status indicates that the data determine the location        *
*    of a touch.  Exit status indicates that the data are the           *
*    last detected location before the touch left the screen.           *
*    No touch present differs from exit status in that no touch         *
*    data immediately precede it and that the accompanying data         *
*    are meaningless.  Both format and range errors are the result      *
*    of data transmission errors and these data must be ignored.        *
*                                                                       *
*************************************************************************
```

```
      SUBROUTINE GET_TOUCH(X,Y,STATUS)

      BYTE BUFF(18)
      INTEGER*4 TERMMASK(2)
      INTEGER*2 X,Y,POS,MISSCOUNT
      CHARACTER STATUS*1,XCHAR*3,YCHAR*3
      LOGICAL TOUCHFLAG

C********
C
C  System definitions and calls used for I/O.
C
C********

      INCLUDE '($IODEF)'
      INTEGER*4 SYS$QIOW

      INTEGER*2 IOSB(4)

      COMMON /CHANNEL/ ICHAN

C********
C
C  The SYS$QIOW call will purge the typeahead buffer of the touch
C  screen and wait one second for input.  If no data are received,
C  the routine will return a status of 'N' or 'E' to the calling
C  routine.  If data are received, 18 ASCII characters will be read
C  into the BUFF array.  The touch data sent by the Microtouch are
C  in the format <SOH>xxx,yyy,<CR>.  The routine will scan the buffer
C  for the first occurrence of CHAR(1) (<SOH>), and then take the next
C  nine characters as data.  This data is then checked for format
C  errors, transformed to integer coordinate data, checked for range
C  errors, and returned to the calling program with the appropriate
C  status.  When no touch data are detected, a status of 'N' is
C  returned unless touch data were present on the previous scan, in
C  which case 'E' is returned.
C
C********

      STATUS='T'

      DO 75 KI=1,18
75    BUFF(KI)=0

      TERMMASK(1)=0
      TERMMASK(2)=1
      IOPURGE=IO$_READPBLK.OR.IO$M_PURGE.OR.IO$M_TIMED
```

```
      NERRFLG=SYS$QIOW(%VAL(19),%VAL(ICHAN),%VAL(IOPURGE),
     1  IOSB,,,%REF(BUFF),%VAL(18),%VAL(1),%REF(TERMMASK),,,,)

      POS=-1

      DO 76 KJ=1,9
          IF (BUFF(KJ).EQ.'01'X) THEN
              POS=KJ
              STATUS='T'
              MISSCOUNT=0
          ENDIF
76    CONTINUE

      IF (POS.LT.0) THEN
          STATUS='N'
          IF (TOUCHFLAG) THEN
              MISSCOUNT=MISSCOUNT+1
              IF (MISSCOUNT.EQ.2) THEN
                  STATUS='E'
                  TOUCHFLAG=.FALSE.
              ENDIF
          ENDIF
          RETURN
      ENDIF

      IF (BUFF(POS+4).NE.'2C'X.OR.BUFF(POS+8).NE.'0D'X) THEN
          STATUS='F'
          RETURN
      ENDIF

      DO 78 KL=1,3
          XCHAR(KL:KL)=CHAR(BUFF(POS+KL))
          YCHAR(KL:KL)=CHAR(BUFF(POS+KL+4))
78    CONTINUE

      X=INTVAL(XCHAR)
      Y=INTVAL(YCHAR)

      IF(X.LT.0.OR.Y.LT.0) STATUS='R'
      IF(X.GT.1023.OR.Y.GT.1023) STATUS='R'

      TOUCHFLAG=STATUS.EQ.'T'

      RETURN
      END
```

```
*****************************************************************
*                                                               *
*       INTVAL, when passed a character string consisting of 3  *
*    hexadecimal digits, will return the integer value of the   *
*    number.                                                    *
*       The string is examined digit by digit.  The ASCII value *
*    of each digit is determined and then converted to an integer *
*    value by subtracting an appropriate offset.  Each digit is *
*    then multiplied by a power of 16, according to its position, *
*    and then added to the total.  If any characters other than *
*    hexadecimal digits are included in the string, the results *
*    will be in error.                                          *
*                                                               *
*****************************************************************

      INTEGER FUNCTION INTVAL(CHVAL)

      INTEGER*2 PLACE,POWER,ASCVAL

      CHARACTER*(*) CHVAL

      INTVAL=0
      POWER=2

      DO 100 I=1,3
          ASCVAL=ICHAR(CHVAL(I:I))
          IF (ASCVAL.GE.48.AND.ASCVAL.LE.57) THEN
              PLACE=ASCVAL-48     ! Value 0-9
          ELSE
              PLACE=ASCVAL-55     ! Value A-F
          ENDIF

