AFATL-TR-74-187

THEORY OF OPERATION AND USE OF CREATE TAPE SYSTEM, VERSION 2

AIR-TO-SURFACE WEAPONS BRANCH
GUIDED WEAPONS DIVISION

NOVEMBER 1974

FINAL REPORT: JANUARY - SEPTEMBER 1974

AIR FORCE ARMAMENT LABORATORY

AIR FORCE SYSTEMS COMMAND • UNITED STATES AIR FORCE

EGLIN AIR FORCE BASE, FLORIDA

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFATL-TR-74-187	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) THEORY OF OPERATION AND USE OF CREATE TAPE SYSTEM', VERSION 2		5. TYPE OF REPORT & PERIOD COVERED Final Report January - September 1974
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) J. David Murley, Captain, USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Guided Weapons Division (DLMM) Air Force Armament Laboratory Eglin Air Force Base, Florida 32542		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 670B1001
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Armament Laboratory Armament Development and Test Center Eglin Air Force Base, Florida 32542		12. REPORT DATE November 1974
		13. NUMBER OF PAGES 20
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Distribution limited to U. S. Government agencies only; this report documents test and evaluation; distribution limitation applied November 1974. Other requests for this document must be referred to the Air Force Armament Laboratory (DLMM), Eglin Air Force Base, Florida 32542.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES Available in DDC		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Cassette Peripherals Source Programs		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report describes the theory of operation and use of the Create Tape System, Version 2. This CTS2 system allows creation and editing of FORTRAN, ALGOL, and Assembly Language programs for use by the HP 9830 Calculator when acting as a paper tape reader for the HP 2100 Mini-Computer		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

UNCLASSIFIED

PREFACE

This report describes the theory of operation and use of the Create Tape System, Version 2. The work was performed in-house in support of Project 670B1001 during the time period January - September 1974 at the Air Force Armament Laboratory, Armament Development and Test Center, Eglin Air Force Base, Florida 32542.

This report has been reviewed and is approved for publication.

FOR THE COMMANDER:

[illegible]

JOHN W. JOHNSON
Deputy Chief, Guided Weapons Division

TABLE OF CONTENTS

(The reverse of this page is blank)

INTRODUCTION

Create Tape System 2 (CTS2) is a part of a larger software/hardware integration package designed for the Mobile Engineering Test System (METS). This integration package uses the Hewlett-Packard 9830A programmable calculator to replace the paper tape peripherals and the teletype in Hewlett-Packard 2100 Mini-Computer systems. CTS2 allows writing and editing of source programs in FORTRAN, ALGOL, or Assembly Language. Furthermore, CTS2 can function in a stand-alone mode so that any properly equipped calculator can be used to write source programs for the METS.

CTS2 requires the following:

9830A Calculator with Matrix ROM, extended IO ROM, and 8K Memory

9866A Thermal Printer

9865A Cassette Memory

CTS2 Special Function Key Overlay

CTS2 Cassette Tape (files 0 and 1 of 9830/2100 Interface Tape)

Premarked cassette tape for source programs

The 9865A Cassette Memory must be set to select Code 5 in accordance with instructions in the Pheripheral Manual.

Source programs are entered from the 9830 Calculator keyboard a line at a time. Each line, as it is entered, may be edited using standard features of the 9830 Calculator. Once entered, a line may not be altered. It may, however, be replaced by a new line, deleted, or exchanged with some other line. Moreover, all or selected portions of a program may be listed and, at any time, the program may be stored on a cassette tape file.

The calculator assigns a line number to each line as it is entered. The line number is added for editing purposes only and is not part of the line itself nor is it transmitted to the 2100 Mini-Computer. Line numbers should not be confused with statement numbers in FORTRAN or with Labels in Assembly Language.

The reader is assumed to be familiar with 9830 Calculator (to the extent of knowing how to operate the keyboard and to load cassettes) and also with cassette tape marking if no pre-marked tapes are available. Keys or combinations of keys to be pressed are set off by dashes. Thus, -PRESS-RUN- means to press the key labeled RUN. PRESS-LOAD 0, 10, 10- means to press the keys labeled LOAD then 0 then 10, then 10, etc. Dashes are used only as a notational convenience in this report and are not to be typed on calculator keyboard. Labels for the special function keys are located under each key on the CTS2 special function key overlay, which should be in place when using CTS2. A facsimile of the overlay is shown in Appendix A.

The remainder of this report is divided into two parts. Part I is an operators manual for using CTS2. Part II is a description of the philosophy and design considerations which went into CTS2 and requires a much more extensive knowledge of 9830 Calculator operation.

Appendix B consists of an operation flowchart that depicts operator actions in using CTS2. This flowchart should be useful in understanding both parts of this report. Appendix C is the CTS2 program listing.

PART I

OPERATION OF CTS2

1. INITIAL START-UP

Begin by placing the cassette tape labeled 9830/2100 INTERFACE 1 in the calculator cassette drive, then press -SCRATCHA-, -LOAD 0, 10, 10-, and -EXECUTE-. The calculator will respond with IF EDIT, INPUT F#, O.W.-1 on the display. This statement is short for IF YOU WISH TO EDIT A PROGRAM, INPUT THE FILE NUMBER ON WHICH IT IS STORED, OTHERWISE ENTER -1. To edit an existing program, place the cassette containing it in peripheral cassette drive, and type the desired program file number followed by -EXECUTE-. To enter a new program, type -1 and press -EXECUTE-. After a short wait, the calculator will display CTS2 COMMAND?. The calculator is now ready to begin entry of a new program or to edit an existing program. Appendix B is a flow chart for the overall CTS2 operation.

Each of the CTS2 special function keys is discussed in the following paragraphs. The three involving program line entry, NEXT, REPLACE, and INSERT BEFORE are treated first, since their operation differs from that of the other CTS2 commands.

2. LINE ENTRY COMMANDS

Figure 1 is a flow chart for using line entry commands. An abbreviated form of this flow chart is on the CTS2 special function key overlay itself (Appendix A). The important thing to remember is that -RUN- must be pressed before any of the keys NEXT, REPLACE, or INSERT BEFORE. This is indicated on both flow charts.

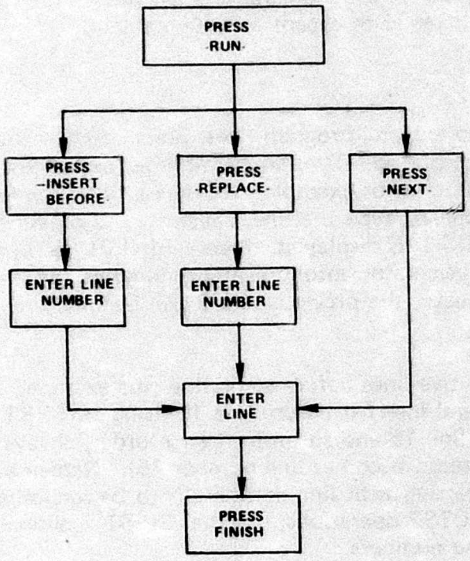


Figure 1. Line Entry Flow Chart

3. NEXT

To enter the first line of a program or to add a line at the end of lines already entered, use NEXT. This is the function normally used to enter a program. Consecutive line numbers are assigned to program lines as they are entered. For example, the calculator assigns the number 1 to the first line entered or, if lines 1 through 15 are already in the calculator, 16 is assigned to the next line entered.

Start the operation by pressing -RUN- and then the special function key -NEXT-. The calculator responds with ENTER LINE on the display. Type in the desired line, and check it carefully for errors. All of the editing functions of the calculator are available for correcting errors at this time. Once -END OF LINE- has been pressed, the line editing functions can no longer be used. When the line has been correctly typed, press END OF LINE-. The calculator will print the line, together with its line number, and will display PRESS FINISH. Press -FINISH-; and after a second or so, CTS2 COMMAND? will be displayed. This means the calculator is waiting for another CTS2 command.