          IF(PLACE.LT.0.OR.PLACE.GT.15)THEN
              INTVAL=-1
              RETURN
          ENDIF

          INTVAL=INTVAL + PLACE*16**POWER
          POWER=POWER-1
100       CONTINUE

      RETURN
      END
```

```
************************************************************************
*                                                                      *
*       FILTER is used by any procedure in which it is desirable       *
*    for touch coordinates to differ by a "significant" amount         *
*    before a change is recognized.                                    *
*                                                                      *
************************************************************************

        SUBROUTINE FILTER(X,Y,OLDX,OLDY)

        INTEGER X,Y,OLDX,OLDY

        IF (ABS(X-OLDX).LT.10.OR.ABS(Y-OLDY).LT.10) THEN
            X=OLDX
            Y=OLDY
        ENDIF

        RETURN
        END

************************************************************************
*                                                                      *
*       SNDMSG will send a parameter-changing command to the           *
*    MicroTouch.  When passed a character string, the routine will     *
*    concatenate this with the command sequence <SOH>command<CR>,      *
*    and then write the command to the screen.  The passed command     *
*    should end with the string '**'.  For instance, the command       *
*    to place the touch screen in hexadecimal is <SOH>FH<CR>.          *
*    This routine should be sent a string of 'FH**' to execute         *
*    this command.  Also note that the terminal port to which the      *
*    touch screen is attached must be set to SPEED=4800, NOECHO,       *
*    and TYPE_AHEAD for writing to screen to be possible.  This        *
*    can be done by executing MICROTCH.COM.                            *
*                                                                      *
************************************************************************

        SUBROUTINE SNDMSG(SPECCMD)

        CHARACTER*45 SPECCMD
        CHARACTER*12 TERMINAL
        CHARACTER*1 COMMAND(45)
        INTEGER*4 SYS$ASSIGN,SYS$QIOW
        INTEGER CMDLEN

        INCLUDE '($IODEF)'

        COMMON /TERMPORT/ TERMINAL

        DATA COMMAND/45*' '/
```

```
        COMMAND(1)=CHAR(1)          ! Begin with <SOH>

        DO 450 IA=1,45
            IF (SPECCMD(IA:IA+1).EQ.'**') GOTO 475
            CMDLEN=IA+2
450     CONTINUE

475     CONTINUE

        DO 500 IB=2,CMDLEN
            COMMAND(IB)=SPECCMD(IB-1:IB-1)
500     CONTINUE

        COMMAND(CMDLEN)=CHAR(13)    ! End with <CR>

        JERRFLAG=SYS$ASSIGN(%DESCR(TERMINAL),ICHAN,,)
        IF(.NOT.JERRFLAG) WRITE(6,*) 'ASSIGN FAILURE'

        KERRFLG=SYS$QIOW(%VAL(4),%VAL(ICHAN),%VAL(IO$_WRITEVBLK),,,,
     1  %REF(COMMAND),%VAL(CMDLEN),,,,)

        RETURN
        END

*****************************************************************
*                                                               *
*       COORDMAP maps the MicroTouch coordinates to the screen  *
*    coordinates used by the Adage-3000.                        *
*                                                               *
*****************************************************************

        SUBROUTINE COORDMAP(X,Y,AX,AY)

        INTEGER X,Y,AX,AY

        AX=X/2
        AY=511-Y/2

        RETURN
        END
```

APPENDIX C

TOUCH EDITOR SAMPLE RUN AND DATA FILE FORMAT

Touch Editor Sample Run

The following is a sample run of the touch editor which displays all necessary terminal inputs required from the user. Inputs are indicated by an asterisk (*). Comments are preceded by an exclamation point (!).

```
        !Render user's graphic display on monitor

        ENTER 1 FOR MAIN MENU, 2 FOR SYMBOLOGY MENU,
        3 FOR STAR MENU, 4 FOR WAYPOINT MENU, OR 0 TO EXIT

*       1

        !Name data file (Do not provide an extension)

        ENTER NAME OF DATA FILE.

*       MAINMU

        IS THIS AN OLD OR NEW FILE? (O/N)

*       N

        PLEASE ASSIGN A NUMBER TO THIS TOUCH DISPLAY.

*       1

        HOW MANY TOUCH AREAS FOR THIS DISPLAY?

*       4

        TOUCH SCREEN AT ONE CORNER OF TOUCH AREA.
        MOVE FINGER TO OPPOSITE CORNER OF AREA.
        WHEN AREA IS SATISFACTORY, REMOVE FINGER.

        DEFINE AREA #1

        !User defines area using touch.
        !Editor assumes definition of area on exit from the screen.

        AREA IS DEFINED. SATISFACTORY? (Y/N)

*       Y
```

```
        !Logical names can be provided to allow for future
        !identification of the data.