4. REPLACE

To change a line previously entered, press -RUN-, and then press -REPLACE-. The calculator will display # OF LINE TO BE REPLACED?. Type the number of the line to be replaced, and press -EXECUTE-. ENTER LINE? is then displayed. Type the line in, and after checking for mistakes, press -END OF LINE-; at this time, the line and its number are printed. When PRESS FINISH appears on the display, press -FINISH-. CTS2 COMMAND? will return when the calculator is ready to accept a new command.

5. INSERT BEFORE

To insert a line between two existing program lines, press -RUN- then -INSERT BEFORE-. INSERT BEFORE LINE #? is displayed. Type in the number of the line that the new line is to precede, and press -EXECUTE-. (For example, to insert a line between lines 4 and 5, type in 5.) When ENTER LINE appears, type the line, check it, and press -END OF LINE-. The line is printed, and PRESS FINISH is displayed. Press -FINISH- after a short wait, CTS2 COMMAND? will return. The calculator automatically renumbers the lines after the line is inserted. Thus, in the example used, the previous line 5 will become line 6, and the inserted line will be line 5.

To insert a block of consecutive lines before some line (for example, before line 15), start with the last line in the block, and insert it before line 15 using INSERT BEFORE. Then insert the next to last line before line 15 and so on in reverse order, always inserting before line number 15 (each line as it is entered becomes line number 15). Remember that this operation has caused all of the old lines starting with line number 15 to be renumbered, so if they are to be referred to in subsequent CTS2 operations, use the SLIST command to obtain a list of the program with the new line numbers.

6. DELETE

To delete either a single line or a block of consecutive lines, press -DELETE-. DELETE: ENTER FIRST AND LAST LINE #S? is then displayed. If only a single line is to be deleted, type its line number twice, separated by a comma, then -EXECUTE-. For example, to delete line 5, press -5, 5- then press -EXECUTE-. On the other hand, to delete a block of lines, type the line number of the first line to be deleted, a comma, and the line number of the last line to be deleted. Then press -EXECUTE-. That is, to delete lines 6 through 9 inclusive, type -6, 9- , then -EXECUTE-. Both of the lines whose numbers were entered will be deleted, together with all lines in between them. The calculator will automatically renumber all program lines after the deleted block to fill in the gap created by deletion. The operation is complete when CTS2 COMMAND? appears.

7. EXCHANGE

To exchange a pair of lines, press -EXCHANGE-. The display will inquire ENTER EXCHANGE LINE #S?. Type the line numbers, separated by a comma, of the two lines to be interchanged, and press -EXECUTE-. When CTS2 COMMAND? returns, the two lines will have traded places, and the next CTS2 command can be given.

8. SAVE

To store a program or a portion of a program on a cassette, press -SAVE-; note that a premarked cassette must be in the peripheral cassette drive. When the display shows FILE # FOR STORAGE?, type the file number on which the program is to be stored and press -EXECUTE-. When the program has been stored, the file on which it is stored is printed along with the program length. CTS2 COMMAND? indicates the calculator is ready to proceed. A long program should be stored from time to time as it is entered, since any momentary power failure will erase the program from the calculator memory.

9. SOURCE LIST

To list program lines already in the calculator memory, press -SLIST-, and LIST: ENTER FIRST AND LAST LINE #S? will appear on the display. Type the line numbers of the first and last lines, separated by a comma, and press -EXECUTE-. Instead of typing the last line number, some large number such as 999 may be used. This will result in a print-out of all program lines in the memory, starting with the first line number entered. The listing produced will have line numbers on the extreme left of each line. The actual line itself begins in column six. Remember that the line numbers are attached for editing purposes only and are not parts of the program lines. Line numbers are not transmitted to the 2100 Mini-Computer. When the listing is complete, CTS2 COMMAND? returns to the display.

10. RESTART

Press -RESTART- to erase all program lines currently entered or to load another program from a cassette. The display shows IF EDIT ENTER F# , O.W. -1. From this point, follow the instructions given under the INITIAL START-UP section of this report. If -RESTART- is pressed inadvertently, any program lines already in the machine will be lost. For this reason, -RESTART- is located in the upper case so that -SHIFT- must be held down while -RESTART- is pressed.

11. TYPING AID KEYS

The typing aid keys are not CTS2 commands but instead provide conveniences and special symbols. Pressing -TAB 6- inserts six spaces at the start of a program line useful for unnumbered FORTRAN statements or unlabeled Assembly Language statements as both then begin in column 7.

The remaining typing aids are special symbols for ALGOL programs and are typed by holding down -SHIFT- and then pressing the special function key below which they appear. The square brackets [and] are displayed and printed as such. The reverse slant \ is printed as it appears and is displayed as a 5 by 7 array of dots. The left arrow ← is displayed as lazy T ⊢ and printed as an underline _ . The symbol @ is obtained by holding down -SHIFT- and pressing -RES-.

12. PROGRAMMING LIMITATIONS

Program lines are limited to 50 characters (including spaces). The maximum number of lines is 220. Quotes may not be used in ALGOL or FORTRAN programs; instead, use an apostrophe wherever quotes would have normally appeared. The ASCII code for quotes will be transmitted when an apostrophe is encountered. This means that the apostrophe cannot be used for any reason other than replacing a quotation mark.

13. PREMARKED TAPES

As previously noted, the tape on which a source program is to be stored should be premarked. The following convention has been established. Even numbered files are used for storing source programs. The odd numbered file following each even file is used to store the corresponding object program. That is, if a certain source program is stored on file number 0, the object program that it produces upon compiling is stored on file number 1. If the number of program lines is known ahead of time, the size file required for the source program is the number of lines plus 8 times 26. Marking the next file two-thirds that size will generally be large enough for the object program. For large programs, mark the even numbered files of a cassette for file length 6000 and the odd files with length 3800. This marking arrangement will be large enough to accept the longest program that can be written using CTS2. It is recommended that programs in a developmental state be stored on a scratch tape with maximum sized files and then be transferred to a tape with custom file sizes when the program reaches production status. For further information on cassette tapes, consult the 9830 Calculator Operating Manual.

14. ERRORS AND ERROR MESSAGES

When entering a line and -END OF LINE- has just been pressed, REDUCE LINE BY 5 ENTER LINE may appear. This means the maximum line length of 50 characters has been exceeded by 5. Rewrite the program line to reduce the length by 5. Type it in, and press -END OF LINE- and -FINISH-.

Another error message which may appear is ERROR 73 IN LINE 130. Often this results from holding the shift key down while typing an alpha character, i.e., A through Z. This creates a letter which is not recognized by CTS2 although the usual alpha character is printed and displayed. Re-enter the line, being careful not to hold the shift key down when it is not needed.

Not pressing -RUN- before pressing -NEXT-, -INSERT BEFORE-, or -REPLACE- results in ERROR 41 IN LINE 30. Press -END-, -RUN-, and then the desired line entry special function key.

For other errors, no specific advice can be given. Just press -END- and continue as though CTS2 COMMAND? has been displayed. If this fails, press -RESTART-, and reload or retype the source program.

15. COPYING SOURCE PROGRAMS

CTS2 may be used to produce additional copies of source programs on other tapes by first loading the source program into the calculator as if to edit it. Then place the cassette onto which the program is to be copied in the peripheral cassette drive and use the SAVE CTS2 command.

16. A TYPICAL USE OF CTS2

Assume that a new FORTRAN program is to be run on the 9830/2100 system. The tape labeled 9830/2100 INTERFACE is placed in the calculator cassette drive, and a premarked tape is placed in the peripheral cassette drive. The premarked tape has files 4 and 5 available for this program. After -SCRATCHA- is pressed, -LOAD 0, 10, 10- is pressed to start CTS2. When IF EDIT, INPUT F# O.W. -1 appears, -1 is entered since a new program is to be input. After a short wait, CTS2 COMMAND? is displayed. The first line FTN, B, T is entered by pressing -RUN-, -NEXT-, and then typing -FTN, B, T- when ENTER LINE appears. The first 20 lines of the program are entered in similar fashion using the NEXT command. For safety, -SAVE- is pressed, and the calculator responds with FILE # FOR STORAGE?; 4 is entered, and the part of the program entered so far is stored on cassette file number 4, a file previously marked with a length of 6000 words. How often SAVE is used is determined by the user but approximately every 20 lines is recommended. When storing is complete, NEXT is used repeatedly to enter the next 20 program lines, SAVE is again used, and so on, 20 lines at a time until the whole program has been entered.