        ENTER LOGICAL NAME OF TOUCH AREA.

*       SYMBOLOGY

        DEFINE AREA #2

        AREA IS DEFINED. SATISFACTORY? (Y/N)

*       Y

        ENTER LOGICAL NAME OF TOUCH AREA.

*       STAR

        DEFINE AREA #3

        !At this time, the user unintentionally defines area #3
        !to overlap area #1.

        AREA IS DEFINED. SATISFACTORY? (Y/N)

*       Y

        ENTER LOGICAL NAME OF TOUCH AREA.

*       OVERLAPPING

        !Diagnostic routines warn the user of the error.

        AN OVERLAP ERROR HAS OCCURRED.
        CONFLICTING AREAS: 1,3

        DO YOU WISH TO REDEFINE AN AREA? (Y/N)

*       Y

        ENTER NUMBER OF AREA TO BE REDEFINED.

*       3

        CORRECT AREA #3

        AREA IS DEFINED. SATISFACTORY? (Y/N)

*       Y

        ENTER LOGICAL NAME OF TOUCH AREA.

*       WAYPOINT
```

```
        !If there are no further conflicts, normal operation resumes.

        DEFINE AREA #4

        AREA IS DEFINED. SATISFACTORY? (Y/N)

*       Y

        ENTER LOGICAL NAME OF TOUCH AREA.

*       FLT PLAN DATA

        FORTRAN STOP
```

Data File Format

The following is an example of a data file created by the touch editor. The data indicate these values:

1) Display number
2) Touch area number
3) Lowest X
4) Highest X
5) Lowest Y
6) Highest Y
7) Logical name of area

```
1  1    0  170  172  204 SYMBOLOGY
1  2    0  179  205  229 STAR
1  3    0  165  241  267 WAYPOINT
1  4    0  172  274  319 FLT PLAN DATA
```

APPENDIX D

```
$ !*****************************************************************
$ !
$ !                    MICROTOUCH COMMAND PROCEDURE
$ !
$ !*****************************************************************
$ !
$ !              Command file to be run before using
$ !                  the MicroTouch touch screen
$ !
$ !*****************************************************************
$ !
$ !      Filename: MICROTCH.COM
$ !
$ !      Set the logical variable touch$screen to indicate the
$ !      terminal port to which the MicroTouch is connected.
$ !
$ assign ttc3: touch$screen
$ !
$ !      Allocate the port and set the appropriate parameters.
$ !      1) Baud rate for the MicroTouch is 4800.
$ !      2) Characters sent by the touch screen should NOT be echoed
$ !         back as this can cause unexpected results.
$ !      3) Type_ahead buffer is enabled so that no data is lost.
$ !
$ allocate touch$screen
$ set term/speed=4800/noecho/type_ahead touch$screen
$ exit
```

APPENDIX E

```
/*********************************************************************

                        TOUCH EDITOR GRAPHICS

Filename: subged.g

    This subroutine, when sent data, will draw an unshaded yellow
rectangle to the screen.  Adage screen coordinates are used directly
so that the window and viewport used in the calling procedure have
no effect.  This also allows the rectangle to be drawn without dis-
turbing the display over which it is rendered.

*********************************************************************/

#include "rdefs.h"              /* Include files contain standard   */
#include "rmacs.h"              /* RAP definitions for colors, etc. */


static int drawenable=0;        /* Flag indicating receipt of data  */

static int corners[]={17,       /* Rectangle data sent here         */
                0,0,0,0,
                0,0,0,0,
                0,0,0,0,
                0,0,0,0};


static int edgcb[2]={corners,corners};

static int edcList[]={1,5,9,13,1,0};

static int toucharea[]={LINESEG + edgcb,   /* Touch area is four yellow */
                edcList,                   /* line segments connecting  */
                edcList,                   /* the points listed in data */
                YELLOW,                    /* structure 'corners'       */
                0};

static int edcGlom[200]={1};
static int edvGlom[600]={1};

static int edscene[]={toucharea,0};

subged()
{

    if (drawenable)         /* If data sent then: */
    {
      edvGlom[0]=1;
      render(edscene);      /* Draw rectangle     */
    };
}
```

```
************************************************************************
*                                                                      *
*                        DATA STORAGE ROUTINE                          *
*                                                                      *
************************************************************************
*                                                                      *
*     Filename: RTOUCHDAT.FOR                                          *
*                                                                      *
*        READ_TOUCH_DATA reads data from the specified data file       *
*     (FILENM) and stores the data in arrays that are passed as        *
*     parameters.  (DISPLAY is the number of the display; AREA is      *
*     the number of areas on the display; LX and RX are the left       *
*     and right X coordinate values; and TY and BY are the top and     *
*     bottom Y coordinate values.)  The total number of items or       *
*     touch areas stored in the data file will be returned in the      *
*     variable NUMITEMS.                                               *
*                                                                      *
************************************************************************

      SUBROUTINE READ_TOUCH_DATA(FILENM,DISPLAY,AREA,LX,RX,TY,BY,
     1                           NUMITEMS)

      CHARACTER*(*) FILENM
      INTEGER*4 DISPLAY(*),AREA(*),LX(*),RX(*),TY(*),BY(*)
      integer*4 NUMITEMS

      OPEN (UNIT=7,FILE=FILENM,READONLY,STATUS='OLD')

      I = 1

      DO WHILE(.TRUE.)