Each line has been printed as it was entered, so a listing of the program is available. This listing is checked for errors, and if none are detected, -SAVE- is pressed to store the program which is then input to the 2100 Computer using the 9830 Calculator as a source tape photoreader. Since this is the first time the program has been compiled, many error messages maybe printed out. The source program listing is again examined, and notations made on it to correct errors. CTS2 is reloaded into the calculator, and this time when IF EDIT, INPUT F# O.W. -1 appears, 4 (the file on which the source program was stored) is entered. When the source program has been loaded, CTS2 requests a command. The functions REPLACE, INSERT BEFORE, DELETE, and EXCHANGE are used to correct the FORTRAN program. Changes are made to the program from the bottom up, since any time a line is inserted or deleted, all lines after it are renumbered; hence, the line numbers on the old listing no longer correspond to the correct lines. If the changes were numerous, -SLIST- would be pressed to get an updated listing of the program to work from. When the corrections have all been made, -SAVE- is pressed; and after the corrected program is stored, SLIST is again used to obtain a listing of the corrected program. Hopefully, the program now compiles correctly, and its object program is stored on file 5 just after the source program on file 4.

PART II

THEORY OF OPERATION

1. OVERVIEW

The CTS2 program sought to solve the problem of storing ASCII program lines in the 9830 Calculator. The form of storage had to be suitable for editing and for output to the 2100 Mini-Computer, and since input is made from the calculator keyboard, the data had to be initially in the form of a string variable. The maximum length of a string is 255, which would be adequate to contain a single line; however, only 26 such strings are available. In addition to this restriction, complicated indexing to input or output these lines would also be required. In particular, indexing would have to be accomplished by passing control to a different program line for each string variable. Packing more than one line into a single string variable produces additional program lines and an even more unwieldy indexing scheme. From these considerations, it was clear that program storage would have to be in an integer array. Further, to make rapid line editing possible, a separate index array was incorporated.

2. FOUR ROLES OF THE SPECIAL FUNCTION KEYS

First, the special function keys are vital to CTS2 operation because they allow the 9830 Calculator memory to be viewed by two different COM statements. It is this feature that allows rapid conversion between the string and integer data types. The three line entry keys, INSERT BEFORE, REPLACE, and NEXT, each have a string variable in their COM statements whereas the main program has an integer array. Pressing -RUN- and then a line entry key causes the calculator to view the memory locations as string data. While in this mode, the desired program line is input into the string variable. The FINISH key is pressed to execute a RUN command which returns control to the COM statement in the main program; thus, converting the line into integer data.

This use of FINISH is an example of the second role of the special function keys. That is, causing a calculator command to be executed. The keys DELETE, SLIST, RESTART, FINISH, SAVE, and EXCHANGE all execute a RUN command at a particular line number of the main program corresponding to their function.

The third role of the special function keys is to store a function subroutine where it is accessible to the programs stored on the line entry keys. This function, FNI, is an input and conditioning routine for entering the string variable, A$.

In their fourth role, the special function keys are used as typing aids for spacing the start of unlabeled program lines and for typing special characters not present on the 9830 keyboard.

3. STRING INPUT

Source program lines are input from the keyboard into a string array. Because quotation marks are used as string delimiters, they cannot be part of input strings; instead, the apostrophe is used in place of quotation marks. Whenever the interface program CTS2 PHOTO READER encounters apostrophes, it replaces them by the ASCII code for quotation marks. This slows the output, but the convenience of quotes in FORTRAN outweighed this disadvantage.

4. STRING TO INTEGER CONVERSION

Once a program line has been input into a string variable, it must be converted into integer format for storage in the program data array. Character by character decoding was used initially but was extremely time consuming. This difficulty was overcome by overlaying the input string array by a one dimensional integer array using the COM statement. Two additional keys must be pressed for each line entry, but this was judged to be less objectionable than waiting 10 seconds or more between line entry for the conversion to take place.

5. PROGRAM LINE STORAGE ARRAY

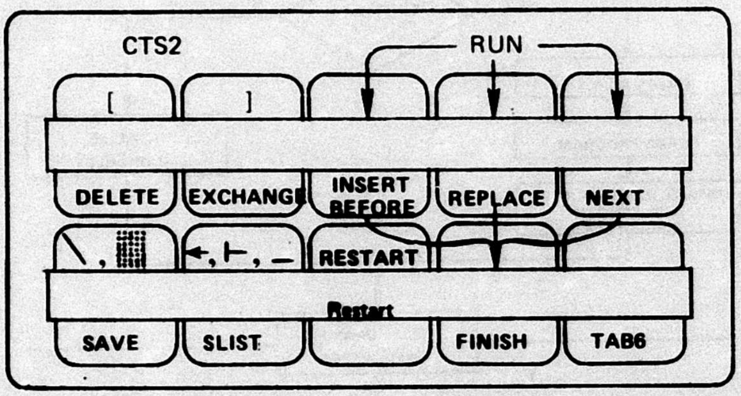
Program lines are stored in an integer array dimensioned, S1(229,26). The first nine rows of this array are used primarily for storing a folded one dimensional index array. The remaining 220 rows are used to store source program lines. One line is stored in each row, together with its line length, which is the last element of the row. Since each row then has 25 integer variables available for program line storage, a program line may have a maximum of 50 characters (two alpha-numeric characters are stored per integer array variable). This means that some space is wasted for short program lines. A denser packing was considered, but, with a maximum one-dimensional array size of 256, indexing becomes complex and time consuming.

Line length is included with each line for the benefit of the output program, which generates a carriage return/line feed whenever the proper number of characters has been transmitted. Line length is specified in number of words. Each integer array variable is one 16 bit word. When an input line has an odd number of characters, the last word is padded with the ASCII code for BLANK; thus, line length is always a whole number.

Program lines need not appear in sequential order in the S array. An index array, dimensioned I1(220), records the ordering of program lines within the S array. For example, the row number in S of the first line is the value of I(1). The second line is in row number I(2) and the n^{th} line in row I(N). This indexing scheme is used because it affords rapid line deletion and insertion, as opposed to an unindexed array in which the lines themselves must be manipulated. When programs are stored or retrieved, the index array is packed into or unpacked from the first nine rows of S. The first element of each of these rows is used as a program variable or left blank. In other words, only the last 25 locations of each row are used to store index information.