20          READ(7,10,END=50) (DISPLAY(I),AREA(I),LX(I),RX(I),
     1                       TY(I),BY(I))
10          FORMAT (I2,1X,I2,1X,I4,1X,I4,1X,I4,1X,I4)

            I = I+1
      END DO

50    CONTINUE

      CLOSE(UNIT=7)

      NUMITEMS = I

      RETURN
      END
```

```
************************************************************************
*                                                                      *
*                    INPUT INTERPRETATION ROUTINE                      *
*                                                                      *
************************************************************************
*                                                                      *
*     Filename: TCHAREA.FOR                                            *
*                                                                      *
*        GET_TOUCH_AREA compares a pair of touch coordinates (TCHX     *
*     and TCHY) to the touch zone coordinate data (LX, RX, TY, and     *
*     BY) read from the data file.  If a zone match occurs, the        *
*     display number (DISPLAY(I)) as well as the area number           *
*     (AREA(I)) of that zone are returned to the calling process in    *
*     the variables DMATCH and AMATCH.  These variables will contain   *
*     a -1 if no match is found.  The number of touch zones stored     *
*     in the data file is also sent as a parameter (ITEMS).            *
*                                                                      *
************************************************************************

      SUBROUTINE GET_TOUCH_AREA(TCHX,TCHY,LX,RX,TY,BY,DISPLAY,
     1                          AREA,ITEMS,DMATCH,AMATCH)

      INTEGER*4 TCHX,TCHY,LX(*),RX(*),TY(*),BY(*)
      INTEGER*4 DISPLAY(*),AREA(*),ITEMS,DMATCH,AMATCH

      DMATCH = -1
      AMATCH = -1
      I = 1

      DO WHILE(I.LE.ITEMS)
          IF(TCHX.GE.LX(I).AND.TCHX.LE.RX(I).AND.
     1       TCHY.GE.TY(I).AND.TCHY.LE.BY(I)) THEN
              DMATCH = DISPLAY(I)
              AMATCH = AREA(I)
              GOTO 50
          END IF
          I = I + 1
      END DO

50    CONTINUE

      RETURN
      END
```

REFERENCES

1. Research Triangle Institute: Reference Manual, RAP, Real-time Animation Package. February 3, 1986.

2. MicroTouch Systems, Inc.: Installation and Operation Guide, MicroTouch Systems Kit, Rev. 3.0. June 1985.

3. MicroTouch Systems, Inc.: Programmer's Guide to the MicroTouch Screen, Rev. 3.0. June 1985.

4. Carroll Touch Technology Corporation: User's Guide, Carroll Touch Input System. 1984.

5. Research Triangle Institute: Manager's Manual, Software Configuration Control and Reporting System. August 28, 1985.

Figure 1. Definition of Touch-Sensitive Zones for a Graphic Display

Standard Bibliographic Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-89136		

4. Title and Subtitle	5. Report Date
An Interactive Editor for Definition of Touch-Sensitive Zones for a Graphic Display	APRIL 1987
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
Burt L. Monroe, III Denise R. Jones	
9. Performing Organization Name and Address	10. Work Unit No.
NASA Langley Research Center Hampton, VA 23665-5225	505-66-11-04
	11. Contract or Grant No.
12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
National Aeronautics and Space Administration Washington, DC 20546	Technical Memorandum
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

In the continuing effort to develop more efficient man-machine communications methods, touch displays have shown potential as straightforward input systems. The development of software necessary to handle such systems, however, can become tedious. In order to reduce the need for redundant programming, a touch editor has been developed which allows a programmer to interactively define touch-sensitive areas for a graphic display. The information produced during the editing process is written to a data file, which can be accessed easily when needed by an application program. This paper outlines the structure, logic, and use of the editor, as well as the hardware with which it is presently compatible.

17. Key Words (Suggested by Authors(s))	18. Distribution Statement
Touch Screen Touch Area Editor Interactive	Unclassified - Limited Subject Category 60

19. Security Classif.(of this report)	20. Security Classif.(of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	37	A03

For sale by the National Technical Information Service, Springfield, Virginia 22161

NASA Langley Form 63 (June 1985)