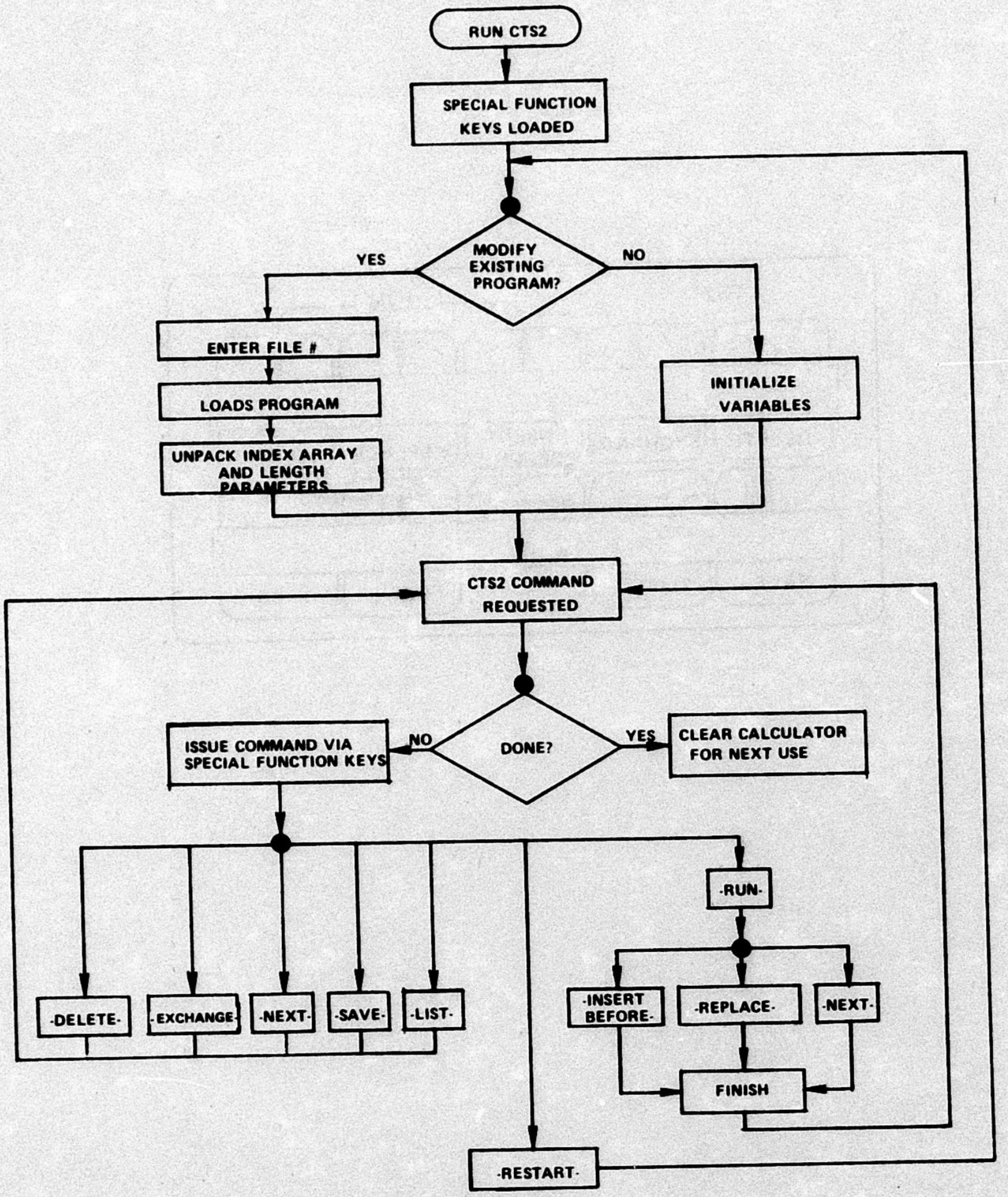
APPENDIX A

SPECIAL FUNCTION KEY OVERLAY



APPENDIX B

OPERATION FLOW CHART



● INDICATES BRANCH DETERMINED BY USE FROM KEYBOARD BY PRESSING

APPENDIX C
PROGRAM LISTING

```
10 COM D[1],T[49],I[220],D1,D2,R,T,S[229,26],J[220],C[64],D$[50]
15 REM THE HURRICANE 'CARMEN' IMPERILED [illegible] SEP 74
20 LOAD  KEY 1
30 D[1]=1
40 REDIM S[229,26]
50 DISP "IF EDIT, INPUT F#, O.W. -1";
60 INPUT N
70 IF N<0 THEN 130
80 LOAD  DATA #5,N,S
90 D1=FNT(1)
100 R=S[1,1]
110 T=S[3,1]
120 GOTO 850
130 FOR I=1 TO 220
140 I[I]=I+9
150 NEXT I
160 R=T=1
170 GOTO 850
180 DEF FNT(X9)
190 D1=1
200 FOR I=1 TO 9
210 FOR J=2 TO 26
220 IF X9 THEN 240
230 S[I,J]=I[D1]
240 I[D1]=S[I,J]
250 D1=D1+1
260 IF D1=221 THEN 290
270 NEXT J
280 NEXT I
290 RETURN 0
300 DEF FNL(X)=X*(X<R)+(R-1)*(X >= R)
310 FOR J=1 TO D1
320 S[D2,J]=T[J]
330 NEXT J
340 S[D2,26]=D1
350 R=R+D[1]
360 S[3,1]=T=T*(T>R)+R*(T <= R)
370 D[1]=1
380 GOTO 850
390 DISP "DELETE: ENTER 1ST & LAST LINE #S";
400 INPUT N,L
410 L=FNLL
420 D1=L-N+1
430 V=R-D1
440 D2=R-1-L
450 MAT J=I
460 FOR I=N TO V-1
470 I[I]=J[I+D1]
480 NEXT I
490 FOR I=V TO R-1
500 I[I]=J[I-D2]
510 NEXT I
520 R=V
530 GOTO 850
540 DISP "ENTER EXCHANGE LINE #S";
```

```
550 INPUT N,L
560 D1=I[L]
570 I[L]=I[N]
580 I[N]=D1
590 GOTO 850
600 DISP "FILE # FOR STORAGE";
610 INPUT N
620 S[1,1]=R
630 REDIM S[8+T,26]
640 D1=FNT0
650 STORE  DATA #5,N,S
660 PRINT "PROGRAM IN FILE"N"LENGTH"(8+T)*26"WORDS"
670 REDIM S[229,26]
680 GOTO 850
690 C$=" !"#$%&'()*+,-./0123456789:;<="
700 C$[31]=">?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_"
710 DISP "LIST: ENTER 1ST AND LAST LINE #S";
720 INPUT N,L
730 PRINT LIN2
740 FOR I=N TO FNLL
750 D1=I[I]
760 FOR J=1 TO S[D1,26]
770 D2=BIAND(ROT(S[D1,J],8),127)-31
780 D$[2*J-1]=C$[D2,D2]
790 D2=BIAND(S[D1,J],127)-31
800 D$[2*J]=C$[D2,D2]
810 NEXT J
820 PRINT I;D$
830 NEXT I
840 PRINT LIN2
850 DISP TAB8"CTS2 COMMAND?"
860 END
```

```
*RUN390*
```

```
*RUN540*
```

```
10 COM DI[2],A$[80],II[220],D1,D2,R
20 DISP "INSERT BEFORE LINE #";
30 INPUT N
35 D2=I[R]
40 D1=FNIN
70 FOR I=R TO N+1 STEP -1
80 I[I]=I[I-1]
90 NEXT I
100 I[N]=D2
110 DISP TAB9"PRESS FINISH"
120 END
```

```
10 COM DI[2],A$[80],II[220],D1,D2,R
20 DISP "# OF LINE TO BE REPLACED";
30 INPUT N
35 D2=I[N]
40 D1=FNIN
60 D[1]=0
70 DISP TAB9"PRESS  FINISH"
80 END
```

```
10 COM DI[2],A$[80],II[220],D1,D2,R
20 D2=I[R]
25 D1=FNIR
45 DISP TAB9"PRESS FINISH"
55 END
```

```
*RUN600*
```

```
*RUN310*


*


*[


*]


*\


*_


*RUN30*



10 DEF FNI(X9)
20 DISP "ENTER LINE";
30 INPUT A$
35 PRINT X9;A$
40 L=LEN(A$)
50 IF L<51 THEN 90
60 DISP "REDUCE LINE BY"L-50"  RE";
80 GOTO 20
90 IF 1-L+INT(L/2+0.01)*2 THEN 120
100 L=L+1
110 A$[L]=" "
120 RETURN L/2
130 END
```

INITIAL DISTRIBUTION

USAF (SAMI)	1
ASD (ENYS)	1
AFWL (LR)	1
AUL (AUL-LSE-70-239)	1
DDC (TC)	2
Ogden ALC (MMNOP)	2
AFIS (INTA)	1
448 SMS (Capt Murley)	2
AMC (AMSMI-REO/Don Smith)	1
Hewlett-Packard (Bob McCoy)	1
ASD (ENYEHM)	1
TAWC (TRADOCLO)	2
AFATL (DL)	1
AFATL (DLMT)	10
AFATL (DLMA)	2
AFATL (DLMM)	2
AFATL (DLOSL)	2

(The reverse of this page is blank)